МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота магістра

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ СИСТЕМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ НА БАЗІ ОС ANDROID

Здобувач освіти гр. ІН.м-01н Т.Г. Харибегашвілі

Науковий керівник,
кандидат технічних наук, доцент,
доцент кафедри комп'ютерних наук І.В. Шелехов

Завідувач кафедри
доктор технічних наук, професор А.С. Довбиш

Суми 2022

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *122 «Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
" _____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Харибегашвілі Т.Г.
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ СИСТЕМ ЕЛЕКТРОННОЇ КОМЕРЦІЇ НА БАЗІ ОС ANDROID*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Інформаційний огляд 2) Вибір програмних засобів 3) Практична реалізація*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми та постановка задачі*		
2.	*Вибір алгоритму*		
3.	*Інформаційне та програмне забезпечення системи*		
4.	*Оформлення пояснювальної записки до дипломної роботи*		

Здобувач вищої освіти _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 68 стор., 15 рис., 5 додатків, 40 джерел.

Об’єкт дослідження — процес проектування систем електронної комерції на базі ОС Android.

Мета роботи — розробка інформаційної технології проектування систем електронної комерції на базі ОС Android..

Методи дослідження — методи моделювання процесів інформаційних систем, методи проєктування інформаційних систем, методи нормалізації структури баз даних .

Результати — розроблено інформаційну технологію проєктування мобільних додатків електронної комерції. При цьому запропоновано ряд методів для проєктування основних складовиз додатків такого типу: інтерфейсу, бізнес-логіки, бази даних тощо. Використано розробленй технологію для проєктування та реалізації мобільного додатку для реалізації комп’ютерної техніки.

МОБІЛЬНИЙ ДОДАТОК, СИСТЕМА УПРАВЛІННЯ ВМІСТОМ,
ІНФОРМАЦІЙНА МОДЕЛЬ, БАЗА ДАНИХ

ЗМІСТ

ВСТУП

Важко представити на сьогоднішній день людину, яка не користується смартфоном, кожен день ми дзвонимо рідним та друзям, дивимося новини про те що відбувається в світі, шукаємо певну інформацію, вивчаємо іноземні мови, спілкуемося через різні чати та месенджери, переглядаємо медіа-контент, робимо інтернет купівлі, та багато чого іншого і все це ми можемо робити на пристрою, який легко помістится у майже будь яку кишеню.

Чим далі розвивається інтернет мережа, тим простіше стає користуватися сайтами, онлайн сервісами, адже всі вони адаптуються під розміри різних гаджетів, тому у тебе не виникає труднощів, у будь який момент, навіть якщо ти не вдома, зайти з мобільного пристрою на сайт та наприклад, зробити інтернет купівлю. Але щоб зайти на сайт потрібен браузер, та потрібно знати адресу сайту, який ти хочеш відвідати, або шукати його у пошуковій системі, не зовсім зручно, так?

Вирішення цієї проблеми для збільшення ефективної продуктивності компанії може стати впровадження інтернет-технологій у систему продажів товарів чи послуг, адже створення мобільного додатку є невід'ємним атрибутом успішного функціонування електронної комерції. Купівля товарів або замовлення послуг через телефон наразі витісняє шопінг по офлайновим магазинам, оскільки додаток магазину дозволяє кожному бажаючому оперативно ознайомитися з асортиментом продукції / послуги та зробити замовлення в один клік.

Метою роботи є створення мобільного додатку електронної комерції для продажу різноманітної електроніки, завдяки якому у користувача буде можливість переглядати каталог товарів, дізнаватися про нові пропозиції, залишати свої відгуки та питання щодо роботи додатку, мати можливість замовити товар та оплатити зручним для них способом. Щоб виконати поставлені задачі необхідно реалізувати наступні кроки: провести інформаційний огляд, обрати методи та засоби вирішення поставленої задачі та виконати програмну реалізацію мобільного додатку електронної комерції.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області

Додаток єлектронної комерції являє собою набір сторінок, яких також по іншому називають активіті, які пов’язані між собою певними шаблонами, та які ідентифікується загальним ідентифікатором і розміщені на сервері. Яскравими прикладами веб-додатків є такі додатки як: rozetka, comfy, moyo, allo, citrus та інші.

Варто зауважити, що всі загальнодоступні мобільні та веб додатки разом складають всесвітню павутину. Існують також приватні додатки, до яких можна отримати доступ тільки в приватній мережі, наприклад, внутрішній сервер компанії для її співробітників. Мобільні аплікації зазвичай присвячені певній темі або цілі, наприклад новинам, освіті, комерції, розвагам або соціальним мережам. Крім того, користувачі можуть отримувати доступ к мобільним додаткам з різних пристроїв, включаючи настільні комп’ютери та ноутбуки з використанням емуляторів, планшети і смартфони. Програмне забезпечення, що працює і використовується на цих пристроях, називається операційною системою [8].

Як свідчать дослідження, Android було засновано у Каліфорнії, у жовтні 2003 року Енді Рубіном, Річардом Майнером (співзасновник WildfireCommunicati ons, Inc.), Ніком Сірсом (колишній працівник компанії мобільного зв’язку TMobil e) і Крісом Уайтом (очолював дизайнерське конструювання інтерфейсуWebTV) для розробки, за словами Рубіна «більш розумних мобільних пристроїв, які краще знають про місце перебування власника і його вподобання»[1].

На сьогоднішній день, мобільні додатки можуть використовуватися по-різному: можуть бути персональними, корпоративними, урядовими. Застосунки можуть бути роботою фізичної особи, бізнесу або іншої організації і зазвичай призначені для конкретної тема або мети [15].

Деякі додатки вимагають реєстрації користувача або підписки для доступу до контенту. Приклади підписки застосунків включають в себе безліч бізнес - додатків, новинних додатків, наукових журнальних додатків, гральних ігор,

додатків загального доступу до файлів, дошки для повідомлень, веб-пошту, соціальні мережі, і т.д. Представимо вигляд класичного додатка електронної комерції на Рис.1.1.

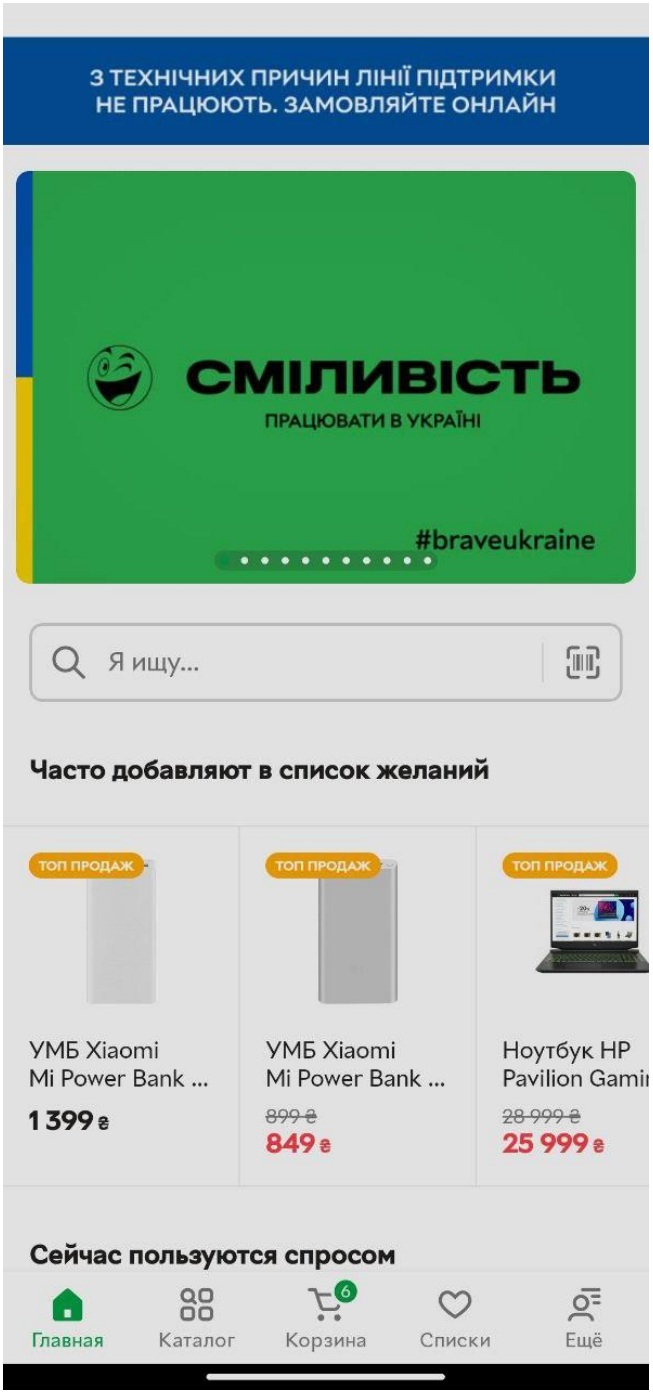

Рисунок 1.1 – Зовнішній вигляд класичного мобільного додатку електронної комерції

1.2 Огляд проблемної області

В залежності від призначення, додатки поділяються на різновиди. Розглянемо детальніше основні групи сучасних додатків[29, с. 129]:

1. Веб-додатки, сайти.

Веб-додаток - це веб-сторінки, що зберігаються на сервері в форматі, який відправляється до клієнтського веб-браузеру. В основному він написаний на мові гіпертекстової розмітки (HTML); Каскадні таблиці стилів (CSS) використовуються для управління зовнішнім виглядом крім базового HTML. Зображення зазвичай використовуються для створення бажаного зовнішнього вигляду і як частина основного контенту. Аудіо або відео також можуть вважатися контентом, якщо вони відтворюються автоматично або зазвичай не є інтерактивними[3]. Цей тип додатку зазвичай відображає одну і ту ж інформацію для всіх відвідувачів. Подібно роздачі друкованої брошури клієнтам або клієнтам, веб-додаток зазвичай надає послідовну стандартну інформацію протягом тривалого періоду часу[4]. Хоча власник веб-додатку може періодично робити оновлення, це ручний процес редагування тексту, фотографій та іншого контенту, який може зажадати базових навичок дизайну і програмного забезпечення. Прості форми або маркетингові приклади веб-додаток, наприклад класичний веб-додаток, п'ятисторінковуий веб-додаток або веб-додаток з брошурою часто є статичними веб-додатками, оскільки вони надають користувачеві заздалегідь задану статичну інформацію. Це може включати інформацію про компанію та її продукти та послуги у вигляді тексту, фотографій, анімації, аудіо / відео і меню навігації[9].

Статичні веб-додатки можуть як і раніше використовувати серверні включення (SSI) для зручності редагування, наприклад, для спільного використання загальної рядки меню на багатьох сторінках[11].

2. Гібридні додатки.

Гібридні додатки - це додатки, які часто всього розроблюються з використанням веб технологій та нативних інструментах розробки мобільніх додатків . Сторінки на гібридних додатках генеруються «на льоту» комп'ютерним кодом, який виробляє HTML (CSS відповідають за зовнішній вигляд і, таким чином, є статичними файлами).

Існує широкий спектр програмних систем, таких як CGI, JavaServlets і JavaServerPages (JSP), ActiveServerPages і ColdFusion (CFML), які доступні для створення гібридних додатків. Для мов програмування загального призначення, таких як Java, доступні різні фреймворки веб-додатків і системи веб-шаблонов. React, Python і Kotlin для прискорення і спрощення створення складних гібридних додатків.

Додаток може відображати поточний стан діалогу між користувачами, відстежувати ситуацію, що змінюється або надавати інформацію, будь-яким чином персоналізовану відповідно до вимог окремого користувача. Наприклад, коли запитується перша сторінка новинного додатку, код, що працює на веб-сервері, може комбінувати збережені фрагменти HTML з новинами, отриманими з бази даних або іншого додатку через RSS, щоб створити сторінку, яка містить найостаннішу інформацію[21].

Гібридні додатки можуть бути інтерактивними, використовуючи HTML-форми, зберігаючи і зчитуючи файли cookie створивши серію сторінок, що відображають попередню історію кліків. Інший приклад відображення контенту - це коли роздрібний веб-додаток з базою даних медіа-продуктів дозволяє користувачеві вводити пошуковий запит, наприклад, за ключовим словом Beatles.

У відповідь вміст сторінки додатку спонтанно змінить свій колишній вигляд, а потім відобразить список продуктів Beatles, таких як компакт-диски, DVD-диски і книги. HTML використовує код Java та JavaScript, щоб вказати веб-

додатку, як інтерактивно змінювати вміст сторінки. Один із способів змоделювати певний тип гібридного додатка, уникаючи втрати продуктивності при запуску динамічного механізму для кожного користувача або кожного з'єднання, - це періодично автоматично відновлювати велику серію статичних сторінок[13].

3. Нативні додатки

Нативні додатки – це додатки, які розроблюються виключно з використанням нативних мов програмування, таких як Java та Kotlin, якщо мова йде на розробку під Andoid, та мова програмування Swift для розробки додатків на девайси з операційною системою IOS.

Це найвимогливіший до ресурсів вид додатків. В той самий час, дозволяє скористатися всіма можливостями операційної системи. Це найбільш функціональний і продуктивний вид мобільних додатків

Крім того, всі додатки можна розділити на дві великі підкатегорії[16]:

- статичні,

- інтерактивні.

Інтерактивні додатки є частиною спільноти Web 2.0 і дозволяють взаємодіяти між власником застосунку і відвідувачами або користувачами додатку. Статичні додатки служать або збирають інформацію, але не дозволяють безпосередньо взаємодіяти з аудиторією або користувачами[30]. Деякі веб-додатки носять інформаційний характер, створені ентузіастами або призначені для особистого використання або розваги. Багато додатків дійсно прагнуть заробляти гроші, використовуючи одну або кілька бізнес-моделей, в тому числі:

- Розміщення цікавого контенту і продаж контекстної реклами або через прямі продажі, або через рекламну мережу.

- Електронну комерцію: товари або послуги купуються безпосередньо через додаток.

- Рекламні продукти або послуги, доступні в звичайному бізнесі.

- Freemium: базовий контент доступний безкоштовно, але преміум-контент вимагає оплати (наприклад, веб-додаток WordPress, це платформа з відкритим вихідним кодом для створення блогу або додатку).

Водночас існує безліч різновидів додатків, кожен з яких спеціалізується на певному типі контенту або використанні в певній галузі, і їх можна довільно класифікувати будь-якою кількістю способів. Ось кілька таких класифікацій[19]:

- Додатки-партнери.

Це застосунок з невеликою кількістю сторінок, метою якого є продаж продукту третьої сторони. Продавець отримує комісію за сприяння продажу.

- Додатки - «партнерські агенції».

Включений портал, який відображає не тільки власну CMS, але і синдикований контент від інших постачальників контенту за узгоджену плату. Зазвичай існує три рівня відносин. (Наприклад, CommissionJunction, eBay,Yahoo!)

- Додатки-архіви.

Використовуються для збереження цінного електронного контенту, що знаходиться під загрозою зникнення (WebArchive)[31].

Додатки, створені спеціально для атаки смарфонів і планшетів, відвідувачів при їх першому відвідуванні застосунку шляхом завантаження файлу (зазвичай троянського файлу). Ці застосунки покладаються на нічого не підозрюють користувачів з поганмим антивірусним захистом або взагалі його відсутністю[17].

- Додатки - блоги.

Застосунки, які зазвичай використовуються для публікації онлайн-щоденників, які можуть включати дискусійні форуми. Багато блогери використовують блоги як редакційні розділи газети, щоб висловлювати свої ідеї з приводу, починаючи від політики і релігії, до відеоігор і батьків, а також по всьому, що між ними. Деякі блогери є професійними блогерами, і їм платять за ведення блогу на певну тему, і їх зазвичай можна знайти на новинних ресурсах (наприклад, WordPress)[14].

- Додатки по розробці бренду.

Застосунки, що створені з метою враження про бренд в Інтернеті. Ці платформи зазвичай нічого не продають, а зосереджені на створенні бренду. Додатки по створенню бренду найбільш поширені для недорогих, масових і швидко просуваються товарів народного споживання (FMCG)[18].

- Додатки відомих діячів.

Застосунки, інформація на яких обертається навколо знаменитостей або громадських діячів. Ці додатки можуть бути офіційними (схваленими знаменитістю) або створеними фанатами (керованими фанатом або шанувальниками знаменитості без явного схвалення) (наприклад, jimcarrey)[20]

- Додатки порівняння покупок.

Додатки з пошуковою системою, яку покупці використовують для фільтрації і порівняння товарів за ціною, характеристикам, відгуками і іншими критеріями (наприклад, ЕК, Hotline, MArena).

- Додатки краундфандингу.

Платформа для фінансування проектів шляхом попередньої покупки продуктів або шляхом звернення до членів аудиторії з проханням зробити пожертвування (наприклад, Kickstarter).

- Додатки для пожертвувань при натисканні.

Мобільні додатки, які дозволяють відвідувачеві робити пожертвування на благодійність, просто натиснувши кнопку або правильно відповівши на питання. Рекламодавець зазвичай жертвує на благодійність за кожну правильну відповідь (наприклад Freerice).

- Додатки співтовариства.

Додатки, на якому люди зі схожими інтересами спілкуються один з одним, зазвичай в чатах або месенджерах (наприклад, Facebook, Telegram, Viber, Instagram).

- Контент-додатки.

Додатки, бізнес яких - створення і поширення оригінального контенту (wikiHow, Youtube, Pinterest).

- Додатки оголошень.

Додатки, які публікують тематичні оголошення (olx, izi).

- Корпоративні додатки.

Використовуються для надання довідкової інформації пробізнес, організації або послугу.

- Додатки знайомств.

Додатки, на яких користувачі можуть знайти інших самотніх людей, що шукають довгострокових відносин, побачень, коротких зустрічей або дружби. Багато з них безкоштовні, але в них зазвичай є платні фунції. (наприклад, Badoo, Tabor, Jaumo).

- Додатки е-комерції.

Додатки, що пропонують товари і послуги для онлайн-продажу і дозволяють здійснювати онлайн-транзакції для таких продажів (наприклад, Amazon, Aliexpress, Rozetka, Allo, Citrus).

- Додатки форумів.

Застосунки, на яких люди можуть вести бесіди у вигляді опублікованих повідомлень (наприклад, 4pda, Xda).

- Додатки галерей.

Застосунки, розробленіспеціально для використання в якості галереї; це може бути художня галерея або фотогалерея комерційного або некомерційного характеру.

- Державні додатки.

Додатки, створені місцевим урядом, державою, департаментом або національним урядом країни. Зазвичай ці додатки також керують сервісами, які призначені для перегляду, зміни даних особи, керування своїми документами та отпримання різних державних послуг (ДІЯ).

- Додатки, що критикують.

Додаток, присвячений критиці людини, місця, корпорації, уряду або установи.

- Ігрові додатки.
- Додатки азартних ігор.
- Гумористичні додатки, які покликані розважати публіку.
- Інформаційні додатки.

Більшість веб-додатків в тій чи іншій мірі потрапляють в цю категорію. Вони не обов’язково мають на комерційні цілі.

- Додатки для медіа-обміну.

Застосунки, що дозволяють користувачам завантажувати і переглядати мультимедіа, наприклад зображення, музику і відео (наприклад, YouTube, Twich, Tiktok).

- Додатки мікроблогів.

Коротка і проста форма ведення блогу. Мікроблоги обмежені певною кількістю символів і працюють аналогічно оновленню статусу в Facebook (наприклад, Twitter).

- Фішінгові додатки.

Застосунки, що створені для обманного отримання конфіденційної інформації, такої як паролі та дані кредитної картки, шляхом маскування під надійне особа або компанію (наприклад, Адміністрацію соціального забезпечення, PayPal, банк) в електронному спілкуванні.

- Додатки для обміну фотографіями.

Застосунок, створений для обміну цифровими фотографіями з онлайн-спільнотою (Flickr, Instagram, Imgur).

- Торрент-додатки, P2P.

Додатки, індексуючі торрент-файли. Цей тип відрізняється від Bittorrent-клієнта, який зазвичай є автономне програмне забезпечення.

- Політичні додатки.

Додатки, на яких люди можуть висловлювати політичні погляди, надавати політичний гумор, агітувати на виборах або надавати інформацію про певний кандидата, політичної партії або ідеології.

- Додатки-«вітрини».

Портали, які використовуються окремими особами і організаціями для демонстрації цікавих або цінних речей.

- Додатки соціальних мереж.

Додатки, на яких користувачі можуть спілкуватися один з одним і обмінюватися мультимедійними даними, такими як зображення, відео, музика, блоги з іншими користувачами. Сюди можуть входити гри і веб-додатки (YouTube, Facebook, Instagram, Pinterest, LinkedIn) [24].

- Додатки - «веб-портали».

Додатки, що забезпечують відправну точку або шлюз до інших ресурсів в Інтернеті або інтрамережі (наприклад, Google ,DuckDuckGo ,Bing, Yahoo!).

Деякі застосунки можуть бути включені в одну або кілька з цих категорій. Наприклад, бізнес-додаток може просувати продукти компанії, але також може містити інформаційні документи, такі як офіційні документи. Є також численні підкатегорії до перерахованих вище[34].

Отже, додатки можуть містити підрозділи, орієнтовані цілком на ту чи іншу аудиторію. Кожен застосунок створюється з якоюсь певною метою. Перш за все, вони потрібні для передачі певної інформації користувачу[32].

Варто акцентувати, що цтворення мобільного додатка – досить великий проект. Навіть якщо проект невеликих розмірів він може зіграти істотну роль у розвитку бізнесу, а тому навіть для додатків з однією сторінкою(макетом) потрібен великий обсяг робіт і ретельне опрацювання всіх деталей. Коли ви наважуєтеся на створення великого проекту, головне – окреслити цілі. Якщо не зробите цього перед створенням застосунку, ваші уявлення про майбутню роботу так і залишаться вельми і вельми туманними. Як і весь проект в підсумку. Не знаючи власних потреб, можна створити додаток, на якому або занадто багато, або занадто мало інформації. Є ще варіант: інформації буде достатньо, але трохи або ж зовсім не по темі[31].

Постановка задач для ресурсу виллється в майбутньому в безліч переваг, серед яких основне – зручність роботи з ним (зрозуміла структура і легка навігація). Завчасне планування також потім дозволить аналізувати, чи окупаються ваші інвестиції в проект і наскільки швидко.

Отже, постановка цілей не займає багато часу. Не пропустіть цей крок. Це важливо.

Для створення додатку для бізнесу, у нього можуть бути такі основні цілі:

- продавати (для e-commerce);
- залучати клієнтів;
- презентувати продукт або лінійку товарів;
- розповісти про послугу;
- збирати ліди (контактні дані);
- розповісти про вашу компанію;
- вибудувати довгострокові відносини з клієнтами;
- інформувати цільову аудиторію.
- підвищити охоплення і впізнаваність бренду, використовуючи digital-канали.

Крім цього ресурс надає додаткові можливості власнику. А саме:

- формувати репутацію бренду;
- додаткові канали збору даних про ЦА;
- оцифровка і аналітика бізнес-процесів і маркетингу.

Таким чином, один додаток може підходити відразу для декількох цілей, існують також складні індивідуальні проекти, які виконують кожну з перерахованих завдань. Дуже важливо правильно визначити, який саме результат потрібен і на основі цієї інформації формувати інші етапи роботи над створенням застосунку.

1.2. Огляд подібних рішень

В процесі проектування і розробки додатку є безліч етапів. Від збору вихідної інформації до створення додатку і, нарешті, до обслуговування для підтримки його в актуальному стані.

Перерахуємо стандартні етапи розробки додатку електронної комерції:

1. Збір інформації.

2. Планування.

3. Дизайн.

4. Розвиток.

5. Тестування і доставка обслуговування.

Перший крок у створенні додатку - це збір інформації. При створенні зовнішнього вигляду мобільного додатку необхідно враховувати безліч чинників. Цей перший крок насправді найважливіший, оскільки він вимагає твердого розуміння компанії, для якої він створений. Це вимагає хорошого розуміння бізнес-цілей, а також того, як можна використовувати Інтернет для досягнення цих цілей.

Слід враховувати наступні моменти: призначення (мета додатку). Ви хочете надати інформацію, просувати послугу чи продавати продукт. Дві найбільш поширені цілі - заробити гроші або поділитися інформацією[35, с. 55].

Цільова аудиторія. Чи є конкретна група людей, яка допоможе вам досягти ваших цілей? Корисно уявити собі «ідеального» людини, якого ви хочете відвідати на свій додаток. Враховуйте їх вік, стать або інтереси - пізніше це допоможе визначити кращий стиль дизайну для вашого додатка[28].

Контент.

Яку інформацію цільова аудиторія буде шукати на вашому застосунку.

Наступним етапом є розробка плану для майбутнього додатку. Це місце, де розробляється мапа застосунку. Мапа застосунку являє собою список всіх основних тематичних областей додатку, а також підтем, якщо є. Це служить керівництвом щодо того, який буде контент, і необхідний для розробки послідовної, простий для розуміння системи навігації[27].

При розробці додатку слід мати на увазі кінцевого користувача додатку, також відомого як ваш клієнт. Зрештою, це люди, які будуть дізнаватися про ваші послуги або купувати ваш продукт. Хороший користувальницький інтерфейс створює зручний для навігації додатку і є основою для цього.

На етапі планування ваш веб-дизайнер також допоможе вам вирішити, які технології слід впровадити. При плануванні вашого додатку обговорюються такі елементи, як якась CMS (система управління контентом), наприклад WordPress, будуть потрібні які-небудь контактні форми. Спираючись на інформацію, зібрану до цього моменту, слід визначити зовнішній вигляд вашого додатку.

Цільова аудиторія - один з ключових чинників, що беруться до уваги. Наприклад, додаток, призначений для підлітків, буде виглядати інакше, ніж додаток, призначений для фінансової установи. На етапі проектування також важливо включити такі елементи, як логотип компанії або кольору, щоб посилити індивідуальність вашої компанії [38, с. 88].

Варто створити один або кілька прототипів для додатку. Зазвичай це зображення уформаті .jpg того, як буде виглядати остаточний дизайн.

Стадія розробки - це момент, коли створюється сам додаток. На цьому етапі розробник візьме все окремі графічні елементи з прототипу і використовує їх для створення реального функціонального застосунка..Зазвичай для цього спочатку створюється домашня сторінка, а потім «оболонка» для внутрішніх сторінок. Оболонка служить шаблоном для сторінок вмісту вашого додатка[2].

Такі елементи, як CMS (система управління контентом), такі як WordPress, інтерактивні контактні форми або візки для покупок електронної комерції, також впроваджуються і стають функціональними на цьому етапі.Весь цей час розробник повинен продовжувати робити незавершений додаток доступним дляперегляду, щоб користувач міг запропонувати будь-які додаткові зміни або виправлення, які ви хотіли б внести[6].

З технічної точки зору успішний застосунок вимагає розуміння інтерфейсної розробки. Це включає в себе написання дійсного коду, який

відповідає поточним стандартам, максимізує функціональність, а також робить його доступним для максимально широкої аудиторії.

На етапі тестування займаються остаточними деталями і тестують майбутній додаток. Тестуванню підлягають такі речі, як повна функціональність форм або інших скриптів, а також останнє тестування на предмет проблем сумісності в останню хвилину (перегляд відмінностей між різними пристоями), гарантуючи, що ваш додаток оптимізований для правильного перегляду в самих різніх версіях операційної системи мобільних пристроїв[22].

1.3 Постановка задачі дослідження

Проведений аналітичний огляд довиодить актуальність практичної задачі розроблення інформаційної технології проєктування проектування систем електронної комерції на базі сичасних оперційних систем, в тому числі і для мобільних пристроїв. Для розв’язання поставленої задачі необхідно вконати ряд завдань:

1. Проаналізувати теоретичні засади дослідження інтернет-технологій;
2. Визначити основні етапи розробки системи електронної комерції на прикладі створення веб-додатку;
3. Дослідити алгоритм створення веб-додатку електронної комерції;
4. Спроектувати модель та описати вибір системи управління вмістом;
5. Розробити мобільний додаток та протестувати всі його розділи та сторінки.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Оцінка технологій для моделювання системи електронної комерції

Основні використовувані в даний час технології – це React, Java, Html і CSS (каскадні таблиці стилів). В рамках тестування розробник повинен переконатися, що весь код, написаний для вашого додатку, проходить перевірку. Дійсний код означає, що додаток відповідає поточним стандартам розробки - це корисно при перевірці таких проблем, як сумісність з різними версіями ос, як згадувалося вище.

Програма FTP (протокол передачі файлів) використовується для завантаження файлів додатку на ваш сервер. Після того, як облікові записи були налаштовані і ваш додаток завантажений на сервер, застосунок повинен пройти останню перевірку. [40].

Інші заключні деталі включають установку плагіна (для WordPress, керованих CMS, і SEO (пошукова оптимізація). SEO - це оптимізація додатку з такими елементами, як заголовок, опис і теги ключових слів, які допомагають додатку досягти більш високих позицій в рейтингу.

Існує безліч доступних плагінів WordPress, які додатково покращують функціональність WordPress за замовчуванням, багато з яких також безпосередньо пов'язані з поліпшенням вашого SEO.Однак розробка додатку не обов’язково завершена. Один із способів привернути постійних відвідувачів на ваш додаток - регулярно пропонувати новий контент або продукти. Більшість розробників будуть більш ніж щасливі продовжити спільну роботу над оновленням інформації в вашому додатку. Багато дизайнерів пропонують пакети обслуговування за зниженими цінами в залежності від того, як часто ви плануєте вносити зміни або доповнення в свому застосунку .

Якщо користувач прикладатиме більше зусиль і оновлюватиме власний контент, на його додатку може бути реалізована так звана CMS (система управління контентом), така як WordPress. Це те, що буде вирішуватися на етапі

планування. Із CMS розробник буде використовувати онлайн-програмне забезпечення для розробки додатку на основі бази даних.

Додаток, керований CMS, дає вам можливість самостійно редагувати області вмісту додатку. Вам надається доступ до внутрішньої адміністративної області, де ви можете використовувати онлайн-текстовий редактор (аналогічний міні-версії Microsoft Word). Таким чином можна редагувати існуючий контент, абододавати нові сторінки і контент.

Далі предметніше розглянемо деякі з популярних систем управління контентом, які використовуються в галузі електронної комерції [26].

- VirtueMart.

VirtueMart, раніше відомий як mambo-phpShop, вільне програмне забезпечення для створення Інтернет-магазину, створене для доповнення таких систем керування вмістом застосунку, як Mambo та Joomla. VirtueMart написаний на PHP і використовує базу даних MySQL. Найбільш підходить для додатків з низьким або середнім рівнем завантаження. VirtueMart захищений ліцензією GNU GPL. VirtueMart з'явився вперше у вигляді автономної програми phpShop для створення інтернет-магазину.

VirtueMartпідтримує необмежену кількість продукції і категорій, необмежену кількість валют для одного продукту, призначення продукції безлічі категорій, можливість продавати продукцію, призначену для скачування, а також надає можливість відключити функцію продажу і використовувати VirtueMart в режимі каталогу. VirtueMart дозволяє мати різні ціни для однієї продукції, засновані на кількості чи приналежності покупця до певної групи, і надає можливість використання різних платіжних систем.

У VirtueMart реалізована система знижок, купонів, а також великий вибір систем оплати та способів доставки.

При перенесенні VirtueMart з Mambo на Joomla розробники не приділили належної уваги відмінностям між Mambo та Joomla, в результаті VirtueMart гілок 1.0.x містить у собі багато старого процедурного коду і компоненти, вже існуючі в

Joomla, але з якихось причин не використовувані (шаблонізатор, з'єднання з базою.

- PrestaShop.

PrestaShop – веб-додаток для електронної комерції з відкритим кодом. Написаний на PHP, для написання шаблонів використовується Smarty, розрахований на MySQL. Цей двигун призначений для малого та середнього бізнесу та має більше 310 стандартних функцій для швидкого створення функціонального магазину. В 2010-2014рр. визнаний кращим веб-додатком для інтернет-магазинів, завоювавши нагороду Best Open-source Business Application.

Перекладений майже на 60 мов, спільнота нараховує більше 500.000 чоловік. Існує платна і безплатна підтримка. Штаб квартири компанії знаходяться в Парижі та Майямі. Команда налічує більше 100 чоловік та складається із розробників, дизайнерів та IT-спеціалістів в електронній комерції.

PrestaShop інколипорівнюютьз Magento та OpenCart. Поступається першому розміром спільноти та кількістю готових рішень, але при цьому легший в освоєнні. OpenCart як і PrestaShop легкий в освоєнні, та набирає популярність з кожним роком.

- Ubercart.

Ubercart є вільним програмним забезпеченням для відкриття торгових майданчиків, інтернет-магазинів, аукціонів, створюваних для доповнення такої системи керування вмістом, як CMFDrupal. Модуль Ubercart може бути встановлений на будь-якому сервері з підтримкою PHP і MySQL і поширюється по ліцензії GNU GPL.

Ubercart повністю інтегрований з Drupal'ом, що означає 100% сумісність з ним будь-якого застосунку, що використовує Drupal.

Модуль може використовуватися як у випадках продажу товарів і послуг, так і при закачування будь онлайнової продукції. Крім того, його можна застосовувати для найму та працевлаштування, створення значущих подій.

Головним же призначенням Ubercart є можливість відкривати необмежену кількість платіжних шлюзів різних електронних світових валют і банківських систем. Ubercart дозволяє проводити цілий ряд комплексних дій всередині CMFDrupal, а саме[38, с. 88]:

- додавати новий продукт,
- здійснювати мультипродаж товарів,
- робити знижки і видавати купони,
- обмінювати валюту,
- здійснювати будь-які внутрішні і міжнародні платежі,
- підтримувати партнерські програми,
- виплачувати комісії,
- видавати купони на ліцензовані товари та послуги,
- проводити комплексну класифікацію продукції підприємства.

Також як і Drupal, Ubercart може працювати на різних мовних платформах. Drupal - популярна вільна модульна система керування вмістом (СКВ) з відкритим сирцевим кодом, написана на мові програмування PHP.

Завдання систем керування вмістом - полегшити створення, наповнення та оновлення застосунку. Drupal може працювати на таких популярних системах як Windows, Mac OS X, Linux, власне, на будь-якій платформі, яка підтримує роботу веб-сервера Apache, Nginx, Lighttpd або Microsoft IIS; також потрібна наявність системи керування базами даних MySQL/MariaDB, PostgreSQL, SQLite чи інших комерційних БД.

Критики Drupal ставлять в докір розробникам слабке використання об'єктних можливостей PHP. API Drupal практично не використовує наявні в PHP можливостей ООП. Розробники аргументують це слабкою реалізацією ООП у мові (особливо до версії PHP 5). Об'єктна модель в Drupal присутня, але в дещо нетрадиційному для PHP вигляді.

До недоліків (але водночас і до переваг) Drupal можна віднести:

- відсутність зворотної сумісності API при досить високій динаміці розробки проекту.

Практично в кожному релізі відбуваються зміни API, коли поряд з додаванням нових функцій прибираються деякі старі або змінюються параметри виклику функцій. Це призводить до необхідності розробникам сторонніх модулів адаптувати їх для роботи з новими версіями Drupal. Проте зміни API і процедура адаптації модулів до нових версій описуються в документації для кожного релізу, а також завжди пропонується механізм автоматизованого апгрейда ядра системи на нову версію. Плюс даної схеми розробки -немає необхідності тягти з версії у версію програмний шар сумісності зі старими API, що полегшує поточний код системи.

У дистрибутив системи входить набір модулів, що дають наступні можливості[34]:

- збір інформаційних стрічок (RSS, RDF, Atom);
- ведення блогів, підшивань і форумів;
- створення форм для відправки повідомлень;
- локалізація системи;
- перейменування посилань (призначення посиланням зрозумілих і зручних псевдонімів);
- проведення опитувань;
- призначені для користувача профілі, що настроюються;
- пошук за змістом (за зміст вважається і повідомлення на форумах, і сторінки, і будь-які інші призначені елементи);
- ведення журналу статистики (відвідуваності);
- таксономія (впорядковування матеріалу за категоріями) - дуже «цінна» можливість;
- включення механізму авторегулятора контролю навантаження.

- Open Cart.

Open Cart - це система інтернет-торгівлі з відкритим кодом, яка розповсюджується за ліцензією GNU General Public License.

Open Cart може бути встановлено на будь який веб-сервер Apache з підтримкою PHP5 і вище та MySQL. Навколо OpenCart сформована велика спільнота (понад 46.000 учасників), завдяки якій створено понад 8.500 безкоштовних розширень у вигляді додаткових модулів.

Найвагомішими перевагами OpenCart над системами Magento, Virtuemart і osCommerce є сучасна MVC-архітектура, підвищена швидкість роботи, vQmod, багатофункціональна адміністративна панель управління контентом, та менше споживання серверних ресурсів.

OpenCart добре зарекомендував себе в комерційному секторі як надійна і недорога в обслуговуванні система електронної торгівлі, що має підтримку розрахунку всіх найвідоміших систем електронної оплати.

Основні можливості:

Технічні переваги:

- Підтримка PHP 5.x і MySQL 4.x, 5.x.
- Код відповідає основним принципам шаблону Model-View-Controller, який дозволяє проводити роботи різної складності незалежно одна від одної.
- Порівняно з конкурентами (Magento, Virtuemart, osCommerce) має кращі показники швидкості і несе менше навантаження на сервер.
- Підтримка багатьох сучасних браузерів.
- Вбудована багатомовність. Доступна українська мова.
- Необмежена кількість категорій і товарів.
- Підтримка шаблонів, модулів і доповнень.

Адміністрування/База:

- Підтримка одного і більше магазинів.
- Необмежена кількість продуктів і категорій.
- Підтримка фізичних і віртуальних (з можливістю завантаження) товарів.
- Легкість резервного копіювання і відновлення бази даних.

- Статистика товарів, замовлень і продажів.
- Багатомовність.
- Підтримка валют і курсів.

Клієнтська частина:

- Реєстрація покупців.
- Усі замовлення зберігаються в базі даних для ефективного пошуку історії покупок.
- Клієнти можуть переглядати історію і статус своїх замовлень.
- Тимчасовий кошик для гостей і постійний для зареєстрованих клієнтів.
- Швидкий і зручний інструмент пошуку.
- Підтримка SSL (Secure Sockets Layer).
- Зручна навігація.
- Клієнт може мати декілька адрес доставки в персональній адресній книзі.

Система оплати і доставки:

- Підтримка багатьох типів платежів (чеки, платіжні доручення).
- Підтримка багатьох платіжних систем за допомогою модулів (2CheckOut, PayPal, Authorize.Net, iPayment, RuPay, Webmoney, Приват24).
- Налаштування методів оплати для різних регіонів.
- Розрахунок доставки на базі ваги і ціни товару та зони доставки.
- Безліч модулів розрахунку доставки.
- Розрахунок податків з урахуванням регіону.

Варто звернути увагу:

- Шаблонізація на базі PHP.
- У цей час не оптимізована робота SEO.
- Швидкі темпи розвитку версій, що потребує частого поновлення ядра.

- osCommerce.

osCommerce («Open Source Commerce») –це движок інтернет-магазину. Він може бути встановлений на будь-якому веб-сервері з підтримкою PHP і MySQL. Це вільне програмне забезпечення з GNU General Public License.

Навколо osCommerce сформувалося величезне співтовариство (більш 140,000 учасників) , завдяки якому існує більше 4000 контрибуцій (різних модулів для osCommerce), що дозволяють змінювати і доповнювати функції магазину самим різним чином. По всьому світу функціонують десятки тисяч магазинів на базі osCommerce (14,063 магазина офіційно зареєстровані в каталозі).

Основні можливості:

- Сумісно з PHP 4.x, 5.x і MySQL 4.x, 5.x.
- Сумісність з усіма основними браузерами.
- Вбудована багатомовність, за замовчуванням встановлені англійський, німецький, іспанська мови. Доступні російська, українська і багато інших.
- Майстер інсталяція (майстер).
- Необмежене число розділів і товарів.

Administration/Base:

- Підтримує необмежену кількість продуктів і розділів категорій.
- Підтримка фізичних і віртуальних (завантажуються) товарів.
- Легкість резервного копіювання і відновлення даних.
- Статистика товарів і замовників.
- Багатомовна підтримка.
- Підтримка декількох валют.

Клієнтська частина:

- Реєстрація покупців.
- Всі замовлення зберігаються в базі даних для швидкого й ефективного пошуку (історія покупок для покупців).
- Клієнти можуть переглядати історію і статуси своїх замовлень.
- Тимчасова корзина для гостей і постійна для клієнтів.
- Швидкий і дружній інтерфейс пошуку.

- Безпека з підтримкою SSL (Secure Sockets Layer).
- Клієнт може мати кілька адрес доставки в своїй адресній книзі.

Система оплати і доставки:

- Підтримка численних типів платежів (чеки, платіжні доручення).
- Підтримка численних платіжних систем (модулям) (2CheckOut, PayPal, Authorize.Net, iPayment, RuPay, Webmoney).
- Налаштування методів оплати для різних областей.
- Розрахунок доставки наоснові ваги та ціни товару, зони доставки.
- Розрахунок податків.

На жаль, osCommerce не володіє системою шаблонів (до версії 3 Alpha 5) для настройки дизайну, але є ряд модулів які вирішують дану задачу. Наприклад, STS і BTS. Також система шаблонізатор є у багатьох клонів osCommerce.

- Magento.

«Magento» є професійною системою для управління інтернет-магазинами. За даними Alexa, Magento - це найпопулярніша система управління інтернет-магазинами в світі на лютий 2019. Ще у 2012р. компанія Magento Inc. була придбана компанією eBay Inc.

Зовнішній вигляд вітрини в Magento визначається так званими темами:

- набір шаблонів (templates) відображення видимих на екрані блоків;
- набір правил, що визначають, які блоки в якому місці конкретної сторінки відображати, і об'єднаних у файли логічної розмітки (layouts);
- набір ресурсних файлів (скінів):

- CSS,

- картинок,

- скрипти на мові JavaScript.

На відміну від більшості інших систем управління

(наприклад, Joomla), в Magento теми можуть бути пов’язані кожна з одним відношенням успадкування: якщо в поточній темі не визначений конкретний

шаблон, правило відображення або ресурсний файл, то система бере їх з базової теми. Так, наприклад, всі три стандартних теми для Magento:

- Default,
- Modern,
- iPhone успадковуються від єдиної базової теми.

Сторонній програміст може створити свою базову тему. На відміну від Joomla, де прямо в адміністративній частині можна призначити місце відображення і параметри конкретного модуля на сторінці, Magento в основному націлена на редагування екранних блоків вітрини не через адміністративну частину, а через файли логічної розмітки і файли шаблонів.

Такий підхід, з одного боку, дає велику гнучкість розробникам магазинів, але, з іншого боку в цьому одна з причин, чому у простих адміністраторів Magento вважається складною у вивченні системою.

Основні можливості:

- З єдиної адміністративної частини можна управляти торговою мережею інтернет-магазинів на різних доменах, мовах, з різним товарним асортиментом.
- Гнучка настройка зовнішнього вигляду («теми» і «шаблонів») вітрини.
- Відображення цін на вітрині в різній валюті (у тому числі за вибором покупця).
- Багатомовність.
- Методи оплати і доставки.

Панель адміністратора:

- Маркетингові інструменти.
- Багаторівневе ціноутворення.
- Купони.
- Можливість організації розпродажів, налаштування оптових знижок
- Звіти з продажів, по корзинах покупця та про список відзначеного товару.
- Звіт про відкликання.

- Звіт за тегами та по Пошуку.
- Визначення податкових ставок.
- Визначення правил для регіону, країни або поштового коду.
- Створення класів податку, наприклад «Нормальний» або «Оптовий».
- Визначення правил податку, наприклад «Податок на одяг», також можна пов'язати різні групи товарів з класами податків.
- Продукти та каталог.
- Можливість завдання різних атрибутів (властивостей) для товару.
- Угруповання товару в комплекти.
- Конфігуровані продукти - продукти з вибірковими властивостями (наприклад колір, розмір, тощо).
- Сортування товару по наперед визначеним атрибутам. Атрибути доступні для сортування визначаються в панелі адміністрування.
- Система управління контентом.
- Групи покупців.
- Імпорт / Експорт (В даний момент імпорт товарів працює в обмеженому варіанті, можливе внесення товарів, але не оновлення).
- Система контролю доступу.
- Валюти.
- Можна вести облік (складський і управлінський) в одній валюті, а ціни на вітрині показувати в інший.
- Можна призначити різним вітринам різні валюти цінників.
- Можна дати відвідувачам можливість перемикатися між валютами.
- Можна налаштувати оновлення валютних курсів за розкладом.
- Налаштування відображення валют визначаються поточною локалізацією (локаллю) відвідувача (використовується бібліотека Zend Locale).

Можливості пошукової оптимізації.

Стандартна збірка Magento:

- Дає повний контроль над розміткою HTML. Є можливість установки індивідуального шаблону HTML для конкретного товару або товарного розділу. Дозволяє для кожної сторінки (товару, розділу або текстової) задавати індивідуальні мета-теги description і keywords[7].

- Дозволяє для кожного товару і товарного розділу задавати індивідуальне і настроювати закінчення адреси (URL Key).

- Дозволяє управляти для кожної сторінки заголовком TITLE (мається як автоматичний режим, так і ручний).

- Автоматично створює карту в форматі XML для пошукових ботів.

- Пошукова оптимізація товарних зображень. Magento при створенні сторінок вітрини описує товарні зображення атрибутом ALT тега IMG. Адміністратор для кожного товарного зображення може або вручну вказати опис, або дозволити Magento описати товарне зображення автоматично[5].

2.2 Підходи до проектування додатку та вибір програмних засобів

Алгоритм - це докладний набір інструкцій для виконання операції або вирішення проблеми. Технічно ПК використовують алгоритми для виведення детальних інструкцій щодо виконання операції. З точки зору ефективності, різні алгоритми можуть легко і швидко виконувати операції або вирішувати проблеми.[5]

Повноцінна розробка нового програмного рішення є доволі ресурсовитратноюта вимагає залучення додаткових фахівців, щонеухильно тягне за собою надмірні витрати часу.

Вищевикладені вимоги можуть бути реалізовані в рамкахіснуючих програмних продуктів.

В даному проекті, незважаючи на труднощі розробки будуть використовуватися нативні іструменти проектування та розробки [37, с. 180].

Впровадження готового рішення дозволить скоротити час, оскількивідсутні етапи розробки та налагодження програми.

Таким чином, поставлена задача зводиться до вибору системи (комплексу систем), що задовольняє заданим вимогам.

Перерахуємо етапи створення додатку.

1. Розробка бази даних.

Представимо на рис. 2.1 типову схему розробленої БД додатку.

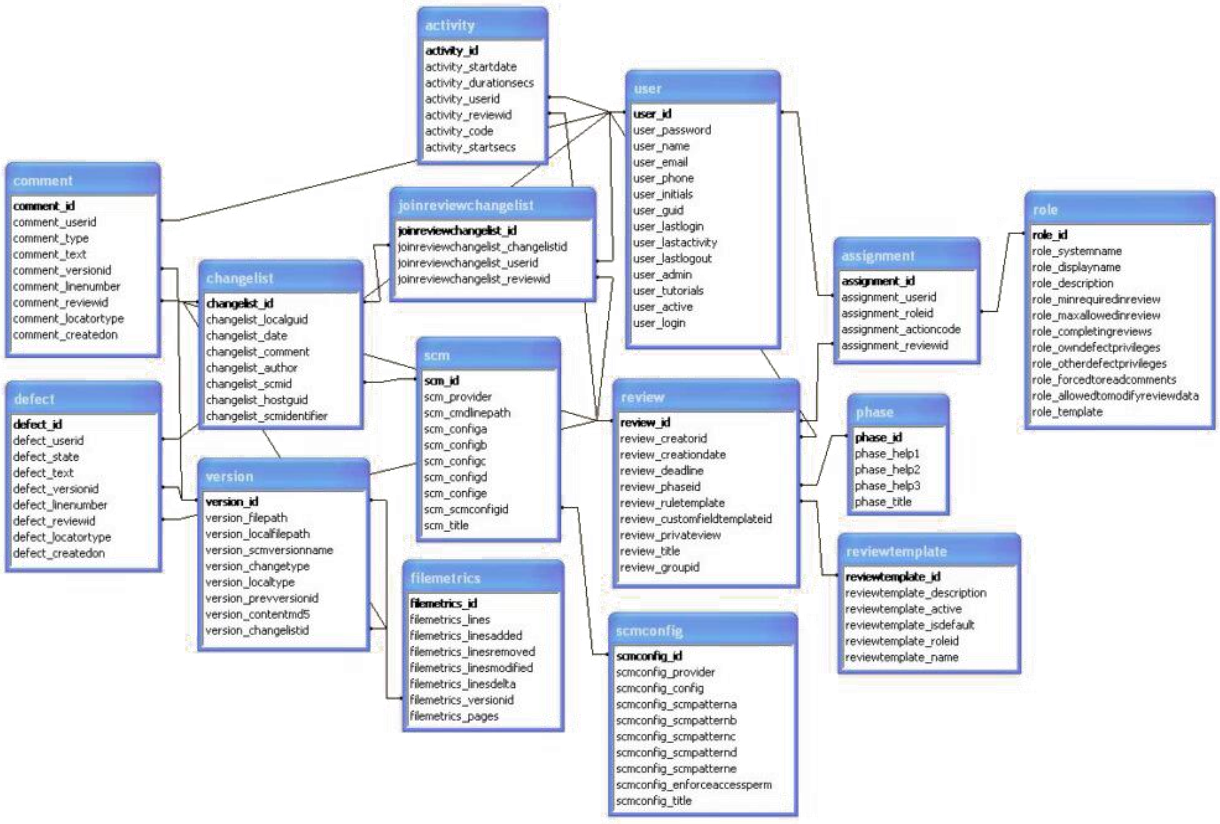

Рисунок 2.1 – Схема бази даних додатку

2. Розробка адміністративної частини

Адміністративна частина повинна включати:

- інформацію про нових користувачів,
- розділ настройок,
- можливість адміністрування магазину,
- можливість виведення статистики по магазину,
- управління обліковими записами користувачів і інші можливості.

Адміністративна частина представлена у вигляді системи управління вмістом. Система управління вмістом (CMS) – комп’ютерна програма, використовувана для управління вмістом чого-небудь (звичайний цей вміст розглядається як неструктуровані дані наочного завдання в протилежність структурованим даним, СУБД, що зазвичай знаходяться під управлінням)[10]. Зазвичай такі системи використовуються для зберігання і публікації великої кількості документів, зображень, музики або відео. Подібні CMS дозволяє управляти текстовим і графічним наповненням додатку, надаючи користувачеві зручні інструменти зберігання і публікації інформації.[25]

Зараз існує безліч готових систем управління вмістом додатку, у тому числі і безкоштовних. Їх можна розділити на три типи, за способом роботи[34]:

- Генерація сторінок за запитом. Системи такого типу працюють наоснові зв'язки «Модуль редагування > База даних > Модуль уявлення».

- Модуль уявлення генерує сторінку із змістом при запиті на нього, на основі інформації з бази даних.

- Інформація в базі даних змінюється за допомогою модуля редагування. Сторінки наново створюються сервером при кожному запиті, а це створює навантаження на системні ресурси. Навантаження може бути багато разів понижена при використанні засобів кешування, які є в сучасних веб-серверах.

- Генерація сторінок при редагуванні . Системи цього типу програми для редагування сторінок, які при внесенні змін до змісту додатку створюють набір статичних сторінок.

- Змішаний тип. Як зрозуміло з назви, поєднує в собі переваги перших два. Може бути реалізований шляхом кешування - модуль уявлення генерує сторінку один раз, надалі вона в рази швидше підвантажується з кеша.

Кеш може оновлюватися як автоматично, по закінченню деякого терміну часу або при внесенні змін до певних розділів, так і вручну по команді адміністратора. Інший підхід - збереження певних інформаційних блоків на етапі

редагування і збірка сторінки з цих блоків при запиті відповідної сторінки користувачем[33].

2.3 Аналіз та обґрунтування вибору системи управління вмістом

Система керування контентом надає адміністративні інструменти для організації авторизації, співпраці, що дозволяють користувачам з незначними знаннями в програмуванні створювати та керувати контентом додатку з відносною простотою.

Більшість систем використовує кешування настороні сервера для покращення швидкодії.

Адміністрування також зазвичай проводиться через інтерфейс, але деякі системи потребують використання «товстого» клієнту.

Подібні системи дозволяють технічно непідготовленим користувачам робити зміни в додатку після проходження тренувального курсу. Для налаштування та додавання нових функцій в додатку зазвичай потрібен системний адміністратор чи розробник.

Існують сотні доступних CMS. Завдяки їхній функціональності, CMS можна використовувати в різних компаніях. Незважаючи на широкий вибір інструментальних та технічних засобів, наявних в CMS, існують загальні для більшості типів систем характеристики[15].

Більшість сучасних CMS мають модульну архітектуру, що дозволяє адміністратору обирати та налаштовувати ті компоненти, які йому необхідні. Типові модулі:

- динамічне меню,
- блог,
- новини,
- опитування,
- пошук,
- статистика відвідувань,

- гостьова книга.

Додатки, організовані засобами систем управління контентом, засновані на наступних технологіях: сервер, сховище даних (часто СУБД, наприклад MySQL чи PostgresSQL, але існують і Firestore Database), застосунок (веб-додаток) для забезпечення роботи системи, візуальний (WYSIWYG – whatyouseeiswhatyouget – що бачиш те й отримуєш) редактор сторінок, файловий менеджер з інтерфейсом для управління файлами та система управління правами користувачів.

Найбільш поширені наступні технологічні платформи які використовуються в якості основи веб-додатку, який реалізує роботу CMS: PHP, Perl, .NET, JAVA, Kotlin[25].

Багато сучасних систем керування вмістом поширюються як безкоштовні і легкі у встановленні (інсталяції) програми, які розробляються групами ентузіастів під вільними ліцензіями такими як GNU/GPL, MIT чи Apache[37].

Існує термін контент-менеджер, який відображає вид професійної діяльності, тобто співробітника який відповідальний за наповнення контентом. CMS дозволяють контент-менеджеру не мати навичок програмування.

Більшість сучасних систем керування контентом мають візуальний редактор, який дозволяє користувачам створювати та редагувати сторінки використовуючи спрощену розмітку.[23] На Рис. 2.2. наведемо приклад модульної архітектури сучасної CMS.

Рисунок 2.2 – Приклад модульної архітектури сучасної CMS

Основні функції CMS:

- Додавання, редагування, видалення інформації.
- Організація спільної роботи з контентом.
- Керування режимом доступу користувачів.
- Формування сторінок.

Способи роботи:

- Генерація сторінок за запитом. Системи такого типу працюють наоснові зв'язки «модуль редагування → база даних → модуль представлення».

Модуль представлення генерує сторінку з контентом при запиті на нього на основі інформації з бази даних. Інформація в БД змінюється за допомогою модуля редагування. Сторінки заново створюються сервером при кожному запиті, а це створює навантаження на сервер. Але це навантаження може бути багатократно зменшене застосуванням різних методів кешування[12].

- Генерація сторінок при редагуванні. Системи цього типу при редагуванні сторінок вносять зміни у вміст додатку та створюють набір статичних сторінок.

- Змішаний тип. Як зрозуміло із назви, цей тип поєднує в собі переваги перших двох. Може бути реалізований шляхом кешування - модуль представлення генерує сторінку один раз, надалі вона через деякий час буде в декілька разів швидше завантажуватися із кешу. Кеш може оновлюватись як автоматично, через деякий час чи при внесенні змін у певні розділи додатку, так і вручну за командоюадміністратора. Другий підхід - збереження певних інформаційних блоків на етапі редагування додатку і збирання сторінок з цих блоків при запиті відповідної сторінки користувачем[39].

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель та стуктура додатку

Проаналізувавши досліжену інформацію було побудовано наступну інформаційну модель.

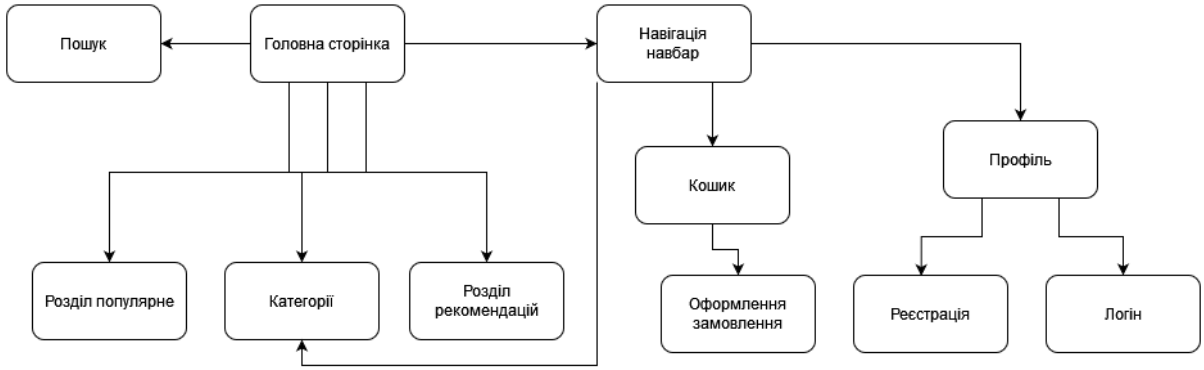

Рисунок 3.1 – Інформаційна модель додатку

Зовнішня структура додатку відповідає сучасному дизайну додатка електронної комерції. В самому низу знаходится навігаційна панель, яка дозволяє здійснювати переміщення між сторінками додатку. Зовнішня структура створюється для полегшення користування додатком користувачами. Цей вид інформаціної моделі відповідає зовнішньому макету додатку.

Основними елементами структури є:

- Шапка додатку(баннер).Зазвичай використовуеться в якості розміщення реклами, або оповіщення про технічні роботи додатка.
- Поле пошуку. Засіб для здіснення пошуку необіхідного товару.
- Центральна частина. Містить три блока, перший з яких це блок з популярними товарами, блок з категоріями товарів, та блок з рекомендувальними товарами.

- Навігаційний навбар. Розміщується у нижній частині додатку та служить для переміщення між сторінками додатку.

Для успішної структури додатку, дуже важливо правильне розміщення елементів, зручний інтерфейс, яскравий дизайн та простий фунціонал.

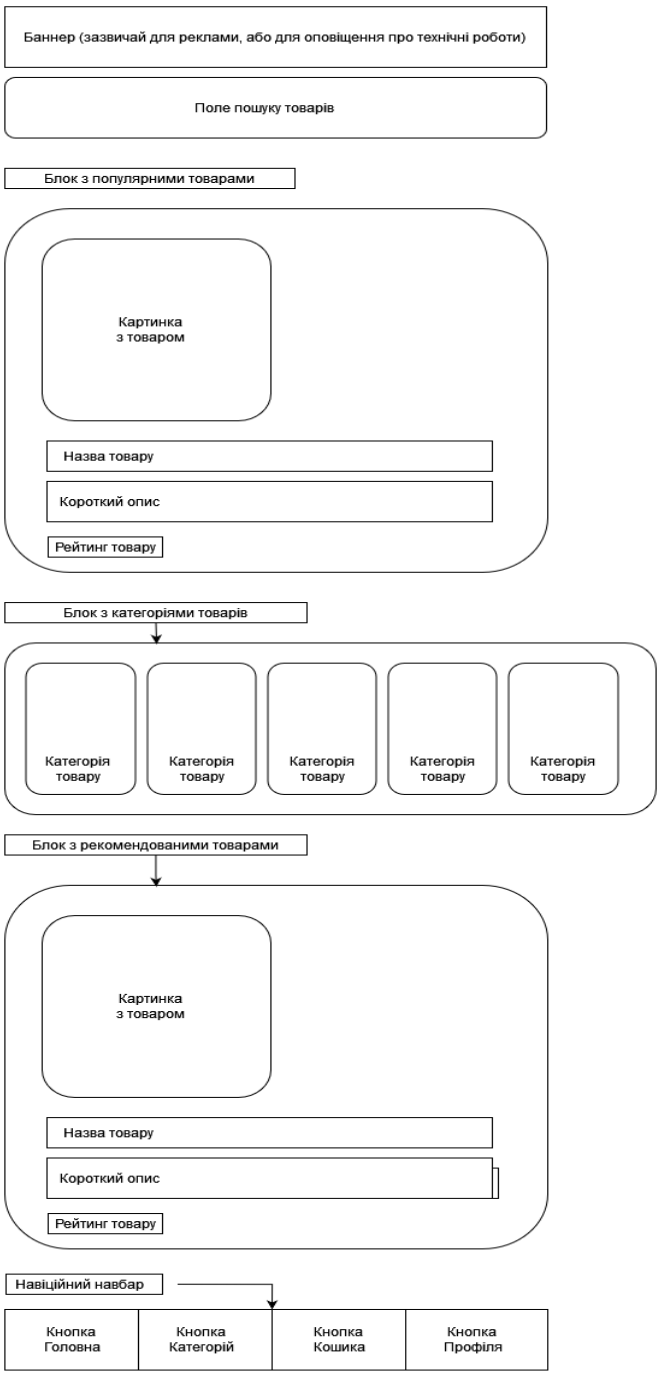

Рисунок 3.2 – Інформаційний макет додатку

3.2 Програмна реалізація

В першу чергу була обрана та розроблена база даних на основі готового рішення від Google. База даних Firebase Database, яка легко налаштовується та підключается до проекту.

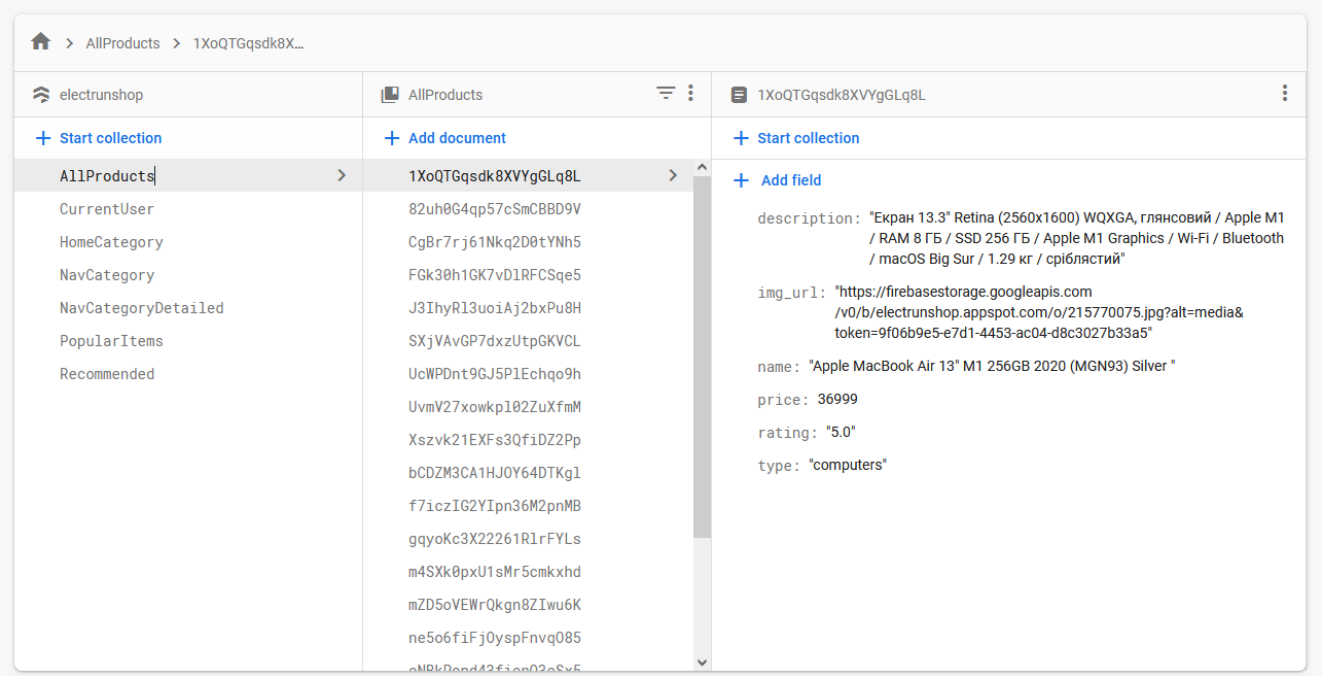

Рисунок 3.3 – Зовнішній вигляд бази даних

```
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

FirebaseFirestore db;
db = FirebaseFirestore.getInstance();

db.collection("НазваКатегорії")
.get()
.addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
   @Override
   public void onComplete(@NonNull Task<QuerySnapshot> task) {
      if (task.isSuccessful()) {
         for (QueryDocumentSnapshot document : task.getResult()) {
    HomeCategory homeCategory document.toObject(HomeCategory.class);
    categoryList.add(homeCategory);
    homeAdapter.notifyDataSetChanged();}
         } else {Toast.makeText(getActivity(),
"Error"+task.getException(), Toast.LENGTH_SHORT).show();}
                  }});
```

В даному коді відбувається підключення до бази даних однієї з блоків сторінки додатку, такий же самий код використовуется для підключення інших блоків.

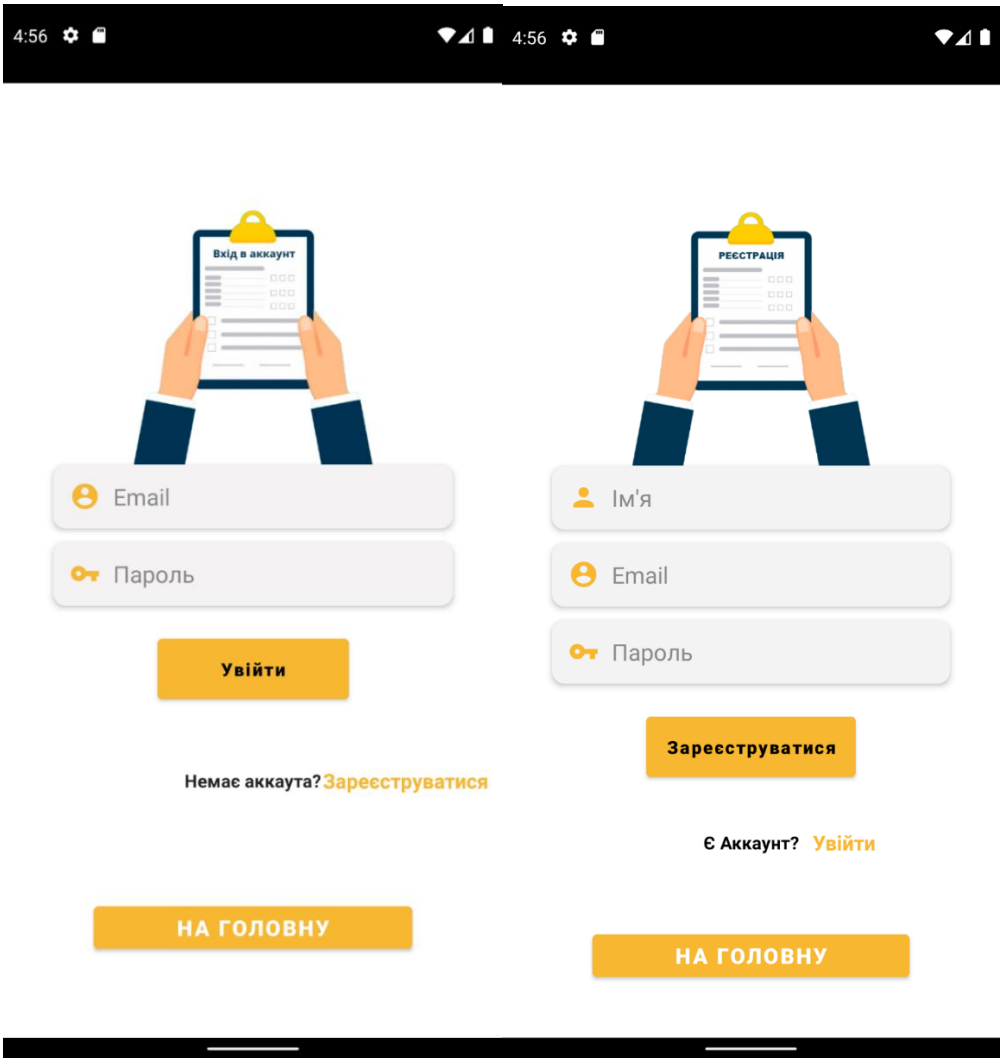

Рисунок 3.4 – Сторінки логіну та реєстрації

```
  auth.createUserWithEmailAndPassword(userEmail,userPassword)
            .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
           @Override
   public void onComplete(@NonNull Task<AuthResult> task) {
          if (task.isSuccessful()){
UserModel userModel = new
UserModel(userName,userEmail,userPassword);
String id = task.getResult().getUser().getUid();

database.getReference().child("Users").child(id).setValue(userModel)
;
```

```
 progressBar.setVisibility(View.GONE);
Toast.makeText(RegistrationActivity.this, "Реєстрація успіх",
Toast.LENGTH_SHORT).show();}
     else {
  progressBar.setVisibility(View.GONE);
                            Toast.makeText(RegistrationActivity.this,
"Помилка:"+task.getException(), Toast.LENGTH_SHORT).show();}
                    }
                });
```

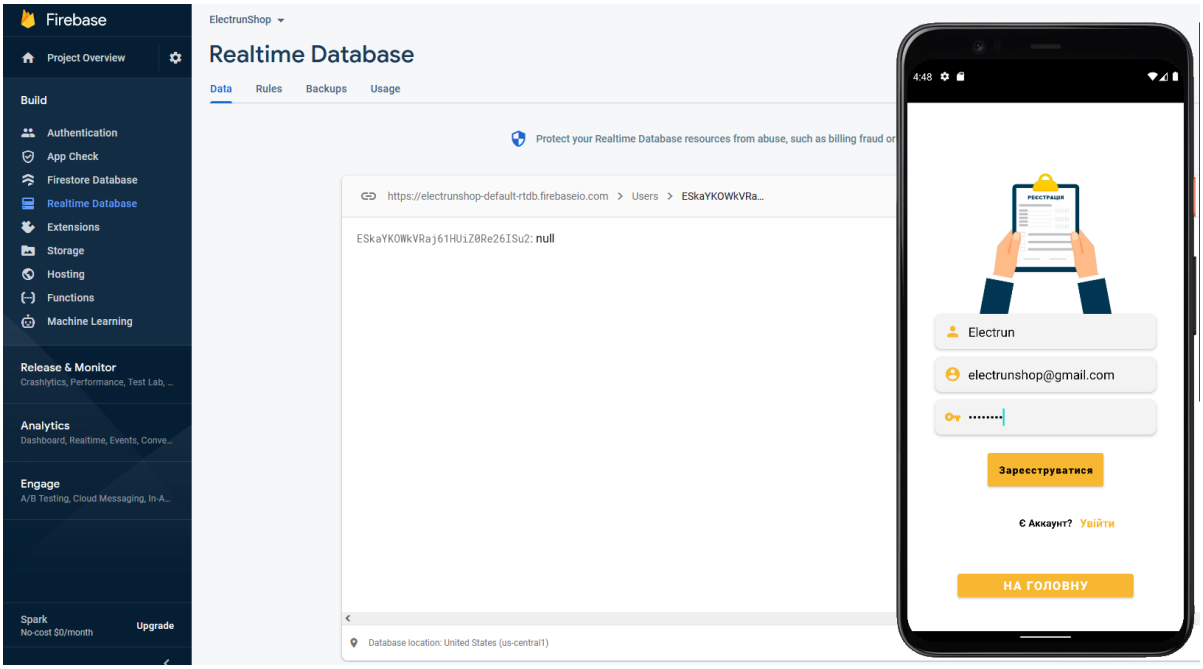

Рисунок 3.5 – Приклад реєстрації користувача

Слід зазначити, що при реєстрації відбувається перевірка даних що були введені у поля, реєстрація неможлива при наступних умовах:

- Якщо поле для введення ім’я користувача порожне
- Якщо пусте поле пошти
- Якщо користувач не ввів пароль
- Якщо пароль меньше 6 символів

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

        if(TextUtils.isEmpty(userName)){
            Toast.makeText(this,"Ім'я пусте!",
Toast.LENGTH_SHORT).show();
            return;}
       if(TextUtils.isEmpty(userEmail)){
```

```
            Toast.makeText(this,"Email пустий!",
Toast.LENGTH_SHORT).show();
            return;}
        if(TextUtils.isEmpty(userPassword)){
            Toast.makeText(this,"Пароль пустий!",
Toast.LENGTH_SHORT).show();
            return;}
        if (userPassword.length() < 6){
            Toast.makeText(this, "Пароль має бути більше 6
символів", Toast.LENGTH_SHORT).show();
            return;}
```

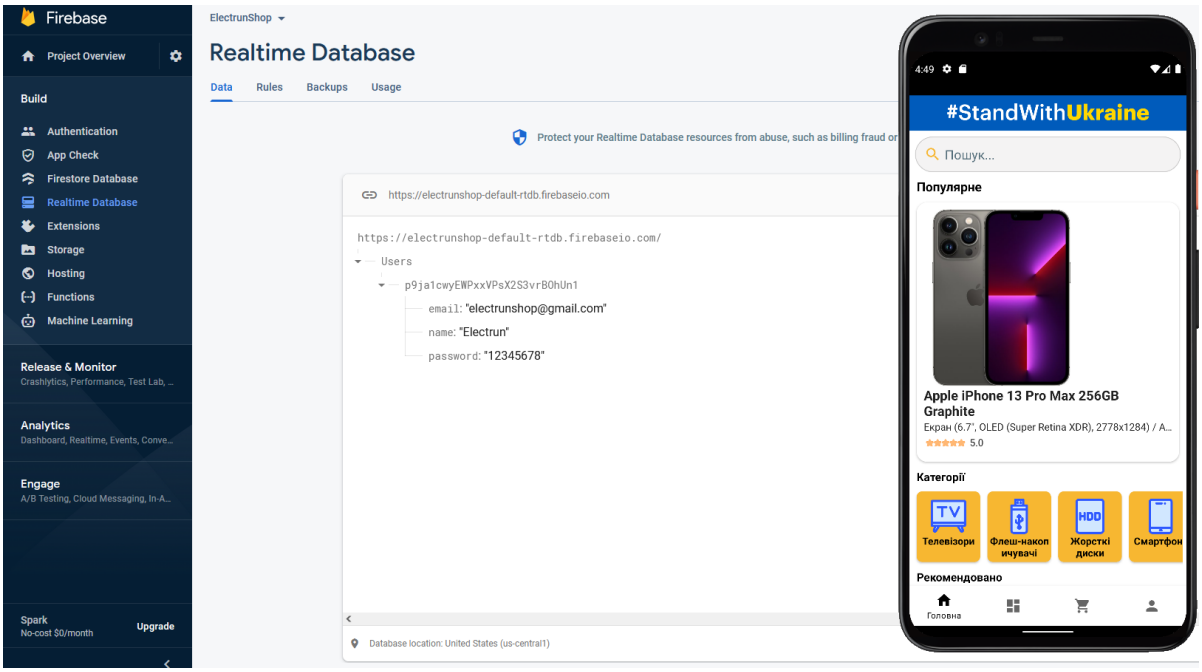

Рисунок 3.5 -–Приклад успішної реєстрації

Після успішної реєстрації виконується код, який в базі даних реального часу створює таблицю “Users” та унікальниий ідентифікатор користувача в який вносяться дані, які були вказані при реєстації, ім’я, пошта та пароль.

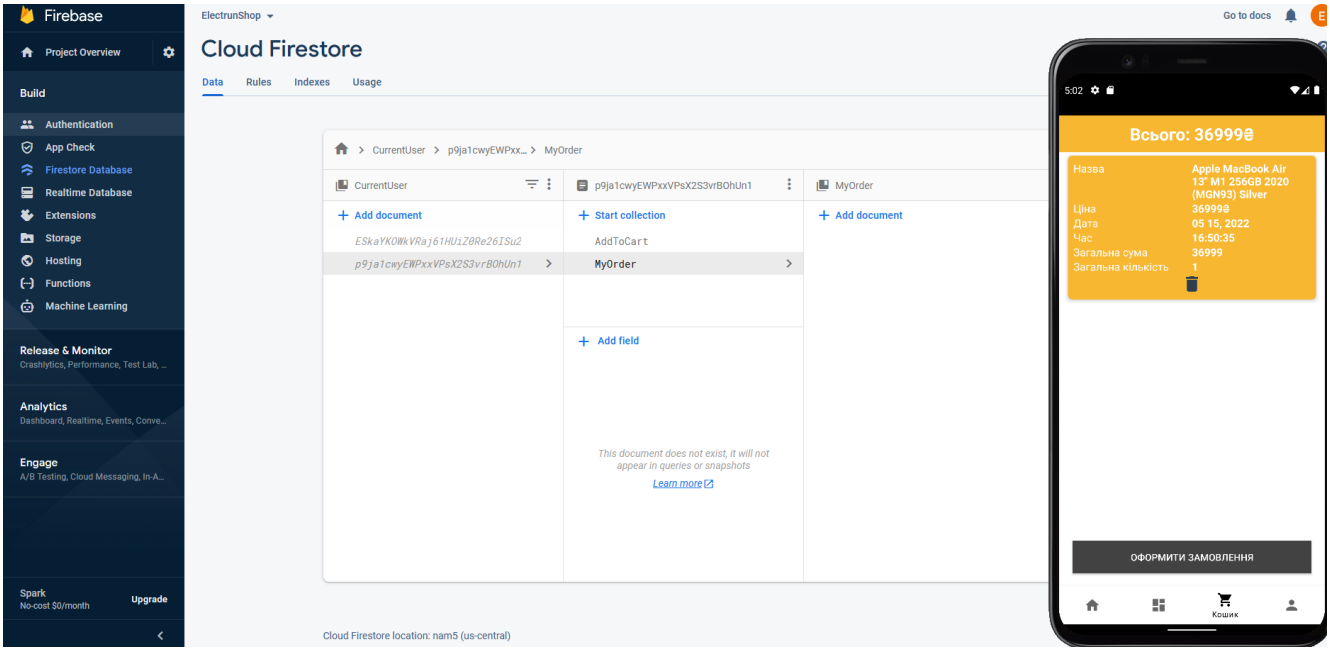

Рисунок 3.6 – Приклад створення замовлення

```
        List<MyCartModel> list = (ArrayList<MyCartModel>)
getIntent().getSerializableExtra("itemList");
        if (list != null && list.size() > 0) {
            for(MyCartModel model : list) {
                final HashMap<String, Object> cartMap = new
HashMap<>();
                cartMap.put("productName", model.getProductName());
                cartMap.put("productPrice",
model.getProductPrice());
                cartMap.put("currentDate", model.getCurrentDate());
                cartMap.put("currentTime", model.getCurrentTime());
                cartMap.put("totalQuantity",
model.getTotalQuantity());
                cartMap.put("totalPrice", model.getTotalPrice());

firestore.collection("CurrentUser").document(auth.getCurrentUser().g
etUid())

.collection("MyOrder").add(cartMap).addOnCompleteListener(new
OnCompleteListener<DocumentReference>() {
                    @Override
                    public void onComplete(@NonNull
Task<DocumentReference> task) {
```

```
Toast.makeText(PlacedOrderActivity.this,"Успіх!",
Toast.LENGTH_SHORT).show();}});
```

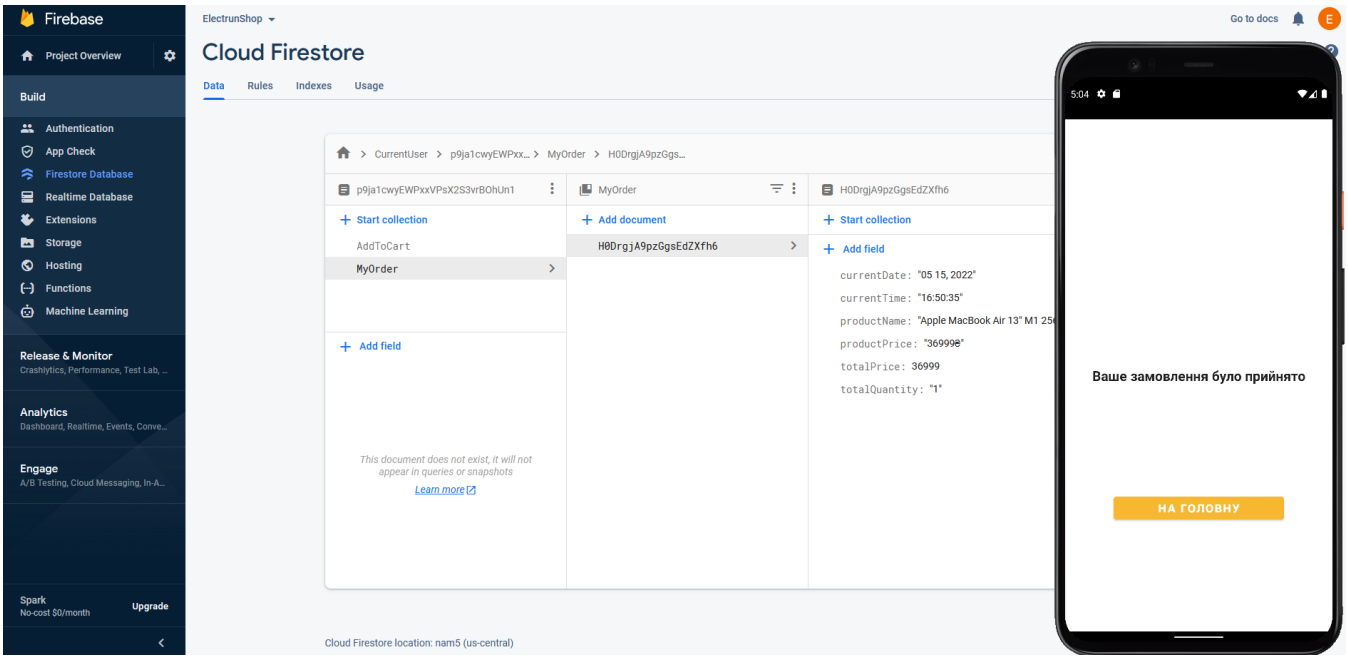

Рисунок 3.7 – Приклад упішного замовлення

```
firestore.collection("CurrentUser").document(auth.getCurrentUser().g
etUid())
 .collection("AddToCart")
```

При успішному замовленні товару, в базі даних Firebase Database створюється дві таблиці, “AddToCart” в яку вноситься спочатку унікальний індентификатор замовлення, а в цей індентифікатор записуються товари, які користувач додав у кошик. Також створюється таблиця “MyOrder”, це таблиця з товарами які користувач замовив, для зручності перегляду, які і коли товари користувач замовив, до таблиці додано такі атрибути:

- `currentDate` – Поточна дата
- `currentTime` – Поточний час
- `productName` – Назва товару

- `productPrice` – Ціна товару
- `totalPrice` – Загальна сума
- `totalQuantit` – Загальна кількість товарів

Рисунок 3.8 – Головна сторінка додатка

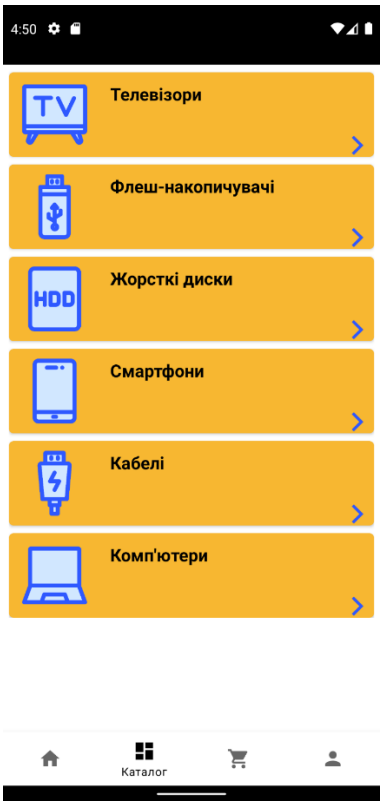

Рисунок 3.9 – Сторінка з категоріями товарів

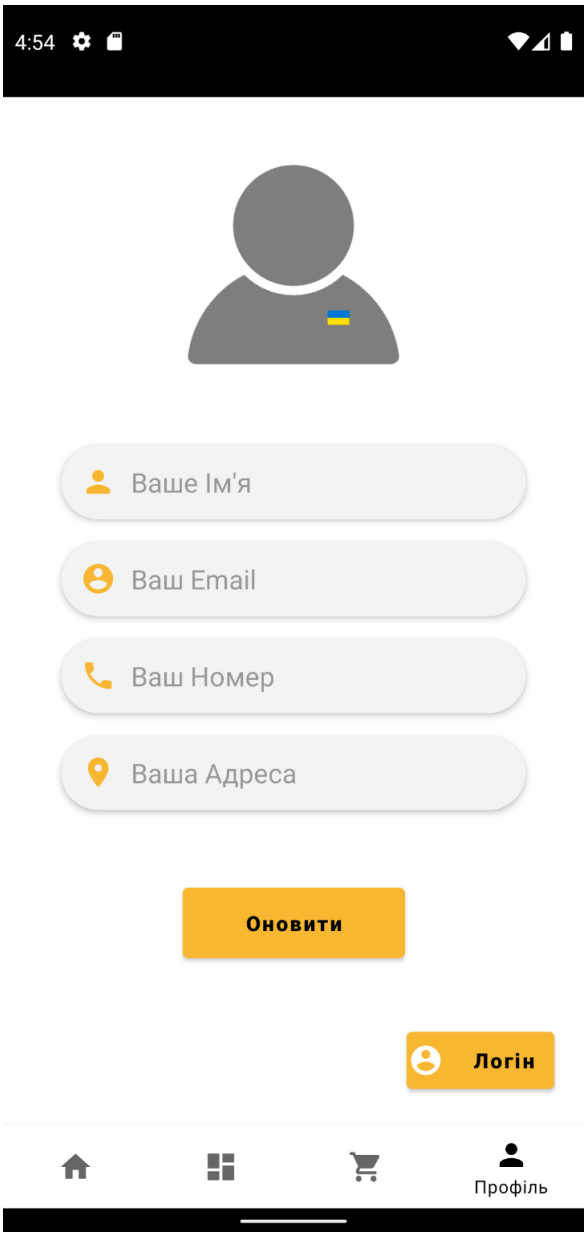

Рисунок 3.10 – Сторінка профіля користувача

Сторінка профіля користувача, на які він має змогу зареєструватися або зробити вхід в аккаунт при натисканні кнопки “Логін”, також на сторінці профіля користувач має змогу написати або внести зміни своєї персональної інформації, такої як:

- Ім’я
- Пошта (Email)
- Номер телефона
- Адреса

ВИСНОВКИ

Під час виконнання роботи було проаналізовано та розгянуто сучасні засоби розробки мобільних додатків, які в свою чергу містять в собі різні види, такі як веб-додатки, гібридні додатки та нативні додатки. На основі розглянутих програмних засобів та аналізу інформаціних технологій було розроблено концептуальну модель додатку.

Використання мобільних додатків електронної комерції зумовлює велику кількість переваг, ніж використання мобільних версій сайтів інтернет магазинів в браузері, це набагато прискорює взаємодію користувача з інтернет магазином, адже для пошуку небхідного товару потрібно лише запустити мобільний застосунок і не шукати необхідний сайт в браузері.

Результатом виконання роботи є створений мобільний додаток електронної комерції. Розроблений додаток має сучаний зовнішний вигляд, та простий і зручний функціонал, який задоволяє всім вимогам комфортного користування мобільним застосунком.

СПИСОК ЛІТЕРАТУРИ

1. Антонюк В. С. Методологія наукових досліджень: [Текст] : навч. посіб./ В.С. Антонюк, Л. Г. Полонський, В. І. Аверченков, Ю. А. Малахов. – К.: НТУУ «КПІ», 2015. – 286 с.

2. Безкоштовні редактори коду для програмістів [Електронний ресурс] – Режим доступу до ресурсу: https://techrocks.ru/2018/09/10/top-6-popular-free-source-code-editors/. (дата звернення: 24.04.2020).

3. Вентцель Е. С. Теорія ймовірностей: Учеб.для вузів. - 6-е вид. стер. - М.: Высш. шк., 1999. – С. 12-54.

4. Дейт К. Дж. Введеннядо системи баз даних - 8-е вид. - М.: Вильямс, 2006. - 1328 с.

5. Іванкевич О.В. Інформаційні системи та структури даних / О.В. Іванкевич, Г.М. Кременецький, В.І. Мазур // Навч.посібник. –К.: НАУ, 2006. –232 с.

6. Інтернет-ресурс. Режим доступу -http://www.opennet.ru/ opennews/art.shtml?num=30814.

7. Інтернет-ресурс. Режим доступу - http://www.cnet.com/news/and-the-winners-of-the-2018-sourceforge-community-choice-awards-are.

8. Інтернет-ресурс. Режим доступу - http://dou.ua/lenta/interviews/interview-magento-2019/

9. Інтернет-ресурс. Режим доступу - http://forbes.ua/company/1476.

10. Інтернет-ресурс. Режим доступу - http://builtwith.com/allo.ua.

11. Інтернет-ресурс. Режим доступу - http://forbes.ua/magazine/forbes/ 1336514-v-nachale-bolshogo-vzryva-top-15-internet-kompanij.

12. Інтернет-ресурс. Режим доступу - http://builtwith.com/mo.ua.

13. Інтернет-ресурс. Режим доступу - http://wikipedia.org.

14. Інтернет-ресурс. Режим доступу - http://www. Magento commerce. com/wiki.

15. Інтернет-ресурс. Режим доступу - http://www. itfb.com.ua /monitoring.html.

16. Інтернет-ресурс. Режим доступу - http://habrahabr.ru/ company/croc/blog/14494.

17. Інтернет-ресурс. Режим доступу – http://www.pingwinsoft.ru /pages/resheniya /resheniya/pwsitmonitoring.

18. Інтернет-ресурс. Режим доступу - http://amigosteam.ru/blog/item/11-zabbix.

19. Інтернет-ресурс. Режим доступу - http://www.opennms.org.

20. Колісніченко Д. М. PHP и SQL. Розробка Web-додатків. / Д. М.Колісніченко. – СПб.: БХВ-Петербург, 2013. – 543 с.

21. Комашинський В. І. Смірнов Д. О. Введення в нейро-інформаційні технології. / В. І. Комашинський, Д. А. Смірнов - СПб, 1999. – С. 33-48.

22. Ліпунцов Ю. П. Управління процесами. М: Компанія АйТі, 2003. – С. 33-42.

23. Лотов О. В., Поспєлова І. І. Багатокритеральні задачі прийняттярішень: навч. посібник. М.: МАКС Прес, 2008. – С. 77-89.

24. Райчев І.Е. Принципи проектування відкритих розподілених систем / І. Е. Райчев, О.Г.Харченко, В. В. Замковий // Навч. посіб. – К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», 2010. – 240 с.

25. Салов В.О. Методичні рекомендації до виконання кваліфікаційних робіт випускників спеціальності 8.080403 «Програмне забезпечення автоматизованихсистем» напряму 0804 «Комп’ютерні науки». Стандарт вищого навчального закладу (СТВНЗ-2070743-КР 2004). / В.О. Салов, М.О. Алексєєв, Г.М. Коротенко,Л. М. Коротенко; М-во освіти і науки України, Нац. гірн. ун-т. – Д.: НГУ, 2004.-55с.

26. Столлінгс В. Передача даних/ В. Столлінгс. - СПб.: Пітер, 2004. – 750 с.

27. Поспєлов Г. С Штучнийінтелект - основа нової інформаційної технології - М.: Вища школа, 1988 – С. 129-154.

28. Растригин Л. А., Эйдук Я. Ю. Адаптивні методи багатокритеріальної оптимізації // Автоматика та телемеханіка. -1985. - № 1. - С. 5-26.

29. Роберт Каллан. Основні концепції нейроних мереж = The Essence of Neural Networks First Edition. - 1-е. - «Вильямс», 2001. - С. 200-233.

30. Рутковська Д. Нейронні мережі, генетичні алгоритми та нечеткі системи / Д. Рутковська, М. Пилиньский, Л. Рутковський. - М. Гаряча лінія - Телеком. 2004. – С. 34-46.

31. Саати Томас Л. Прийняттярішень при залежностяхтазворотніхзв’язках: Аналітичнімережі [Текст] Пер. з англ. / Наук. Ред. О. В. Андрейчіков, О. М. Андрейчікова. – М.: Вид.-во ЛКІ, 2008. – С. 28-39.

32. Системи автоматизації діяльності організації [Електронний ресурс] - Режим доступу: http://www.in-line.ru/solutions/business_appl.

33. Совєтов Б. Я. Інформаційні технології / Б.Я. Совєтов, В.В Цехановський - М.: Вища школа, 2005 – С. 55-63.

34. Тархов Д. А. Нейронні мережі. Моделі а алгоритми. – М.: Радіотехніка, 2010. – С. 65-70.

35. Терехов В. А., Єфімов Д. В., Тюкин И. Ю. Нейромережні системи керування. - 1-е. - Высшая школа, 2002. - С. 180-184.

36. Уосермен Ф. Нейрокомп’ютерна техніка: Теорія і практика. Переклад українською І. Ю. Юрчак, 2001. – С. 88-94.

37. Чорноруцький І. Г. Методи прийняттярішень [Текст]/І. Г. Черноруцький.– СПб.: БХВ-Петербург, 2005. – С. 388-395.

38. Ясницький Л. Н. Введення в штучний інтелект. - 1-е. – Видавничий центр «Академія», 2005. - С. 170-176.

39. Guttman Antonin. R - trees: a dynamic index structure for spatial searching. ACM SIGMOD International Conference on Management of Data, 1984. – P. 43-52.

40. Richard C. Larson. Perspectives on Queues: Social Justice and the Psychology of Queueing. – INFORMS, 1987 – P. 895-905.

ДОДАТОК А

MainActivity.java;

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import com.example.electrun.databinding.ActivityMainBinding;
public class MainActivity extends AppCompatActivity {

    private ActivityMainBinding binding;

    public void onClick(View v) {
        Intent intent = new Intent(this, LoginActivity.class);
        startActivity(intent);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
```

```
        setContentView(binding.getRoot());


        BottomNavigationView navView = findViewById(R.id.nav_view);

        // Passing each menu ID as a set of Ids because each

        // menu should be considered as top level destinations.

        AppBarConfiguration appBarConfiguration = new
AppBarConfiguration.Builder(

                R.id.navigation_home, R.id.navigation_catalog,
R.id.navigation_cart, R.id.navigation_account)

                .build();

        NavController navController =
Navigation.findNavController(this,
R.id.nav_host_fragment_activity_main);


        //This is remove actionbar

        // NavigationUI.setupActionBarWithNavController(this,
navController, appBarConfiguration);


        NavigationUI.setupWithNavController(binding.navView,
navController);

    }

}
```

ДОДАТОК Б

RegistrationActivity.java

```
public class RegistrationActivity extends AppCompatActivity {

    Button signUp;
    TextView name,email,password;
    TextView signIn;
    FirebaseAuth auth;
    FirebaseDatabase database;
    ProgressBar progressBar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);

        auth = FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance();

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

        signUp = findViewById(R.id.reg_btn);
        name = findViewById(R.id.name);
        email = findViewById(R.id.email_reg);
        password = findViewById(R.id.password_reg);
```

```
        signIn = findViewById(R.id.sign_in);


        signIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new
Intent(RegistrationActivity.this,LoginActivity.class));
            }
        });
        signUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                createUser();
                progressBar.setVisibility(View.VISIBLE);


            }
        });
    }
    public void onClickHome(View v) {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
    private void createUser() {


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

```
        if(TextUtils.isEmpty(userName)){

            Toast.makeText(this,"Ім'я пусте!",
Toast.LENGTH_SHORT).show();

            return;

        }


        if(TextUtils.isEmpty(userEmail)){

            Toast.makeText(this,"Email пустий!",
Toast.LENGTH_SHORT).show();

            return;

        }


        if(TextUtils.isEmpty(userPassword)){

            Toast.makeText(this,"Пароль пустий!",
Toast.LENGTH_SHORT).show();

            return;

        }

        if (userPassword.length() < 6){

            Toast.makeText(this, "Пароль має бути більше 6
символів", Toast.LENGTH_SHORT).show();

            return;

        }


        //Create User

        auth.createUserWithEmailAndPassword(userEmail,userPassword)

                .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {

                    @Override

                    public void onComplete(@NonNull Task<AuthResult>
task) {
```

```
                        if (task.isSuccessful()){


                            UserModel userModel = new
UserModel(userName,userEmail,userPassword);

                            String id =
task.getResult().getUser().getUid();


database.getReference().child("Users").child(id).setValue(userModel)
;

                            progressBar.setVisibility(View.GONE);



Toast.makeText(RegistrationActivity.this, "Реєстрація успіх",
Toast.LENGTH_SHORT).show();

                        }

                        else {

                            progressBar.setVisibility(View.GONE);


Toast.makeText(RegistrationActivity.this,
"Помилка:"+task.getException(), Toast.LENGTH_SHORT).show();

                        }

                    }

                });


    }

}
```

ДОДАТОК В

ViewAllActivity.java

```
public class ViewAllActivity extends AppCompatActivity {


    FirebaseFirestore firestore;
    RecyclerView recyclerView;
    ViewAllAdapter viewAllAdapter;
    List<ViewAllModel> viewAllModelList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_all);


        firestore = FirebaseFirestore.getInstance();
        String type = getIntent().getStringExtra("type");
        recyclerView = findViewById(R.id.view_all_rec);
        recyclerView.setLayoutManager(new
LinearLayoutManager(this));


        viewAllModelList = new ArrayList<>();
        viewAllAdapter = new ViewAllAdapter(this, viewAllModelList);
        recyclerView.setAdapter(viewAllAdapter);


        //get phones
        if (type != null && type.equalsIgnoreCase("phones")) {
            firestore.collection("AllProducts").whereEqualTo("type",
"phones").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
```

```
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot>
task) {
                    for (DocumentSnapshot documentSnapshot :
task.getResult().getDocuments()) {
                        ViewAllModel viewAllModel =
documentSnapshot.toObject(ViewAllModel.class);
                        viewAllModelList.add(viewAllModel);
                        viewAllAdapter.notifyDataSetChanged();
                    }
                }
            });
        }
            //get tv
            if (type != null && type.equalsIgnoreCase("tv")) {

firestore.collection("AllProducts").whereEqualTo("type",
"tv").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull
Task<QuerySnapshot> task) {
                        for (DocumentSnapshot documentSnapshot :
task.getResult().getDocuments()) {
                            ViewAllModel viewAllModel =
documentSnapshot.toObject(ViewAllModel.class);
                            viewAllModelList.add(viewAllModel);
                            viewAllAdapter.notifyDataSetChanged();
                        }
                    }
                });
```

```
            }


                //get usb_flash

                if (type != null &&
type.equalsIgnoreCase("usb_flash")) {


firestore.collection("AllProducts").whereEqualTo("type",
"usb_flash").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {

                        @Override

                        public void onComplete(@NonNull
Task<QuerySnapshot> task) {

                            for (DocumentSnapshot
documentSnapshot:task.getResult().getDocuments()){

                                ViewAllModel viewAllModel =
documentSnapshot.toObject(ViewAllModel.class);

                                viewAllModelList.add(viewAllModel);


viewAllAdapter.notifyDataSetChanged();

                            }

                        }

                    });

        }


        //get hard_disk

        if (type != null && type.equalsIgnoreCase("hard_disk")) {

            firestore.collection("AllProducts").whereEqualTo("type",
"hard_disk").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {

                @Override

                public void onComplete(@NonNull Task<QuerySnapshot>
task) {
```

```
                    for (DocumentSnapshot
documentSnapshot:task.getResult().getDocuments()){

                        ViewAllModel viewAllModel =
documentSnapshot.toObject(ViewAllModel.class);

                        viewAllModelList.add(viewAllModel);

                        viewAllAdapter.notifyDataSetChanged();

                    }

                }

            });

        }


        //get cables

        if (type != null && type.equalsIgnoreCase("cables")) {

            firestore.collection("AllProducts").whereEqualTo("type",
"cables").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {

                @Override

                public void onComplete(@NonNull Task<QuerySnapshot>
task) {

                    for (DocumentSnapshot
documentSnapshot:task.getResult().getDocuments()){

                        ViewAllModel viewAllModel =
documentSnapshot.toObject(ViewAllModel.class);

                        viewAllModelList.add(viewAllModel);

                        viewAllAdapter.notifyDataSetChanged();

                    }

                }

            });

        }
```

```
        //get computers
        if (type != null && type.equalsIgnoreCase("computers")) {
            firestore.collection("AllProducts").whereEqualTo("type",
"computers").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot>
task) {
                    for (DocumentSnapshot
documentSnapshot:task.getResult().getDocuments()){
                        ViewAllModel viewAllModel =
documentSnapshot.toObject(ViewAllModel.class);
                        viewAllModelList.add(viewAllModel);
                        viewAllAdapter.notifyDataSetChanged();
                    }
                }
            });
        }
    }
}
```

ДОДАТОК Г

DetailedActivity.java

```
public class DetailedActivity extends AppCompatActivity {

    TextView quantity;
    int totalQuantity = 1;
    int totalPrice = 0;
    ImageView detailedImg;
    TextView price, rating, description;
    Button addToCart;
    ImageView addItem, removeItem;

    FirebaseFirestore firestore;
    FirebaseAuth auth;

    ViewAllModel viewAllModel = null;

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

        firestore = FirebaseFirestore.getInstance();
        auth = FirebaseAuth.getInstance();

        final Object object =
getIntent().getSerializableExtra("detail");
```

```
        if (object instanceof ViewAllModel) {
            viewAllModel = (ViewAllModel) object;

        }
        quantity = findViewById(R.id.quantity);
        detailedImg = findViewById(R.id.detailed_img);
        addItem = findViewById(R.id.add_item);
        removeItem = findViewById(R.id.remove_item);

        price = findViewById(R.id.detailed_price);
        rating  = findViewById(R.id.detailed_rating);
        description  = findViewById(R.id.detailed_description);

        if (viewAllModel != null) {

Glide.with(getApplicationContext()).load(viewAllModel.getImg_url()).
into(detailedImg);
            rating.setText(viewAllModel.getRating());
            description.setText(viewAllModel.getDescription());
            price.setText(viewAllModel.getPrice()+"₹");

            totalPrice = viewAllModel.getPrice() * totalQuantity;
        }

        addToCart = findViewById(R.id.add_to_cart);
        addToCart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addedToCart();
```

```
            }
        });
        addItem.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(totalQuantity < 10) {
                    totalQuantity++;
                    quantity.setText(String.valueOf(totalQuantity));
                    totalPrice = viewAllModel.getPrice() *
totalQuantity;
                }
            }
        });
        removeItem.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(totalQuantity > 1) {
                    totalQuantity--;
                    quantity.setText(String.valueOf(totalQuantity));
                    totalPrice = viewAllModel.getPrice() *
totalQuantity;
                }
            }
        });
    }
    private void addedToCart() {
        String saveCurrentDate, saveCurrentTime;
        Calendar calForDate = Calendar.getInstance();
```

```
        SimpleDateFormat currentDate = new SimpleDateFormat("MM dd,
yyyy");

        saveCurrentDate = currentDate.format(calForDate.getTime());

        SimpleDateFormat currentTime = new
SimpleDateFormat("HH:mm:ss");

        saveCurrentTime = currentTime.format(calForDate.getTime());


        final HashMap<String, Object> cartMap = new HashMap<>();


        cartMap.put("productName", viewAllModel.getName());

        cartMap.put("productPrice", price.getText().toString());

        cartMap.put("currentDate", saveCurrentDate);

        cartMap.put("currentTime", saveCurrentTime);

        cartMap.put("totalQuantity", quantity.getText().toString());

        cartMap.put("totalPrice", totalPrice);



firestore.collection("CurrentUser").document(auth.getCurrentUser().g
etUid())


.collection("AddToCart").add(cartMap).addOnCompleteListener(new
OnCompleteListener<DocumentReference>() {

            @Override

            public void onComplete(@NonNull Task<DocumentReference>
task) {

                Toast.makeText(DetailedActivity.this,"Додано у
кошик", Toast.LENGTH_SHORT).show();

                finish();

            }

        });

    }
```

```
}
```

ДОДАТОК Д

PlacedOrderActivity.java

```
public class PlacedOrderActivity extends AppCompatActivity {

    FirebaseAuth auth;
    FirebaseFirestore firestore;



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

        auth = FirebaseAuth.getInstance();
        firestore = FirebaseFirestore.getInstance();

        List<MyCartModel> list = (ArrayList<MyCartModel>)
getIntent().getSerializableExtra("itemList");

        if (list != null && list.size() > 0) {
            for(MyCartModel model : list) {

                final HashMap<String, Object> cartMap = new
HashMap<>();

                cartMap.put("productName", model.getProductName());
                cartMap.put("productPrice",
model.getProductPrice());
```

```
                cartMap.put("currentDate", model.getCurrentDate());

                cartMap.put("currentTime", model.getCurrentTime());

                cartMap.put("totalQuantity",
model.getTotalQuantity());

                cartMap.put("totalPrice", model.getTotalPrice());



firestore.collection("CurrentUser").document(auth.getCurrentUser().g
etUid())


.collection("MyOrder").add(cartMap).addOnCompleteListener(new
OnCompleteListener<DocumentReference>() {

                    @Override

                    public void onComplete(@NonNull
Task<DocumentReference> task) {


Toast.makeText(PlacedOrderActivity.this,"Успіх!",
Toast.LENGTH_SHORT).show();



                    }

                });

            }


        }

    }

}
```