НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено
Завідувач кафедри
_____ *С.Г.Стіренко*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект

освітньо-кваліфікаційного рівня « бакалавр »

з напряму підготовки(спеціальності) 6.050103 «Програмна інженерія»

на тему «Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android»

Виконав: студент *4* курсу, групи *ІП-53*
Голуб Роман Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Алещенко О. В.* _____
(прізвище, ім’я, по батькові) (підпис)

Консультант нормоконтроль *Симоненко В.П.* _____
(назва розділу) (прізвище, ім’я, по батькові) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

Обчислювальної техніки
(повна назва кафедри)

Освітньо-кваліфікаційний рівень **бакалавр**

Напрям підготовки **6.050103 « Програмна інженерія »**

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(ініціали, прізвище) (підпис)

“___”_____2019р.

ЗАВДАННЯ

на бакалаврську дипломну роботу студента

Голуб Роман Олександрович
(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android
керівник проекту (роботи) *Алещенко Олексій Вадимович, старший викладач*,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23” 04 2019 року №1180-С
2. Термін здачі студентом закінченого проекту (роботи) 20 червня 2019р.
3. Вихідні дані до проекту (роботи) технічна документація, теоретичні дані та функціонал для виконання замовлень в системі придбання товарів
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) Опис та аналіз предметної області, аналіз аналогів у даній області, дослідження основних вимог до ПЗ, конструювання архітектури системи та створення компонентів системи придбання товарів з вискористанням місцезнаходження користувача, інструкція користувача
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) діаграма класів, структурна схема системи, узагальнена схема процесу придбання товару, узагальнена схема процесу фільтрації товару

6. Консультанта проекту (роботи), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	Затвердження теми роботи	10.12.2018 - 15.12.2018	
2.	Вивчення та аналіз завдання	15.12.2018 - 15.03.2019	
3.	Розробка архітектури та загальної структури системи	15.03.2019 - 25.03.2019	
4.	Розробка структур окремих підсистем	25.03.2019 - 05.04.2019	
5.	Програмна реалізація системи	05.04.2019 - 15.04.2019	
6.	Оформлення пояснювальної записки	15.04.2019 - 25.05.2019	
8.	Передзахист	29.05.2019	
9.	Захист	20.06.2019	

Студент-дипломник _____
(підпис)

Керівник роботи _____

(підпис)

	Формат	Позначення	Найменування	Кільк сторінок	№ екземпляру	Примітка
1	А4		Завдання на дипломний проект	2		
2	А4	ІАЛЦ.467200.001 ВП	Відомість на дипломний проект	1		
3	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	4		
4	А4	ІАЛЦ.467200.003 ПЗ	Пояснювальна записка	80		
5	А4	ІАЛЦ.467200.004 Д1	Діаграма класів	4		
6	А4	ІАЛЦ.467200.005 Д2	Діаграма компонентів	1		
7	А4	ІАЛЦ.467200.006 Д3	Діаграма послідовності	2		
8	А4	ІАЛЦ.467200.007 Д4	Лістинг програми	42		
9	А4	ІАЛЦ.467200.008 Д5	Скриншоти програми	6		
10	А4	ІАЛЦ.467200.009 Д6	Алгоритм програми	1		

Изм	Лист	№ докум.	Подп	Дата	ІАЛЦ.467200.001 ВП		
Розробив		Голуб Р.О.			Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android		
Перевір.		Алещенко О.В.					
						НТУУ «КПІ» ФІОТ ІП-53	
Н. контр.		Сімоненко В.П					
Затв.		Стіренко С. Г					

АНОТАЦІЯ

В дипломній роботі розроблено платформу під операційну систему Android для покращення ефективності пошуку необхідних товарів у сфері спорту, за допомогою використання місцезнаходження користувача, розширеної та гнучкої системи фільтрів, актуалізації інформації за допомогою використання платформи Firebase для обробки аутентифікаціїї користувачів, та використання віддаленої бази данних, виокремлення товарів з найвищою кількістю купівель, а також рекламних акцій.

Платформа дозволяє продивитися асортимент товарів конкретної категорії, конкретного магазину, відфільтрувати товари по характеристикам та субхарактеристикам, отримувати нотифікації про гарячі пропозиції при проходженні в радіусі 1 кілометра від конкретного магазину, продивитися інформацію про конкретний магазин, продивитися інформацію про конкретний товар, в тому числі продивитися серію фотографій, відео та характеристик про цей товар. Мобільний додаток був розроблений на мові Kotlin, яка являється однією з основних мов для розробки нативних додатків під Android. Версії, що підтримуються 5.1 та вище, у середовище для розробки Android Studio v3.4.

Для використання користувацької інформації в додатку використовуються uses-permissions(користувацькі дозволи), які підтверджуються користувачем при спробі використовувати функціонал, що залежний від цих користувацьких дозволів. У випадку, коли користувач не підтверджує включення конкретних користувацьких дозволів, платформа обмежує користувача у використанні цього функціоналу.

КЛЮЧОВІ СЛОВА: ЕЛЕКТРОННА КОМЕРЦІЯ, АВТОМАТИЗАЦІЯ ПРОЦЕСІВ, ПРОФАЙЛІНГ.

ABSTRACT

I developed platform for mobile operation system Android in my diploma for increasing search efficiency needed stuffs in sport sphere, with help of checking user location, wide and flexible filters, actualizing information with using Firebase platform for processing user authentication, using remote database, showing top sold stuffs and discount offers.

The platform allows go through the range of stuffs in specific category, specific store, filter stuffs by characteristics and subcharacteristics, get notifications about hot offers while passing the store in radius less than 1km, look through the information about specific store, specific stuff, including photo series, movie series and characteristics about specific stuff.

Mobile app was developed with Kotlin programming language, one of the basic languages for developing native apps for Android. Supported Android versions — 5.1 and higher, in IDE Android Studio v3.4.

For using user`s data I use uses-permissions, which are accepted by user in case of using functions that are related to this uses-permissions. In case of declining uses-permission, platform`ll restrict user to use this functions.

KEYWORDS: E-COMMERCE, AUTOMATION, PROFILING

АНОТАЦИЯ

В моей дипломной работы разработанна платформа под операционную систему Android для улучшения эффективности поиска необходимых товаров в сфере спорта, с помощью использования местонахождения пользователя, расширенной и гибкой системы фильтров, актуализации информации с помощью использования платформы Firebase для обработки аутентификации, использования удаленной базы данных, выделения товаров с высшим количеством покупок, а также рекламных акций.

Платформа разрешает просмотреть ассортимент товаров конкретной категории, конкретного магазина, отфильтровать товары по характеристикам и субхарактеристикам, получать нотификации о горячих предложениях при прохождении в радиусе 1 километра от конкретного магазина, просмотреть информацию про конкретный магазин, конкретный товар, в том числе просмотреть серию фотографий, видео и характеристик этого товара. Мобильное приложение было разработано на языке Kotlin, который является одним из основных языков для нативной разработки под Android. Версии, что поддерживаются — 5.1 и выше, в среде для разработки Android Studio v3.4.

Для использования пользовательской информации в приложении используются uses-permissions(пользовательские разрешения), которые подтверждаются пользователем при попытке использовать конкретный функционал, который зависит от этих пользовательских разрешений, В случае, когда пользователь не подтверждает конкретное пользовательское разрешение, платформа ограничивает пользователя от использования данного функционала.

КЛЮЧЕВЫЕ СЛОВА: ЭЛЕКТРОННАЯ КОММЕРЦИЯ, АВТОМАТИЗАЦИЯ, ПРОФАЙЛИНГ

Пояснювальна записка
до дипломного проекту

на тему: «Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android»

Київ – 2019 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ..……..2
2. ПІДСТАВИ ДЛЯ РОЗРОБКИ……..…….....2
3. МЕТА ТА ПРИЗНАЧЕННЯ РОБОТИ…….……..3
4. ДЖЕРЕЛА РОЗРОБКИ…….....3
5. ТЕХНІЧНІ ВИМОГИ……...3
5.1. Вимоги до продукту, що розробляється …….....3
5.2. Вимоги до програмного забезпечення…….……...3
5.3. Вимоги до апаратної частини…….....4
6. ЕТАПИ ДЛЯ РОЗРОБКИ……....4

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку мобільного додатку «Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android». Область застосування: прогресивна платформа для покращення ефективності пошуку необхідних товарів у сфері спорту, за допомогою використання місцезнаходження користувача, розширеної та гнучкої системи фільтрів, актуалізації інформації за допомогою використання платформи Firebase для обробки аутентифікаціїї користувачів, та використання віддаленої бази данних, виокремлення товарів з найвищою кількістю купівель, а також рекламних акцій.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Завдання на виконання дипломного проекту освітньо-кваліфікаційного рівня «бакалавр», затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут».

Була поставлена задача розробити програмне забезпечення, яке дозволяло б підвищити ефективність пошуку необхідних товарів у сфері спорту, за допомогою розширеної системи фільтрів, виокремлених товарів з найвищою кількістю купівель та використання місцезнаходження користувача.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного мобільного додатку є економія часу для користувача та підвищення ефективності пошуку необхідних товарів. Призначення додатку: покращення комунікації спортивного бізнесу та споживача, діжиталізація

спортивного бізнесу, що дозволить зменшити кількість ресурсів бізнесу та користувачів для пошуку один одного.

4. ДЖЕРЕЛА РОЗРОБКИ

На сучасному ринку існують web аналоги такі як prom.ua, але дана платформа дозволяє постійно комунікувати з користувачем через знаходження платформи у мобільного девайсі користувача та можливості у будь-який момент відправити актуальну інформацію користувачу.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до продукту, що розробляється

- Захищеність даних, зумовленій операційною системою Android.
- Можливість фільтрувати товари за характеристиками та субхарактеристиками.
- Використання місцезнаходження користувача для забезпечення користувача актуальними даними про актуальні пропозиції.
- Забезпечення користувача нотифікаціями з інформацією про гарячі пропозицію при проходженні в радіусі 1 кілометра від конкретного магазину.
- Можливість оформити замовлення в різних магазинах.
- Можливість заплатити за замовлення.
- Забезпечення мультидевайсності за допомогою винесення користувацької інформації на видалену базу даних на платформі Firebase.

5.2 Вимоги до програмного забезпечення

- Операційна система Android 5.1 і вище.

5.3 Вимоги до апаратної частини

Мінімальні технічні вимоги:

- Частота процессора: 1.5 ГГц.
- Мінімальний об’єм оперативної пам’яті: 1ГБ.
- Вільне місце не менш ніж 20МБ.

Бажані технічні вимоги:

- Частота процессора: 2 ГГц.
- Об’єм оперативної пам’яті: 2ГБ.
- Вільне місце не менш ніж 50МБ.

6. ЕТАПИ РОЗРОБКИ

Етап	Дата
1. Вивчення літератури за тематикою проекту	24.11.2018
2. Складання і узгодження технічного завдання	25.12.2018
3. Аналіз області застосування та особливостей системи	15.01.2019
4. Розробка загальної архітектури системи	15.03.2019
5. Реалізація програмного забезпечення	05.04.2019
6. Тестування програмного забезпечення	13.04.2019
7. Оформлення документації дипломної роботи	15.04.2019

Технічне завдання
до дипломного проекту

на тему: «Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android»

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 2
ВСТУП 3
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 5
1.1 Загальні положення 5
1.2 Змістовний опис і аналіз предметної області 11
1.2.1. E-commerce – основні поняття і аналіз 11
1.2.2. Дослідження портрету користувача своєї платформи 12
1.3. Аналіз успішних IT-проектів 14
1.4. Аналіз вимог до програмного забезпечення 16
1.4.1 Ролі та функціонал 16
1.4.2 Функціональні вимоги 20
1.4.3. Постановка комплексного завдання 20
ВИСНОВКИ ДО РОЗДІЛУ 1 22
РОЗДІЛ 2 МОДЕЛЮВАННЯ ТА АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 23
2.1 Моделювання та аналіз програмного забезпечення 23
2.2 Моделі програмної платформи 30
2.3 Проміжні модулі програмної платформи 34
2.4 Розробка бази даних 35
ВИСНОВКИ ДО РОЗДІЛУ 2 37
РОЗДІЛ 3 ІНСТРУКЦІЯ КОРИСТУВАЧА 38
ВИСНОВКИ ДО РОЗДІЛУ 3 49
ВИСНОВКИ 50
ПЕРЕЛІК ПОСИЛАНЬ 52

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IT — Інформаційні технології

ПП — Програмний продукт

IDE — Програмне середовище для розробки

GUI- Graphical user interface

XML — eXtensible Markup Language

DI — Dependency Injection

OOP — Object Oriented Programming

ADB — Android Debug Bridge

API — Application Programming Interface

REST — Representational State Transfer

UI — User interface

UX — User Experience

IO — Input/Output

DB — Database

App - Application

ВСТУП

Ринок мобільних додатків сьогодні залишається одним зі стрімкозростаючих сегментів ринку програмного забезпечення. Це зумовлено тим, що зростають можливості мобільних девайсів, що дозволяє переносити все більше справ на мобільні додатки замість залишення цих додатків у форматі desktop рішень. Також розвиваються нові фреймворки для розробки кросплатформенних додатків для зменшення витрат на розробку мобільних додатків під різні платформи і покриття більшого сегменту ринку.

Але треба зазначити, що кросплатформенна розробка має ряд недоліків у порівнянні з нативною розробкою(нативна розробка – розробка додатку з використанням базового інструменту та SDK(software development kit)) під конкретну операційну систему. Серед найзначущих недоліків — неможливість створення складних графічних інтерфейсів через різні підходи створення користувацького інтерфейсу на різних платформах. Також кросплатформена розробка вимагає присутність експертизи у нативній розробці під кожну з операційних платформ при роботі з NDK (Native development kit), що дозволяє комунікувати з hardware(апаратне забезпечення) девайсу, наприклад: використання Bluetooth, різних сенсорів(в тому числі сенсору відбитку пальця), адже буде необхідно писати так звані bridges(мости) окремо під кожну систему, що б комунікувати з інтерфейсом hardware окремо на кожній операційній системі.

Все більше і більше ринок пропонує користувачу рішення у сфері eCommerce. Ecommerce — сфера економіки, що включає у себе усі фінансові транзакції, які виконуються за допомогою комп'ютерних, мобільних та інших віртуальних додатків. Поява сервісів-агрегаторів, що збільшують ефективність пошуку товарів, допомагають власникам бізнесів покращити продаж своїх товарів.

На даний момент проводиться популяризація тренду спорту, створюються велика кількість спортивних організацій, магазинів, комплексів.

Сьогодні бізнес прагне діджиталізуватися для покриття більшого сегменту ринку, в зв'зку з цим створюються віртуальні платформи в кожній зі сфер. Спорт не є вийнятком. Наприклад компанія Nike давно створила свої мобільні додатки NRC(Nike Running Club) та NTC(Nike Training Club) для публічних забігів та приватних тренувань відповідно. Спортивна продукція також еволюціонує, наприклад тенісні ракетки використовують технології, шо створювалися для виробництва космічних ракет.

Створення Android додатків є дуже цікавим сегментом IT світу, адже тут поєднуються творчість у створенні гнучкого, зрозумілого і естетичного дизайну та великі можливості апаратних забезпечень, що дозволяють забезпечувати користувача великою кількістю різного функціоналу.

Створення додатків саме на мові Kotlin є нових витком розвитку розробки мобільних додатків, адже мова Kotlin достатньо нова і команда, що працює над цією мовою є дуже прогресивною і активною, це дозволяє цій мові запозичувати велику кількість сучасних, зручних та потужних підходів, які були створені в інших мовах.

Така велика потреба у мобільних телефонах та відповідно мобільних додатках є мотиватором для маркетингових компаній створювати свої рішення для клієнтів, щоб зробити більш ефективною рекламу та збільшити кількість кінцевих користувачей сервісів.

Сьогодні користувач потребує різноманітний функціонал для забезпечення широкого сегменту бізнес-задач.

Статистика показує, що частка користувачів, що користуються мобільними девайсами на операційній системі Android складає 74.85%(дані взяті з сайту statcounter http://gs.statcounter.com/os-market-share/mobile/worldwide, дані взяті у травні 2019).

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні положення

Сьогодні існує велика кількість операційних систем, ці системи працюють на різних типах пристроїв, деякі призначені виключно для одного типу пристроїв, деякі призначені для повної лінійки пристроїв, починаючи з мобільних девайсів закінчуючи стаціонарними компьютерами.

На даний момент існують такі найпоширеніші системи: Android, iOS, MacOS, Windows, Linux, TizenOS, WindowsPhoneOS, FuchsiaOS, ChromeOS і т.д.

Такі системи як Linux, Windows, MacOS призначені виключно під стаціонарні комп'ютери, більшість користувачів стаціонарних комп'ютерів використовують саме ці операційні системи. Багато сервісів розробляють свої сервіси та платформи під стаціонарні операційні системи, але в час глобалізації, в час швидкого розвитку бізнесу, люди повинні мати постійний зв'язок з цими сервісами, щоб мати актуальну інформацію і давати актуальну інформацію іншим людям. Тому популярним рішенням сьогодні є створювати додатки під різні операційні системи, щоб захопити більший сегмент користувачів і зробити використання їх сервісу більш ефективним і постійним.

Приведемо приклад таких сервісів:

Youtube – платформа, операційна система, що дозволяє дивитися та ділитися відеоматеріалами з іншими користувачами.

Youtube пропонує своїм користувачам мобільні додатки під операційні системи Android та iOS, що покриває 91% користувачів мобільних девайсів. Також Youtube має свою desktop версію в браузері, адаптовану під більшість браузерів, такі як Safari, Mozilla Firefox, Google Chrome, Opera і т.д.

Youtube є медіаплатформою, присутні різні матеріали, як розважальні так і навчальні. Але існують сервіси, які адаптовані під різні платформи і не є розважальними, такими сервісами користуються в корпораціях для збільшення ефективності.

Приведемо приклад такого сервісу:

Trello – платформа, яка дозволяє систематизувати роботи в маленьких та великих компаніях. Існує така методологія розробки як Scrum. Вона призначена для того, щоб організувати роботу команди над певним продуктом. Існує така людина, як ScrumMaster яка має керувати процесом розробки, створювати задачі для розробників і перевіряти, що команда виконує задачі по плану.

Trello має мобільні додатки під операційні платформи Android, iOS, та браузери Google Chrome, Safari, Opera, Mozilla Firefox.

Кожна компанія розробляючи продукт стає перед питанням яку частину користувачів, а означає і ринку треба покрити. Відповідаючи на це питання, компанія вирішує під які операційні системи необхідно створювати додатки.

Вирішуючи, що компанії необхідно розробити мобільний додаток чи мобільні додатки, компанія розглядає портрет свого користувача і процесі цього розглядання, компанія може прийняти рішення розробляти мобільний додаток тільки під Android або тільки під iOS, адже абсолютна більшість їх користувачів використовує одну операційну систему.

Приведемо статистику користувачів різних мобільних операційних систем:

Рис. 1.1 – Перша частина статистики користувачів

Рис. 1.2 – Друга частина статистики користувачів

Розглянемо динаміку зросту та падіння рівня користування кожною з операційних систем за 2018-2019 роки:

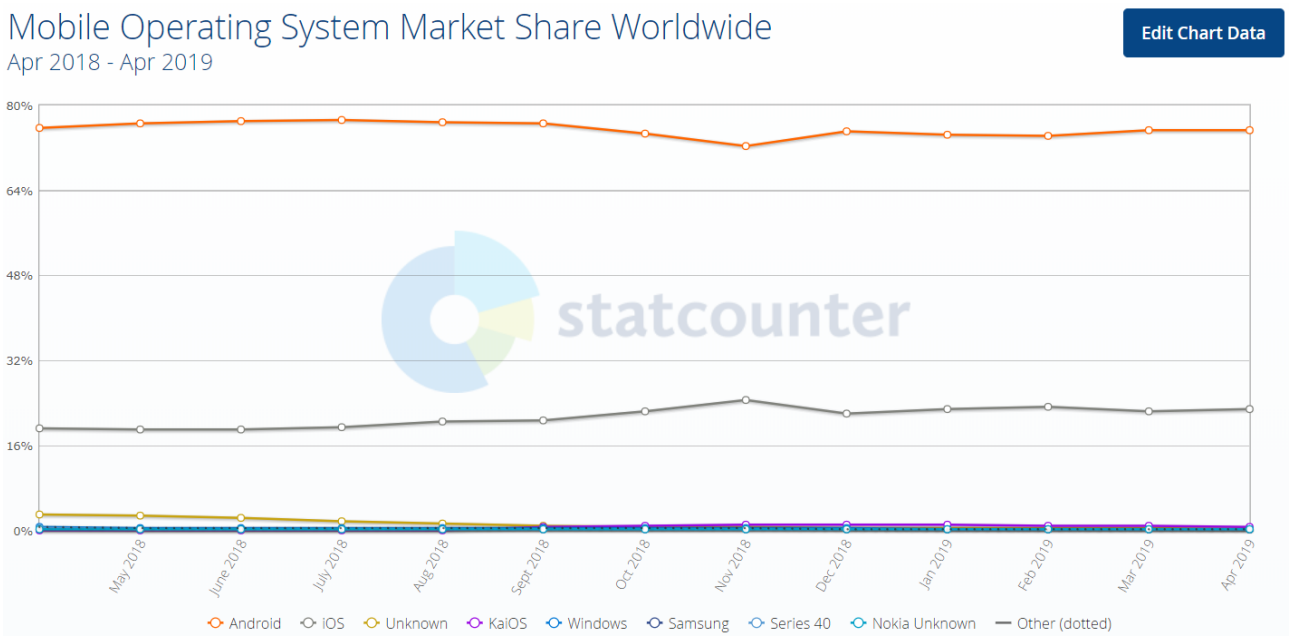


Рис.1.3 – Графік популярності операційних систем

Вибравши мобільну операційну систему для розробки, компанія стає перед вибором типу розробки під цю операційну систему. Сьогодні існує два типи розробки під операційну систему:

1. Створення додатку у нативний спосіб
2. Створення додатку у кросплатформений спосіб

Нативний спосіб розробки означає розробку додатку у базовий спосіб з використанням цільової мови, що зазвичай використовується для розробки під цю операційну систему, та використання нативного SDK(Software

Development Kit), що був створений спеціально для розробки додатків під цю операційну систему.

Кросплатформений спобів розробки означає, що для розробки додатку буде використовуватися сторонній інструмент, що дозволяє створювати одночасно додаток працюючий на декількох операційних системах.

Вибір того чи іншого шляху для розробки мобільного додатку не є однозначним. Не один з шляхів не є ультимативним, адже кожен з варіантів має свої переваги та недоліки. Розглянемо їх:

Переваги нативної розробки:

1. Повний доступ до апаратних можливостей девайсу.
2. Можливість створювати прогресивний дизайн, через повний доступ до SDK(Software Development Kit) операційної системи.
3. Ширша спільнота розробників, які стикаються з різними питаннями і дають на них відповіді.
4. Більша підтримка програмних бібліотек через ширшу спільноту розробників.
5. Витрата меншої кількості часу на підтримку додатку та створення нових функціональностей, через ширшу спільноту розробників.
6. Вища працездатність та показники роботи додатків, розроблених у нативний спосіб.

Недоліки нативної розробки:

1. Витрата більшої кількості часу на створення різних додатків під різні операційні системи.

Розглянемо переваги та недоліки кросплатформеної розробки:

Переваги кросплатформеної розробки:

1. Нижча ціна розробки програмного забезпечення у кросплатформений спосіб.
2. Менша кількість людино-годин для розробки програмного забезпечення у кросплатформений спосіб.

Недоліки кросплатформеної розробки:

1. Нижча працездатність та показники роботи додатків, розроблених у кросплатформений спосіб.
2. Складність при розробці складних користувацьких інтерфейсах, через непрямий доступ до нативного SDK(Software Development Kit).
3. Витрата більшої кількості часу при використанні апаратних можливостей девайсів, адже при кросплатформеній розробці немає прямого доступу до нативного SDK(Software Development Kit) і розробники мають розробляти так звані bridges(мости) для кожної операційної системи окремо.

Свою програмну платформу я вирішив розробляти у нативний спосіб, адже я не маю потреби розробляти аналогічний додаток для операційної системи iOS чи інших, а також я маю потребу використовувати апартні можливості девайсу, а саме модуль GPS для отримання місцеположення користувача.

Для розробки програмної платформи я використовував основний інструмент для розробки Android додатків, а саме – Android Studio.

Android Studio – IDE (Intelligent Development Environment) для розробки Android додатків, що була розроблена компанією Google з використанням IntelijiIDEA від компанії JetBrains. До речі, IntelijiIDEA дає той самий інструмент для розробки Android додатків, але цей інструмент багато інших цілей і функціональностей не націлених на розробку Android додатків, Android Studio навпаки націлена виключно на розробку Android додатків, тому Android Studio має більшу підтримку від компанії Google і випускає нові версії швидше і ефективніше.

Розробляючи додатки під операційну систему Android розробник має можливість вибрати різні мови програмування. Серед найпопулярніших мов програмування розробки під Android є такі мови:

1. Java

2. Kotlin
3. C#
4. C++
5. Scala
6. Python
7. JavaScript

Основними мовами для розробки мобільних додатків для операційної системи Android є Java та, з недавнього часу, Kotlin.

Я вибрав для розробки програмної платформи мову Kotlin.

Мова Kotlin була розроблена компанією JetBrains на базі мови програмування Java. Очевидними перевагами мови Kotlin перед Java є те, що Kotlin був розроблений командою компанії JetBrains, ця компанія відрізняється виключною активністю у розробці нових продуктів і підтримки та розвитку створених продуктів. Мова Kotlin кожен місяць отримує оновлення, нові функціональності. Мова Java навпаки достатньо стабільна та масштабна мова, існує така організація, яка займається прийманням рішень по додаванню нових функціональностей у наступних версіях, всі члени цією організації можуть мати різні думки щодо розвитку мови, це стримує розвиток. Також мова Java використовуються у багатьох enterprise проектах, для яких прийняття нових інструментів створює додаткові ризики, це також стримує розвиток мови.

Мова Kotlin навпаки необтяжена такою організацією та enterprise проектами, завдяки цьому ця мова може черпати найкращі практики з інших мов. Існує таке поняття як «Синтаксичний цукор», що означає певну обгортку над стандартними інструментами, що дозволяє зробити інтерфейс цих інструментів більш зрозумілим, простим, витрачати меншу кількість часу на інтеграцію цих інструментів.

Мови такого віку як Java, C++, C# і т.д. використовують ООП(Об’єктно-Орієнтовану Парадигму) як основну, це накладає відбиток на вигляд всіх інструментів у мові. Сьогодні стала популярна Функціональна Парадигма програмування. Вона на відміну від ООП(Об’єктно-Орієнтованої Парадигми) змушує програміста працювати з абстракціями як з математичними функціями та змінними. Функціональна Парадигма програмування дозволяє створювати ту ж саму функціональність, але за меншу кількість рядків коду, адже при використанні ООП парадигми ми маємо об’єкт над якими виконуються дії, на відміну від ООП Функціональна Парадигма дозволяє як в математиці наклади на змінну послідовність певних операторів(функцій), що дозволяє створювати закриту послідовність дій над змінною, такий код виглядає більш лаконічним і зрозумілим, на мій погляд.

1.2 Змістовний опис і аналіз предметної області

1.2.1. E-commerce – основні поняття і аналіз

E-commerce – електронна комерція. Український IT-сектор активно розвивається. Ми можемо спостерігати динаміку не тільки проектів, спрямованих на розвиток компьютерних технологій, інтернет індустрії, але й стрімкий розвиток конкуренції в сегменті проектів, який розуміє під собою B2C та B2B – рішення.

Важливим стимулов для розвитку IT-стартапів в Україні, стала девольвація гривні. Не дивлячись на відносну дороговизну проживання, Київ має ряд переваг для життя та розвитку IT-бізнесу.

По-перше, це розвинута інфраструктура, по-друге – зручна транспортна система з будь-якою країною Європи, по-третє – значна кількість кваліфікованих трудових ресурсів.

Тобто створення IT-продукту може коштувати в 5 разів дешевше, ніж в Америці або країні західної Європи.

Інші економічні розвинуті міста України – Одеса, Харків, Дніпропетровськ, Львів – також цікаві для інвесторів.

Ринок e-commerce достатньо молодий в Україні – особливо активно він почав розвивається тільки останні 5-6 років. За 2018 рік він виріс на 25%, але всеодно ще не насичений.

Думаю протягом 10 років він буде стрімко розвиватися, не дивлячись на відчутну конкуренцію.

Для бізнесменів, бажаючих працювати саме в цьому напряму, відкрите велике поле для діяльності.

Існує ряд причин, який дає можливість для розвитку e-commerce в Україні. Серед них два самих значущих фактору – вплив інфляції на зріст користувацького попиту. Перед тим як перейти в магазин, в 8 з 10 випадків користувачі вивчають інформацію про продукт в інтернеті, порівнюючи характеристики, ціни, умови купівлі.

Ще один важливий етап розвитку ринку – омнікальність торгівлі. Згідно з інформацією Google, для 85% користувачів достатньо природньо використовувати декілька пристроїв і каналів продаж для створення однієї покупки.

При цьому користувач шукає товар в соціальних мереж, онлайн магазинах, маркетплейсах, тобто максимально вивчають всі канали продажів, які йому пропонує ринок.

1.2.2. Дослідження портрету користувача своєї платформи

Запускаючи e-commerce бізнес власник має провести соціологічне дослідження ринку, та сформувати портрет свого користувача.

Таке дослідження буває трьох видів, таких як:

1. Розвідувальне
2. Описове
3. Аналітичне

Розвідувальне соціологічне дослідження можна зустріти за назвою пілотне опитування.

Описове соціальне дослідження представляє собою отримання загальної інформації про об'єкт. Таке соціологічне дослідження можна назвати вільним.

Тому представлений результат не прислідує ніякої точності в досліджуваних явищах. Описовий соціологічний метод приміняється дуже не часто і має багато протиріч.

Аналітичне соціологічне дослідження надає перевагу вивченню системи в якості виникнення різних можливих ситуацій.

Опитування за масштабом

Соціологічне дослідження згідно критерій поділяється на:

1. Міжнародні
2. Загальнонаціональні
3. Регіональні
4. Галузеві
5. Локальні

Приклад програми соціологічного дослідження

Під програмою розуміється виділення суті питання, які виступають основною базою для проведення соціологічного дослідження. Питання в ній логічно зв'язані між собою і мають єдину ціль – вирішення проблеми соціального дослідження.

Перед ствердженням вона досліджується на можливий інтерес у населення.

Буває, що проводяться соціальні пілотні дослідження в планах яких виділення рівня знань можливої тематики у людей.

Приклади

Програма соціологічного дослідження:

Формування платіжної здатності в бідних сім'ях країни, вплив реклами на людину і т.д.

Здогадки соціологічного дослідження:

Існує здогадка про те, що чим вище у людини забезпеченність, тим більший об'єм товару вона буде купляти протягом цього періоду.

Соціологічні дослідження:

Створюється програма дослідження і подається на ствердження. Потім проводиться соціальне опитування з метою збору необхідної інформації.

1.3. Аналіз успішних IT-проектів

Аналіз успішності IT-проекту виконується за допомогою бізнес аналізу в IT.

Бізнес-аналіз і його дефеніція

Бізнес-аналіз в IT – це постійний пошук потреб клієнту цього бізнес аналізу до програмних продуктів в IT і пошук вирішення бізнес-завдань відносно бізнес-потреб.

Бізнес-аналіз був створений з метою проявлення та формулювання необхідних змін, що мають статися в роботі організації, корпорації тощо, також метою бізнес аналізу є оптимізація інтеграції цих змін в роботу організації.

IT бізнес-аналітики знаходять вирішення, що поліпшують додану цінність продукту, яку дає організація і може приймати участь на різних рівнях роботи організації, наприклад: побудова стратегії, створення архітектури організації щодо лідерства, поставлення мети та вимог до програмних продуктів чи проектів.

Задача бізнес-аналізу – порівняти умови та архітектуру наявних процесів та програмних продуктів з необхідностями замовника та сформувати можливі варіанти, як можна оптимізувати поточний процес, програмний продукт відповідно до задач організації.

Яке завдання виконавчої одиниці бізнес-аналізу в IT?

Виконавча одиниця бізнес-аналізу в IT досліджує ситуацію клієнту, проводить аналіз і пошук ефективного вирішення проблеми та оформлює цей аналіз згідно з тим, на що можуть орієнтуватися розробники при створенні програмної платформи. Ця виконавча одиниця є посередником між клієнтом та командою протягом всього проміжку роботи проекту.

Бізнес-аналіз сильно впливає на хід розробки, адже він створює необхідні milestone(знаки пройденої милі, що означає певну значущу точку), які спрямовують команду в певне русло.

Етапи бізнес-аналізу

1. Проведення аналізу над потребами клієнту
2. Представлення сформованої ідеї або рішення проблеми клієнту
3. Оформлення цієї ідея або рішення в технічне завдання згідно з концепцією программного продукту.
4. Створення деталей специфікацій для вимог клієнту
5. Постійні консультації та brainstorms(мозгові штурми) серед розробників та інших учасників проекту.
6. Контакт з клієнтом під час розробки програмного продукту.

Платформа **Prom.ua**

Основний конкурент даної платформи це Prom.ua, адже він знаходиться в тій самій галузі бізнесу, що і моя платформа. Серед переваг над даною платформою у Prom.ua:

1. Покриття всіх галузей товарів. Це дозволяє отримати більше користувачів ніж в даній платформі.
2. Prom.ua має веб-клієнт, що дозволяє користувачам користуватися цією платформою не тільки на мобільному телефоні, а й на компьютері.

Серед недоліків перед даною платформою у Prom.ua:

1. Відсутність можливості отримання гео-координати користувача, щоб дати йому актуальну інформацію по тому де користувач може отримати необхідний для нього товар.

Відсутність нативного мобільного додатку, а тільки веб-клієнту, що оптимізован для мобільного телефону, що не дозволяє використовувати апаратні можливості смартфону користувача, такі як модуль GPS або Bluetooth.

1.4. Аналіз вимог до програмного забезпечення

1.4.1 Ролі та функціонал

Для створення архітектури продукту необхідно проаналізувати набір функціональностей та ролі користувачів нашої платформи.

В даній системі присутні такі типи користувачів:

- авторизований користувач;
- не авторизований користувач;

Не авторизований користувач може пройти процес авторизації та отримати доступ до всієї публічної інформації в платформі.

Авторизований користувач може отримати доступ до всієї публічної інформації в платформі, зробити замовлення, продивитися асортимент, продивитися список гарячих пропозицій, продивитися список магазинів, подивитися список замовлень, подивитися інформацію про себе в екрані профілю, продивитися відео матеріал товару, продивитися фото матеріал товару, продивитися список характеристик товару.

Платформа матиме таку функціональність:

- авторизація користувача
- Перегляд асортименту товарів конкретного магазину

- Перегляд асортименту товарів конкретної категорії
- Перегляд інформації конкретного магазину
- Перегляд інформації користувача в екрані профілю
- Створення замовлення в системі платформи
- Пошук конкретного магазину по назві
- Пошук конкретного категорії по назві
- Фільтрація товарів конкретної категорії по характеристикам

Таблиця 1.1 - Перегляд асортименту товарів конкретного магазину

Назва	Перегляд асортименту товарів конкретного магазину
Опис	Користувач може продивитися асортимент товарів конкретного магазину і побачити інформацію про цей товар.
Учасники	Авторизований користувач
Передумови	Користувач авторизований
Постумови	Користувач отримав інформацію про асортимент товарів
Основний сценарій	Користувач авторизується, Користувач переходить на головний екран, Користувач переходить на екран конкретного магазину, Користувач бачить асортимент конкретного магазину

Таблиця 1.2 - Перегляд асортименту товарів конкретної категорії

Назва	Перегляд асортименту товарів конкретної категорії

Опис	Користувач може продивитися асортимент товарів конкретної категорії і побачити інформацію про цей товар.
Учасники	Авторизований користувач
Передумови	Користувач авторизований
Постумови	Користувач отримав інформацію про асортимент товарів конкретної категорії

Продовження таблиці 1.2

Основний сценарій	Користувач авторизується, Користувач переходить на головний екран, Користувач переходить на екран конкретної категорії, Користувач бачить асортимент конкретної категорії

Таблиця 1.3 - Перегляд інформації конкретного магазину

Назва	Перегляд інформації конкретного магазину
Опис	Користувач може продивитися інформацію про конкретний магазин.
Учасники	Авторизований користувач
Передумови	Користувач авторизований
Постумови	Користувач отримав інформацію про асортимент товарів конкретної категорії

Основний сценарій	Користувач авторизується, Користувач переходить на головний екран, Користувач переходить на екран конкретного магазину, Користувач бачить інформацію конкретного магазину

Таблиця 1.4 - Створення замовлення в системі платформи

Назва	Створення замовлення в системі платформи
Опис	Користувач може створити замовлення.

Продовження таблиці 1.4

Учасники	Авторизований користувач
Передумови	Користувач авторизований
Постумови	Користувач успішно створив замовлення, і це замовлення показане в списку замовлень
Основний сценарій	Користувач авторизується, Користувач переходить на головний екран, Користувач переходить на екран конкретного магазину, Користувач додає товари в кошик Користувач переходить до екрану кошика Користувач переходить до екрану підтвердження замовлення Користувач додає інформацію до замовлення Користувач оплачує товар

	Користувач бачить створене замовлення в списку замовлень.

1.4.2 Функціональні вимоги

Таблиця 1.5 - Авторизація в системі

Назва	Авторизація в системі
Опис	Платформа має зберегти користувацькі дані в платформі, і використовувати їх при створенні замовлення та повторному заході в додаток.

Таблиця 1.6 - Створення замовлення

Назва	Створення замовлення
Опис	Платформа має створити замовлення, провести успішну оплату замовлення та відобразити замовлення в списку замовлень.

Таблиця 1.7 - Фільтрація товарів категорії

Назва	Фільтрація товарів категорії
Опис	Платформа має зберегти конфігурацію фільтрів по характеристикам, та відобразити відфільтрований товар для користувача. Дана конфігурація повинна бути відтворена при повторному переході до конкретної категорії.

1.4.3. Постановка комплексного завдання

Платформа що розробляється призначена для підвищення ефективності користувача робити покупки в інтернеті, допомагати власникам бізнесу знаходити комунікацію з їхніми клієнтами.

Мета створення даної платформи – підвищення ефективності користувача при купівлі товарів в інтернеті.

Для досягнення мети даної платформи, робота повинна мати наступну функціональність:

- Авторизація користувача в системі
- Перегляд асортименту товарів конкретного магазину
- Перегляд асортименту товарів конкретної категорії
- Перегляд інформації конкретного магазину
- Перегляд інформації користувача в екрані профілю
- Створення замовлення в системі платформи
- Пошук конкретного магазину по назві
- Пошук конкретного категорії по назві
- Фільтрація товарів конкретної категорії по характеристикам
- Дана платформа має працювати на операційній системі Android з версією 5.1 та вище. Бізнес-логіка реалізована локально на клієнті, серверна частина відсутня, для синхронізації даних і роботи мультидевайсності використовується платформа Firebase, яка призначення для зберігання інформації про товари, магазини, категорії, гарячі пропозиції.

ВИСНОВКИ ДО РОЗДІЛУ 1

E-commerce бізнес захоплює більше і більше галузей та більше і більше користувачів. Бізнеси діджиталізуються та пропонують своїм користувачам функціонал для підвищення ефективності їх купівель в інтернеті та офлайн. Моя платформа призначена для підвищення ефективності купівель користувача в інтернеті, а також покращення комунікації власників бізнесів та їх клієнтів.

Розробка платформи, яка містить вище описану функціональність, задоволняє всі потреби власників бізнесів в e-commerce та покращує комунікацію власників бізнесу та їх клієнтів.

Даний мобільний додаток підтримує мультидевайсність та синхронізацію даних завдяки використанню інтегрованої платформи Firebase, дозволяє оплачувати свої покупки та порівнювати товари різних виробників.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Буде доцільно детальніше розглянути важливі аспекти аналізу програмного забезпечення, особливо користувацький інтерфейс, про його концепцію і про те які патерни в UI/UX були використані.

Взаємодія віртуального додатку та користувача відбувається саме через графічний інтерфейс цього віртуального додатку. Тож дизайн має відповідати таким функціональним потребам користувача, як – зрозумілість, простота, гнучкість і опціонально естетична складова інтерфейсу.

Дизайнери розподіляють інтерфейс на дві складові – UI та UX.

UI(User Interface) – складова користувацького інтерфейсу, яка відповідає за графічну та естетичну складові. Через те, що графічні інтерфейси з’явилися дуже давно, спільнота дизайнерів багато експерементувала, що дозволило дизайну еволюціонувати і створювати нові патерни у графічному дизайні.

UX(User Experience) – складова користувацького інтерфейсу, яка відповідає за архітектуру компонентів графічного інтерфейсу, зрозумілість графічного інтерфейсу та функціональність графічного інтерфейсу. Так як і UI(User Interface) UX(User experience) розвивався паралельно, тому дизайнерська спільнота сформувала також велику кількість дизайнерських патернів, які використовуються і сьогодні.

Дизайн мобільних додатків є окремою галуззю дизайну, тому мобільний дизайн має свої унікальні дизайнерські патерни, які використовуються саме в мобільних додатках.

Говорячи про мобільний дизайн треба зазначити, що кожна операційна система має свою спільноту дизайнерів, яка працює над еволюціонуванням дизайну відповідної операційної системи.

Серед найпопулярніших операційних систем Android (75% користувачів у світі) та iOS (16% користувачів у світі) мають свої відповідні дизайнингові концепції Material Design і Flat Design відповідно.

Material Design – стиль дизайна, що був розроблений у великій корпорації Google у 2014 році. Цей дизайн був представлений на головній публічній конференції Google, Google I/O. Ця концепція дизайну була придумана завдяки використанню реальних матеріалів, реальних тіней.

Material Design використовується на сучасних версіях Android, а також на більш старих(починаючи з версії Lollipop) через Support версії графічних біблотек. Треба зазначити, що у 2018 році була розроблена нова версія концепції Material Design`у Material Design 2.0.

Для користувачів девайсів на операційній системі Android Material Design став стандартним і найзрозумілішим інтерфейсом серед усіх додатків. Приведемо приклади найпопулярніших додатків з використанням Material Design – Google Play (стандартний додаток на операційній системі Android необхідний для завантаження нових додатків. Користувач операційної системи Android має також можливість встановлювати додатки завантажуючи файли з розширенням .apk через browser або сторонні ресурси, але другорядні способи встановлення додатків не гарантують безпеку користувацьких даних і можуть бути використані зловмисниками для завантаження приватної інформації про користувача. Google Play пропонує користувачам тільки перевіренні додатки, адже в систему Google Play встановлена система перевірки кожної версії кожного додатку на надійність та неможливість використовувати дані не згідно з захистом користувацьких даних), Youtube(соціальна мережа, що дозволяє дивитися та викладати відео матеріали, ділитися корисною та розважальною інформацією та навчальним контентом), Telegram(месенжер, що являється одним з найзахищеніших месенжерів у світі, завдяки використанню несиметричних алгоритмів шифрування даних).

Приведемо низку рисунків з демонстрацією головних компонентів Material Design та їх використання в додатках перерахованих вище:

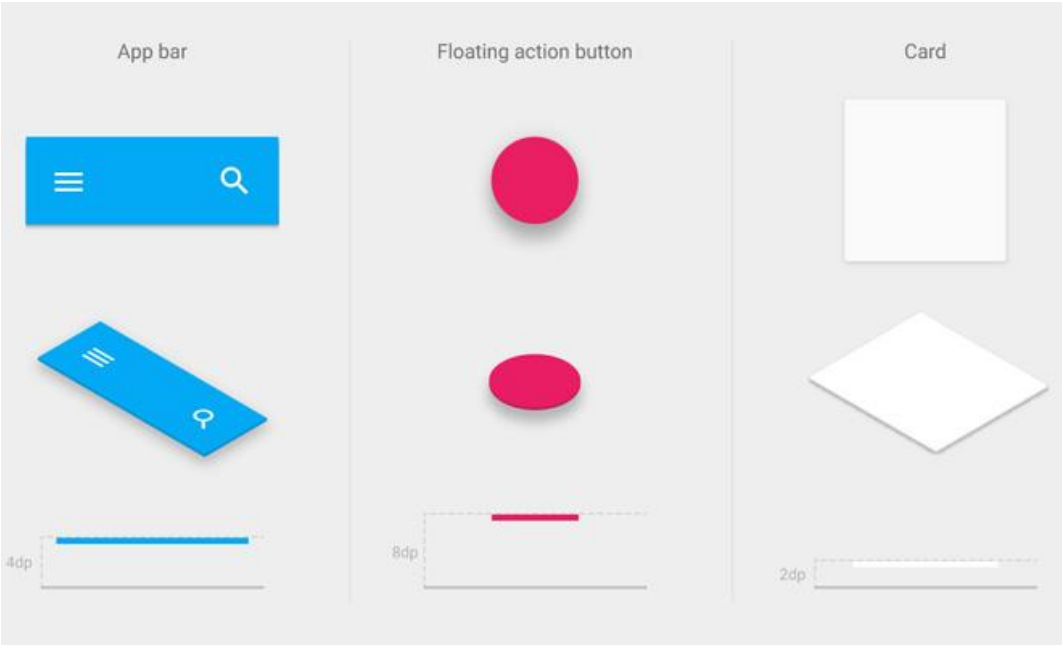


Рис. 2.1 – Головні елементи матеріального дизайну

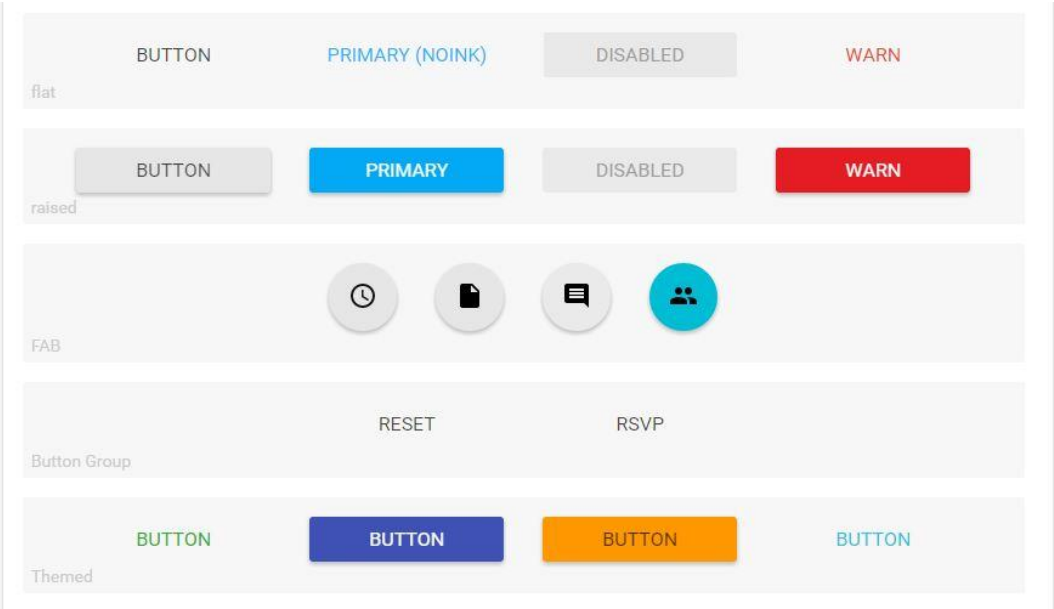


Рис. 2.2 – Вигляд різних типів кнопок матеріального дизайну

Наведу приклад використання Material Design у додатку Google Play. Серед патернів Material Design у додатку Google Play є такі патерни: ViewPager горизонтальлний слайдер компонентів, де компоненти можуть виступати самостійними модулями або бути простими візуальними елементами.

SearchBar – пошукова стрічка, яка виділяється в окрему картку та має власне поле для вводу, яке довзоляє ввести пошуковий запит, та кнопку мікрофону, щоб користувач мав можливість продиктувати пошуковий запит. Hamburger Button – один з найпоширеніших патернів Material Design, що виглядає як три горизонтальні стрічки, розташовані одна над одною, натискаючи на неї користувач матиме можливість побачити меню додатку з усіма можливими функціями даного додатку.

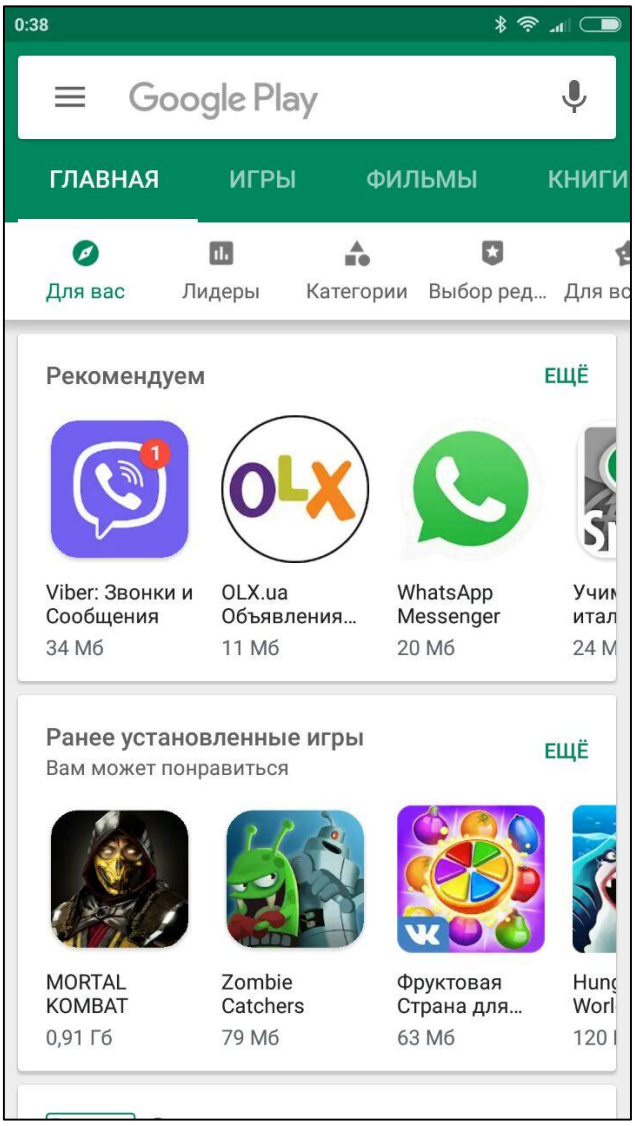


Рис. 2.3 – Головний екран

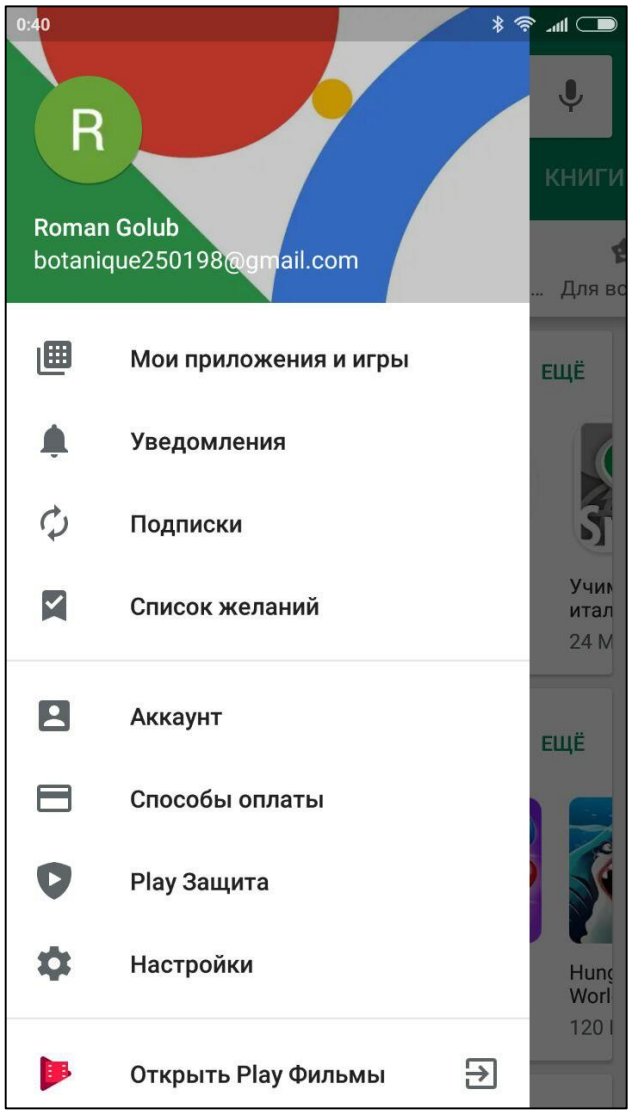


Рис. 2.4 – Головне меню

Операційна система iOS створила свою реалізацію Flat Design концепції, вони поглибили і розширили цю концепцію, та продовжують робити це з кожною новою версією.

Можна зазначати, що ще 10 років тому дизайн у більшій мірі представляв ультра-реалістичні елементи дизайну зі спробою повторити реальні матеріали, такі як: шкіра, метал, скло і т.д.

Зараз Flat Design(плоский дизайн) є майже найпоширенішим, адже навіть у дизайні Material Design присутні елементи плоского дизайна.

Flat Design – плоский дизайн, стиль, що був наступною етапом розвитку реалістичного дизайну, де основними деталями концепції є високо-реалістичні графічні елементи, жирні шрифти та контрастність кольорів.

Вважається, що Flat Design був створений компанією Microsoft, коли ця компанія створила свою концепцію Metro Design, але плоский дизайн був створений ще до першого виртуального интерфейсу. Походження плоского дизайну пішло при створенні дизайну під перший мобільний девайс компанії Apple, а саме Apple Iphone. Apple створила революцію у світі мобільних девайсів, створивши нові стандарти того, як девайси мають виглядати і вести себе. Подивившись на перші мобільні інтерфейси, ми зможемо помітити, що більшість мобільних додатків мають скеуроморфний дизайн. Скеуроморфний дизайн намагався перенести реальне життя і вигляд на екран, такий дизайн використовує реальні текстури, падаючі тіні та інші характеристики реального світу.

Коли був створений перший айфон, екрани з тачскрином були не тривіальними для користувачів. Скеуроморфний дизайн грав головну роль у адаптації користувачів до інтерфейсів нового типу.

Через те, що Material Design має схожі елементи з Flat Design. Є мета висвітлити головні деталі, які являються спільними, щоб подальше було зрозуміло які основні концепції було використано у створенні мобільного дизайна.

Такі основні концепції, які являються спільними для Flat Design та Material Design:

2. Менше декоративних елементів – у плоскому дизайні всі орнаментні елементи дизайну вважаються необов’язковими і заважаючими, адже у матеріальному дизайні і плоскому дизайні відносяться до мінімалістичного дизайну.
3. Більше фокусу на інформації, яка має бути донесена до користувача – знову ж таки, через те, що матеріальний і плоский дизайни відносяться до мінімалістичного дизайну, що ж на інформації, що являється головною на екрані, має бути основний фокус.
4. Адаптивність дизайну – девайси на операційних системах Android та iOS мають різні розміри екрану. Тому дизайнер має створювати адаптивний дизайн під різні розміри девайсів.
5. Уникання максимально плоских елементів дизайну – хоча плоский дизайн названий плоским, це не означає, що елементи не мають тіней, навпаки, елементи мають тіні, але правильних розмірів. Те ж саме у Material Design.
6. Уникання низьких контрастів – не є популярним у плоскому та матеріальному дизайнах створювати дизайн з низьким контрастом кольорів елементів, адже це ускладнює читання даних користувачем, що створює незадоволення користувача від мобільного додатку і дискомфорт від використання цього додатку.

7. Обмежена кількість шрифтів у додатку – популярним серед починаючих дизайнерів створювати контрастні, яскраві дизайни. Частина з цих дизайнерів досягає цієї яскравості за допомогою великої кількості шрифтів у додатку. Це не є правильний шлях для дизайнера, що працює у матеріальному або плоскому дизайнах.

Адже користувач буде асоціювати кожен зі шрифтів з функцією того графічного елементу, де він цей шрифт побачив, а не з конкретним елементом і його оформленням.

8. Візуальна приорітизація важливих графічних елементів – ви можете використовувати зміну розміру важливих елементів або використовувати контрастні кольори для виділення конкретного елементу і збільшення фокусу і приорітизації на цьому об’єкті.
9. Зрозуміла індикація інтерактивності елементу – в плоскому дизайні, як і в матеріальному дизайні існують інтерактивні і не інтерактивні елементи. Завдання UX(User Experience) складової дизайну є в тому, щоб користувач одразу розумів, що цей елемент може бути натиснутий, перетягнутий і т.д., а інший не є інтерактивним, але несе за собою завдання пронотифікувати користувача корисною інформацією.
10. Серед важливих елементів індикації інтерактивності елементів існують такі:
    a. Простір. Необхідно додавати простір між інтерактивними елементами та контентом, щоб відокремити інтерактивний та не інтерактивний елемент
    b. Контраст. Необхідно використовувати контрастні кольори для інтерактивних елементів, які допоможуть користувачу зрозуміти, що конкретний елемент є інтерактивним.
    c. Тіні. В процесі еволюції дизайну та програмного забезпечення, з’явилася можливість створювати псевдо-об’ємні ефекти(в даному випадку тіні), щоб допомогти користувачу зрозуміти доступність певної дії конкретного інтерактивного елементу.
11. Використання анімацій задля пояснення просторових відносин між елементами – візуальна простота плоского дизайну дуже

добре працює з рухами. Рухи допомагають пояснити зміну стану графічних елементів користувачу.

Правильне використання анімацій у плоскому та матеріальному дизайнах дозволяє зробити дизайн більш інтерактивним, що зробить додаток більш зручним, гнучким і зрозумілим.

Навівши дані спільні частини матеріального і плоского дизайну, можно зрозуміти, що вони хоча були створені у різних компаніях та у різні часи, але запозичують один в одного багато частин, що дозволяє створювати конкуренцію і в результаті еволюцію дизайну.

2.2 Моделі програмної платформи

Створюючи програмне забезпечення, програміст стикається з питанням схеми бази даних, схема бази даних часто є не однозначною. При роботі з різними базами даних, схеми даних будуть відрізнятися, адже різні бази даних, використовують різні види концепцій.

Розділяють такі види баз даних:

1. Реляційні
2. Не реляційні

Реляційні бази даних означають табличне представлення даних в базі даних. Назва до реляційних баз даних прийшла з англійської мови, адже Relation(реляція) означає відношення.

Одна з переваг реляційних баз даних над не реляційними, це те, що кожен об’єкт бази даних це таблиця, а це означає, що немає ніякої вкладенності. Але об’єкти можуть зберігати в собі теж об’єкти. Яким чином це реалізується в реляційних базах даних? Все просто! Це відношення.

Існують відношення один до одного, один до багатьох, багато до багатьох. Якщо об’єкт зберігає в собі список інших унікальних об’єктів, то це один до багатьох. Якщо ж об’єкт зберігає в собі список інших необов’язково

унікальних об’єктів, то це багато до багатьох. Якщо об’єкт зберігає в собі унікальний об’єкт, то це один до одного.

Також існують не реляційні бази даних, такі бази даних можуть відрізнятися один від одної. Наприклад існують такі концепції нереляційних баз даних, як документний тип представлення, об’єктний тип представлення і т.д.

Складність мого проекту складається в тому, що локальна база даних у додатку у мене реляційна, а віддалена база даних на платформі Firebase документна, тому на стороні додатку необхідно конвертувати дані у документному представлені з Firebase до реляційного представлення для локальної бази даних.

Проаналізувавши взаємодію сутностей даної платформи, було створені такі моделі в базі даних:

CategoryDto – data transfer object, модель категорії(має наступні поля: id: String, name: String, photoUrl: String, deleted: Boolean, characteristics: List<CharacteristicDto>), що отримується з платформи Firebase

Category - модель категорії(має наступні поля: (має наступні поля: id: String, name: String, photoUrl: String, deleted: Boolean, characteristics: List<Characteristic>))

CharacteristicDto – data transfer object, що отримується з платформи Firebase, представляє собою модель характеристики (має наступні поля:characteristicId: String, categoryId: String, name: String, subCharacterisitcs: List<SubCharacteristicDto>)

Characteristic - представляє собою модель характеристики (має наступні поля:characteristicId: String, categoryId: String, name: String, subCharacterisitcs: List<SubCharacteristicDto>)

DiscountOfferDto – data transfer object, що отримується з платформи Firebase, представляє собою модель гарячої пропозиції(має наступні поля:id: string, photoUrl: String, stuffId: String)

DiscountOffer - представляє собою модель гарячої пропозиції(має наступні поля:id: string, photoUrl: String, stuffId: String)

StoreCategoryDto – data transfer object, що отримується з платформи Firebase, представляє собою модель категорії магазину(має наступні поля:id: String, storeId: String, parentCategoryId: String)

StoreCategory - представляє собою модель категорії магазину(має наступні поля:id: String, storeId: String, parentCategoryId: String

StoreDto – data transfer object, що отримується з платформи Firebase, представляє собою модель магазин(має наступні поля: var id: String, var merchantId: String, var name: String, var photoUrl: String, var address: String, var phone: String, var description: String, var longitude: Double, var latitude: Double, var categories: List<StoreCategoryDto>, var stuffs: List<StuffDto>, var startWorkingSession: Long, var endWorkingSession: Long, var deleted: Boolean, var blocked: Boolean)

Store - представляє собою модель магазин(має наступні поля: var id: String, var merchantId: String, var name: String, var photoUrl: String, var address: String, var phone: String, var description: String, var longitude: Double, var latitude: Double, var categories: List<StoreCategoryDto>, var stuffs: List<StuffDto>, var startWorkingSession: Long, var endWorkingSession: Long, var deleted: Boolean, var blocked: Boolean)

StuffCategoryDto – data transfer object, що отримується з платформи Firebase, представляє собою модель категорія товару(має наступні поля: var id: String, var stuffId: String, var categoryId: String, var stuffCharacteristics: List<StuffCharacteristicDto>)

StuffCategory - представляє собою модель категорія товару(має наступні поля: var id: String, var stuffId: String, var categoryId: String, var stuffCharacteristics: List<StuffCharacteristicDto>)

StuffDto – data transfer object, що отримується з платформи Firebase, представляє собою модель товару(має наступні поля: val stuffId: String, val

storeCategoryId: String, val stuffCategoryId: String, val parentStoreId: String, val name: String, val photoUrl: String, val description: String, val price: Int, var salesCount: Int, var blocked: Boolean, var discountPercentage: Int, val deleted: Boolean, val stuffCategory: StuffCategoryDto, val stuffMovies: List<StuffMovieDto>,val stuffPhotos: List<StuffPhotoDto>)

Stuff - представляє собою модель товару(має наступні поля: val stuffId: String, val storeCategoryId: String, val stuffCategoryId: String, val parentStoreId: String, val name: String, val photoUrl: String, val description: String, val price: Int, var salesCount: Int, var blocked: Boolean, var discountPercentage: Int, val deleted: Boolean, val stuffCategory: StuffCategoryDto, val stuffMovies: List<StuffMovieDto>,val stuffPhotos: List<StuffPhotoDto>)

StuffMovieDto – data transfer object, що отримується з платформи Firebase, представляє собою модель відео товару(має наступні поля: val movieId: String, val youtubeMovieId: String, val youtubeMoviePhotoUrl: String, val stuffId: String)

StuffMovie - представляє собою модель відео товару(має наступні поля: val movieId: String, val youtubeMovieId: String, val youtubeMoviePhotoUrl: String, val stuffId: String)

StuffPhotoDto – data transfer object, що отримується з платформи Firebase, представляє собою модель фото товару(має наступні поля: val photoId: String, val photoUrl: String, val stuffId: String)

StuffPhoto - представляє собою модель фото товару(має наступні поля: val photoId: String, val photoUrl: String, val stuffId: String)

StuffSubCharacteristicDto – data transfer object, що отримується з платформи Firebase, представляє собою модель суб характеристики товару(має наступні поля: var stuffSubCharacteristicId: String, var stuffId: String, var stuffCharacteristicId: String, var parentSubCharacteristicId: String, var name: String, var type: String, var floatValue: Float, var intValue: Int, var stringValue: String, var booleanValue: Boolean)

StuffSubCharacteristic - представляє собою модель суб характеристики товару(має наступні поля: var stuffSubCharacteristicId: String, var stuffId: String, var stuffCharacteristicId: String, var parentSubCharacteristicId: String, var name: String, var type: String, var floatValue: Float, var intValue: Int, var stringValue: String, var booleanValue: Boolean)

SubCharacteristicDto – data transfer object, що отримується з платформи Firebase, представляє собою модель суб характеристики(має наступні поля: var subCharacteristicId: String, var name: String, var type: String, var floatStart: Float, var floatEnd: Float, var floatRangeStart: Float, var floatRangeEnd: Float, var intStart: Int, var intEnd: Int, var intRangeStart: Int, var intRangeEnd: Int, var stringValue: String, var booleanValue: Boolean, var parentCharacteristicId: String, var dropList: List<StringValueDto>)

SubCharacteristic - представляє собою модель суб характеристики(має наступні поля: var subCharacteristicId: String, var name: String, var type: String, var floatStart: Float, var floatEnd: Float, var floatRangeStart: Float, var floatRangeEnd: Float, var intStart: Int, var intEnd: Int, var intRangeStart: Int, var intRangeEnd: Int, var stringValue: String, var booleanValue: Boolean, var parentCharacteristicId: String, var dropList: List<StringValueDto>)

2.3 Проміжні модулі програмної платформи

Проміжні модулі – сутності, які відповідають за обробку інформації, сортування інформації, перетворення однією інформаціїї в іншу.

SubCharacteristicConverters – клас, який відповідає за перетворення типу субхарактеристики в String, і навпаки, а також у випадку типу субхарактеристики DROP(тобто ця субхарактеристика має конкретні варіанти вибору з Drop списку) конвертація списку стрічок в одну стрічку і навпаки.

AppModule – сутність, яка відповідає за побудування головних залежностей у відповідності до правил послідновності, що були задані автоматично.

AppComponent – сутність, яка використовує AppModule, як метод для побудування залежностей і віддачу конкретних залежностей по конкретним методам.

App – головний клас у додатку, він відповідає за ініціалізацію головних і другорядних залежностей, ініціалізацію додатку у відповідності до конфігурацій з файлу Manifest.json, ініціалізацію статичних даних.

StoreInteractor – сутність, що грає роль орбітра і виконавця усіх запитів з протилежних сторін, що відносяться до операції з магазином і товарами, з однієї сторони – IO шар(шар вводу-виводу), з іншої сторони – клієнтська частина(конкретні екрани і графічні сутності).

2.4 Розробка бази даних

В даній платформі було використано поширену нині базу даних – RoomDb. Дану базу даних було розроблено компанією Google, і через короткий час ця база даних стала стандартним інструментом для роботи з даними. Звичайно в мене є можливість використовувати интерфейс SQLite, така можливість є, але цей підхід застарів через велику кількість boilerplate(шаблонного) коду, мені як програмісту необхідно створити велику кількість сутностей, методів по заповненню таблиць і т.д. Замість цього можливо використовувати інструмент, який економить мій час і являється в тій же мірі гнучким, як і SQLite, тим більше, що RoomDb являється лише надстройкою над SQLite і не являється окремою базою даних.

Для створення бази даних в Android додатку, мені необхідно створити RoomDatabase класс, що відповідає за конфігурацію бази даних.

```
@Database(entities = [Store::class, Stuff::class, Category::class, StoreCategory::class, CartStuff::cl
    DiscountOffer::class, OrderStuff::class, Order::class, Card::class, StuffMovie::class, StuffPhoto:
    SubCharacteristic::class, Characteristic::class, StuffCategory::class,
    StuffCharacteristic::class, StuffSubCharacteristic:: class],
    version = BuildConfig.VERSION_CODE, exportSchema = false)
@TypeConverters(SubCharacteristicConverters::class)
abstract class RoomDatabase : android.arch.persistence.room.RoomDatabase() {

    abstract fun getStoreDao(): StoreDao

    abstract fun getOrderDao(): OrderDao

    abstract fun getPaymentDao(): PaymentDao
}
```

Рис. 2.5 – Скриншот класу бази даних для RoomDB

Після цього, необхідно створити сутності Dao, що відповідатимуть за обробку запитів до бази даних, в даному випадку це StoreDao, OrderDao, PaymentDao, які відповідають за операції з магазину і товарами, замовленнями та платіжними картками відповідно.

```
@Dao
interface StoreDao {

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun upsertStore(store: Store): Long

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun upsertDiscountOffer(discountOffer: DiscountOffer): Long

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun upsertStuffPhoto(stuffPhoto: StuffPhoto): Long

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun upsertStuffCategory(stuffCategory: StuffCategory): Long

    @Delete
    fun removeStuffCategory(stuffCategory: StuffCategory)
```

Рис. 2.6 – Скриншот інтерфейсу Dao для бази даних на RoomDB

В Dao кожен метод інтерфейсу має свою анотацію, вона може бути @Query, @Insert чи @Delete, де @Query відповідає за надання доступу до будь якого SQL запиту, @Insert дозволяє додати об’єкт до бази даних і вибрати стратегію у випадку конфлікту, @Delete видаляє об’єкт.

ВИСНОВКИ ДО РОЗДІЛУ 2

В цьому розділу було зроблено акцент на графічній та програмній частині додатку, провів аналіз порівняння двох концепцій дизайнів(Android-Material Design та iOS-Flat Design), пояснення які головні деталі обох концепцій створюють гнучкий, простий та зрозумілий інтерфейс.

Я розкрив загальну структуру додатку за допомогою написання схеми бази даних та внутрішньої взаємодії, а також сутностей, що обробляють ці дані і дозволяють, різним компонентам та шарам комунікувати один між одним.

Мною було зроблено висновок, що набір тих інструментів та схеми бази даних, що вибрані для розробки, повністю відповідають технічному завданню.

РОЗДІЛ 3
ІНСТРУКЦІЯ КОРИСТУВАЧА

При запуску додатку користувач буде ознайомлений з головними функціями платформи.

Рис. 3.1 – Перша ознайомлююча функція

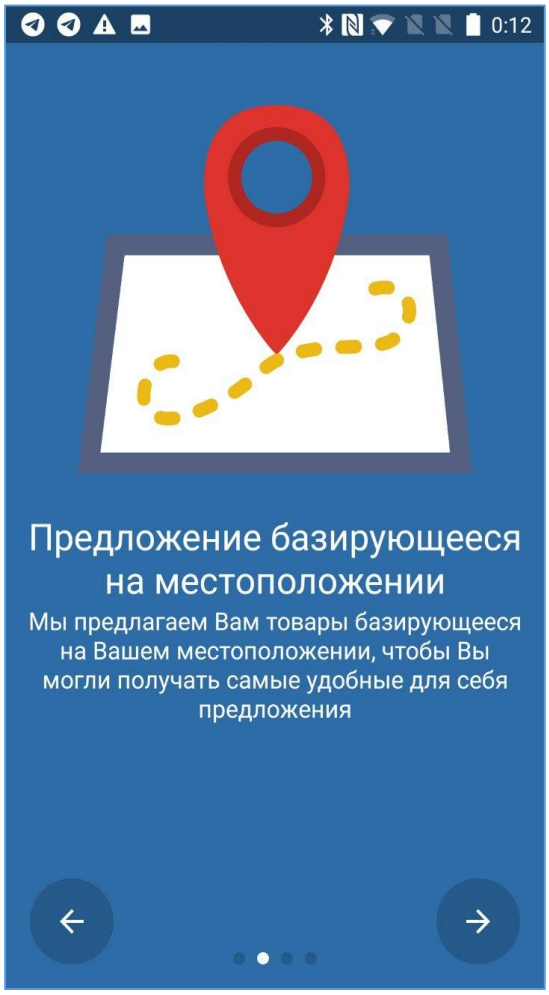


Рис. 3.2 – Друга ознайомлююча функція

Рис. 3.3 – Третя ознайомлююча функція

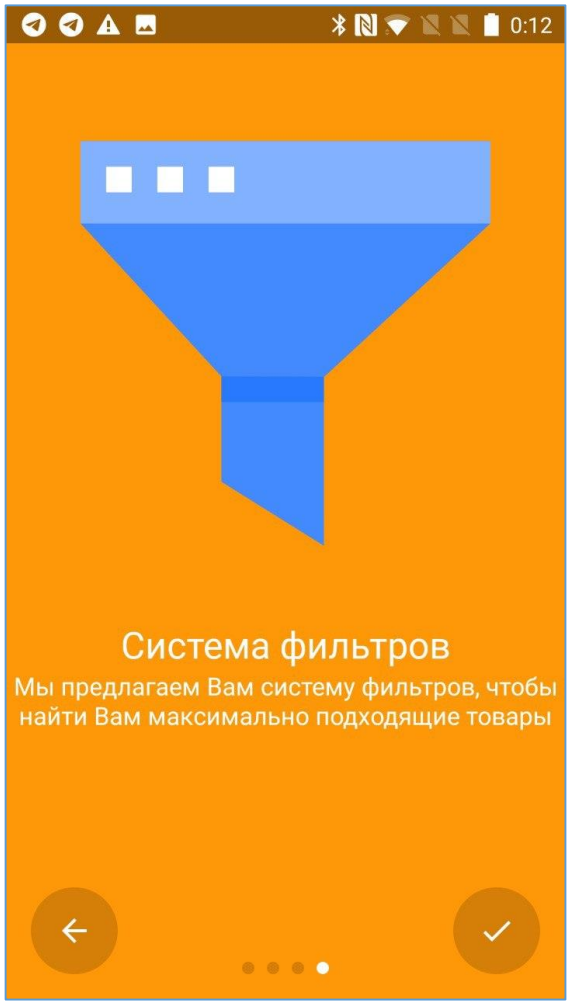


Рис. 3.4 – Четверта ознайомлююча функція

Після ознакоймлення з основними функціями платформи, користувач перейде до екрану аутентифікації, де користувач може аутентифікуватися за допомогою Google або Facebook аккаунтів.

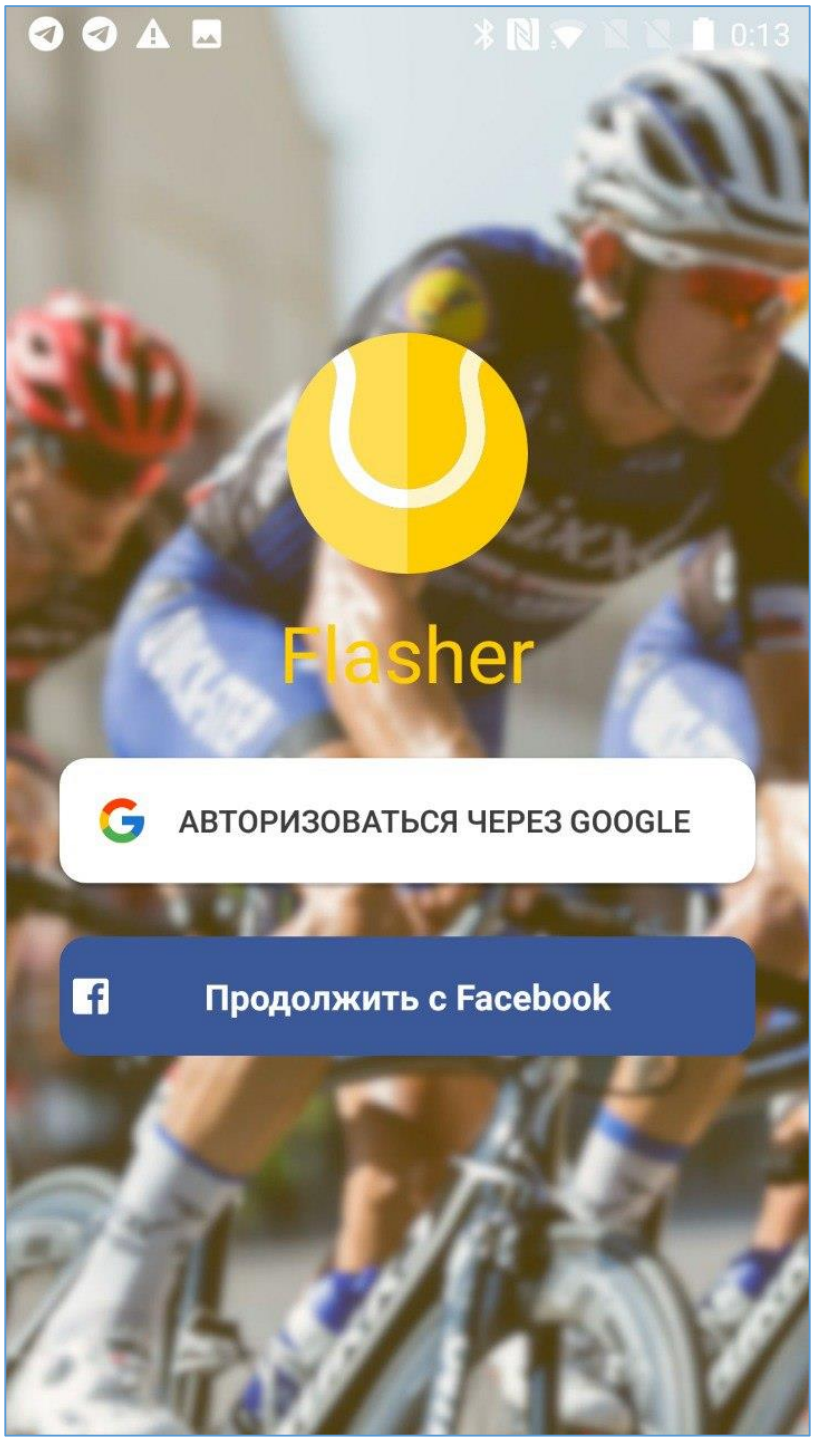


Рис. 3.5 – Екран авторизації

Після успішної аутентифікації користувач перейде до головного екрану з такими вкладками: головна, мапа, замовлення, профіль та кошик.

На головній сторінці користувач може побачити такі блоки: гарячі пропозиції, список магазинів, список категорій, найпопулярніші товари. Натиснувши на конкретну гарячу пропозицію, користувач перейде до екрану конкретного товару, з яким зв’язана ця гаряча пропозиція. Натиснувши на магазин, користувач перейде до конкретного магазину. Натиснувши на конкретну категорію, користувач перейде до екрану конкретної категорії. Натиснувши на конкретний товар серед найпопулярніших товарів, користувач перейде до екрану конкретного товару.

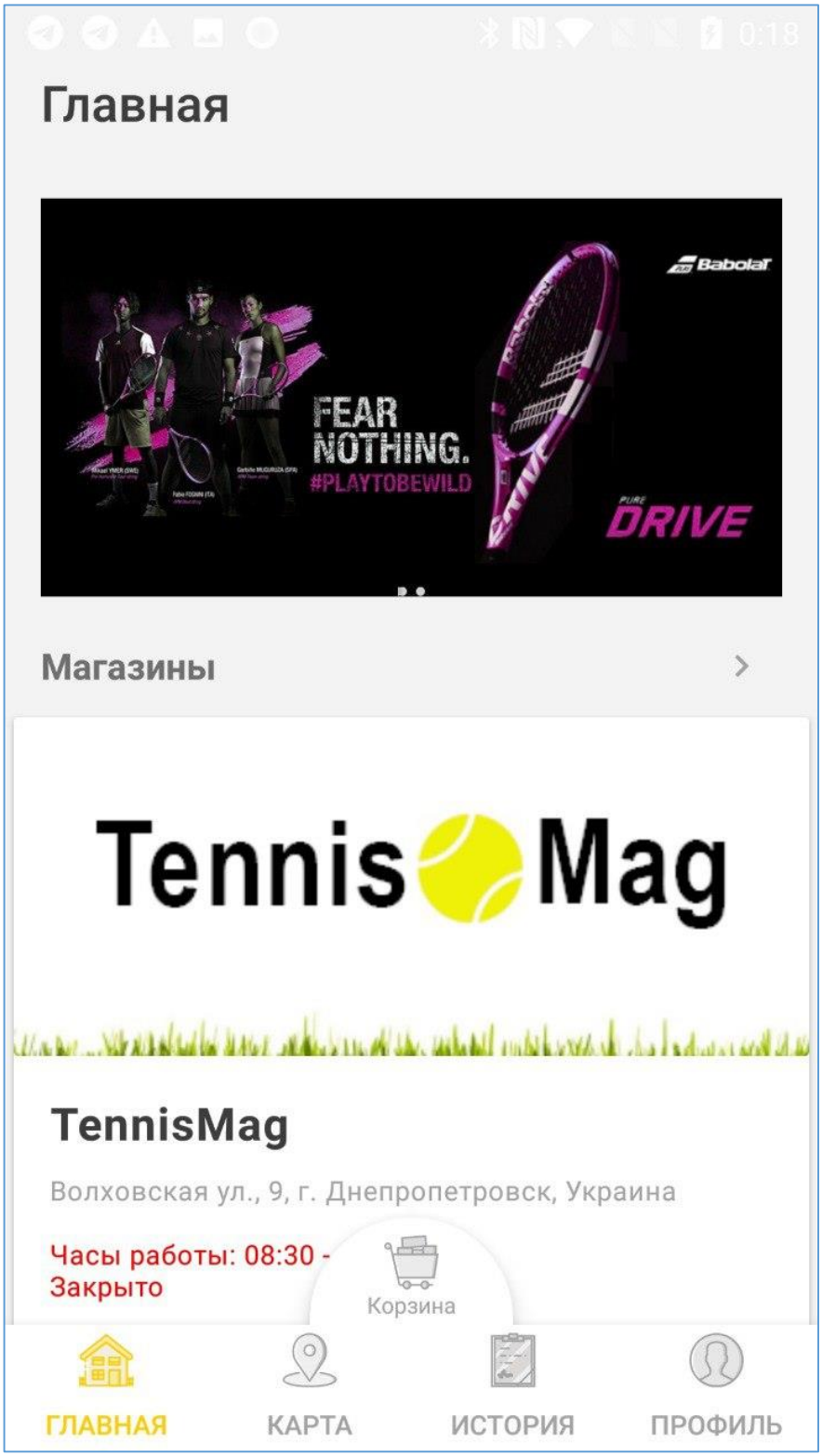


Рис. 3.6 – Головний екран

Натиснувши на вкладку мапа, користучач перейде до вкладки мапи. На якій він зможе побачити саму мапу та маркери на ній, кожен маркер відповідає конкретному магазину. Натиснувши на маркер, користувач зможе побачити картку магазину, на якій буде розташована коротка інформація про магазин. Натиснувши на картку, користувач перейде до екрану конкретного магазину.

Рис. 3.7 – Екран мапи

Натиснувши на вкладку замовлення, користувач перейде до вкладки зі списком замовлень. Користувач зможе побачити інформацію про кожне замовлення, що він створював раніше.

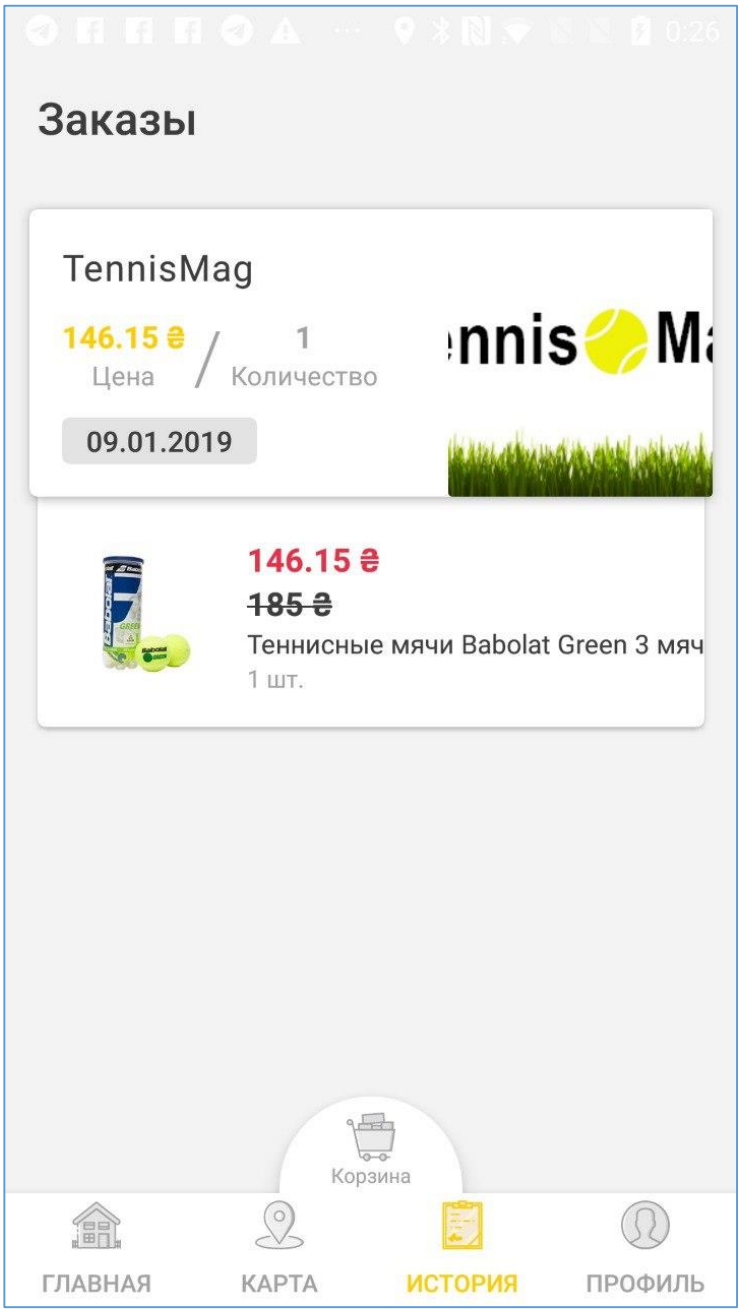


Рис. 3.8 – Екран замовлень

Натиснувши на вкладку профілю, користувач перейде до вкладки профіля, де зможе побачити інформацію про себе, а саме: аватар, повне ім’я, телефон, поштову адресу. Також на екрані профілю є список платіжних карт, які користувач може використовувати в процесі оплати замовлення. Разом зі списком карт, на екрані присутня кнопка для додавання платіжної карти, щоб користувач міг додати нову карту. Натиснувши на хрестик навпроти кожної карти, користувач може видалити відповідну карту.

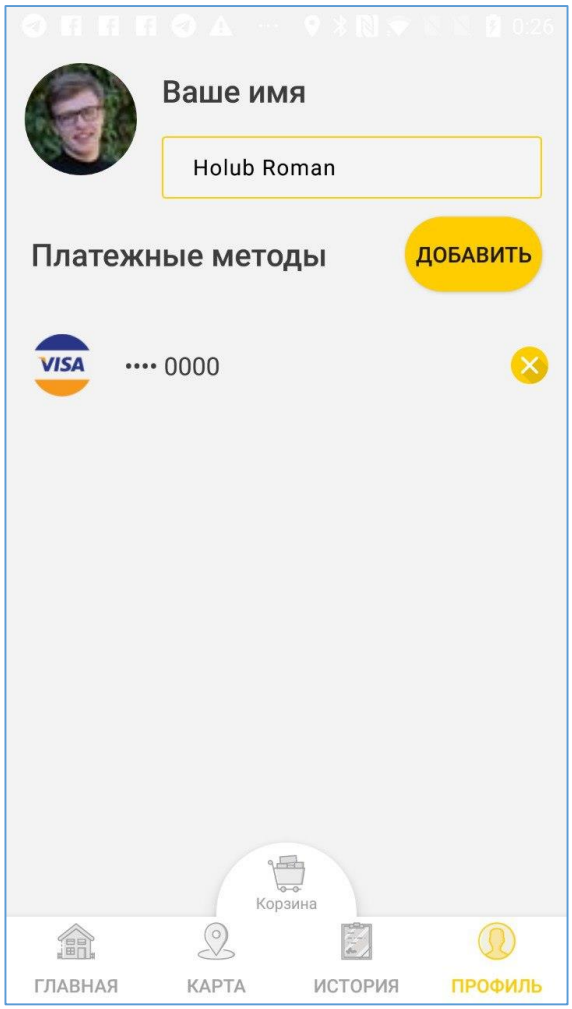


Рис. 3.9 – Екран профілю

Натиснувши на кнопку кошик, користувач перейде до екрану кошика, де він може побачити набір товарів відповідних до конкретного магазину, кнопку оформити замовлення, щоб оформити замовлення в конкретному магазині.

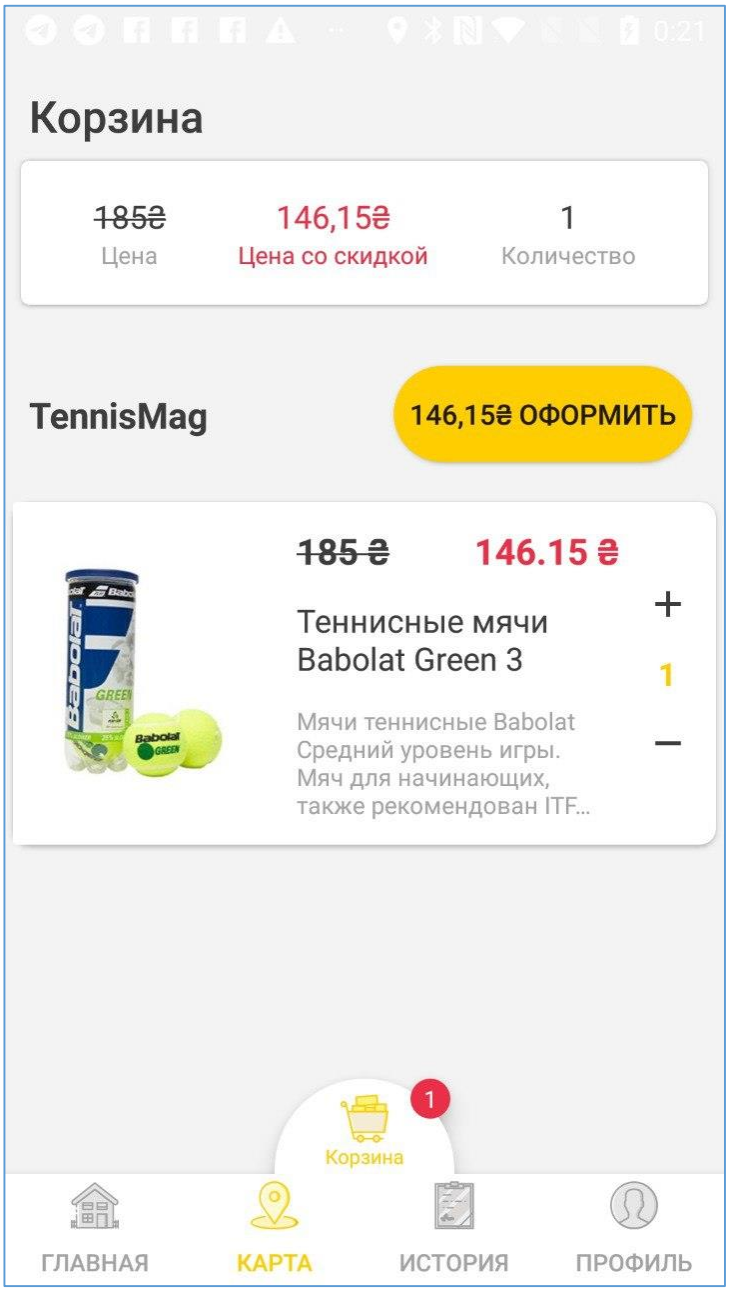


Рис. 3.10 – Екран кошика

Натиснувши на кнопку оформити замовлення в екрані кошику, користувач перейде до екрану оформлення замовлення, де він матиме можливість додати інформацію до замовлення, а саме: ім’я, телефон, адресу доставки. Також користувач побачить список товарів і кнопку заплатити за замовлення, натиснувши на яку користувач зможе перейти до екрану оплати замовлення.

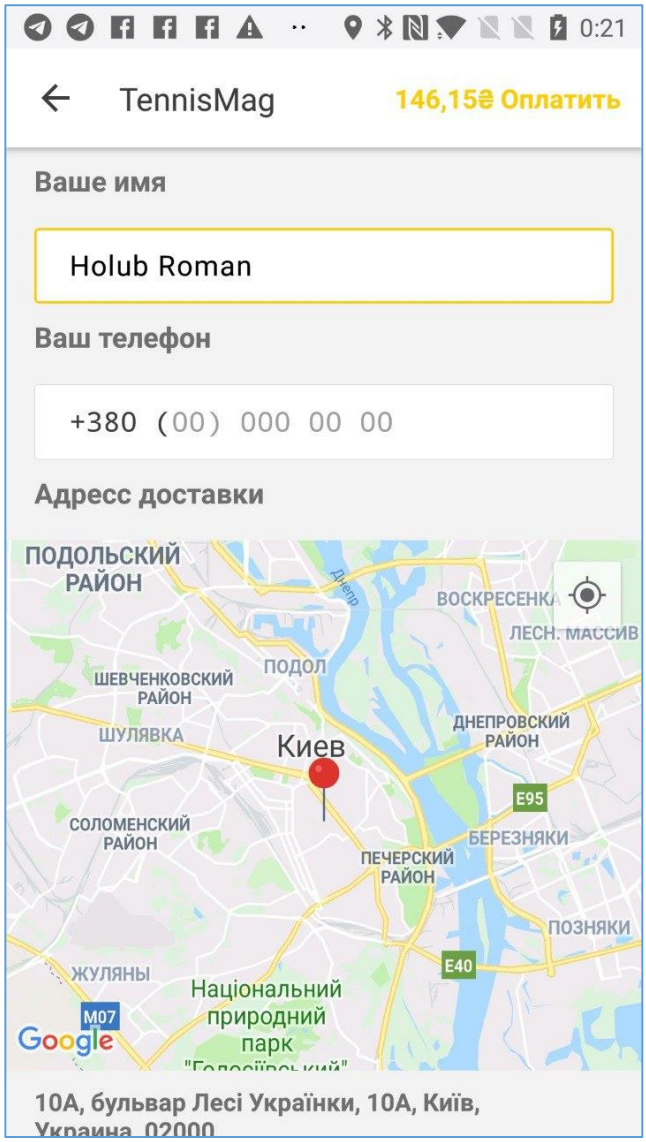


Рис. 3.11 – Екран оформлення замовлення

Натиснувши на кнопку заплатити за замовлення, користувач перейде до екрану оплати замовлення. На цьому екрані присутні список платіжних карт, кнопка для додавання нової платіжної карти. Натиснувши на конкретну карту, користувачу буде показане вікно підтвердження оплати від банку, яким користується користувач. Після успішної оплати замовлення, користувач перейде до головного екрану.

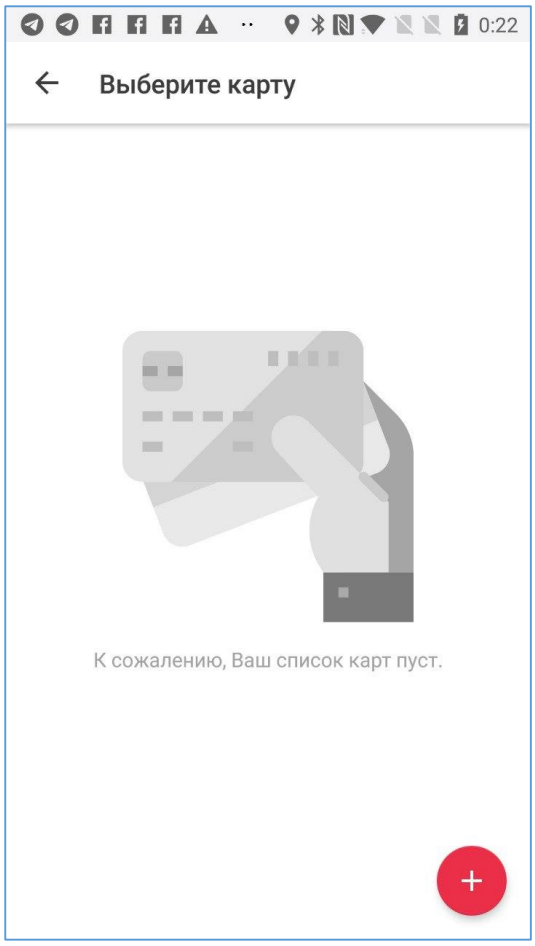


Рис. 3.12 – Екран оплати замовлення

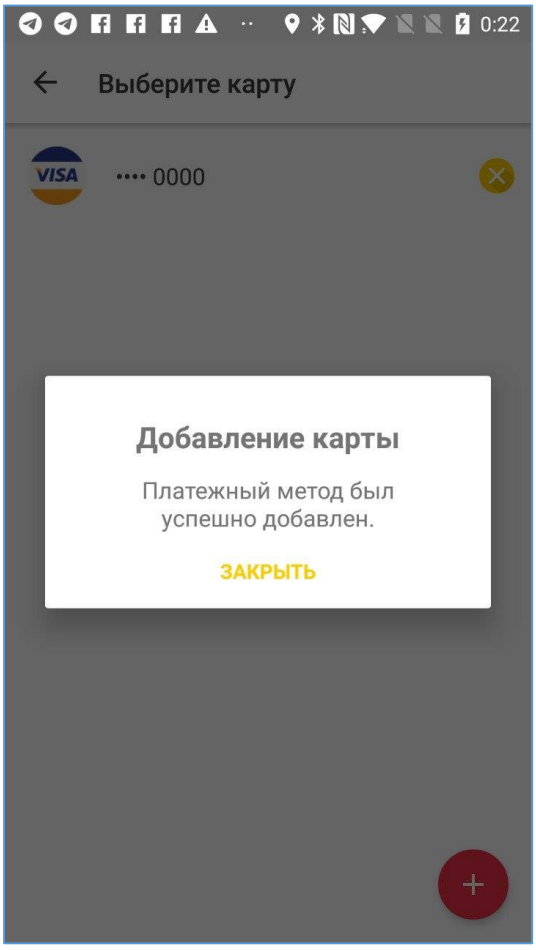


Рис. 3.13 – Екран успішного додавання карти

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було продемонстровано інструкцію користувача клієнтської частини програмної платформи. Ця інструкція достатньо детальна для того, щоб сформувати розуміння про функціонал програмного забезпечення, а також використовувати його.

ВИСНОВКИ

В даній дипломній роботі було розроблене технічне завдання, було проаналізовано всі аспекти технічної задачі при розробці мобільної платформи по придбанню товарів з урахуванням місцезнаходження користувача для Android.

В процесі створення програмної платформи було розроблено також універсальні модулі, які в подальшому були використані у великій кількості місць мобільної платформи.

Також, розробляючи дане технічне завдання, було розроблено схему бази даних, яка дозволяє розширювати асортимент товарів у мобільній платформі і витрачати мінімум часу при редагуванні існуючих товарів в платформі.

Щодо дизайну, то було проаналізовано структуру користувацького інтерфейсу у сучасних патернах графічного дизайну для мобільних додатків, що дозволило мені власноруч створити сучасний, зручний, гнучкий та простий інтерфейс для користувача. Є популярна точка зору серед програмістів, що дизайн це другорядна частина програмного забезпечення, думка автора не співпадає з цим, тому приділив особливе значення UI/UX дизайну платформи. Якщо звертатися до значення слова ІНТЕРФЕЙС(загальна границя між окремими системами), то можна зрозуміти, що графічний інтерфейс є головним каналом взаємодії користувача та реалізованого програмістом функціонала.

На мій погляд, область в якій працює дана платформа є прогресивною, тому було приділено окрему увагу архітектурі програмного забезпечення, тому що при розвитку даної платформи будуть залучені додаткові учасники команди. Якщо архітектура програмного забезпечення не буде слідувати популярним рішенням та патернам, то цій команді буде тяжко розібратися і розвивати дану платформу. Тому було використано два популярних патерна архітектури програмного забезпечення, а саме VIPER та MVVM.

Ці патерни дозволили мені створити програмне забезпечення, де певні модулі використовуються повторно в різних місцях програмної платформи, і немає необхідності реалізовувати одну і ту ж функціональність декілька разів.

Розглядаючи окремо патерн VIPER, можна сказати, що головною перевагою цього патерну є розшарування програмної платформи на окремі шари, що дозволяє розподілити відповідальність кожного шару, що також сильно пересікається з патерном SRP(Single Responsibility Principle) — принцип однієї відповідальності, адже кожен шар має свою, тільки одну відповідальність.

Якщо говорити окремо про патерн MVVM, то цей патерн дозволяє перейти від ООП принципу, коли ми виконуємо дію над об’єктом, до принципу Функціонального програмування, що дозволяє працювати з даними не як з об’єктами, а як з потоками даних. В ООП даний патерн називається Observer.

У процесі створення програмної платформи було вивчено велику кількість технічної літератури з програмування, наприклад “Реактивне програмування з використанням RxJava”, на основі цієї літератури було обрано архітектуру программного забезпечення, а також основні деталі концепції дизайну та користувацького інтерфейсу.

Розробляючи дану дипломну роботу були розглянуті всі частини, що стосуються розробки мобільної платформи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Reactive Programming with RxJava 1st Edition[Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2016 – Режим доступу: https://www.amazon.com/Reactive-Programming-RxJava-Asynchronous-Applications/dp/1491931655

2. The book of VIPER[Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2017 – Режим доступу: https://github.com/strongself/The-Book-of-VIPER

3. Software Architecture Patterns [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2015 – Режим доступу: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/

4. Design Patterns: Elements of Reusable Object-Oriented Patterns [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 1994 – Режим доступу: https://www.oreilly.com/library/view/design-patterns-elements/0201633612/

5. Reactive Streams in Java — Concurrency with RxJava, Reactor, and Akka Streams [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2013 – Режим доступу: https://www.apress.com/us/book/9781484241752

6. Kotlin in Action [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2017 – Режим доступу: https://www.manning.com/books/kotlin-in-action

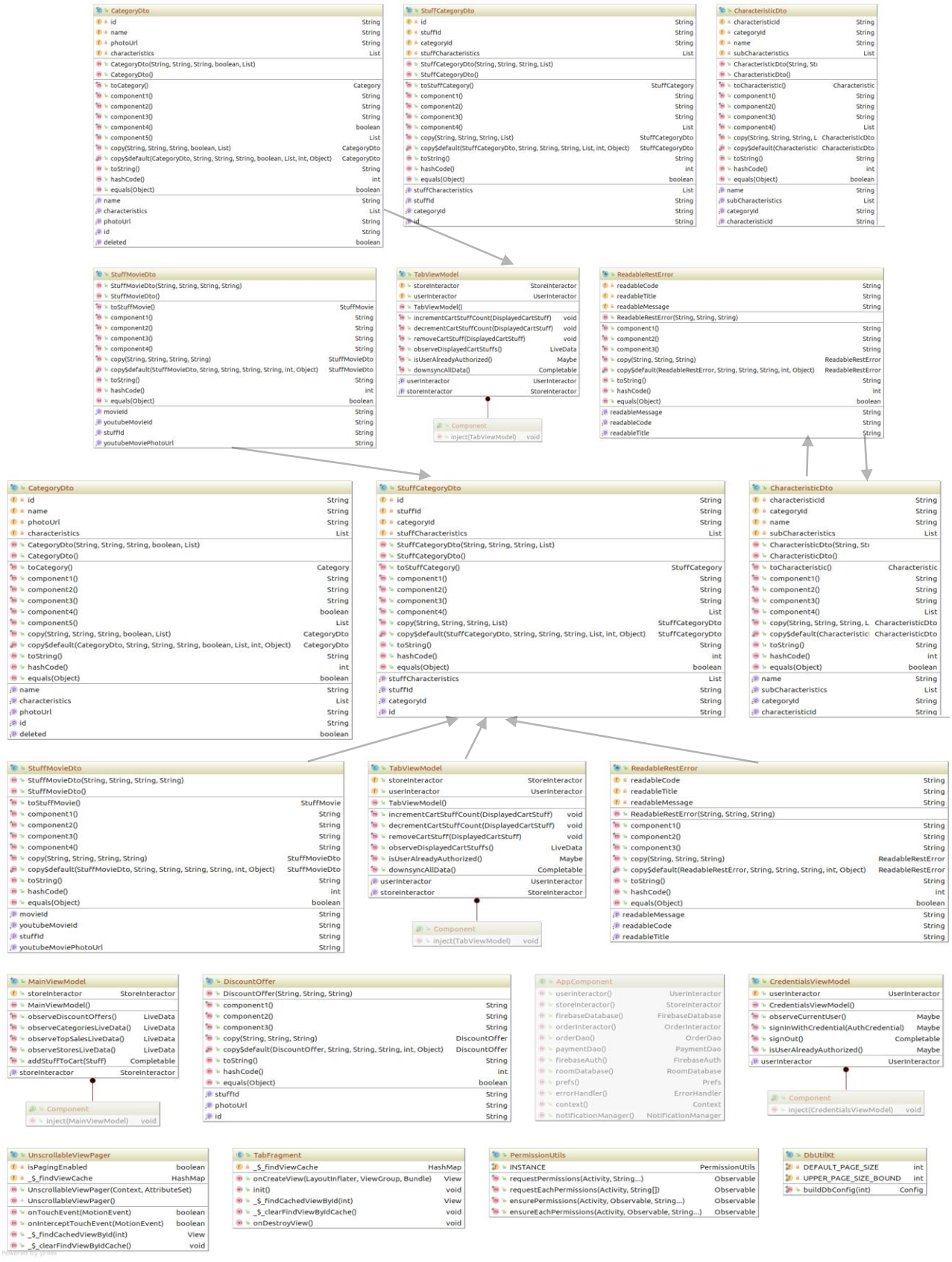
7. Kotlin for Android Developers [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2017 – Режим доступу: https://leanpub.com/kotlin-for-android-developers

8. Programming Kotlin [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2017 – Режим доступу: https://www.packtpub.com/application-development/programming-kotlin

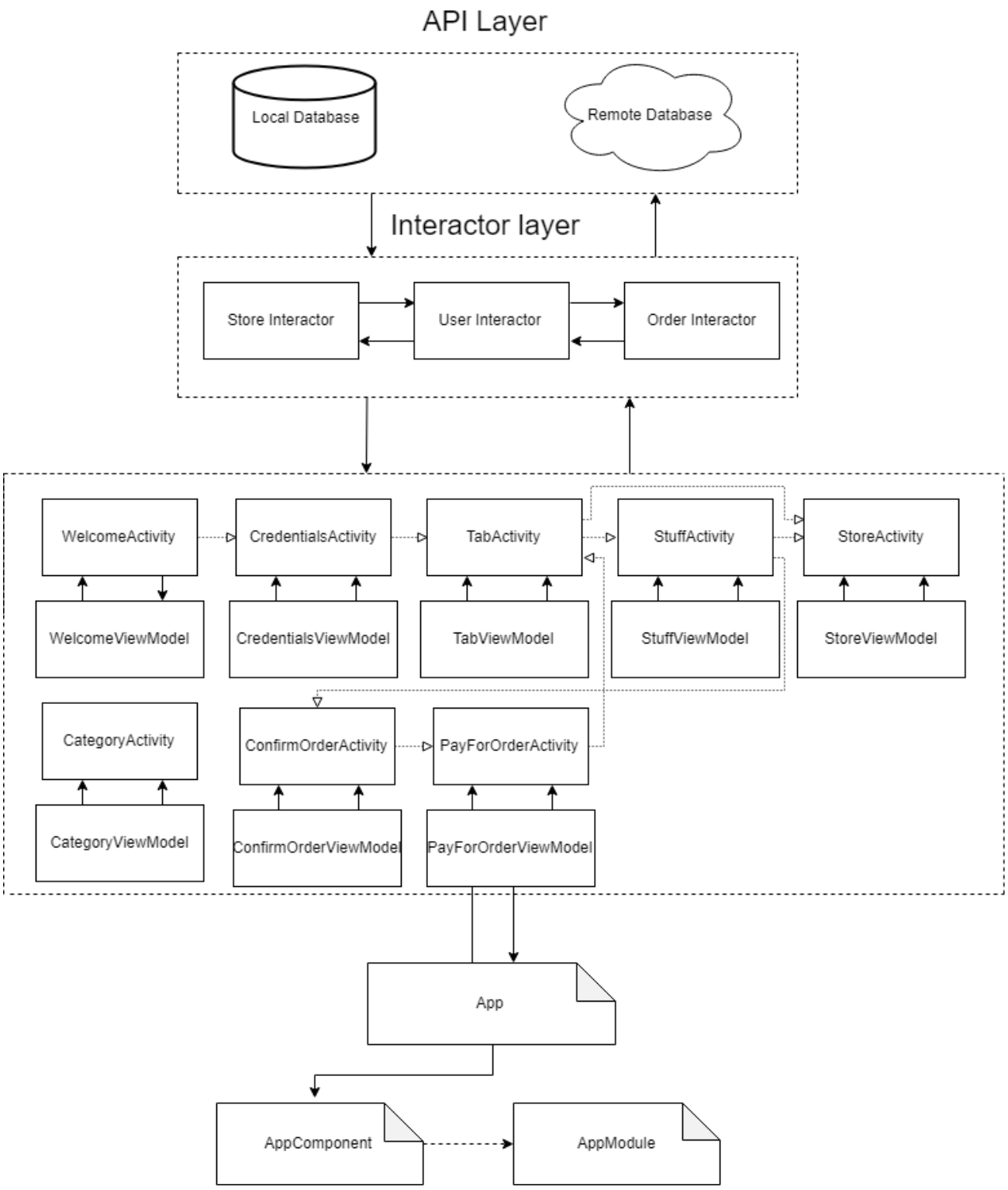
9. Android Development with Kotlin [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2017 – Режим доступу: https://www.packtpub.com/application-development/android-development-kotlin

10. Dependency Injection Principles, Practices, and Patterns [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2019 – Режим доступу: https://livebook.manning.com/#!/book/dependency-injection-principles-practices-patterns

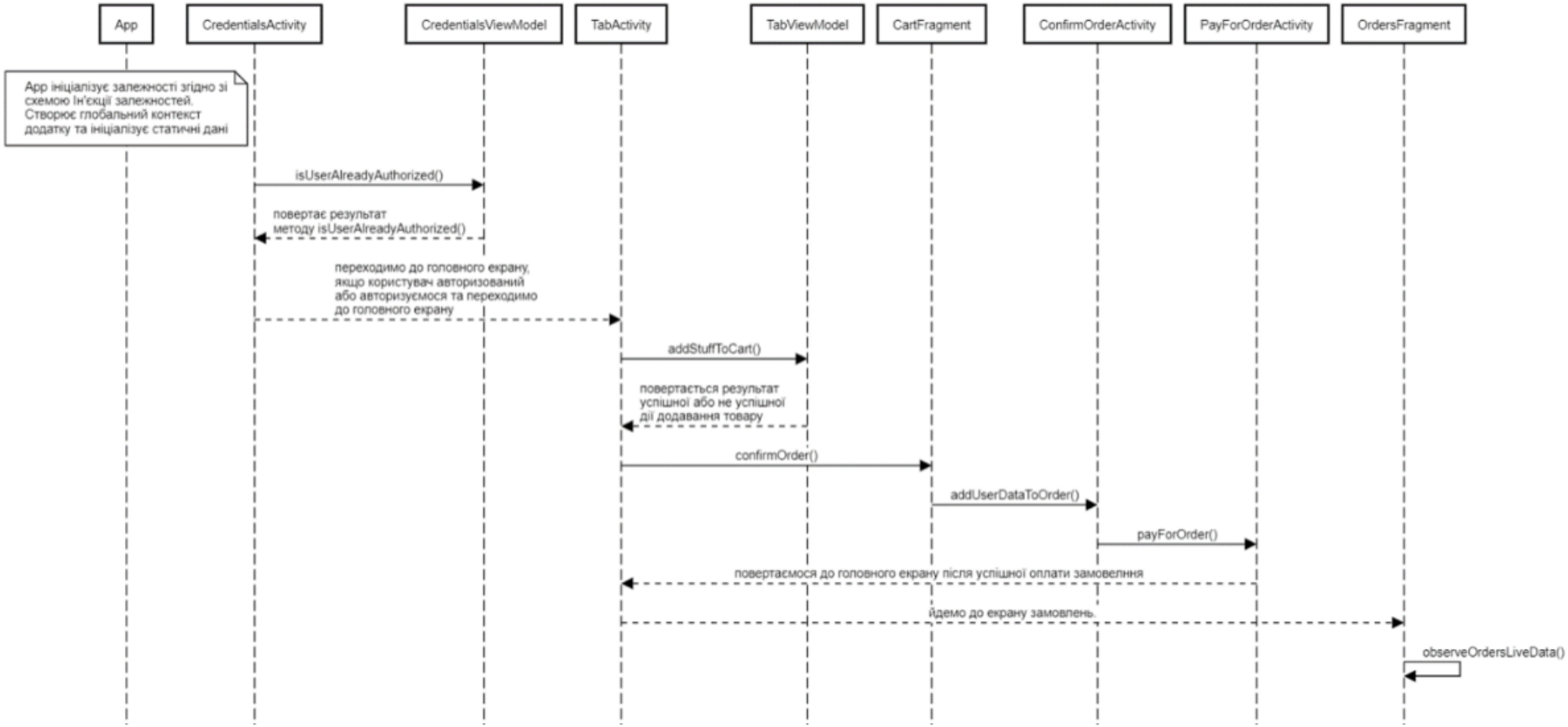
11. Android Apprentice: Beginning Android Development with Kotlin 1.2 [Електронний ресурс]:(Книга) – Електрон. Дан. (1 файл). – 2018 – Режим доступу: https://www.amazon.com/Android-Apprentice-Beginning-Development-Kotlin/dp/1942878494



					ІАЛЦ.467200.004 Д1	
					Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android *Діаграма класів*	*Літ.* / *Маса* / *Масштаб*
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Голуб Р.О.*				*Аркуш 1* / *Аркушів 1*
Перевірив		*Алещенко О.В.*				
Н. Контр.		*Сімоненко В.П.*				*НТУУ «КПІ», ФІОТ, ІП-53*
Затв.		*Стіренко С.Г.*				



					ІАЛЦ.467200.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android *Діаграма компонентів*	Літ. / Маса / Масштаб
Розробив		Голуб Р.О.				
Перевірив		Алещенко О.В.				Аркуш 1 / Аркушів 1
Н. Контр.		Сімоненко В.П.				НТУУ «КПІ», ФІОТ, ІП-53
Затв.		Стіренко С.Г.				



					ІАЛЦ.467200.006 Д3	
Зм.	Арк.	№ документа	Підпис	Дата	Система придбання товарів з урахуваннямміснезнаходження користувача для мобільної платформи Android	Літера / Маса / Масштаб
Розробив		Голуб Р.О.			Діаграма послідовності	Аркуш 1 / Аркушів 1
Перевірив		Алещенко О.В.				
Н. кон.		Сімоненко В.П.				НТУУ "КПІ ім. І. Сікорського" кафедра ОТ, гр. ІП-53
Затвердив		Стіренко С.Г.				

ДОДАТОК 4

Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android

Лістинг програми

ІАЛЦ.467200.007 Д4

Аркушів 23

2019

```
class App : Application() {

  private lateinit var appComponent: AppComponent

  @SuppressLint("CheckResult")
  override fun onCreate() {
    super.onCreate()
    Traceur.enableLogging()
    Fabric.with(this, Crashlytics())
    instance = this

    warmUpCache()
    appComponent = DaggerAppComponent.builder()
      .appModule(AppModule(applicationContext))
      .build()
    FirebaseMessaging.getInstance().isAutoInitEnabled = true
    generateDeviceId(appComponent.prefs())

    registerReceiver(LocationChangeReceiver(),
IntentFilter(GoogleService.LOCATION_BROADCAST_RECEIVER_ACTION))
    startService(Intent(this, GoogleService::class.java))
  }


private fun generateDeviceId(prefs: Prefs) {
    val deviceIdOpt = prefs.get<String>(Prefs.Key.DEVICE_ID)
      .map { Optional.ofNullable(it) }
      .toSingle(Optional.empty())
      .blockingGet()
    if (!deviceIdOpt.isPresent) {
      prefs.blockingPut(Prefs.Key.DEVICE_ID,
Settings.Secure.getString(this.contentResolver, Settings.Secure.ANDROID_ID))
    }
```

```
    }

    private fun warmUpCache() {
        FacebookSdk.sdkInitialize(this)
        AppEventsLogger.activateApp(this)
        ISOChronology.getInstance()
    }

    fun appComponent(): AppComponent {
        return appComponent
    }

    companion object {

        lateinit var instance: App
            private set
    }
}
class CredentialsActivity : AppCompatActivity() {

    private var binding: ActivityCredentialsBinding? = null
    private var viewModel: CredentialsViewModel? = null
    private var googleSignInClient: GoogleSignInClient? = null
    private var callbackManager: CallbackManager? = null


    @SuppressLint("CheckResult")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        window.setFlags(
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS
        )
```

```
        navigateToTabsOrInitActivity()
    }

    @SuppressLint("CheckResult")
    private fun navigateToTabsOrInitActivity() {
        binding = DataBindingUtil.setContentView(this, R.layout.activity_credentials)
        viewModel =
ViewModelProviders.of(this).get(CredentialsViewModel::class.java)
        viewModel?.isUserAlreadyAuthorized()
            ?.subscribe({
                TabActivity.start(this, true)
            },
                {
                    ErrorHandler.logError(it) },
                {
                    initGoogleAuth()
                    initFacebookAuth()
                    hideProgressBarAndEnableButton()
                })
    }

    public override fun onStart() {
        super.onStart()
        navigateToTabsOrInitActivity()
    }

    @SuppressLint("CheckResult")
    private fun processCurrentUser() {
        viewModel?.observeCurrentUser()
            ?.subscribe({
                hideProgressBarAndEnableButton()
                TabActivity.start(this, true)
            }, {
```

```
            ErrorHandler.logError(it)
            showAuthenticationFailed()
        })
    }

    public override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

        if (requestCode == RC_GOOGLE_SIGN_IN) {
            val task = GoogleSignIn.getSignedInAccountFromIntent(data)
            try {
                val account = task.getResult(ApiException::class.java)
                firebaseAuthWithGoogle(account!!)
            } catch (e: ApiException) {
                showAuthenticationFailed()
            }
        }



        callbackManager?.onActivityResult(requestCode, resultCode, data)
    }

    private fun showAuthenticationFailed() {
        binding?.view?.let {
            Snackbar.make(
                it,
                resources.getString(R.string.authentication_failed),
                Snackbar.LENGTH_SHORT
            ).show()
        }
    }
```

```
    private fun showLogoutSuccessful() {
        binding?.view?.let {
            Snackbar.make(
                it,
                resources.getString(R.string.logout_successful),
                Snackbar.LENGTH_SHORT
            ).show()
        }
    }

    @SuppressLint("CheckResult")
    private fun firebaseAuthWithGoogle(acct: GoogleSignInAccount) {
        showProgressBarAndDisableButton()


viewModel?.signInWithCredential(GoogleAuthProvider.getCredential(acct.idToke
n, null))
            ?.subscribe({
                it.addOnCompleteListener(this) { task ->
                    if (task.isSuccessful) {
                        processCurrentUser()
                    } else {
                        showAuthenticationFailed()
                    }
                    hideProgressBarAndEnableButton()
                }
            }, { ErrorHandler.logError(it) })
    }

    private fun googleSignIn() {
        val signInIntent = googleSignInClient?.signInIntent
        startActivityForResult(signInIntent, RC_GOOGLE_SIGN_IN)
    }
```

```
  @SuppressLint("CheckResult")
  private fun googleSignOut() {
    viewModel?.signOut()
      ?.subscribe({}, { ErrorHandler.logError(it) })
    googleSignInClient?.signOut()?.addOnCompleteListener(this) {
showLogoutSuccessful() }
  }

  private fun initFacebookAuth() {
    callbackManager = CallbackManager.Factory.create()
    binding?.btnFacebookSignIn?.setReadPermissions("email", "public_profile")
    binding?.btnFacebookSignIn?.registerCallback(callbackManager, object :
FacebookCallback<LoginResult> {
      override fun onSuccess(loginResult: LoginResult) {
        signInWithFacebook(loginResult.accessToken)
      }

      override fun onCancel() {
        showAuthenticationFailed()
      }

      override fun onError(error: FacebookException) {
        showAuthenticationFailed()
      }
    })
  }

  private fun initGoogleAuth() {
    val gso =
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
      .requestIdToken(getString(R.string.default_web_client_id))
      .requestEmail()
```

```
            .build()

        googleSignInClient = GoogleSignIn.getClient(this, gso)
    }

    @SuppressLint("CheckResult")
    private fun signInWithFacebook(token: AccessToken) {
        showProgressBarAndDisableButton()


viewModel?.signInWithCredential(FacebookAuthProvider.getCredential(token.tok
en))
            ?.subscribe({
                it.addOnCompleteListener(this) {
                    if (it.isSuccessful) {
                        processCurrentUser()
                    } else {
                        showAuthenticationFailed()
                    }
                    hideProgressBarAndEnableButton()
                }
            }, { ErrorHandler.logError(it) })
    }

    private fun onBtnGoogleClicked() {
        googleSignIn()
    }

    private fun onBtnFacebookClicked() {
    }


    private fun showProgressBarAndDisableButton() {
```

```
        binding?.let { binding ->
            binding.flProgressBar.visibility = View.VISIBLE
            binding.flProgressBar.setOnClickListener {}
            binding.btnGoogleSignIn.isEnabled = false
            binding.btnGoogleSignIn.setOnClickListener {}
            binding.btnFacebookSignIn.isEnabled = false
            binding.btnFacebookSignIn.setOnClickListener {}
        }
    }

    private fun hideProgressBarAndEnableButton() {
        binding?.let { binding ->
            binding.flProgressBar.visibility = View.INVISIBLE
            binding.btnGoogleSignIn.isEnabled = true
            binding.btnGoogleSignIn.setOnClickListener { onBtnGoogleClicked() }
            binding.btnFacebookSignIn.isEnabled = true
            binding.btnFacebookSignIn.setOnClickListener { onBtnFacebookClicked()
}
        }
    }

    companion object {

        private const val RC_GOOGLE_SIGN_IN = 9001

        fun start(currentActivity: AppCompatActivity, finishPrevious: Boolean) {
            if (finishPrevious) {
                currentActivity.finishAffinity()
            }
            currentActivity.startActivity(Intent(currentActivity,
CredentialsActivity::class.java))
        }
    }
```

```
class CredentialsViewModel : ViewModel() {

  @Inject
  lateinit var userInteractor: UserInteractor

  init {
    DaggerCredentialsViewModel_Component.builder()
      .appComponent(App.instance.appComponent())
      .build()
      .inject(this)
  }

  fun observeCurrentUser(): Maybe<FirebaseUser> {
    return userInteractor.observeCurrentUser()
      .subscribeOn(Schedulers.io())
      .observeOn(AndroidSchedulers.mainThread())
  }

  fun signInWithCredential(credential: AuthCredential):
Maybe<Task<AuthResult>> {
    return userInteractor.signInWithCredential(credential)
      .subscribeOn(Schedulers.io())
      .observeOn(AndroidSchedulers.mainThread())
  }

  fun signOut(): Completable {
    return userInteractor.signOut()
      .subscribeOn(Schedulers.io())
      .observeOn(AndroidSchedulers.mainThread())
  }

  fun isUserAlreadyAuthorized(): Maybe<FirebaseUser> {
    return userInteractor.isUserAlreadyAuthorized()
```

```
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
    }

    @AppScope
    @dagger.Component(dependencies = [(AppComponent::class)])
    interface Component {
        fun inject(credentialsViewModel: CredentialsViewModel)
    }
}


class WelcomeActivity : MaterialIntroActivity() {

    private var viewModel: WelcomeViewModel? = null
    private var batteryDialog: Dialog? = null

    @SuppressLint("CheckResult")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        viewModel =
ViewModelProviders.of(this).get(WelcomeViewModel::class.java)
        viewModel?.isUserAlreadyAuthorized()
            ?.subscribe({
                TabActivity.start(this, true)
            },
                { ErrorHandler.logError(it) },
                {
                    welcomeSlides()
                    prepareBatteryOptimizationDialog(this)
                    showBatteryDialog()
                })
```

```
    }

    private fun showBatteryDialog() {
        if (isBatteryOptimized() && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP_MR1) {
            batteryDialog?.show()
        }
    }

    private fun isBatteryOptimized(): Boolean {
        val pwrm =
applicationContext.getSystemService(Context.POWER_SERVICE) as
PowerManager
        val name = applicationContext.packageName
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            return !pwrm.isIgnoringBatteryOptimizations(name)
        }
        return false
    }

    private fun welcomeSlides() {
        addSlide(SlideFragmentBuilder()
            .backgroundColor(R.color.best_stuff_offers_background)
            .buttonsColor(R.color.next_button_background)
            .image(R.drawable.ic_tennis_racket)

.title(resources.getString(R.string.activity_welcome_best_stuff_offers_title))

.description(resources.getString(R.string.activity_welcome_best_stuff_offers_desc
ription))
            .build())

        addSlide(SlideFragmentBuilder()
```

```
            .backgroundColor(R.color.location_based_offers_background)
            .buttonsColor(R.color.next_button_background)
            .image(R.drawable.ic_location_based)

.title(resources.getString(R.string.activity_welcome_location_based_offers_title))

.description(resources.getString(R.string.activity_welcome_location_based_offers
_description))
            .build())

        addSlide(SlideFragmentBuilder()
            .backgroundColor(R.color.shop_aggregation_background)
            .buttonsColor(R.color.next_button_background)
            .image(R.drawable.ic_shop_aggregation)

.title(resources.getString(R.string.activity_welcome_shop_aggregation_title))

.description(resources.getString(R.string.activity_welcome_shop_aggregation_des
cription))
            .build())

        addSlide(SlideFragmentBuilder()
            .backgroundColor(R.color.filters_system_background)
            .buttonsColor(R.color.next_button_background)
            .image(R.drawable.ic_filters)
            .title(resources.getString(R.string.activity_welcome_filters_system_title))

.description(resources.getString(R.string.activity_welcome_filters_system_descript
ion))
            .build())

    }
```

```
  override fun onFinish() {
     super.onFinish()
     CredentialsActivity.start(this, true)
  }

  override fun onDestroy() {
     super.onDestroy()
     if (batteryDialog?.isShowing == true) {
        batteryDialog?.dismiss()
     }
  }

  private fun prepareBatteryOptimizationDialog(context: Context) {
     val dialogBuilder = MaterialDialog.Builder(context)

     val dialogView =
LayoutInflater.from(context).inflate(R.layout.dialog_battery_optimization, null,
false)
     dialogBuilder.customView(dialogView, false)

     val btnClose = (dialogView.findViewById(R.id.btn_close) as TextView)
     val btnDisableOptimization =
(dialogView.findViewById(R.id.btn_battery_optimization) as TextView)

     batteryDialog = dialogBuilder.build()
     batteryDialog?.setCancelable(false)

     btnClose.setOnClickListener {
        batteryDialog?.dismiss()
     }
     btnDisableOptimization.setOnClickListener {
        val intent =
Intent(Settings.ACTION_IGNORE_BATTERY_OPTIMIZATION_SETTINGS)
```

```
            startActivity(intent)
            batteryDialog?.dismiss()
        }

    }

    companion object {

        fun start(currentActivity: AppCompatActivity, finishPrevious: Boolean) {
            if (finishPrevious) {
                currentActivity.finishAffinity()
            }
            currentActivity.startActivity(Intent(currentActivity,
WelcomeActivity::class.java))
        }
    }
class WelcomeViewModel : ViewModel() {

    @Inject
    lateinit var userInteractor: UserInteractor

    init {
        DaggerWelcomeViewModel_Component.builder()
            .appComponent(App.instance.appComponent())
            .build()
            .inject(this)
    }

    fun isUserAlreadyAuthorized(): Maybe<FirebaseUser> {
        return userInteractor.isUserAlreadyAuthorized()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
    }
```

```
    @AppScope
    @dagger.Component(dependencies = [(AppComponent::class)])
    interface Component {
        fun inject(welcomeViewModel: WelcomeViewModel)
    }
}
class StoreActivity : AppCompatActivity() {

    private var binding: ActivityStoreBinding? = null
    private var viewModel: StoreViewModel? = null

    @SuppressLint("CheckResult")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        initStatusBar()
        binding = DataBindingUtil.setContentView(this, R.layout.activity_store)
        viewModel = ViewModelProviders.of(this).get(StoreViewModel::class.java)


        if (intent != null && intent?.getStringExtra(STORE_ID) != null) {
            intent?.getStringExtra(STORE_ID)?.let {
                val ft = supportFragmentManager.beginTransaction()
                ft.replace(R.id.cl_fragment_container,
StoreContainerFragment.newInstance(it))
                ft.commit()
            }
        }
    }

    private fun initStatusBar() {
        window.setFlags(
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
```

```
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS
        )
    }

    companion object {

        val STORE_ID = "STORE_ID"

        fun start(currentActivity: AppCompatActivity, storeId: String, finishPrevious:
Boolean) {
            if (finishPrevious) {
                currentActivity.finishAffinity()
            }
            val intent = Intent(currentActivity, StoreActivity::class.java)
            intent.putExtra(STORE_ID, storeId)
            currentActivity.startActivity(intent)
        }
    }
}

class StoreViewModel : ViewModel() {

    @Inject
    lateinit var storeInteractor: StoreInteractor

    init {
        DaggerStoreViewModel_Component.builder()
            .appComponent(App.instance.appComponent())
            .build()
            .inject(this)
    }

    @AppScope
```

```
  @dagger.Component(dependencies = [(AppComponent::class)])
  interface Component {
    fun inject(storeViewModel: StoreViewModel)
  }
}
class StoreInteractor(
  private val prefs: Prefs,
  private val storeDao: StoreDao,
  private val firebaseDatabase: FirebaseDatabase,
  private val notificationManager: NotificationManager
) {

  val onFiltersChangedPublishSubject = PublishSubject.create<Boolean>()

  fun observeAllStoresList(latLng: LatLng, pageSize: Int):
LiveData<PagedList<Store>> {
    return LivePagedListBuilder(
      storeDao.getAllStoresDataSourceFactory(latLng.latitude, latLng.longitude),
      buildDbConfig(pageSize)
    ).build()
  }

  fun setOrder(
    fondyOrderId: String, storeId: String, name: String, phone: String,
    deliveryLatitude: Double, deliveryLongitude: Double, deliveryAddress: String
  ): Completable {
    return storeDao.observeSpecificStoreForId(storeId)
      .flatMapCompletable {
        Completable.fromRunnable {
          var order = Order(
            UUID.randomUUID().toString(), Date().time,
            storeId, it.name, it.photoUrl, 0,
```

```
                0, 0, deliveryLatitude, deliveryLongitude, deliveryAddress, name,
phone
            )
            var orderStuffs =
storeDao.getDisplayedCartStuffsForStoreId(order.storeId)
                .map {
                    val orderStuff = OrderStuff(
                        UUID.randomUUID().toString(),
                        order.orderId,
                        it.stuff.stuffId,
                        it.stuff.stuffCategoryId,
                        it.stuff.name,
                        it.stuff.photoUrl,
                        it.stuff.description,
                        it.stuff.price,
                        (it.stuff.price * (1 - it.stuff.discountPercentage * 0.01)).toInt(),
                        it.cartStuff.count * it.stuff.price,
                        it.cartStuff.count * (it.stuff.price * (1 -
it.stuff.discountPercentage * 0.01)).toInt(),
                        it.cartStuff.count
                    )
                    storeDao.upsertOrderStuff(orderStuff)
                    storeDao.removeCartStuff(it.cartStuff)
                    orderStuff
                }
            order.stuffsCount = orderStuffs.size
            order.priceWithDiscount = orderStuffs.sumBy {
it.totalPriceWithDiscount }
            order.price = orderStuffs.sumBy { it.totalPrice }
            storeDao.upsertOrder(order)
        }
    }
}
```

```
  fun observeStuffsForStoreCategoryIdLiveData(storeCategoryId: String,
pageSize: Int): LiveData<PagedList<Stuff>> {
      return LivePagedListBuilder(
          storeDao.observeStuffsForStoreCategoryIdLiveData(storeCategoryId),
          buildDbConfig(pageSize)
      ).build()
  }

  fun observeStoreCategoryForCategoryId(categoryId: String):
Maybe<StoreCategory> {
      return storeDao.observeStoreCategoryForCategoryId(categoryId)
  }

  fun dropAllFiltersForCategoryId(categoryId: String): Completable {
      return storeDao.observeCharacteristicsForCategoryIdMaybe(categoryId)
          .flatMap {
              Maybe.fromCallable {
                  it.forEach {

storeDao.getSubCharacteristicsForCharacteristicId(it.characteristicId)
                          .forEach {
                              when (it.type) {
                                  SubCharacteristicType.STRING -> {
                                      it.stringValue = ""
                                  }
                                  SubCharacteristicType.FLOAT -> {
                                      it.floatEnd = it.floatRangeEnd
                                      it.floatStart = it.floatRangeStart
                                  }
                                  SubCharacteristicType.INT -> {
                                      it.intEnd = it.intRangeEnd
                                      it.intStart = it.intRangeStart
```

```
                        }
                        SubCharacteristicType.BOOLEAN -> {
                            it.booleanValue = false
                        }
                        SubCharacteristicType.DROP -> {
                            it.stringValue = ""
                        }
                        else -> {
                            it.stringValue = ""
                        }
                    }
                    storeDao.upsertSubCharacteristic(it)
                }
            }
            onFiltersChangedPublishSubject.onNext(true)
        }
    }.ignoreElement()
}

fun observeSubCharacteristicsForCharacteristicsId(
    characteristicsId: List<String>,
    pageSize: Int
): LiveData<PagedList<SubCharacteristicWithCharacteristic>> {
    return LivePagedListBuilder(
        storeDao.observeSubCharacteristicsForCharacteristicIds(characteristicsId),
        buildDbConfig(pageSize)
    ).build()
}

fun observeCharacteristicsForCategoryId(categoryId: String):
Maybe<List<Characteristic>> {
    return storeDao.observeCharacteristicsForCategoryIdMaybe(categoryId)
}
```

```
  @SuppressLint("CheckResult")
  fun upsertSubCharacteristic(subCharacteristic: SubCharacteristic) {
     Completable.fromCallable {
        storeDao.upsertSubCharacteristic(subCharacteristic)
     }.subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe({ onFiltersChangedPublishSubject.onNext(true) }, {
ErrorHandler.logError(it) })
  }

  fun observeCategoryForCategoryId(categoryId: String): Maybe<Category> {
     return storeDao.observeCategoryForCategoryId(categoryId)
  }

  @SuppressLint("CheckResult")
  fun addStuffToCart(stuff: Stuff): Completable {
     return storeDao.observeCartStuffForStuffId(stuff.stuffId)
        .defaultIfEmpty(CartStuff(UUID.randomUUID().toString(), stuff.stuffId,
0))
        .flatMap {
           Maybe.fromCallable {
              it.count++
              storeDao.upsertCartStuff(it)
           }
        }.ignoreElement()
        .subscribeOn(Schedulers.io())
  }

  fun observeSubCharacteristicsForCharacteristicId(characteristicId: String):
Maybe<List<SubCharacteristic>> {
     return storeDao.observeSubCharacteristics(characteristicId)
  }
```

```
  fun observeSubCharacteristicsForStuffId(
      stuffId: String,
      pageSize: Int
  ):
LiveData<PagedList<StuffSubCharacteristicWithSubCharacteristicWithCharacteri
stic>> {
      return LivePagedListBuilder(
          storeDao.observeSubCharacteristicsForStuffIdLiveData(stuffId),
          buildDbConfig(pageSize)
      ).build()
  }

  fun observeStuffCharacteristicsLiveData(
      stuffId: String,
      pageSize: Int
  ): LiveData<PagedList<StuffCharacteristicWithCharacteristic>> {
      return LivePagedListBuilder(
          storeDao.observeStuffCharacteristicsLiveData(stuffId),
          buildDbConfig(pageSize)
      ).build()
  }
```

ДОДАТОК 5

Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android

Скриншоти програми

ІАЛЦ.467200.008 Д5

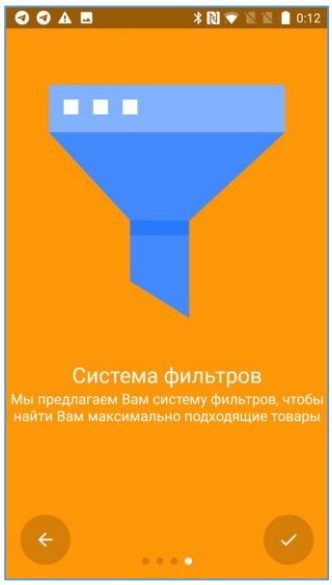
Аркушів 6

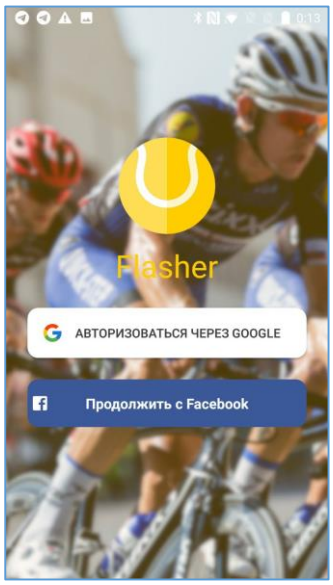
2019

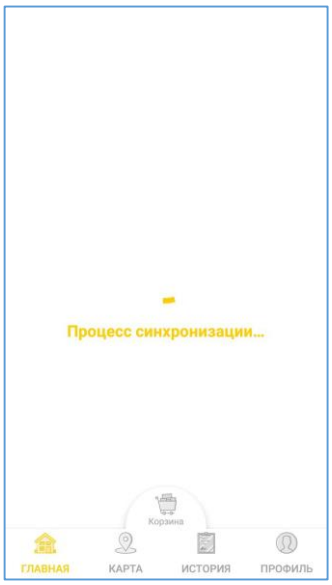

0:12
Только лучшие товары
Мы проверяем каждый магазин в нашем сервисе, чтобы Вы могли быть уверенны в том, что покупаете только оригинальные товары

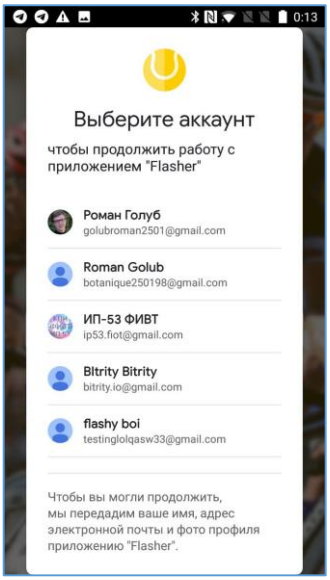

0:12
Предложение базирующееся на местоположении
Мы предлагаем Вам товары базирующееся на Вашем местоположении, чтобы Вы могли получать самые удобные для себя предложения

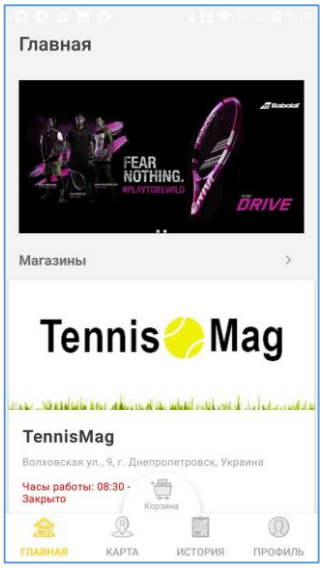

0:12
Агрегация магазинов
Мы собираем разные магазины, чтобы увеличить возможность Вам найти подходящий магазин для себя

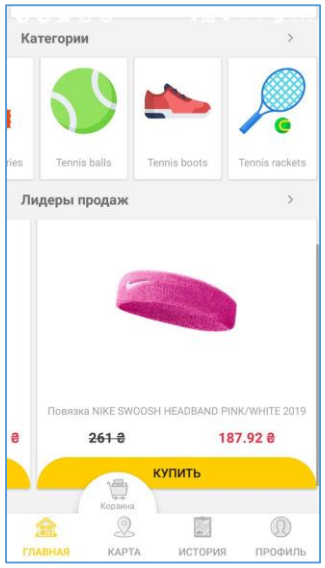

0:12
Система фильтров
Мы предлагаем Вам систему фильтров, чтобы найти Вам максимально подходящие товары

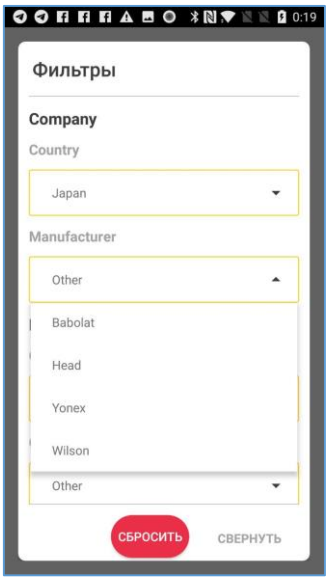

Flasher
АВТОРИЗОВАТЬСЯ ЧЕРЕЗ GOOGLE
Продолжить с Facebook

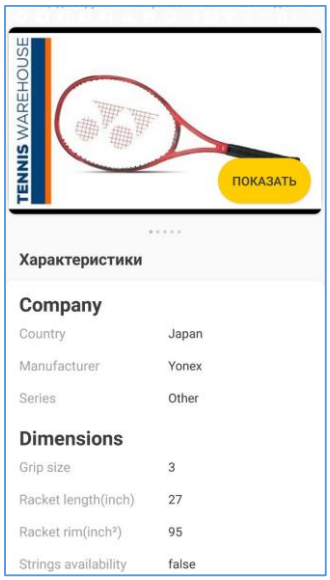

Процесс синхронизации...
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ

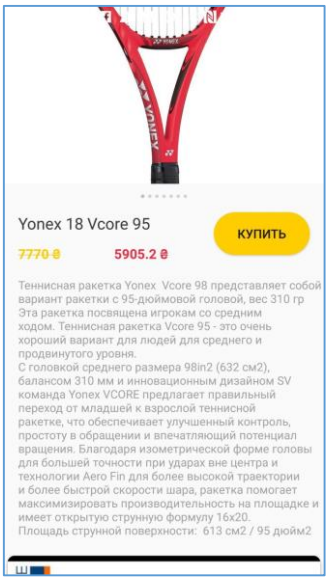

Выберите аккаунт
чтобы продолжить работу с приложением "Flasher"
Роман Голуб
golubroman2501@gmail.com
Roman Golub
botanique250198@gmail.com
ИП-53 ФИВТ
ip53.fiot@gmail.com
Bitrity Bitrity
bitrity.io@gmail.com
flashy boi
testinglolqasw33@gmail.com
Чтобы вы могли продолжить, мы передадим ваше имя, адрес электронной почты и фото профиля приложению "Flasher".

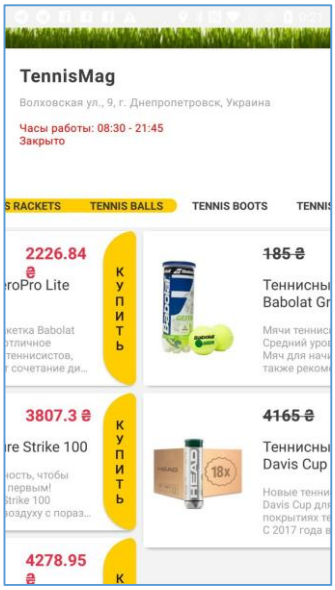

Главная
FEAR NOTHING.
#PLAYTOBEWILD
DRIVE
Магазины
TennisMag
TennisMag
Волховская ул., 9, г. Днепропетровск, Украина
Часы работы: 08:30 - Закрыто
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ


Категории
Tennis balls
Tennis boots
Tennis rackets
Лидеры продаж
Повязка NIKE SWOOSH HEADBAND PINK/WHITE 2019
261 ₴
187.92 ₴
КУПИТЬ
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ


Фильтры
Company
Country
Japan
Manufacturer
Other
Babolat
Head
Yonex
Wilson
Other
СБРОСИТЬ
СВЕРНУТЬ

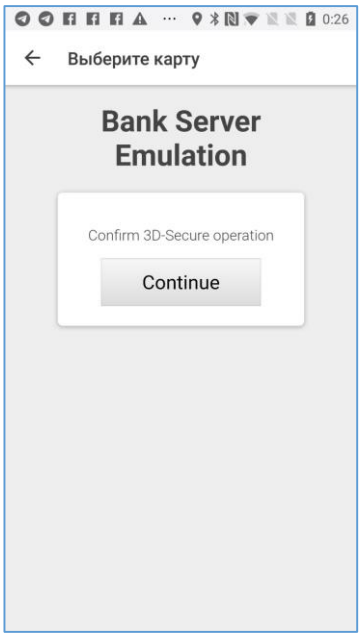

TENNIS WAREHOUSE
ПОКАЗАТЬ
Характеристики
Company
Country Japan
Manufacturer Yonex
Series Other
Dimensions
Grip size 3
Racket length(inch) 27
Racket rim(inch²) 95
Strings availability false


Yonex 18 Vcore 95
КУПИТЬ
7770 ₴
5905.2 ₴
Теннисная ракетка Yonex Vcore 98 представляет собой вариант ракетки с 95-дюймовой головой, вес 310 гр
Эта ракетка посвящена игрокам со средним ходом. Теннисная ракетка Vcore 95 - это очень хороший вариант для людей для среднего и продвинутого уровня.
С головкой среднего размера 98in2 (632 см2), балансом 310 мм и инновационным дизайном SV команда Yonex VCORE предлагает правильный переход от младшей к взрослой теннисной ракетке, что обеспечивает улучшенный контроль, простоту в обращении и впечатляющий потенциал вращения. Благодаря изометрической форме головы для большей точности при ударах вне центра и технологии Aero Fin для более высокой траектории и более быстрой скорости шара, ракетка помогает максимизировать производительность на площадке и имеет открытую струнную формулу 16x20.
Площадь струнной поверхности: 613 см2 / 95 дюйм2

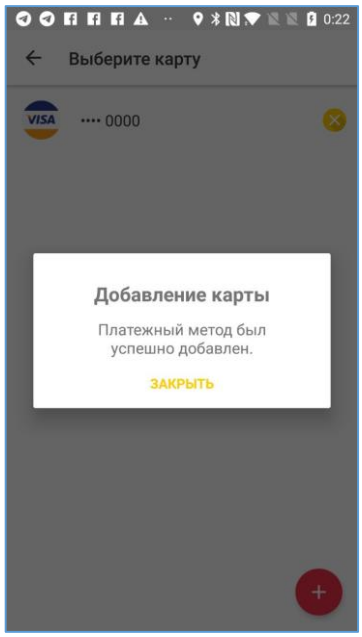

TennisMag
Волховская ул., 9, г. Днепропетровск, Украина
Часы работы: 08:30 - 21:45
Закрыто
RACKETS
TENNIS BALLS
TENNIS BOOTS
2226.84 ₴
КУПИТЬ
185 ₴
3807.3 ₴
КУПИТЬ
4165 ₴
4278.95 ₴

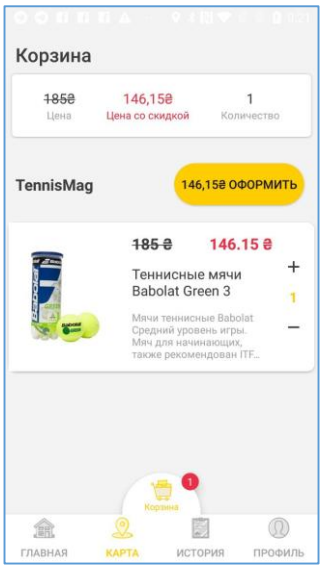

TennisMag
TennisMag
Волховская ул., 9, г. Днепропетровск, Украина
Часы работы: 08:30 - 21:45
Закрыто
TENNIS RACKETS
TENNIS BALLS
TENNIS BOOTS
6748 ₴
2226.84 ₴
Babolat AeroPro Lite pink
Теннисная ракетка Babolat AeroPro Lite -отличное решение для теннисистов, которые ищут сочетание ди...
КУПИТЬ
7770 ₴
3807.3 ₴

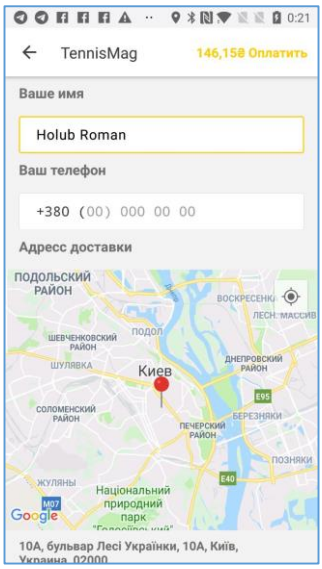

0:21
Babolat
GREEN
Slazenger £10 / $15


0:26
Выберите карту
Bank Server Emulation
Confirm 3D-Secure operation
Continue


Заказы
TennisMag
146.15 ₴
Цена
1
Количество
09.01.2019
146.15 ₴
185 ₴
Теннисные мячи Babolat Green 3 мяч
1 шт.
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ

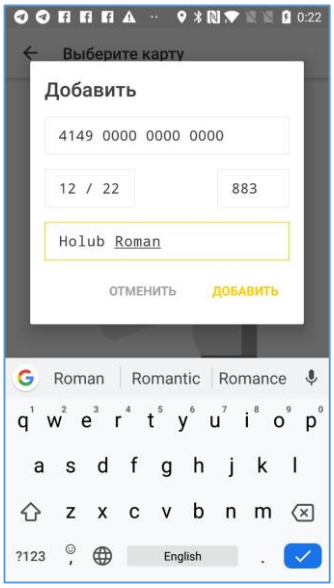

0:22
Выберите карту
VISA
···· 0000
Добавление карты
Платежный метод был успешно добавлен.
ЗАКРЫТЬ

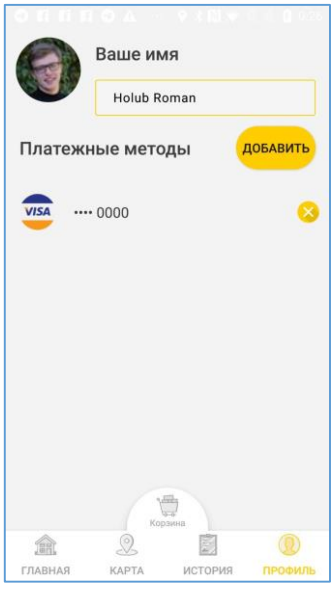

Корзина
185₴
Цена
146,15₴
Цена со скидкой
1
Количество
TennisMag
146,15₴ ОФОРМИТЬ
185 ₴
146.15 ₴
Теннисные мячи Babolat Green 3
Мячи теннисные Babolat Средний уровень игры. Мяч для начинающих, также рекомендован ITF...
1
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ


TennisMag
146,15₴ Оплатить
Ваше имя
Holub Roman
Ваш телефон
+380 (00) 000 00 00
Адресс доставки
Киев
10А, бульвар Лесі Українки, 10А, Київ, Украина, 02000

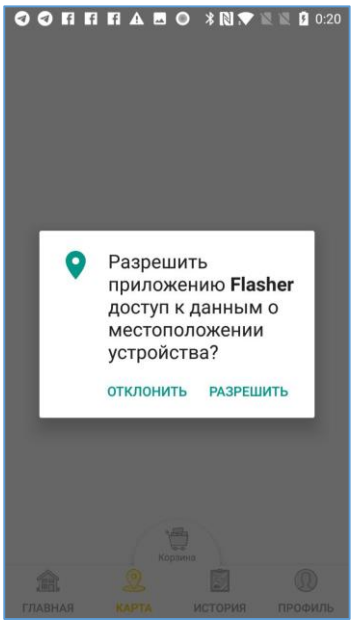

Дитячий будинок № 3
TennisMag
TennisMag
382,6 км
Волховская ул., 9, г. Днепропетровск, Украина
Часы работы: 08:30 -
Закрыто
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ

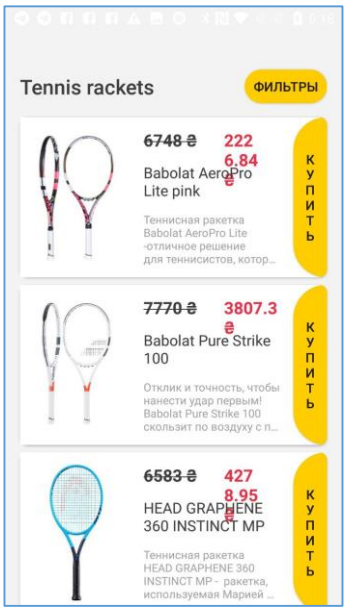

Выберите карту
К сожалению, Ваш список карт пуст.

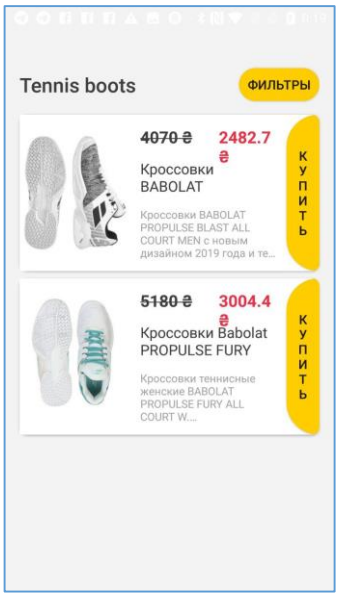

Выберите карту
Добавить
4149 0000 0000 0000
12 / 22
883
Holub Roman
ОТМЕНИТЬ
ДОБАВИТЬ
Roman
Romantic
Romance
English


Ваше имя
Holub Roman
Платежные методы
ДОБАВИТЬ
VISA
···· 0000
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ

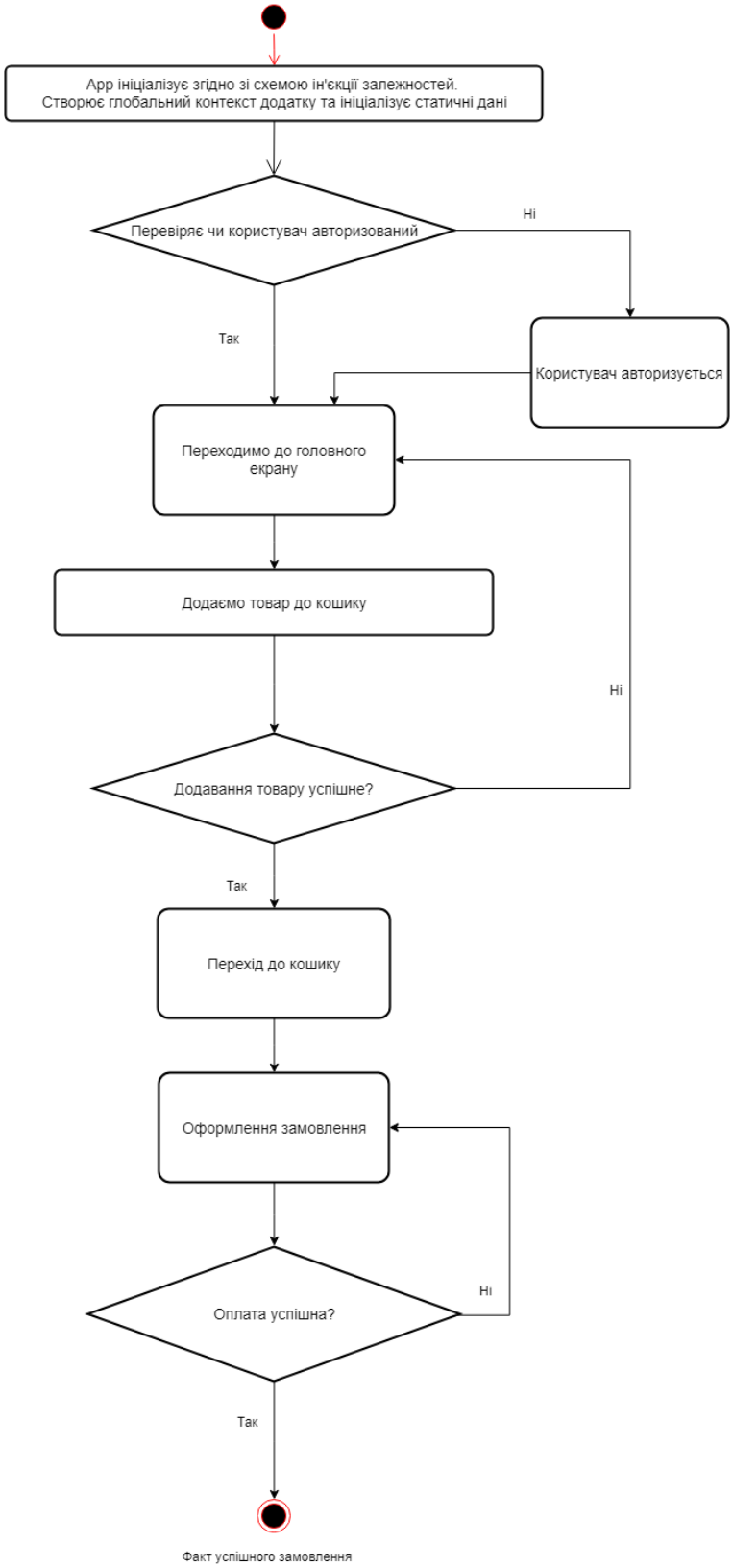

Review


0:20
Разрешить приложению **Flasher** доступ к данным о местоположении устройства?
ОТКЛОНИТЬ
РАЗРЕШИТЬ
Корзина
ГЛАВНАЯ
КАРТА
ИСТОРИЯ
ПРОФИЛЬ


Tennis rackets
ФИЛЬТРЫ
6748 ₴
2226.84 ₴
Babolat AeroPro Lite pink
Теннисная ракетка Babolat AeroPro Lite -отличное решение для теннисистов, котор...
КУПИТЬ
7770 ₴
3807.3 ₴
Babolat Pure Strike 100
Отклик и точность, чтобы нанести удар первым! Babolat Pure Strike 100 скользит по воздуху с п...
КУПИТЬ
6583 ₴
4278.95 ₴
HEAD GRAPHENE 360 INSTINCT MP
Теннисная ракетка HEAD GRAPHENE 360 INSTINCT MP - ракетка, используемая Марией ...
КУПИТЬ


Tennis boots
ФИЛЬТРЫ
4070 ₴
2482.7 ₴
Кроссовки BABOLAT
Кроссовки BABOLAT PROPULSE BLAST ALL COURT MEN с новым дизайном 2019 года и те...
КУПИТЬ
5180 ₴
3004.4 ₴
Кроссовки Babolat PROPULSE FURY
Кроссовки теннисные женские BABOLAT PROPULSE FURY ALL COURT W....
КУПИТЬ


Tennis accessories
ФИЛЬТРЫ
261 ₴
187.92 ₴
Повязка NIKE SWOOSH
Повязка NIKE SWOOSH HEADBAND - созданная из современного высококачественного м...
КУПИТЬ



					ІАЛЦ.467200.009 Д6	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Голуб Р.О.			Система придбання товарів з урахуванням місцезнаходження користувача для мобільної платформи Android Алгоритм програми	Літ. / Маса / Масштаб
Перевірив		Алещенко О.В.				Аркуш 1 / Аркушів 1
Н. Контр.		Сімоненко В.П.				НТУУ «КПІ», ФІОТ, ІП-53
Затв.		Стіренко С.Г.				