

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА ANDROID-ЗАСТОСУНКУ МАГАЗИНУ З ДОСТАВКОЮ**

Виконав: студент 4 курсу, групи КНТ-519
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____
«Комп'ютерна інженерія»

СОКОЛОВ М.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГРУШКО С.С.

(ПРИЗВИЩЕ та ініціали)

Рецензент КУЛЯБА-ХАРИТОНОВА Т.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

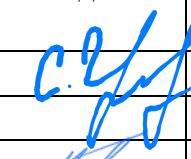
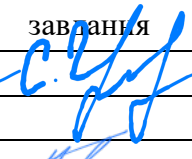
ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

СОКОЛОВ Михайло Олександрович
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка Android-застосунку магазину з доставкою
- керівник проєкту (роботи) к. т. н., доцент, Грушко Світлана Сергіївна
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)
- затверджені наказом вищого навчального закладу від “24” березня 2023 року № 71
2. Строк подання студентом проєкту (роботи) 10 травня 2023 року
3. Вихідні дані до проєкту (роботи) наявні методи реалізації застосунку магазину з доставкою на платформі Android
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз технічного завдання, дослідження наявних методів реалізації застосунку магазину з доставкою на платформі Android;
2) Розробка структури та алгоритму роботи магазину з доставкою на платформі Android;
3) Реалізація та дослідження програми.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Схема взаємодії між елементами архітектурного паттерну MVC
Схема-алгоритм роботи додатку
Алгоритм процесу покупки товару
Макет каталогу товарів

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ГРУШКО С.С., к.т.н., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.03.2023 р.

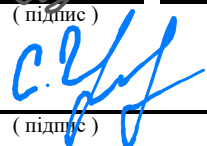
КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз технічного завдання. Огляд сучасних рішень	01.03.2023 р.	
2	Розробка структури системи	15.03.2023 р.	
3	Розробка алгоритму роботи системи	20.03.2023 р.	
4	Реалізація проєкту	01.04.2023 р.	
5	Дослідження роботи проєкту	10.04.2023 р.	
6	Оформлення отриманих результатів у ПЗ	01.05.2023 р.	
7	Оформлення графічного матеріалу	10.05.2023 р.	
8	Оформлення допоміжного матеріалу	17.05.2023 р.	

Студент (ка)

 Михайло СОКОЛОВ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

 Світлана ГРУШКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 61 с., 13 рис., 26 джерел.

ANDROID SDK, ANDROID STUDIO, CLOUD STORAGE FIREBASE, FIREBASE REALTIME DATABASE, GOOGLE MAPS, БАЗА ДАНИХ, БІБЛІОТЕКА PICASSO, МОБІЛЬНИЙ ЗАСТОСУНОК, ОПЕРАЦІЙНА СИСТЕМА ANDROID, ХМАРНЕ СХОВИЩЕ

Об'єкт розробки – створення мобільного застосунку для магазину «MonoSell» з функцією доставки на платформі Android.

Метою роботи є розробка мобільного застосунку на платформі Android для магазину «MonoSell» з функцією доставки, проведення аналізу наявних програмних засобів, проектування архітектури та дизайну інтерфейсу застосунку, , визначення подальших напрямків вдосконалення розробленого застосунку.

Проект складається з трьох розділів. У першому розділі буде розглянуто вже існуючі рішення на ринку, зокрема, буде проаналізовано, які додатки для мобільних пристроїв існують і як вони реалізовані. У другому розділі буде розроблено архітектуру застосунку, включно з вибором архітектурних патернів, розподілом функціоналу на компоненти та описом взаємодії між ними. Також буде розроблено дизайн інтерфейсу застосунку, включно з розробкою макетів, вибором колірної гами, шрифтів та іконок. У третьому розділі описується процес створення самого мобільного додатку - розробка програмного коду, використання певних технологій, фреймворків, проведення тестування, аналіз результатів та виправлення помилок і дефектів застосунку.

В результаті виконання дипломної роботи проведено аналіз, моделювання та розробку Android-застосунку для магазину з функцією доставки. Також приведені рекомендації щодо його модернізації та розширення функціоналу.

ЗМІСТ

Скорочення та умовні позначки	7
Вступ	8
1 Аналіз ринку	9
1.1 Огляд наявних рішень в галузі магазинів з доставкою на платформі Android	9
1.2 Аналіз основних технологій, фреймворків та інструментів, використовуваних для розробки мобільних додатків на платформі Android	13
1.3 Виявлення потреб та очікувань користувачів від магазину з доставкою на платформі Android	28
2 Проектування архітектури та функціоналу застосунку	30
2.1 Визначення вимог до функціоналу застосунку для магазину з функцією доставки на основі потреб користувачів	31
2.2 Розробка архітектури застосунку, включаючи вибір архітектурних паттернів, розподіл функціоналу на компоненти та опис взаємодії між ними	33
2.3 Розробка дизайну інтерфейсу користувача застосунку, включаючи розробку макетів, вибір кольорової гами, шрифтів та іконок	35
3 Реалізація та тестування застосунку	43
3.1 Розробка програмного коду застосунку з використанням вибраних технологій та фреймворків	44
3.2 Виконання тестування та аналіз результатів для перевірки функціональності, стабільності та ефективності застосунку	54
3.3 Виявлення та виправлення помилок та дефектів застосунку	55
Висновки	56
Перелік джерел посилання	59
Додаток А	62
Додаток Б	65
Додаток В	70

Графічний матеріал:

Плакат 1 - Схема взаємодії між елементами архітектурного паттерну MVC

Креслення 1 - Схема-алгоритм роботи додатку

Креслення 2 - Алгоритм процесу покупки товару

Плакат 2 - Макет каталогу товарів

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ОС – операційна система

ПЗ – програмне забезпечення

БД – база даних

JSON – JavaScript Object Notation

URI – Uniform Resource Identifier

ВСТУП

Магазини з продуктами є затребуваними для користувачів будь-якого віку. Функція доставки робить подібні заклади більш зручними та популярними серед клієнтів. Нажаль, в Україні не так багато магазинів, які мають зручний додаток з можливістю зробити замовлення для доставки на якусь адресу. А ті бренди, які врахували дану функцію користуються сторонніми сервісами, такими як Bolt, Glovo та інші кур'єри. Ціна в цьому випадку є завищеною. Розробка мобільного додатку для магазину з доставкою є актуальною темою, адже це дозволить підвищити клієнтообіг. Також дана функція надасть змогу користувачам не витратити зайвий час на перебування у чергах, а просто зробити замовлення та дочекатися прибуття кур'єра.

В якості операційної системи для мобільного додатку була обрана платформа Android. Цей вибір аргументується популярністю вищеназваної ОС. Судячи зі статистики сервісу Statcounter, в березні 2023 року Android займає приблизно 71% світового ринку та близько 73% українського. Пристрої з цією операційною системою включають доступніші за ціною моделі, що значно підвищує аудиторію магазину. Також не варто недооцінювати велику інтеграцію з сервісами Google, такими як Gmail, Google Drive, Google Maps та інші, що робить використання пристрою більш зручним. Більше того, Android є привабливішим для розробників, бо робота з iOS потребує жорсткішого контролю з боку Apple.

Мета дослідження – створити зручний додаток на платформі Android з унікальними функціями для магазину продуктів з доставкою. Попередня назва – Monosell.

Завдання дослідження – розробка програмного забезпечення на платформі Android для магазину продуктів з функцією доставки без посередників та кількома унікальними можливостями.

1 АНАЛІЗ РИНКУ

Android платформа це популярна операційна система для мобільних пристроїв, розробником якої є компанія Google. Враховуючи статистику, на кінець 2022 року користувачами даної ОС є 75% власників смартфонів [1]. Однією з основних причин популярності Android є його відкрита архітектура, яка дозволяє розробникам створювати додатки та адаптувати їх. Це відкриває великі можливості для магазинів та торгових площадок, які можуть використовувати вищевказану ОС для розробки програм, що забезпечують доступ до своїх продуктів та послуг через мобільні пристрої.

Реалізація застосунку для магазину з доставкою потребує аналізу всіх схожих продуктів на ринку. Це необхідно для того, аби врахувати всі недоліки та переваги кожної програми. Для прикладу можна взяти такі мережі магазинів як Rozetka, АТБ-Маркет, Сільпо, COMFY, VARUS.

1.1 Огляд наявних рішень в галузі магазинів з доставкою на платформі Android

У наш час, дуже популярним стало використання мобільних додатків для замовлення та доставки продуктів харчування. На платформі Android існує безліч додатків, які надають такі послуги, огляд найпопулярніших з них наведено нижче.

«Rozetka» - один з найпопулярніших інтернет-магазинів в Україні. Додаток дозволяє користувачам замовляти різноманітні товари для дому, включаючи продукти харчування, техніку, меблі та багато іншого. Клієнти можуть вибирати спосіб доставки та оплати, а також відстежувати статус своїх замовлень. Доставка виконується від самого бренду. Варто зазначити зручний інтерфейс та

пропрацьований функціонал застосунку. Недоліки: замовлення будуть доставлятися з Києва [2].

Zakaz.ua - додаток, який дозволяє замовляти продукти харчування з магазинів. Користувачі можуть вибирати товари зі списку доступних магазинів, додавати їх до кошика та здійснювати оплату онлайн. Zakaz.ua пропонує різні способи доставки, включаючи доставку до дому та самовивіз. Крім того, додаток пропонує можливість зберігати списки замовлень та користуватися промокодами. Недоліки: довгий час очікування, через співпрацю з кількома магазинами у служби доставки немає можливості перевірити якість товару, тому трапляються випадки, коли покупцю привозять прострочені продукти [3].

«АТБ-Маркет» - це українська мережа супермаркетів, яка пропонує різноманітний асортимент товарів за доступними цінами. У «АТБ» використовуються сучасні технології для зручності покупців. Вони пропонують безкоштовну доставку замовлень від певної суми, а також можливість замовити товари онлайн на офіційному сайті мережі. Однак основним недоліком є відсутність актуального мобільного додатку, здатного задовольнити всі потреби споживачів. Тобто єдина функція доставки – через замовлення на сайті [4].

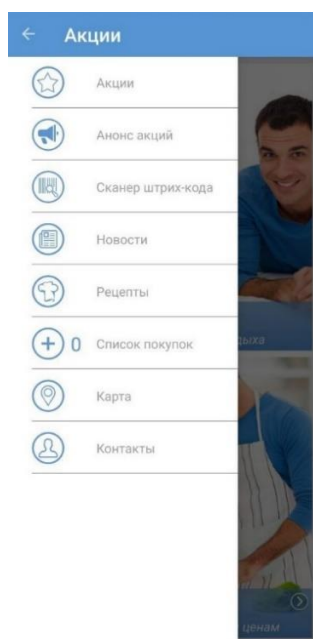


Рисунок 1.1 – Застосунок мережі магазинів «АТБ» не отримував оновлення з лютого 2022 року

«Сільпо» - мережа магазинів, яка створила свій додаток для мобільних пристроїв. Основними перевагами є зручність, приємний інтерфейс, бонуси для користувачів. Доставка реалізується службою самого магазину та кількома іншими компаніями. Головним недоліком є довге очікування прибуття замовлення [5].

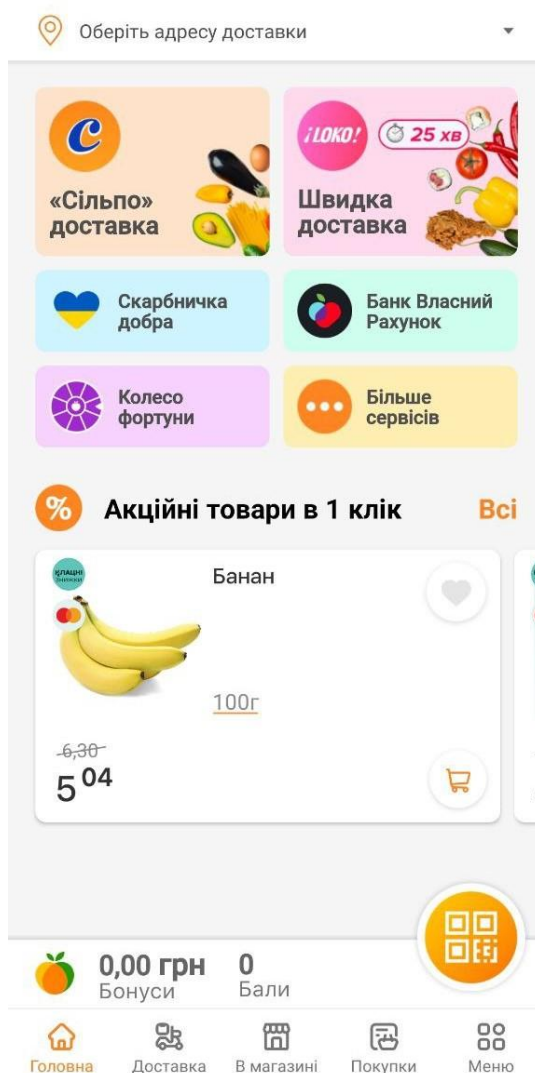


Рисунок 1.2 – Інтерфейс мобільного застосунку мережі супермаркетів «Сільпо»

«Varus» - мережа супермаркетів, яка пропонує своїм клієнтам зручний мобільний додаток, з функцією збереження списку покупок, сканером штрих-кодів, бонусами за купівлю та різноманіттям товарів. Нововведення Scan&Go дозволяє покупцям не витрачати час, стоячи у черзі, а відсканувати штрих-коди товарів та оплатити їх у касі самообслуговування. Недоліків при користуванні не виявлено.

«Fozzy» - мережа гіпермаркетів, яка діє в Києві, Дніпрі, Одесі, Житомирі та кількох інших містах України. Реалізована функція доставки на будь-яку адресу у межах одного населеного пункту, є мобільний додаток, а також можливість безкоштовного транспортування у разі замовлення вартістю більше ніж 250 гривень. Недоліком є відсутність доставки до населених пунктів в області та невеликий список міст, де працює гіпермаркет [6].

Супермаркет «Novus» діє виключно на території Києва, Тернополя, Рівного та київської області. Основними перевагами є можливість замовлення доставки продуктів з магазину, а головним недоліком є відсутність супермаркетів у інших містах України.

«METRO» - міжнародна мережа оптових магазинів для бізнесу зі штаб-квартирою в Німеччині. В Україні «METRO» працює з 2003 року і пропонує широкий асортимент продуктів харчування, товарів для готелів, ресторанів та кафе, а також інших виробів та послуг. Крім того, у цієї мережі магазинів є власна онлайн-платформа для замовлення продукції, включаючи доставку до офісу або складу клієнта. Додаток «METRO» для платформи Android дозволяє користувачам переглядати асортимент товарів, замовляти товари онлайн, переглядати історію замовлень та виконувати інші дії в мобільному форматі. Реалізована функція «METRO FAST LINE», що дозволяє самостійно сканувати штрих-коди на товарі та оплачувати все на касі самообслуговування.

«Foga» - українська мережа супермаркетів, що належить до групи компаній Fozzy Group, яка також включає супермаркети «Сільпо», гіпермаркети Fozzy та інші магазини. У мережі «Foga» представлений широкий асортимент продуктів харчування, товарів для дому, косметики, гігієнічних засобів та багато іншого.

У «Foga» є власна служба доставки, яка дозволяє клієнтам замовляти товари онлайн та отримувати їх безпосередньо до дому. Покупці можуть скористатися мобільним додатком для платформи Android, щоб зробити замовлення та відслідкувати статус доставки своєї покупки. Програма також дозволяє переглядати промо-акції, зберігати улюблені товари та створювати списки бажаного.

«Auchan» - це одна з найбільших мереж супермаркетів у світі, який працює в Україні. У цих магазинах клієнти можуть знайти широкий асортимент продуктів харчування, товарів для дому, електроніки, одягу та багато іншого.

«Auchan» має власну службу доставки, яка дозволяє покупцям замовляти товари онлайн та отримувати їх безпосередньо до дому. Клієнти можуть скористатися мобільним додатком для платформи Android, щоб зробити замовлення, відслідкувати статус доставки, скористатися сканером цін та переглянути актуальні промо-акції та знижки. Додаток також має функції, які дозволяють клієнтам створювати списки покупок та зберігати улюблені товари.

«Таврія В» є яскравим прикладом інтернет-супермаркету з доставкою. Цей магазин працює в південній частині України, в таких містах як Одеса, Миколаїв, Ізмаїл, тощо. Супермаркет пропонує велику кількість товарів, включаючи продукти харчування, товари для дому, побутову хімію, канцелярію та інше. Функція доставки реалізована, проте головним недоліком є те, що додаток на Android не оновлювався, тож користувачам мобільної платформи неможливо ним скористатися.

1.2 Аналіз основних технологій, фреймворків та інструментів, використовуваних для розробки мобільних додатків на платформі Android

Для розробки мобільних додатків на платформі Android використовуються різні технології, фреймворки та інструменти.

Почати необхідно з того, що створення програм на вищевказаній операційній системі можливе за допомогою кількох мов програмування, а саме:

- Java;
- Kotlin;
- C++;
- C#;

- JavaScript;
- Python.

Java до 2019 року була однією з основних мов програмування для розробки додатків на платформі Android. Android SDK (Software Development Kit) містить багато інструментів, які дозволяють розробляти додатки на Java. За допомогою цієї мови програмування, розробники можуть створювати різноманітні мобільні застосунки, включаючи ігри, програми для роботи з мультимедіа, соціальні мережі, торгові платформи та інші.

Один з головних принципів розробки додатків на Android з використанням Java - це паттерн проектування Model-View-Controller (MVC), який є архітектурним шаблоном програмного забезпечення, використовується для розробки веб- та десктоп-програм. В основі цього шаблону лежить розділення компонентів програми на три основних рівні: модель (Model), представлення (View) та контролер (Controller). Кожен рівень має свою функцію та взаємодіє з іншими пунктами шляхом передачі повідомлень.

Модель (Model) відповідає за управління даними додатку та його бізнес-логікою. Цей рівень забезпечує методи для доступу до даних та їх збереження, а також виконання операцій над ними.

Представлення (View) відображає дані моделі та відповідає за взаємодію з користувачем. Цей рівень відображає дані в зручному форматі та забезпечує можливість взаємодії з юзером.

Контролер (Controller) відповідає за обробку вхідних даних та взаємодію між моделлю та представленням. Цей рівень обробляє запити користувача та передає їх моделі для виконання необхідних дій, а також забезпечує оновлення представлення після змін даних.

MVC дозволяє розділити логіку додатку на окремі компоненти, що спрощує розробку та розширення програмного забезпечення. Він забезпечує більшу гнучкість управління даними та представленням, що дозволяє змінювати окремі компоненти без впливу на решту системи. Це робить код більш організованим та зрозумілим [7].

Загалом, Java є однією з найбільш популярних мов програмування для розробки додатків на платформі Android завдяки своїй простоті та зручності, а також багатій екосистемі інструментів та ресурсів, що підтримують її використання на даній платформі.

Kotlin - це молода мова програмування, яка стала офіційною для розробки додатків на платформі Android в 2019 році. Вона була розроблена компанією JetBrains, яка також стоїть за розробкою популярного інтегрованого середовища розробки IntelliJ IDEA.

Kotlin відрізняється від Java більш простою синтаксичною структурою, а також набором функціональних можливостей. Вона пропонує коротші і більш читабельні вирази коду, а також можливості для безпечного програмування завдяки системі перевірки на стадії компіляції, що дозволяє виявляти більше помилок в процесі розробки.

Також Kotlin підтримує взаємодію з Java, що дозволяє розробникам поступово переходити на нову мову без необхідності повністю переписувати код своїх додатків. Вона здобуває все більшу популярність серед розробників на платформі Android завдяки своїм перевагам та підтримкою від компанії Google [8].

Програмування C++ реалізується завдяки Android NDK (Native Development Kit) - інструменту, який дозволяє розробникам використовувати ці мови програмування для розробки додатків на платформі Android.

Подібний інструмент дозволяє розробникам писати код на мовах високого рівня, які компілюються в нативний код для роботи на мобільних пристроях, а значить програмісти отримують більшу ефективність та швидкість виконання, що особливо важливо для додатків з великим обчислювальним навантаженням, наприклад, ігор чи програм для обробки відео [9].

Проте, використання Android NDK та мови програмування C++ для розробки додатків на платформі Android може бути складнішим, ніж Java чи Kotlin. Це пов'язано з тим, що розробник повинен мати додаткові знання та навички роботи з мовою C++ та бібліотеками, що використовуються в Android NDK.

Мова програмування C# може бути використана для розробки додатків як на платформі Android, так і на iOS. Для цього треба використовувати Xamarin - фреймворк для створення мобільних програм, який дозволяє писати код на вищевказаній мові.

Xamarin забезпечує зручний спосіб для розробки кросплатформних додатків, а значить один і той же код можна використовувати для створення застосунків на різних платформах. Крім того, цей фреймворк надає доступ до бібліотек та інструментів, що дозволяють програмістам робити більш складні та продуктивні програми [10,11].

Хоча JavaScript не є найбільш поширеною мовою програмування для розробки додатків на платформі Android, проте за допомогою деяких інструментів та фреймворків, таких як Cordova, PhoneGap та NativeScript така функція можлива.

Ці інструменти дозволяють створювати гібридні додатки, які можуть бути запущені на платформі Android за допомогою веб-технологій, таких як HTML, CSS та JavaScript. Однак, їхнє використання може призвести до певних обмежень щодо швидкодії та можливостей програм, порівняно з нативними, розробленими на мовах Java, Kotlin або C++.

Отже, хоча JavaScript може використовуватися для створення додатків на платформі Android за допомогою гібридних фреймворків, вона не є найкращим вибором з точки зору продуктивності та можливостей.

Python також є мовою програмування, завдяки якій можлива розробка мобільних додатків, але її використання не є дуже популярним варіантом.

Існує кілька способів встановлення Python на Android, наприклад, використання сторонніх додатків, таких як Pydroid 3 або Termux, або встановлення спеціальних операційних систем, таких як KivyPie, які вже містять у собі цю мову програмування та додаткові бібліотеки.

Зазвичай Python використовується на Android для розробки скриптів, виконання обчислень або як допоміжний інструмент при створенні програм на інших мовах.

Знати мову програмування це найнеобхідніший аспект при створенні додатку, але не менш важливо мати середовище розробки. IDE (Integrated Development Environment) - це інтегроване середовище розробки програмного забезпечення, що об'єднує різні інструменти, такі як редактори, компілятори, відлагоджувачі, засоби побудови, засоби контролю версій та інші. Це дозволяє розробникам працювати з кодом, відлагоджувати програми, тестувати їх та збирати готові додатки, що забезпечує зручність та ефективність.

Android Studio - це офіційна інтегрована середовище розробки для платформи Android. Воно підтримується компанією Google та активною спільнотою розробників. Android Studio забезпечує повний набір інструментів для розробки, від редактора коду до симуляторів пристроїв та інструментів профілювання.

На офіційному сайті можна знайти багато матеріалів, включаючи документацію, посібники та відеоуроки для новачків та досвідчених розробників. Також там є посилання на останні версії Android Studio та його складових, таких як Android SDK та Gradle.

Це середовище забезпечує підтримку різних мов програмування, включаючи Java, Kotlin та C++. Воно також надає можливість розробляти додатки для різних версій Android, від старих до найновіших.

Однією з найбільш цінних особливостей Android Studio є його інтеграція з платформою Google Cloud, що дозволяє розробникам швидко та легко підключати до своїх додатків такі сервіси, як Firebase та Google Maps.

Також Android Studio надає інструменти для побудови та тестування додатків. Зокрема, це середовище містить такі інструменти, як Android Virtual Device Manager для налаштування віртуальних пристроїв Android та Android Debug Bridge для зв'язку з фізичними пристроями [12].

Отже, Android Studio - це потужне та зручне середовище розробки, яке забезпечує широкі можливості та інструменти для створення як простих, так і складних додатків.

Eclipse - це інтегроване середовище розробки, яке було створене на початку 2000-х років, і призначене для програмістів, що працюють з різноманітними

мовами програмування. Ця платформа належить фонду Eclipse Foundation, який розробляє та підтримує цей проект.

Eclipse був створений в 2001 році в рамках проекту VisualAge в IBM. Ідея полягала в тому, щоб створити інструментальну платформу, яка дозволяла б розробляти програмне забезпечення на різних мовах програмування.

У 2004 році Eclipse Foundation була створена з метою розвитку вищевказаного середовища та забезпечення його безкоштовним розповсюдженням. З того часу, проект постійно розвивався та набував все більшої популярності серед розробників програмного забезпечення.

Eclipse має широкий спектр функціональності для програмування різними мовами, такими як Java, C/C++, Python, PHP, та інші. На даний момент, платформа має понад 2500 доступних плагінів, що дозволяють збільшити функціональність та покращити продуктивність розробки. За допомогою плагінів, розробники можуть створювати власні інструменти та додавати їх до середовища.

Крім того, Eclipse підтримує плагіни для підключення до інших інструментів, таких як системи контролю версій (наприклад, Git), системи збірки (наприклад, Maven та Gradle), сервери додатків та багато іншого. Всі ці інструменти допомагають розробникам покращити продуктивність і ефективність роботи [13].

Також важливо зазначити, що Eclipse має велику спільноту розробників, яка активно працює над покращенням та розширенням функціональності цієї IDE. Вона забезпечує безкоштовну та відкриту розробку, що дозволяє розробникам використовувати та змінювати код IDE згідно з їхніми потребами.

Розбіжності між цими двома середовищами:

- Eclipse підтримує більше мов програмування, включаючи Java, C++, PHP, Python та багато інших, тоді як Android Studio спеціалізується на розробці додатків на мові Java та Kotlin;
- Android Studio має більш зручний та простий інтерфейс користувача порівняно з Eclipse;

- Android Studio має вбудовані інструменти для аналізу коду, що дозволяють розробникам виявляти та виправляти помилки в коді. Eclipse також має інструменти для аналізу коду, але вони менш розвинені порівняно з Android Studio;

- Android Studio працює швидше та продуктивніше, ніж Eclipse. Це пов'язано з тим, що Android Studio була створена спеціально для розробки додатків для Android, тоді як Eclipse має більше функціональності та можливостей, що призводить до сповільнення роботи в деяких випадках;

- Android Studio є офіційною IDE для розробки додатків для платформи Android, тому вона має більшу підтримку та оновлення. Eclipse більше не оновлюється, і підтримка від компанії-розробника завершена.

IntelliJ IDEA є інтегрованим середовищем розробки для програмістів. Воно розроблене компанією JetBrains і підтримує створення додатків на різних мовах програмування, включаючи Java, Kotlin, Python, JavaScript, інші.

IntelliJ IDEA надає широкі можливості для розробки програм на платформі Android. Для початку, це середовище дозволяє розробникам використовувати мову програмування Kotlin для створення додатків. Крім того, IntelliJ IDEA підтримує інтеграцію з платформою Android SDK, що дозволяє розробникам програмувати застосунки, які працюють з багатьма компонентами платформи, такими як Activity, Service, Content Provider і BroadcastReceiver.

Окрім базових функцій, таких як автодоповнення коду, підсвічування синтаксису, побудова та налагодження програми, IntelliJ IDEA також має деякі унікальні функції, які роблять його більш ефективним для розробки додатків на платформі Android. Наприклад, він має вбудований дизайнер інтерфейсу користувача, що дозволяє створювати інтерфейс додатку безпосередньо в середовищі розробки. Крім того, IntelliJ IDEA має інструменти для оптимізації роботи додатків та детектування помилок у коді [14].

Visual Studio - це інтегроване середовище розробки від компанії Microsoft, яке підтримує створення програмного забезпечення для різних платформ, включаючи Android. Для програмування на Android у IDE використовують фреймворк Xamarin, який дає змогу писати код на мові програмування C# та

використовувати .NET-бібліотеки для створення переносних додатків для різних пристроїв.

Завдяки використанню Xamarin, розробники можуть зосередитись на розробці функціоналу додатку, а не на проблемах сумісності з різними версіями ОС або пристроями. Більше того, програмісти можуть працювати з інструментами Visual Studio, такими як IntelliSense та відлагоджувач, для підвищення ефективності своєї роботи.

Окрім цього, Visual Studio має широкі можливості для роботи з системою контролю версій, такі як Git, SVN та TFS, що дозволяє легко співпрацювати з іншими розробниками та зберігати історію змін коду [15].

NetBeans - це середовище розробки з відкритим вихідним кодом для розробки програмного забезпечення на різних мовах програмування, включаючи Java, PHP, C++ та інші. Також воно підтримує розробку додатків на платформі Android за допомогою спеціального плагіна.

Основні особливості NetBeans для розробки додатків на платформі Android:

- підтримка Java та Kotlin;
- вбудований емулятор Android, що дозволяє тестувати додаток без підключення до фізичного пристрою;
- підтримка збірки та підпису додатків для Android;
- можливість створення різних версій додатків залежно від SDK та рівня API;
- підтримка розширення за допомогою плагінів, що дозволяє додатково налаштувати робоче середовище під власні потреби;
- однією з переваг NetBeans є його простота використання, що робить його доступним для початківців;
- це середовище має добре розвинену спільноту користувачів, яка надає підтримку та забезпечує змістовну документацію.

Проте, на відміну від Android Studio чи IntelliJ IDEA, NetBeans має менше інструментів для розробки додатків на платформі Android. Однак, для розробки додатків на інших платформах, наприклад, Java чи PHP, NetBeans є одним з найбільш популярних та ефективних робочих середовищ [16].

Фреймворк - це набір програмних компонентів, бібліотек, правил та інструкцій, які забезпечують розробникам загальну архітектуру для створення додатків або систем. Він може включати в себе інструменти для розробки, тестування та відлагодження, а також інші компоненти, які полегшують програмування. Використання фреймворку дозволяє розробникам швидше створювати програмне забезпечення, зменшуючи кількість коду, що потрібен для написання, та полегшуючи підтримку програмного забезпечення.

Фреймворки часто використовують для створення застосунків на різних платформах, таких як веб-сайти, мобільні додатки, настільні додатки та інші. Їх можна використовувати для різних цілей, наприклад розробка інтерфейсу користувача, роботи з базами даних, серверною частиною додатка, підключення до сторонніх сервісів та інше. Використання фреймворку може дозволити розробникам програмного забезпечення ефективніше використовувати свій час та зосередитися на розробці нового функціоналу замість вирішення технічних проблем.

React Native - це відкритий фреймворк для розробки мобільних додатків, який створений компанією Facebook. Він дозволяє програмувати мовою JavaScript та використовувати бібліотеку React.

Однією з основних переваг React Native є те, що він дозволяє розробляти гібридні додатки, які працюють швидко та мають високу продуктивність. Крім того, він має велику спільноту розробників, що активно підтримують та розширюють фреймворк.

Для розробки додатків з використанням React Native в Android Studio необхідно встановити спеціальний плагін, який забезпечує підтримку фреймворку в IDE. Далі можна створювати нові проекти та розробляти додатки з використанням JavaScript та React [17].

Xamarin - це відкритий фреймворк для розробки мобільних додатків, який дозволяє створювати кросплатформні застосунки для Android, iOS та Windows з використанням мов програмування C# та .NET.

Xamarin дозволяє використовувати спільний код для різних платформ, що зменшує затрати часу та зусиль на розробку. Фреймворк включає в себе різні інструменти для програмування, такі як Xamarin Studio та Visual Studio, які підтримують інтеграцію з різними сервісами, наприклад Azure та HockeyApp.

Flutter - це фреймворк для розробки мобільних додатків для Android та інших платформ, який був створений компанією Google. Його основою є мова програмування Dart, яка була розроблена тим же самим брендом. Flutter використовує власну систему віджетів, яка дозволяє швидко створювати красиві та динамічні інтерфейси користувача.

Основні переваги Flutter перед іншими фреймворками:

- швидкість розробки. Flutter дозволяє розробляти додатки для Android та інших платформ швидко та ефективно завдяки власній системі віджетів та багатофункціональному інструментарію.

- кросплатформеність. Flutter дозволяє створювати додатки для Android, iOS, веб-браузерів та інших платформ з використанням одного коду, що економить час та зусилля розробників.

- гнучкість та налаштування. Flutter має велику кількість готових віджетів та інструментів, що дозволяє розробникам налаштувати додаток під свої потреби.

- висока продуктивність. Flutter використовує вбудований двигун Skia для відображення графіки, що дозволяє досягати високої продуктивності та роботи на різних платформах.

- активна спільнота. Flutter має велику активну спільноту розробників, що дозволяє швидко знайти відповіді на будь-які запитання та проблеми, які можуть виникнути під час розробки [18].

Ionic - це відкритий фреймворк для розробки гібридних мобільних додатків, що дозволяє використовувати веб-технології, такі як HTML, CSS і JavaScript, для створення додатків для різних платформ, включаючи Android.

Розглянемо основні переваги Ionic:

- кросплатформеність: Ionic дозволяє створювати додатки, які працюють на різних платформах, включаючи Android, iOS, та Windows Phone.

- можливості налаштування: Іонік дозволяє налаштовувати різні елементи дизайну, такі як кольори, фони, та інші.

- підтримка Cordova: Іонік використовує фреймворк Apache Cordova, який дозволяє додаткам використовувати різні функції пристроїв, такі як камера, GPS, та інші.

- швидкість розробки: Іонік має багато готових компонентів, які можна використовувати для створення додатків, що дозволяє зменшити час розробки.

- спільнота розробників: Іонік має велику спільноту розробників, яка надає підтримку та розв'язує проблеми, що можуть виникнути під час розробки [19].

PhoneGap - це відкритий фреймворк для розробки гібридних мобільних додатків, який дозволяє використовувати технології веб-розробки, такі як HTML, CSS та JavaScript, для створення кросплатформних застосунків для різних мобільних платформ, включаючи Android.

PhoneGap був розроблений Adobe, але потім був переданий Apache Software Foundation і перейменований в Apache Cordova. Цей фреймворк дозволяє розробникам створювати мобільні додатки, що мають доступ до різноманітних функцій пристроїв, таких як камера, геолокація, контакти та інші.

Основні переваги PhoneGap включають:

- кросплатформеність. додатки, розроблені з використанням PhoneGap, можуть бути запущені на різних мобільних платформах, включаючи Android, iOS, Windows та інші;

- відкритий код: PhoneGap є відкритим фреймворком, тому його можна безкоштовно використовувати та модифікувати;

- легкість використання. PhoneGap дозволяє використовувати веб-технології для розробки мобільних додатків, що робить його легким для вивчення та використання;

- широкий спектр плагінів: PhoneGap має велику кількість плагінів, що дозволяють розробникам отримувати доступ до різноманітних функцій пристроїв, таких як камера, геолокація, контакти та інші;

- підтримка спільноти. PhoneGap має велику спільноту розробників, яка надає підтримку та розвиває фреймворк.

Існують деякі недоліки використання PhoneGap, які можуть вплинути на розробку мобільних додатків. Розглянемо основні з них:

- відсутність доступу до функціоналу платформи. хоча PhoneGap дозволяє створювати мобільні додатки для різних платформ, він не надає повний доступ до функціоналу платформи, такого як датчики пристрою, GPS та інші;

- обмежена продуктивність. PhoneGap використовує веб-технології для створення мобільних додатків, що може призвести до обмежень в продуктивності додатку, особливо для складних додатків з великою функціональністю;

- проблеми з відтворенням. PhoneGap може мати проблеми з відтворенням деяких функцій на різних платформах, що може призвести до помилок в додатках;

- залежність від сторонніх розробників. PhoneGap використовує різноманітні плагіни, які створюються сторонніми розробниками, що може призвести до залежності від стороннього програмного забезпечення і вразливостей в ньому.

Apache Cordova - це відкритий фреймворк для розробки гібридних мобільних додатків на основі веб-технологій, таких як HTML, CSS та JavaScript. Cordova дозволяє створювати додатки, що можуть працювати як на Android, так і на інших платформах, таких як iOS та Windows.

Cordova дозволяє програмувати додатки для різних мобільних платформ, що зменшує час та зусилля, необхідні для розробки та тестування застосунків для кожної окремої платформи. Оскільки це відкритий фреймворк, то розробники можуть налаштувати його під свої потреби та внести зміни в код за необхідності. Cordova використовує відомі веб-технології, такі як HTML, CSS та JavaScript, що дозволяє веб-розробникам з легкістю перейти до створення мобільних додатків.

На жаль, Cordova не є ідеальним фреймворком, тож варто розглянути деякі недоліки, а саме:

- швидкість - гібридні додатки, розроблені за допомогою Cordova, можуть бути повільнішими в порівнянні з нативними додатками;

- обмеження функціональності - Cordova може мати обмеження щодо доступу до певних функцій, які можуть бути доступні тільки в застосунках;

- залежність від інших бібліотек - для деяких функцій може бути необхідно встановлювати додаткові бібліотеки, що може збільшити розмір програми та знизити її продуктивність [20].

NativeScript - це відкритий фреймворк для розробки нативних мобільних додатків для платформ Android та iOS з використанням мов програмування JavaScript або TypeScript, а також XML-подібної мови маркування. Одна з його особливостей - це можливість повторного використання коду між платформами, що дозволяє прискорити процес та зменшити затрати.

NativeScript надає доступ до нативних API, дозволяючи розробникам створювати програми, що мають повний доступ до функціоналу пристрою. Крім того, він робить можливим використання популярних фреймворків, таких як Angular, Vue.js та React, для розробки інтерфейсу користувача [21].

Одним з недоліків NativeScript є його складність та обмежена документація порівняно з іншими фреймворками. Також може бути важко розробляти складні анімації, оскільки вони можуть бути повільними.

Sencha Touch - це HTML5-фреймворк для мобільних додатків, який дозволяє розробляти кросплатформні застосунки для різних платформ, включаючи Android. Він використовує мову програмування JavaScript та різноманітні HTML5-технології, такі як CSS3 та SVG, для створення високопродуктивних інтерфейсів для користувачів.

Основні переваги Sencha Touch:

- кросплатформеність: Sencha Touch дозволяє розробляти додатки для різних мобільних платформ, включаючи Android, iOS та BlackBerry, за допомогою єдиного коду;

- компоненти і шаблони: фреймворк має багатий набір компонентів та шаблонів, що дозволяє розробникам швидко створювати різноманітні елементи інтерфейсу користувача;

- висока продуктивність: Sencha Touch забезпечує високу продуктивність завдяки оптимізованій роботі з DOM та CSS;

- підтримка плагінів: фреймворк підтримує різноманітні плагіни, що дозволяє розширювати його можливості [22].

Однак використовувати фреймворк Sencha Touch не так просто як здається, оскільки він має високі вимоги до ресурсів, отже не кожен пристрій зможе забезпечити максимальну продуктивність. Також, варто зазначити, що Sencha Touch не безкоштовний, тому це може обмежити його ефективність. Вивчення Sencha Touch може зайняти більше часу, через те, що він має власну архітектуру та систему компонентів.

jQuery Mobile - це фреймворк для розробки мобільних додатків з використанням HTML, CSS та JavaScript. Він надає можливість швидко створювати застосунки, які працюють на більшості популярних мобільних платформ, включаючи Android.

Основні переваги jQuery Mobile:

- легко вивчати та використовувати для розробки мобільних додатків;
- надає можливість створювати мобільні додатки, які працюють на більшості популярних мобільних платформ, включаючи Android;
- підтримує різну локалізацію;
- надає велику кількість готових елементів, які можна використовувати у своїх додатках;
- має документацію та підтримку від спільноти.

Основні недоліки jQuery Mobile:

- не забезпечує таку високу продуктивність та швидкість роботи, яку можна отримати з використанням нативних рішень;
- не має такої гнучкості, яку можуть надати інші фреймворки, наприклад, React Native або Flutter;
- може бути складніше використовувати для розробки складних додатків, які потребують більшої контролю над роботою мобільного додатку.

Оскільки найбільш популярним рішенням для розробки мобільних додатків є Android Studio, то варто розглянути додаткову функцію, яку надає Google, а саме – технологію Firebase.

Firebase – це платформа для розробки мобільних і веб-додатків, яка пропонує розробникам широкий спектр сервісів, що допомагають розробляти високоякісні та функціональні додатки. Вона була заснована у 2011 році і придбана компанією Google у 2014. Платформа стала надзвичайно популярною у розробників завдяки своїм інструментам для роботи з базами даних, хмарними сервісами та інструментами аналітики.

Основні компоненти Firebase включають наступне:

- Realtime Database;
- Cloud Firestore;
- Authentication;
- Cloud Functions;
- Cloud Storage;
- Cloud Messaging;
- Analytics;
- Test Lab;
- Crashlytics;
- Performance Monitoring.

Realtime Database - сервіс для збереження та синхронізації даних в режимі реального часу між всіма підключеними клієнтами.

Cloud Firestore – це хмарна база даних з підтримкою документів та колекцій, що забезпечує багатofункціональні запити та можливість реалізації складних операцій.

Authentication - сервіс для аутентифікації користувачів за допомогою електронної пошти, пароля, номера телефону, облікового запису Google та інших методів.

Cloud Functions - сервіс для реалізації функцій на стороні сервера з використанням середовища виконання Node.js.

Cloud Storage - хмарний сервіс для зберігання та керування мультимедійними файлами.

Cloud Messaging - сервіс для надсилання повідомлень на пристрої користувача.

Analytics - інструмент для збору та аналізу даних про використання додатків.

Test Lab - сервіс для тестування додатків на різних пристроях та в середовищах.

Crashlytics - інструмент для виявлення та відстеження помилок та відмов додатків.

Performance Monitoring - інструмент для вимірювання продуктивності додатків [23].

1.3 Виявлення потреб та очікувань користувачів від магазину з доставкою на платформі Android

Виявлення потреб та очікувань є важливим кроком при розробці певного додатку, адже маючи розуміння, чого саме бажають клієнти, можна створити той продукт, який буде мати вищий попит на ринку, а значить стане чудовим об'єктом для фінансування і підвищення клієнтообігу будь-якого магазину.

Найкращий спосіб виявлення потреб та очікувань це аналіз всіх технологій, використовуваних конкурентами, проведення опитувань серед користувачів, обов'язкове тестування продукції на наявність помилок. Після всіх вищевказаних процедур було виявлено, що основними потребами користувачів до магазинів з доставкою на платформі Android є:

- зручність і швидкість замовлення: користувачі хочуть, щоб процес замовлення був простим та зрозумілим;
- безпечність: їхні особисті дані та оплата будуть захищені від зловмисників;

- широкий вибір продуктів: великий асортимент товарів, щоб знайти саме те, що їм потрібно;

- якість продуктів: користувачі хочуть високої якості продуктів, які вони замовляють;

- доставка: клієнти очікують швидкої та надійної доставки, а також можливості відстежувати статус доставки;

- ціна: доступна ціна, отримання знижок та спеціальних пропозицій;

- корисність та зручність: користувачі хочуть, щоб магазин з доставкою був корисним та зручним для використання, з можливістю переглядати відгуки та рейтинги продуктів, зберігати свої улюблені продукти, тощо.

Нові функції, які покупці бажають бачити у мобільному додатку для магазину з доставкою, можуть залежати від конкретного магазину та його асортименту, а також від потреб та очікувань самого користувача. Проте, основні функції, які зазвичай бажають бачити користувачі, включають:

- зручний та швидкий пошук товарів;

- можливість здійснювати покупки зі швидкою реєстрацією за допомогою соціальних мереж;

- можливість зберігати товари у власних списках бажань або кошику та синхронізувати їх між різними пристроями;

- детальна інформація про товари, включаючи опис, фотографії, характеристики, відгуки користувачів та ціни;

- зручні способи оплати та доставки;

- можливість відстежувати статус замовлення та доставки;

- програма лояльності та можливість отримувати знижки та спеціальні пропозиції.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ФУНКЦІОНАЛУ ЗАСТОСУНКУ

Мобільний застосунок для магазину має мати просту архітектуру, для того щоб приваблювати користувачів. Перша ступінь в проектуванні архітектури – розробка форми реєстрації для користувача. Повинно бути три різні види створення облікового запису:

- для звичайного клієнта. Цей вид передбачає перегляд каталогу товарів, створення списку улюблених продуктів, занесення їх у торговий кошик з можливістю подальшого замовлення доставки. Також є функція замовлення окремо приготовлених страв, або швидка покупка всіх необхідних інгредієнтів одним кліком;

- для кур'єра. Цей спосіб створення облікового запису в базі даних передбачає ті самі функції, що і для клієнта, проте, при переході на вкладку «Доставка», перед користувачем з'являється карта, на якій відображені всі найближчі замовлення, за рахунок чого кур'єр сам обирає зручний і швидкий маршрут;

- для адміністратора. Такий спосіб реєстрації надає привілеї, завдяки яким користувач може змінювати опис товару, додавати нові продукти або видаляти вже існуючі.

Визначення ролі користувача проводиться завдяки вводу особливого зашифрованого коду і перевірки його з наявними в базі даних. Функціонал повинен надавати всі можливості для перегляду товарів, їх купівлі та зручного оформлення доставки.

2.1 Визначення вимог до функціоналу застосунку для магазину з функцією доставки на основі потреб користувачів

Визначення вимог до функціоналу застосунку є важливим етапом розробки будь-якого програмного забезпечення, включаючи мобільні додатки для магазинів з доставкою. Для цього потрібно провести аналіз потреб користувачів.

У випадку мобільного додатка для магазину з доставкою, основні вимоги до функціоналу включають:

- наявність реєстрації та авторизації користувачів. Цей пункт є очевидним, адже завдяки ньому клієнти магазину можуть робити особисті налаштування, проводити верифікацію замовлення, зберігати улюблені продукти;
- перегляд каталогу товарів, включаючи фото, опис, ціни, кількість на складі, вагу продукції та абсолютно унікальним для цього додатку є наявність інформації про вміст алергенів;
- додавання товарів у кошик та у обране, а також можливість видалення їх звідти;
- можливість оформлення замовлення для доставки на певну адресу;
- оплата замовлення зручним способом;
- заповнення особистого профілю з даними про ім'я та прізвище, адресу електронної пошти для отримання різноманітних промокодів. Також варто зазначити, що в особистому кабінеті застосунку для магазину «MonoSell» є можливість обрання продуктів, вітамінів, або елементів, які викликають у користувача алергічну реакцію, для виключення цих продуктів з каталогу товарів;
- можливість зворотного зв'язку з підтримкою магазину;

Всі побажання та вимоги користувачів реалізуються завдяки сервісам Firebase. Наприклад, реєстрація та логін можуть бути зроблені за допомогою Firebase Authentication. Проте такий спосіб є не дуже зручним для збереження даних про клієнтів, тому в розробленому застосунку використовується внесення

логіну та пароллю до БД Firebase Realtime Database. Електронна пошта стає унікальним ідентифікатором, адже мати дві однакові скриньки фізично неможливо.

Каталог товарів, як і скринька покупок та обране у користувача зберігається у БД Firebase Realtime Database. Ця інформація представлена у форматі JSON, що наглядно показано на рисунку 2.1

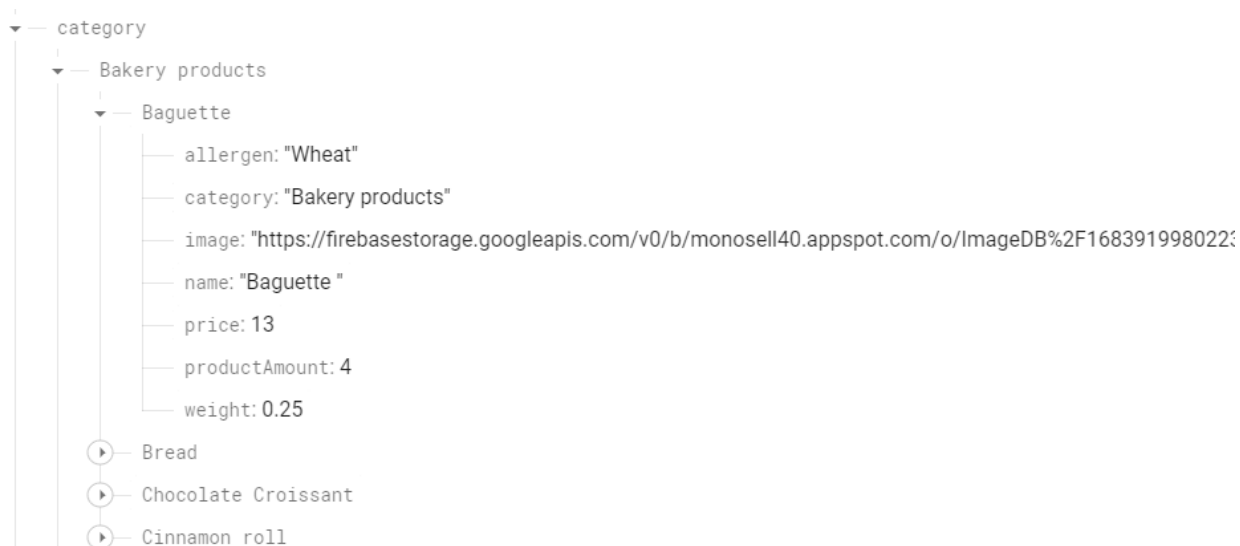


Рисунок 2.1 – Структура даних у БД Firebase Realtime Database

JSON - це формат обміну даними, який використовується для передачі структурованих даних між різними програмними системами. Він базується на синтаксисі об'єктів JavaScript і складається з пар ключ-значення, які групуються в об'єкти і масиви. Формат JSON є легким, текстовим і простим у використанні, що робить його ідеальним для обміну даними між веб-серверами та клієнтами [24].

Для зручного відображення використовується бібліотека Picasso.

Бібліотека Picasso є однією з найпопулярніших бібліотек для роботи з зображеннями на платформі Android. Вона надає простий та зручний інтерфейс для завантаження фото з мережі Інтернет або з локального сховища, обробки та відображення їх у відповідних ImageView-елементах у мобільному додатку.

Бібліотека підтримує кешування зображень, що дозволяє зменшити кількість запитів до сервера та забезпечити швидке відображення навіть при поганому

з'єднанні з Інтернетом. Крім того, вона підтримує обрізання, зміну розміру, обертання та інші операції для редагування фото [25].

2.2 Розробка архітектури застосунку, включаючи вибір архітектурних паттернів, розподіл функціоналу на компоненти та опис взаємодії між ними

Існує багато різних архітектур застосунків, кожна з яких має свої переваги та недоліки.

Model-View-Controller (MVC) - це архітектура, де дані, логіка та представлення поділені на три різні компоненти. Модель представляє дані та бізнес-логіку, Представлення відповідає за відображення даних та користувацький інтерфейс, а Контролер обробляє запити користувача та взаємодіє з Моделлю та Представленням.

Model-View-Presenter (MVP) - це модифікація MVC, де Презентер відповідає за взаємодію між Моделлю та Представленням, забезпечуючи також розділення логіки відображення та бізнес-логіки.

Model-View-ViewModel (MVVM) - це архітектура, яка базується на MVP та використовує прив'язку даних. ViewModel відповідає за зберігання та управління даними, які відображаються в Представленні.

Clean Architecture - це архітектура, що розділяє програмне забезпечення на рівні залежності та гарантує, що бізнес-логіка є незалежною від інфраструктури та зовнішніх факторів.

Domain-Driven Design (DDD) - це архітектурний підхід, що ставить бізнес-логіку на перший план, забезпечуючи розділення бізнес-логіки від інфраструктури та інших технічних аспектів.

Event-Driven Architecture (EDA) - це архітектура, що базується на обміні повідомленнями між компонентами системи. Кожен компонент може генерувати та споживати події, які відбуваються в системі [26].

Вибір архітектури застосунку залежить від багатьох факторів, таких як тип застосунку, його складність, масштаб, вимоги до безпеки, швидкодії, можливості розширення, доступність ресурсів і знань команди розробників та інші. Наприклад, якщо застосунок має бути простим і не великим, можна використати MVP або MVVM архітектури, які забезпечують достатньо хорошу швидкодію та легкість розробки. Але якщо мова йде про складну багатокористувацьку програму зі складним функціоналом, то, наприклад, варто звернути увагу на Clean Architecture або Hexagonal Architecture, щоб забезпечити високу масштабованість та можливість розширення. Також важливо враховувати відповідність архітектури вимогам безпеки та надійності, які можуть бути критичними для певних типів ПЗ.

Оскільки застосунок для магазину MonoSell зараз знаходиться на початковій стадії розробки і ще не має багато користувачів, то була обрана архітектура Model-View-Controller.

В якості моделі в додатку MonoSell використовуються класи User, Dish та Product.

У класі User для представлення інформації про нього використовується тип даних String, а також інтерфейс List. За допомогою першого зберігаються ім'я, електронна пошта, пароль та роль клієнта. Інтерфейс List використовується для збереження улюблених продуктів, кошика купівель та алергенів.

У класі Product та Dish використовується тип даних String, double, int. Перший тип даних для збереження категорії, назви, алерген та URI зображення. Double необхідний для збереження ціни та ваги продукту. Int представляє кількість продукції на складі.

У якості View використовуються різні класи Adapter та ViewHolder. Вони необхідні для представлення каталогу продукції, списку обраного, кошика куплених товарів, а також гарячої продукції.

Контролер реалізований за допомогою класів AddProductActivity, ProductDetailInfActivity, CustomerMapActivity, DeliveryMapsActivity, FavouriteUserActivity, HotDishesActivity, HotDishesInfo, LoginActivity, MainActivity, RegisterActivity, ShoppingCart, ShopActivity, ActivityShopAdmin. Кожен з цих класів

відповідає за редагування, додавання або видалення інформації і взаємодію між користувачем та елементами архітектури Model і View. Весь дизайн створений за допомогою xml-файлів.

2.3 Розробка дизайну інтерфейсу користувача застосунку, включаючи розробку макетів, вибір кольорової гами, шрифтів та іконок

Оскільки додаток розрахований на велику кількість споживачів, адже магазин має бути популярним серед клієнтів, то він повинен мати простий та інтуїтивно зрозумілий інтерфейс.

Першочерговим завданням при розробці був вибір гами, оскільки колір напряму впливає на дизайн і привабливість застосунку. Після перегляду кількох можливих варіантів оформлення, та аналізу поєднань кольору у додатках конкурентів, було прийнято рішення про використання зеленого, білого та їх відтінків у якості основи для композиції.

Основні причини вибору такої кольорової гами:

- асоціації з природою і свіжістю: зелений колір часто асоціюється з природою, зростанням і свіжістю. Він може створювати атмосферу комфорту, спокою і природності, що є привабливим для користувачів магазину;

- позитивний психологічний вплив: науково доведено, що кольори чинять психологічний вплив на людей, і зелений вважається заспокійливим і розслаблюючим. Він може створювати відчуття гармонії і стимулювати позитивні емоції у користувачів програми;

- відмінність від конкурентів: більшість магазинів-конкурентів використовують інші колірні гами, тому зелений колір може допомогти додатку виділитися і запам'ятатися користувачам.

Вибір шрифту був наданий користувачам, адже у додатку використовується стандартний шрифтовий набір, обраний системою Android смартфона, яким користується споживач.

Для розробки іконок використовувався веб-сайт <https://icons8.com/>, який дозволяє легко та зручно знаходити та створювати необхідні зображення, які відповідають критеріям, введеним користувачем.

Тепер більше детально розберемо макет та зовнішній вигляд додатку MonoSell.

Перша сторінка - MainActivity - складається з логотипу, який розташований зверху, завдяки чому він привертає увагу користувача і водночас не займає зайвого місця. Під логотипом є слоган, який означає «MonoSell – магазин у телефоні». Знизу розташовані дві кнопки призначені для реєстрації нового користувача, або логіну для вже зареєстрованих покупців. Вони розміщені на оптимальній відстані одна від одної, адже клієнтам так буде зручніше натискати на них, і водночас людина не заплутається в кнопках.

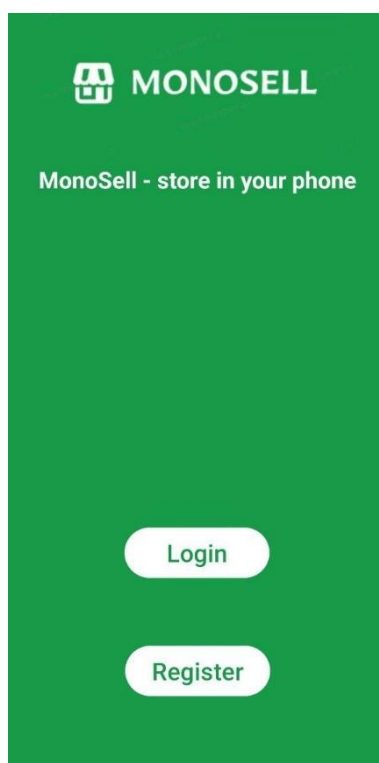


Рисунок 2.2 – Головна сторінка додатку MonoSell

Наступний пункт – RegisterActivity. Зверху є логотип магазину, по центру розташовані три поля EditText для вводу особистих даних, необхідних при реєстрації, а саме: електронна пошта, ім'я та пароль. Також користувачу надається можливість для обрання ролі, що реалізується за допомогою елементу Spinner, який дозволяє обробляти натискання на один з трьох можливих варіантів.

Якщо користувач є звичайним клієнтом, то йому потрібно обрати пункт User. У разі обрання пунктів Delivery або Admin, з'являється ще один елемент EditText, в якому необхідно ввести секретний код, який зберігається у базі даних Firebase Realtime Database.

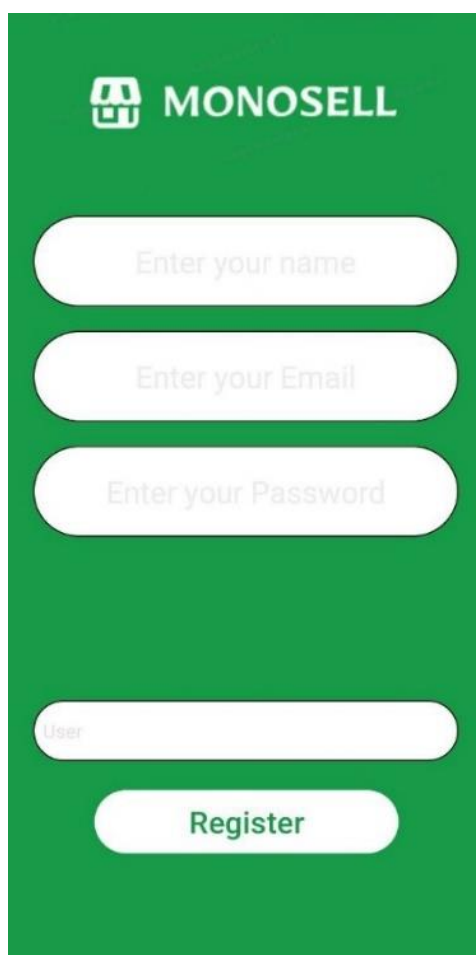


Рисунок 2.3 – Екран реєстрації у разі обрання ролі User

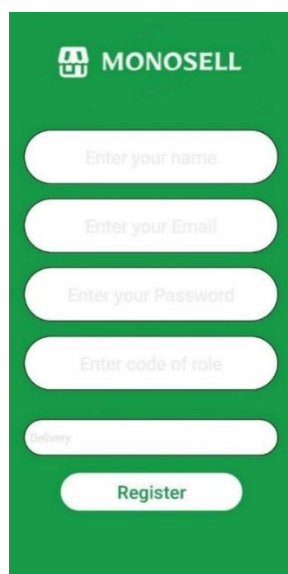
The image shows a registration screen for the MONOSELL app. At the top, there is a logo consisting of a stylized 'M' and 'S' inside a square, followed by the text 'MONOSELL'. Below the logo, there are five white input fields with rounded corners, each containing a placeholder text: 'Enter your name', 'Enter your Email', 'Enter your Password', 'Enter code of role', and 'Delivery'. At the bottom of the form, there is a white button with rounded corners labeled 'Register'.

Рисунок 2.4 – Екран реєстрації у разі обрання ролі Delivery або Admin

Після успішної реєстрації проводиться перехід до LoginActivity, де користувачеві потрібно ввести електронну пошту та пароль, для подальшого користування магазином. Варто зазначити зручне розташування полів для вводу та кнопки логіну.

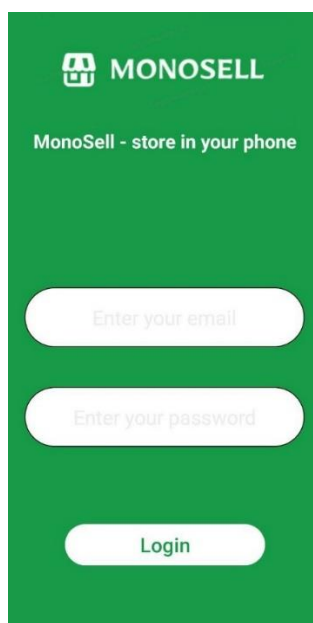
The image shows a login screen for the MONOSELL app. At the top, there is a logo consisting of a stylized 'M' and 'S' inside a square, followed by the text 'MONOSELL'. Below the logo, there is a line of text: 'MonoSell - store in your phone'. Below this text, there are two white input fields with rounded corners, each containing a placeholder text: 'Enter your email' and 'Enter your password'. At the bottom of the form, there is a white button with rounded corners labeled 'Login'.

Рисунок 2.5 – Екран логіну для переходу до каталогу магазину

Тепер розглянемо вигляд каталогу у магазині. Категорії продуктів знаходяться зверху, що дозволяє зручно перегортати їх та обирати необхідний вид товарів.

Кожна продукція обведена рамкою для зручності та чіткості зображення. На початковій сторінці каталогу є лише основна інформація, тобто назва продукту, вага, ціна та кількість на складі. Біля кожного продукту розташована кнопка для додання його у список обраного. Список товарів з легкістю гортається вгору та вниз. В нижній частині екрану розташовані кнопки навігації, до яких входять такі розділи:

- каталог;
- список обраного;
- кошик покупок;
- доставка;
- особистий кабінет;
- гарячі страви.

При натисканні на будь-яку кнопку навігації вона міняє колір, аби користувачеві було зручно орієнтуватися в додатку.

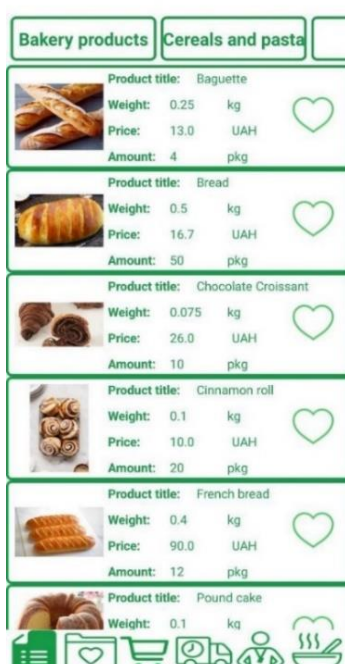


Рисунок 2.6 – Екран каталогу магазину MonoSell

При натисканні на товар відкривається окреме Activity, на якому відображено фото товару, його назву, ціну, вагу, наявність на складі, а також вказується алерген, який міститься у продукції. Крім того, у користувача з'являється можливість обрати кількість продукту та оформити замовлення, після чого воно з'явиться у кошику.

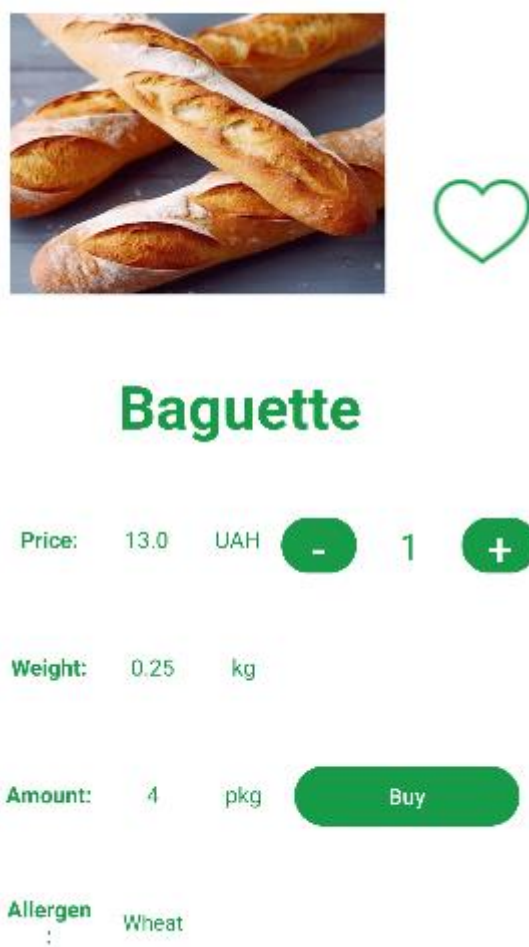


Рисунок 2.7 – Екран з детальною інформацією про товар

Перейдемо до екрану зі списком обраних товарів. Механізм прокрутки реалізований за допомогою Recycler View. Кнопка додавання до списку обраного замінена на кнопку видалення, при цьому, після натискання дані оновлюються не тільки в додатку, а і у БД.

	Product title: Buckwheat	
	Weight: 0.8 kg	DELETE
	Price: 75.0 UAH	
	Amount: -6 pkg	

	Product title: Bulgur	
	Weight: 0.5 kg	DELETE
	Price: 40.0 UAH	
	Amount: 20 pkg	

	Product title: CousCous	
	Weight: 1.0 kg	DELETE
	Price: 100.0 UAH	
	Amount: 20 pkg	

	Product title: Macaroni	
	Weight: 0.75 kg	DELETE
	Price: 60.0 UAH	
	Amount: 19 pkg	



Рисунок 2.8 – Екран зі списком обраного

Наступна вкладка – це корзина покупок, на ній відображені всі покупки, які зробив користувач під час сеансу. Натиснувши на товар можна перейти на вкладку з більш детальною інформацією. Також, на кнопці, яка знаходиться знизу, пишеться остаточна вартість покупки, що є зручним функціоналом, адже клієнту не доведеться вираховувати всю суму самотійно. При натисканні на неї здійснюється перехід на екран з доставкою.

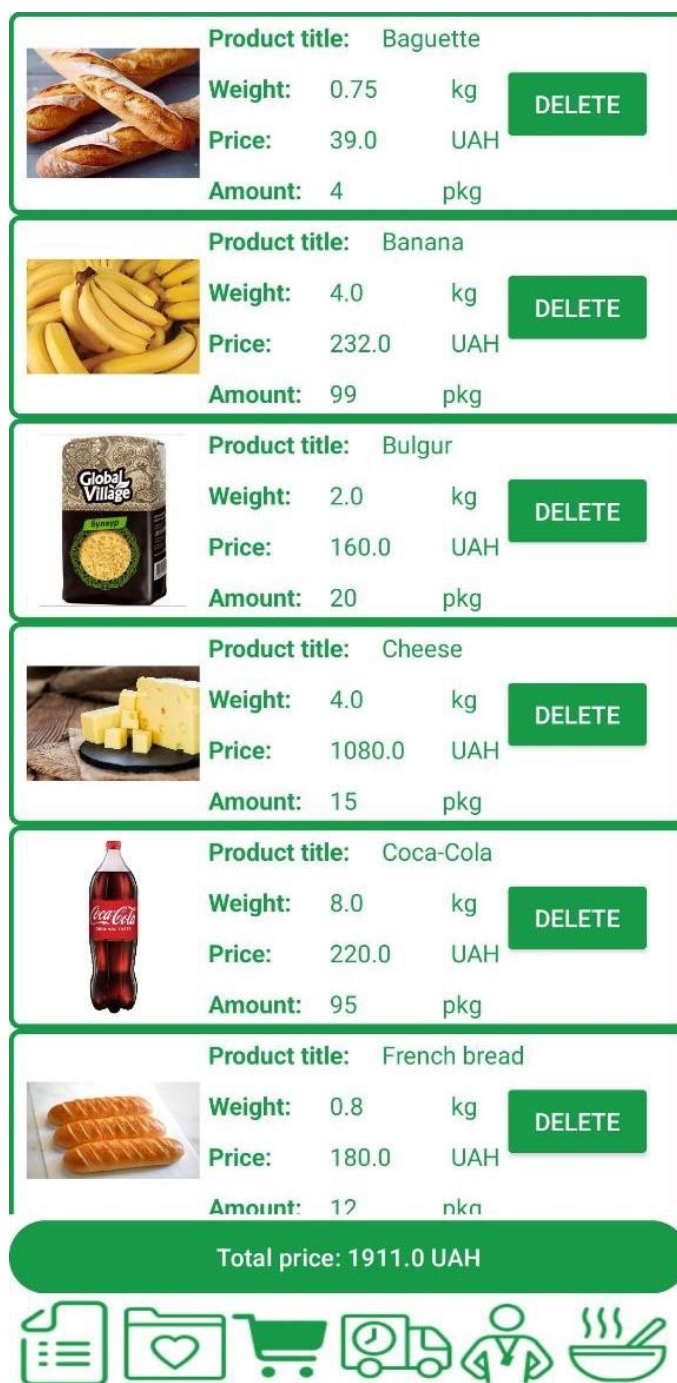


Рисунок 2.9 – Екран корзини покупок

На рисунку 2.10 зображений екран доставки, на якому зліва зображений вигляд у звичайного користувача, а праворуч – у кур'єр. Користувач після замовлення робить виклик, який видно на мапі у людини, яка має роль Delivery. Кур'єр має всі ресурси для прокладання зручного маршруту завдяки сервісам Google Maps.

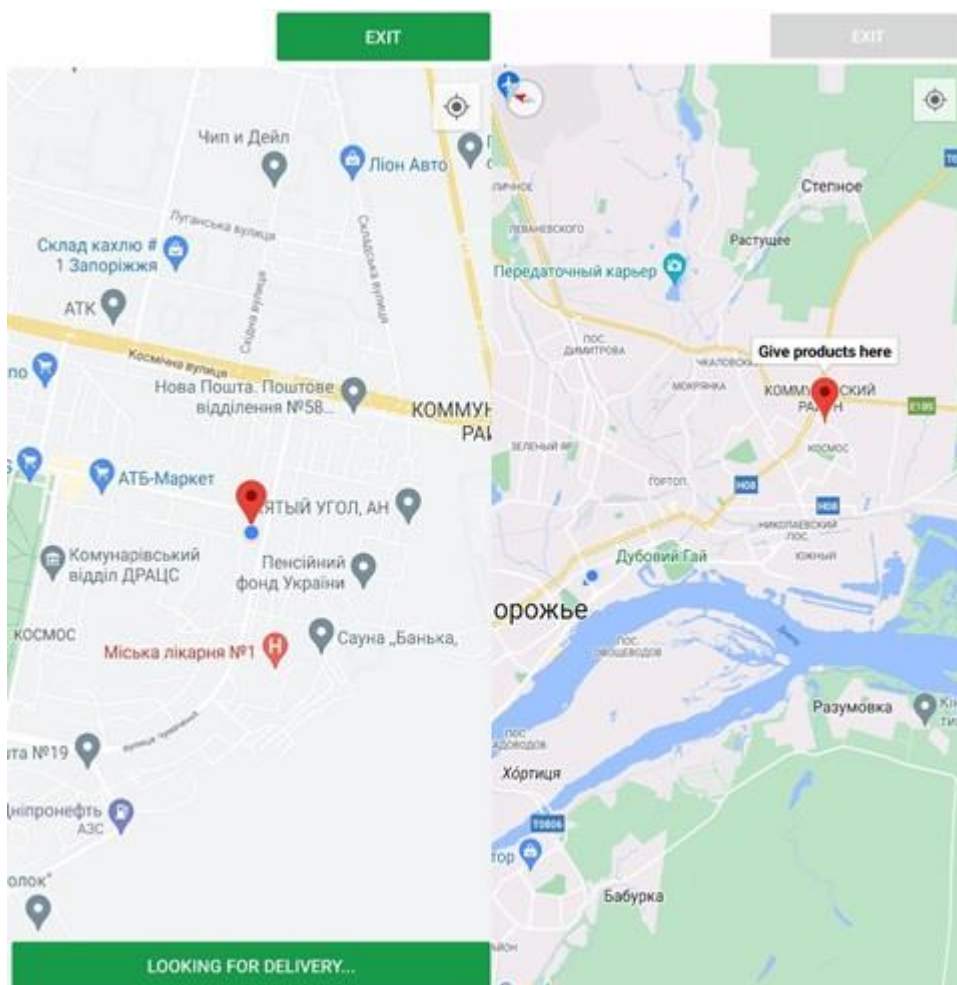


Рисунок 2.10 – Екран для доставки

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЗАСТОСУНКУ

Після аналізу ринку та проектування архітектури і дизайну майбутнього застосунку слід перейти до його реалізації, бо всі підготовчі етапи вже пройдено. Основною задачею на цьому проміжку створення додатку є остаточне рішення про мову програмування, якою буде написана програма, середовище розробки, яке буде підходити під можливості розробника та потреби при написанні коду. Також варто приділити увагу створенню алгоритму роботи застосунку і опрацюванню взаємодії між компонентами.

3.1 Розробка програмного коду застосунку з використанням вибраних технологій та фреймворків

Першочерговим етапом створення мобільного додатку для магазину з доставкою на платформі Android є додавання всіх необхідних залежностей у файл `build.gradle`. В процесі розробки будуть використовуватися бази даних `Firebase Realtime Database`, хмарне сховище `Firebase Storage`, бібліотека `GeoFire-Android` для збереження місцезнаходження користувача і кур'єра, бібліотека `AppCompat` для безперебійної роботи сучасного зовнішнього вигляду та функціоналу на старих версіях ОС Android, бібліотека `Picasso` для роботи з зображеннями та `ImageView`, відображення, а також сервіси `Google Play Services Location` та `Google Play Services Maps` для роботи з картами. Перший сервіс - це набір API та сервісів від Google, які дозволяють додаткам для Android отримувати доступ до інформації про місцезнаходження з різних джерел, включаючи GPS, Wi-Fi та стільникові мережі. Він надає розробникам уніфікований і послідовний спосіб отримання поточного місцезнаходження користувача, а також отримання оновлень при зміні місцезнаходження. `Google Play Services Maps` - це набір API та сервісів від Google, які дозволяють розробникам інтегрувати карти та сервіси на основі визначення місцезнаходження у свої додатки для Android. Він надає повний набір інструментів і функцій для відображення карт, додавання маркерів, накладок та інших візуальних елементів, а також для взаємодії з картою і отримання даних про місцезнаходження. Тому в файл `build.gradle` рівня Модуль потрібно додати код, який відображено на лістингу 3.1.

Лістинг 3.1 - Додавання бібліотек у файл `build.gradle`

```
dependencies {  
    implementation 'com.firebase:geofire-android:3.2.0'  
    implementation platform('com.google.firebase:firebase-bom:31.0.0')  
    implementation 'androidx.appcompat:appcompat:1.6.1'  
    implementation 'com.google.firebase:firebase-database:20.2.1'  
    implementation 'com.google.firebase:firebase-storage:20.2.0'
```

```
implementation 'com.google.android.gms:play-services-maps:18.1.0'
implementation 'com.google.android.gms:play-services-
location:21.0.1'}
```

На головному екрані MainActivity в методі onCreate проходить ініціалізація двох кнопок. Далі, за допомогою методу setOnClickListener, обробляється натискання кожної кнопки, та проводяться певні дії. У даному випадку – відбувається перехід з MainActivity на RegisterActivity, або LoginActivity, в залежності від натиснутої кнопки (лістинг 3.2).

Лістинг 3.2 - Перехід з MainActivity на RegisterActivity, або LoginActivity

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    initialization();
    ...
    registerButton.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            registerButtonClick();
        }
    });
    loginButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            loginButtonClick();
        }
    });
    private void registerButtonClick() {
        Intent intent = new Intent(MainActivity.this,
        RegisterActivity.class);
        startActivity(intent);
    }
    private void loginButtonClick() {
        Intent intent = new Intent(MainActivity.this, LoginActivity.class);
        startActivity(intent);
    }
    private void initialization() {
        loginButton = (Button) findViewById(R.id.buttonLoginMain);
        registerButton = (Button) findViewById(R.id.buttonRegisterMain);
    }
}
```

Після переходу на RegisterActivity користувачеві потрібно ввести свої реєстраційні дані у поля EditText, після чого обрати роль через об'єкт Spinner. Далі виконується обробка ролі і налаштовується відображення поля EditText, де потрібно ввести секретний код (лістинг 3.3).

Лістинг 3.3 - Spinner з ролями

```

ArrayAdapter<String> adapter = new
ArrayAdapter<String>(RegisterActivity.this,
com.google.android.material.R.layout.support_simple_spinner_dropdown
_item, roleOfUser);
adapter.setDropDownViewResource(com.google.android.material.R.layout
.support_simple_spinner_dropdown_item);
roleSpinner.setAdapter(adapter);
roleSpinner.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
role = parent.getItemAtPosition(position).toString();
if(role.equals("Admin")){
codeRegister.setVisibility(View.VISIBLE);}else
if(role.equals("Delivery")){codeRegister.setVisibility(View.VISIBLE)
;}else{codeRegister.setVisibility(View.INVISIBLE);}
@Override
public void onNothingSelected(AdapterView<?> parent) {}
});

```

Після натискання кнопки реєстрації викликається метод `createNewUser()`, де проходить обробка всіх введених даних, перевірка відповідності секретного коду, написаного користувачем та збереженого в базі даних. Якщо вся інформація правильна, викликається метод `checkGeneralInformation()`, куди передається ім'я клієнта, адреса електронної пошти, id та пароль. У вказаному методі проходить перевірка введених даних та створення нового користувача, якого одразу заносять у базу даних `Firebase Realtime Database`. Якщо вся інформація була успішно збережена, викликається метод `checkDataSaved()` і виконується перехід на екран логіну (додаток А.1).

На `LoginActivity` користувач повинен ввести свою електронну пошту та пароль в два поля `EditText`. Після натискання на кнопку логіну, дані з вищевказаних полів записуються в об'єкти класу `String`. Завдяки методу `isEmpty()` рядок перевіряється на наявність літер. Якщо користувач ввів всю необхідну інформацію викликається метод `checkCorrectData()`, де проходить пошук на наявність вказаного клієнта у БД. Далі перевіряється роль і виконується перехід на певне `Activity` за допомогою `Intent`, через якого передається об'єкт класу `User` (додаток А.2).

Якщо роль користувача – User, то на екрані каталогу відбувається збереження даних про переданого за допомогою Intent об'єкта класу User. До речі, для передачі об'єктів класу User між Activity, потрібно імплементувати інтерфейс Serializable. Далі відбувається ініціалізація всіх елементів на Activity, та перевірка ролі, якщо вона дорівнює Admin, то кнопка для її зміни стає видимою. Цей процес зображено на лістингу 3.4.

Лістинг 3.4 - Перевірка ролі користувача

```
Bundle arguments = getIntent().getExtras();
if(arguments!=null){
    userInf = (User) arguments.getSerializable("user");
    role = userInf.getRole();}
initiation();
checkRole();
...
private void checkRole() {
    if(role.equals("Admin")){
        changeRole.setVisibility(View.VISIBLE);
    }else{changeRole.setVisibility(View.INVISIBLE);}}
```

Потім відбувається перевірка всіх категорій продуктів в БД і вони додаються у список List, після чого оновлюється Adapter (лістинг 3.5).

Лістинг 3.5 - Оновлення інформації у адаптері

```
mDatabase.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        for (DataSnapshot categorySnapshot : dataSnapshot.getChildren()) {
            categoryValue = categorySnapshot.getKey();
            mainListCategory.add(categoryValue);
            myCategoryListAdapter.notifyDataSetChanged();}}
    @Override
    public void onCancelled(DatabaseError databaseError) {}));
```

Для правильного відображення всіх продуктів та категорій у каталозі використовується RecyclerView. Для кожного такого елементу використовується свій ViewHolder - клас, який зберігає посилання на візуальні елементи (віджети) одного елемента списку. Цей клас допомагає знизити частоту виклику методу

`findViewById()`, що є затратною в плані використання ресурсів операцією, оскільки він шукає віджети в ієрархії кожного елемента списку під час прокрутки.

При створенні адаптера для списку `RecyclerView`, ми визначаємо клас `ViewHolder`, який розширює клас `RecyclerView.ViewHolder`. У цьому класі оголошуються змінні для збереження посилань на візуальні елементи (віджети) одного елемента списку. При першому створенні елемента списку, `ViewHolder` об'єкт створюється і посилання на візуальні елементи присвоюються відповідним змінним в `ViewHolder`. Далі, при переробці списку, коли відображається новий елемент, метод `onBindViewHolder()` адаптера використовує цей `ViewHolder` для отримання доступу до візуальних елементів і оновлення їх залежно від даних, які потрібно відобразити. Використання `ViewHolder` дозволяє ефективно використовувати пам'ять і зменшує навантаження на процесор при прокрутці списку, оскільки затратні операції знаходження віджетів виконуються тільки один раз під час створення `ViewHolder`, а не при кожному оновленні елемента списку (лістинг 3.6).

Лістинг 3.6 - ViewHolder

```
public class ProductsListViewHolder extends RecyclerView.ViewHolder
{public ImageView imageOfProduct;
public TextView
nameOfProduct,weightOfProduct,priceOfProduct,amountOfProduct;
public Button favouriteButton;
public ProductsListViewHolder(@NonNull View itemView) {
super(itemView);
initiation();}
private void initiation() {
imageOfProduct = itemView.findViewById(R.id.imageOfProduct);
nameOfProduct = itemView.findViewById(R.id.textNameOfProduct);
weightOfProduct = itemView.findViewById(R.id.textWeightOfProduct);
priceOfProduct = itemView.findViewById(R.id.textPriceOfProduct);
amountOfProduct = itemView.findViewById(R.id.textProductAmount);
favouriteButton = itemView.findViewById(R.id.favouriteRecycler);}}
```

Для прокрутки і обробки натискань на кнопку використовується клас `Adapter`. `Adapter` - це клас, який використовується для зв'язку даних із візуальними елементами інтерфейсу користувача, такими як списки, ґріди, спінери тощо.

Adapter виступає посередником між даними і візуальними елементами, дозволяючи відображати дані з джерела даних у відповідному форматі візуальним елементам.

Головна відповідність адаптера полягає в тому, що він надає методи, які дозволяють відображати дані візуальним елементам та забезпечують інтеракцію з ними. Наприклад, адаптер може надавати методи для визначення кількості елементів у списку, отримання даних для кожного елемента, створення візуального представлення для кожного елемента, обробки подій натискання на елементи тощо.

В розроблюваному застосунку використовуються такі адаптери, як ArrayAdapter для списків, RecyclerView.Adapter для RecyclerView (додаток Б.1).

Для обробки натискань потрібно імплементувати два інтерфейси. В контексті розробки на платформі Android, Interface - це колекція абстрактних методів, які визначаються, але не реалізуються в самому інтерфейсі. Інтерфейс виступає як контакт між класами, вказуючи, які методи повинні бути реалізовані в класах, що імплементують цей інтерфейс. У розробці Android-додатків, інтерфейси використовуються для опису взаємодії між компонентами програми. У цьому випадку, інтерфейс був визначений для обробки подій натискання на кнопку, і всі класи, які хочуть реагувати на натискання кнопки, повинні реалізувати цей інтерфейс та його методи. Інтерфейси дозволяють досягти поліморфізму та забезпечують можливість розширення функціоналу. Вони також допомагають забезпечити чітку роздільність між різними компонентами додатку та підвищують його модульність (лістинг 3.7).

Лістинг 3.7 - Інтерфейси

```
public interface ItemClickListener{  
    void onItemClick(Product product);}  
public interface ButtonClickListener{  
    void onButtonClick(Product product);}
```

Реалізація створення списку продуктів та категорій за допомогою RecyclerView детально показана на лістингу 3.8.

Лістинг 3.8 - RecyclerView зі списком продуктів

```

mainListProd= new ArrayList<Product>();
favListProd = new ArrayList<Product>();
myProductListAdapter = new
ListOfProductAdapter(getApplicationContext(), mainListProd, new
ListOfProductAdapter.ItemClickListener() {
@Override
public void onItemClick(Product product) {
Intent intent = new Intent(ShopActivity.this,
ProductDetailInfActivity.class);
intent.putExtra("productDetail",product);
intent.putExtra("user",userInf);
startActivity(intent); }}, new
ListOfProductAdapter.ButtonClickListener() {
@Override
public void onClick(Product product) {
favListProd.add(product);
FirebaseDatabase userDBFav=FirebaseDatabase.getInstance();
DatabaseReference userFavDR =
userDBFav.getReference().child("users");
userFavDR.child(userInf.getEmail().replace(".",","))
.child("favourite").child(product.getName()).setValue(product);}});
mainListCategory = new ArrayList<>();
myCategoryListAdapter = new
ListOfCategoryAdapter(getApplicationContext(), mainListCategory, new
ListOfCategoryAdapter.ItemClickListener() {
@Override
public void onItemClick(String str) {mainListProd.clear();
mDatabase.child(str).addListenerForSingleValueEvent(new
ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {
if(snapshot.exists()){
for (DataSnapshot productSnapshot : snapshot.getChildren()) {
productToShow = productSnapshot.getValue(Product.class);
mainListProd.add(productToShow);
myProductListAdapter.notifyDataSetChanged();}}else{
Toast.makeText(ShopActivity.this, "Something went wrong",
Toast.LENGTH_SHORT).show();}}
@Override
public void onCancelled(@NonNull DatabaseError error) {}}}));
categoriesRecycler.setLayoutManager(new LinearLayoutManager(this,
LinearLayoutManager.HORIZONTAL,false));
categoriesRecycler.setAdapter(myCategoryListAdapter);
productsRecycler.setLayoutManager(new LinearLayoutManager(this));
productsRecycler.setAdapter(myProductListAdapter);

```

Вікно ActivityShopAdmin працює за тим самим принципом, проте має ряд відмінностей – можливість переключання між виглядом для адміністратора та клієнту (лістинг 3.9), окрема кнопка для додавання категорій товару (лістинг 3.10),

та для роботи з продуктами. Додавання категорій проходить завдяки PopupWindow, а для продуктів потрібно перейти на нове AddProductActivity. До того ж, після повернення з AddProductActivity потрібно, щоб адаптер оновлював список, тому використовується метод startActivityForResult().

Метод startActivityForResult() є одним з способів запуску іншої Activity. Він використовується для запуску іншої активності з метою отримання результату виконаної дії. При використанні startActivityForResult(), ми передаємо Intent, який описує, яку активність потрібно запустити. У відповідь на запуск, отримується результат, який повертається з цієї активності після її закриття (лістинг 3.11).

Лістинг 3.9 - Перемикання вигляду з адміністратора на користувача

```
changeRole.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(ActivityShopAdmin.this,
            ShopActivity.class);
        intent.putExtra("user",userInfo);
        startActivity(intent);});
```

Лістинг 3.10 – Редагування категорій продукції

```
addCategories.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        popupAddCategory();});
private void popupAddCategory() {
    LayoutInflater inflater = (LayoutInflater)
        getSystemService(LAYOUT_INFLATER_SERVICE);
    View popupView = inflater.inflate(R.layout.popup_add_category_admin,
        null);
    PopupWindow popupWindow = new PopupWindow(
        popupView,
        ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.WRAP_CONTENT,true);
    popupWindow.setBackgroundDrawable(new
        ColorDrawable(Color.TRANSPARENT)); // Установлюємо прозорий фон
    Button addCategoryButtonPopup =
        popupView.findViewById(R.id.addCategoryButton);
    EditText addCategoryEditTextPopup =
        popupView.findViewById(R.id.addCategoryEditText);
    addCategoryButtonPopup.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            if(addCategoryEditTextPopup.getText().toString().equals("")){
```

```

Toast.makeText(ActivityShopAdmin.this, "Empty name of category",
Toast.LENGTH_SHORT).show();
}else{
String AddCategory = addCategoryEditTextPopup.getText().toString();
mDatabase.child(AddCategory).setValue(AddCategory);
mainListCategory.clear();
myCategoryListAdapter.notifyDataSetChanged();
popupWindow.dismiss();}}});
View rootView = findViewById(android.R.id.content);
int[] location = new int[2];
rootView.getLocationOnScreen(location);
popupWindow.showAtLocation(rootView, Gravity.CENTER, 0, 0);}

```

Лістинг 3.11 – Перехід на Activity з додаванням продукту

```

addProducts.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Intent intent = new Intent(ActivityShopAdmin.this,
AddProductActivity.class);
mainListCategory.clear();
startActivityForResult(intent, REQUEST_CODE_FOR_ADD_PRODUCT);}
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
super.onActivityResult(requestCode, resultCode, data);
if (requestCode == REQUEST_CODE_FOR_ADD_PRODUCT && resultCode ==
Activity.RESULT_OK) {myProductListAdapter.notifyDataSetChanged();}}

```

Для того, щоб додати продукт на AddProductActivity, потрібно заповнити всі поля, включаючи встановлення зображення товару. Далі вся інформація відправляється у БД Realtime Database після натискання кнопки Add product, а фото зберігається на хмарному сховищі Firebase Storage (додаток Б.2).

Після натискання на продукт викликається активність ProductDetailInfActivity, на якій користувачеві надається детальна інформація про товар, включаючи наявні алергени. Також, у клієнта є можливість додати предмет у обране за допомогою кнопки, визначити кількість товару для покупки та оформити своє замовлення (додаток Б.3).

Для навігації між Activity в нижній частині екрану додається навігаційна панель, яка складається з 6 кнопок. Після натискання на будь-яку з них створюється Intent, куди передаються дані про користувача (лістинг 3.12). Також варто зазначити, що для зручності відображення після натискання на кнопку змінюється

її колір, аби клієнту було комфортніше орієнтуватися на якій вкладці він на даний момент знаходиться.

Лістинг 3.12 – Навігація між Activity:

```
deliveryChoose.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(userInf.getRole().equals("User")){
            Intent intent = new Intent(ShopActivity.this,
            CustomerMapActivity.class);
            intent.putExtra("user",userInf);
            startActivity(intent);}else{
            Intent intent = new Intent(ShopActivity.this,
            DeliveryMapsActivity.class);
            intent.putExtra("user",userInf);
            startActivity(intent);}}});
favouriteChoose.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(ShopActivity.this,
        FavouriteUserActivity.class);
        intent.putExtra("user",userInf);
        startActivity(intent);}});
shoppingCartChoose.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(ShopActivity.this, ShoppingCart.class);
        intent.putExtra("user",userInf);
        startActivity(intent);}});
hotDishesChoose.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(ShopActivity.this,
        HotDishesActivity.class);
        intent.putExtra("user",userInf);
        startActivity(intent);}});
personalCabinetChoose.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(ShopActivity.this,
        PersonalCabinetActivity.class);
        intent.putExtra("user",userInf);
        startActivity(intent);}});
```

Функціонал корзини покупок та списку обраного є дуже схожим, тому розглянемо лише ShoppingCart. Для зручного відображення продуктів використовується RecyclerView, адаптер має аналогічну до ListOfProductAdapter

структуру. Головною відмінністю є наявність кнопки знизу, на якій відображається остаточна ціна товару та надається можливість зробити замовлення. Всі дані про куплені продукти беруться з БД, зміни проходять в реальному часі (додаток В.1).

Доставка працює таким чином, що кур'єр бачить замовлення на мапі, та завдяки сервісам Google Maps прокладає найбільш зручний маршрут. При цьому дані про місцезнаходження користувача та кур'єра зберігаються в режимі реального часу в БД Realtime Database. Також, при виконанні замовлення кур'єр зі статусу Driver Available переходить до Driver Working, а після успішної доставки знов повертається у попередній статус (додаток В.2).

3.2 Виконання тестування та аналіз результатів для перевірки функціональності, стабільності та ефективності застосунку

Для фінального тестування було залучено вбудований в Android Studio емулятор, та три діючі пристрої. Розмір арк-файлу становить 10.1 мегабайти, що вдалося досягти завдяки використанню хмарної БД Firebase Realtime Database та сховища Firebase Storage. Всі класи працюють без перебоїв, навігація легка, кнопки розташовані зручно, всі необхідні функції враховані. Стабільність роботи додатку та кількість ресурсів, які необхідні для застосунку вказані на рисунку 3.1. Тестування проводилося за допомогою вбудованої функції Android Profiler.

Android Profiler - це інструмент, що надається Android Studio для аналізу та оптимізації продуктивності Android-додатків. Він дозволяє розробникам відстежувати різні показники і ресурси програми під час виконання, такі як використання процесора, розподіл пам'яті, мережева активність і споживання заряду акумулятора. Android Profiler надає дані та візуалізації в режимі реального часу, що дозволяє розробникам виявляти вузькі місця в роботі, витoki пам'яті та інші проблеми. Це важливий інструмент для оптимізації продуктивності додатків і забезпечення безперебійної роботи користувачів на пристроях Android.

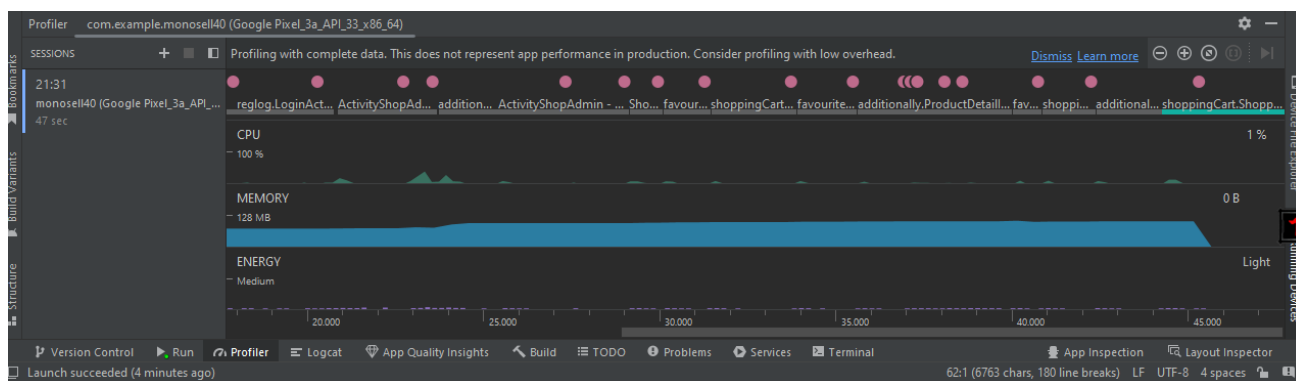


Рисунок 3.1 – Кількість оперативної пам'яті, ресурсів процесора та батареї, необхідних для роботи з додатком MonoSell

3.3 Виявлення та виправлення помилок та дефектів застосунку

Під час розробки було виявлено та виправлено наступні дефекти:

- дефект: в якості логіну використовується електронна пошта, проте в БД неможливо занести дані з крапкою. Виправлення: використання методу `replace(".", "", ",")` для заміни певних символів або рядків у текстовому рядку. Зазвичай він приймає два параметри: перший параметр - рядок або символ, який потрібно замінити, і другий параметр - рядок або символ, на який потрібно замінити;

- дефект: при додаванні продукту або категорії, список не оновлювався та відображалися як старі, так і нові продукти. Виправлення: використання методів `clear()` та `notifyDataSetChanged()`. Метод `clear()` використовується для очищення даних, які знаходяться в адаптері. Це означає, що всі елементи, які були додані до адаптера, будуть видалені. Цей метод зазвичай використовується перед додаванням нових даних до адаптера, щоб впевнитись, що старі дані будуть видалені перед додаванням нових. Метод `notifyDataSetChanged()` використовується для повідомлення адаптеру про зміни в даних, що відображаються.

ВИСНОВКИ

Застосування мобільних пристроїв з Android-операційною системою стає все більш поширеним в сучасному світі. Одним з найбільш популярних використань є замовлення товарів та послуг через мобільні додатки.

Android-застосунок це важливий аспект для бізнесу, що займається торгівлею. Враховуючи зріст популярності онлайн-покупок та зручність використання мобільних пристроїв для замовлення товарів, розробка додатку магазину з доставкою є важливим кроком для підвищення ефективності та задоволення потреб клієнтів.

Під час створення застосунку було враховано ряд важливих факторів, таких як практичність та легкість використання, швидкість доставки, безпека та конфіденційність даних клієнтів, можливість користування додатком на слабких пристроях.

Практичність та легкість використання зумовлена правильно підібраним дизайном та зручним розташуванням кнопок та елементів, за рахунок чого користувачеві не доведеться тягнутися до різних країв екрану для виконання бажаних дій. Також варто зазначити приємну кольорову гаму, яка не викликає роздратування, а тільки створює атмосферу затишку та комфорту.

Швидкість доставки реалізована завдяки тому, що кур'єр бачить всі можливі замовлення і може сам визначати, за яким маршрутом йому везти товари.

Безпека інформації клієнтів гарантується через використання технологій Google, а саме Realtime Database. Ця база даних фігурує у багатьох проектах, тож переживати за конфіденційність не потрібно.

Оскільки вся інформація зберігається на хмарному сховищі та у БД, а сам додаток не займає багато місця, не витрачає велику кількість ресурсів процесора, батареї та оперативної пам'яті, тому використання застосунку на слабких пристроях буде комфортним та не викликатиме проблем.

Метою дослідження було розроблення функціонального та зручного в користуванні Android-додатку для магазину з доставкою, який задовольнить потреби користувачів та покращить ефективність роботи бізнесу.

Для досягнення мети були поставлені такі завдання:

- провести аналіз літератури та ринку, щодо магазинів з доставкою на платформі Android;
- дослідити основні технології, фреймворки та інструменти, використовувані для розробки мобільних додатків на платформі Android;
- виявити потреби та очікування користувачів від магазину з доставкою на платформі Android;
- визначити вимоги до функціоналу застосунку на основі аналізу потреб користувачів;
- розробити архітектуру застосунку, включаючи вибір архітектурних паттернів, розподіл функціоналу на компоненти та опис взаємодії між ними;
- розробити дизайн інтерфейсу користувача застосунку, включаючи розробку макетів, вибір кольорової гами, шрифтів та іконок;
- розробити програмний код застосунку з використанням вибраних технологій та фреймворків;
- виконати тестування та аналіз результатів для перевірки функціональності, стабільності та ефективності застосунку;
- виявити та виправити помилки та дефекти додатку.

Під час дослідження було проведено огляд наявних рішень в галузі торговельних платформ. Було виявлено, що на сьогоднішній день існує велика кількість мобільних додатків, які надають можливість замовляти товари та послуги з доставкою.

Деякі з розглянутих додатків пропонують зручні інструменти для пошуку товарів, фільтрації за різними параметрами, а також можливість відслідковувати статус замовлення та доставки. Застосунки також можуть мати інтеграцію з різноманітними платіжними системами, що спрощує процес оплати.

Проте, були виявлені певні недоліки, такі як складність навігації, нестабільна робота додатків на деяких пристроях та відсутність необхідних функцій. Також, деякі застосунки можуть не мати достатньої кількості пунктів самовивозу або можливості самовивозу товарів, що ускладнює процес отримання замовлення.

Отже, можна зробити висновок, що наявні рішення в галузі магазинів на платформі Android мають свої переваги та недоліки. Враховуючи їх, було розроблено новий додаток, який поєднує в собі переваги наявних рішень та вирішує деякі з їх недоліків, що дозволяє забезпечити максимально комфортний та безпечний процес замовлення та доставки товарів та послуг.

Варто зазначити, що основними напрямками для розвитку та вдосконалення додатку є:

- розробка аналогічного застосунку на операційну систему iOS;
- покращення інтерфейсу та дизайну;
- додавання більш детальної інформації про товар;
- можливість відправляти сформований список покупок через месенджери;
- додавання онлайн-чату зі службою підтримки, щоб користувачеві було зручно повідомляти про несправності або недоліки в роботі додатку;
- покращення сортування продуктів за наявних в них алергенах;
- додавання налаштувань в особистому кабінеті, таких як зміна теми додатку з нічної на денну, заповнення інформації про користувача;
- опрацювання функції розсилки сповіщень про улюблені товари, акції та новинки через електронну пошту.

В реальних умовах додаток працює без проблем, здатний забезпечити користувачів всім необхідним функціоналом. Головне – додати можливість оплати товару онлайн і тоді застосунок можна виставляти в Google Play. Користь для клієнтів представлена у вигляді додаткових функцій, таких як сортування товарів за наявністю алергенів, та замовлення готових гарячих страв, або можливість додавання всіх необхідних інгредієнтів для самостійного приготування у кошик покупок одним натисканням на рецепт.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mobile Operating System Market Share Worldwide | Statcounter Global Stats. StatCounter Global Stats. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата звернення: 20.04.2023).
2. Офіційний сайт комерційного проекту ROZETKA. URL: <https://rozetka.com.ua/ua/> (дата звернення: 18.04.2023).
3. Офіційний сайт сервісу з доставки продуктів. URL: <https://zakaz.ua/uk/> (дата звернення: 18.04.2023).
4. Офіційний сайт мережі супермаркетів АТБ. URL: <https://www.atbmarket.com/> (дата звернення: 18.04.2023).
5. Офіційний сайт мережі супермаркетів Сільпо. URL: <https://shop.silpo.ua/> (дата звернення: 18.04.2023).
6. Офіційний сайт інтернет-магазину продуктів Fozzy. URL: <https://fozzyshop.ua/> (дата звернення: 18.04.2023).
7. Herbert Schildt. Java: The Complete Reference Twelfth Edition – Видавництво "McGraw Hill", 2021. – 1280 с. – ISBN: 978-1260463415.
8. Dmitry Jemerov, Svetlana Isakova. Kotlin in Action First Edition – Видавництво "Manning", 2017. – 360 с. – ISBN: 978-1617293290.
9. Sylvain Ratabouil. Android NDK: Beginner's Guide - Second Edition – Видавництво "Packt Publishing", 2015. – 494 с. - ISBN: 978-1783989645.
10. Офіційна сторінка платформи для розробки мобільних додатків. URL: <https://visualstudio.microsoft.com/xamarin/> (дата звернення: 26.04.2023).
11. Dan Hermes. Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals 1st ed. Edition – Видавництво "Apress", 2015. - 470 с. - ISBN: 978-1484202159.
12. Офіційна сторінка IDE Android Studio. URL: <https://developer.android.com/studio/intro> (дата звернення: 26.04.2023).

13. Офіційний сайт Eclipse. URL: <https://www.eclipse.org/> (дата звернення: 26.04.2023).

14. Офіційний сайт IDE IntelliJ IDEA. URL: <https://www.jetbrains.com/idea/> (дата звернення: 26.04.2023).

15. Ockert J. du Preez. Visual Studio 2019 In Depth: Discover and make use of the powerful features of the Visual Studio 2019 IDE to develop better and faster mobile, web, and desktop applications 1st Edition, Kindle Edition - Видавництво "BPB Publications", 2019. - 348 с. - ISBN: 978-9389328325.

16. David Salter, Rhawi Dantas. NetBeans IDE 8 Cookbook – 2014. - 386 с. - ISBN: 978-1782167761.

17. Офіційна сторінка платформи для розробки мобільних додатків за допомогою JavaScript React Native. URL: <https://learn.microsoft.com/uk-ua/windows/dev-environment/javascript/react-native-for-android> (дата звернення: 26.04.2023).

18. Офіційна документація про Flutter. URL: <https://docs.flutter.dev/> (дата звернення: 26.04.2023).

19. Arvind Ravulavaru. Learning Ionic - Second Edition: Hybrid mobile apps with HTML5, CSS3, and Angular 2nd Edition, Kindle Edition, 2017. – 635 с. - ISBN: 978-1786466051.

20. John M. Wargo. Apache Cordova 4 Programming (Mobile Programming) 1st Edition, Kindle Edition, 2015. – 558 с. - ISBN: 978-0134048192.

21. Nathan Walker, Nathanael J. Anderson. NativeScript for Angular Mobile Development: Creating dynamic mobile apps for iOS and Android 1st Edition, Kindle Edition, 2017. – 574 с. - ISBN: 978-1787125766.

22. Adrian Kosmaczewski. Sencha Touch 2 Up and Running: Building Enterprise Cross-Platform Mobile Web Applications 1st Edition, Kindle Edition, 2013. – 385 с. - ISBN: 978-1449339388.

23. Офіційна сторінка хмарного сервісу Firebase від Google. URL: <https://firebase.google.com/docs> (дата звернення: 27.04.2023).

24. Tom Marrs. JSON at Work: Practical Data Integration for the Web 1st Edition, O'Reilly Media, 2017. – 376 с. - ISBN: 978-1449358327.

25. Офіційна сторінка проекту Picasso на Github. URL: <https://square.github.io/picasso/> (дата звернення: 27.04.2023).

26. Eran Boudjnah. Clean Architecture for Android: Implement Expert-led Design Patterns to Build Scalable, Maintainable, and Testable Android Apps (English Edition) 1st Edition, BPB Publications, 2022. – 397 с. - ISBN: 978-9355510495.

ДОДАТОК А

Лістинг А.1 - Метод створення користувача на RegisterActivity

```

private void createNewUser() {
String nameOfUser = nameRegister.getText().toString().trim();
String emailOfUser = emailRegister.getText().toString().trim();
String emailAsAnId = emailOfUser.replace(".", "");
String passwordOfUser =
passwordRegister.getText().toString().trim();
if(role.equals("User")){
checkGeneralInformation(nameOfUser,emailOfUser,emailAsAnId,passwordOfUser);}else if(role.equals("Admin")){
String roleCode = codeRegister.getText().toString().trim();
if(roleCode.equals("")){
Toast.makeText(this,"Input code for admin",
Toast.LENGTH_SHORT).show();
}else{
mDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {
if(snapshot.child("secret
code").child("Admin").child(roleCode).exists()){
Toast.makeText(RegisterActivity.this, "It's true!",
Toast.LENGTH_SHORT).show();
checkGeneralInformation(nameOfUser,emailOfUser,emailAsAnId,passwordOfUser);
}else{Toast.makeText(RegisterActivity.this, "You're not an admin!",
Toast.LENGTH_SHORT).show();}}
@Override
public void onCancelled(@NonNull DatabaseError error)
{Toast.makeText(RegisterActivity.this, "Error",
Toast.LENGTH_SHORT).show();}});
}else{String roleCode = codeRegister.getText().toString().trim();
if(roleCode.equals("")){
Toast.makeText(this,"Input code for delivery",
Toast.LENGTH_SHORT).show();
}else{mDatabase.addListenerForSingleValueEvent(new
ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {
if(snapshot.child("secret code").
child("Delivery").child(roleCode).exists()){
Toast.makeText(RegisterActivity.this, "It's true!",
Toast.LENGTH_SHORT).show();
checkGeneralInformation(nameOfUser,emailOfUser,emailAsAnId,passwordOfUser);}else{
Toast.makeText(RegisterActivity.this, "You're not a delivery!",
Toast.LENGTH_SHORT).show();}}
@Override

```

```

public void onCancelled(@NonNull DatabaseError error)
{Toast.makeText(RegisterActivity.this, "Error",
Toast.LENGTH_SHORT).show();});}}
private void checkGeneralInformation(String nameOfUser, String
emailOfUser,String emailAsAnId,String passwordOfUser) {
if(emailAsAnId.isEmpty()||nameOfUser.isEmpty()||emailOfUser.isEmpty(
)||passwordOfUser.isEmpty()){
Toast.makeText(this, "Wrong DATA, please check your input data",
Toast.LENGTH_SHORT).show();}else{
User user = new User(nameOfUser, emailOfUser, passwordOfUser, role,
null, null, null, null);
mDatabase.child("users").child(emailAsAnId).setValue(user);
checkDataSaved();}}
private void checkDataSaved() {
ValueEventListener registrationListener = new ValueEventListener() {
@Override
public void onDataChange(DataSnapshot dataSnapshot) {
if (dataSnapshot.exists()) {
Intent intent = new Intent(RegisterActivity.this,
LoginActivity.class);
startActivity(intent);
finish();}
else {Toast.makeText(RegisterActivity.this, "Your DATA doesn't save
in DB", Toast.LENGTH_SHORT).show();}}
@Override
public void onCancelled(DatabaseError databaseError) {}};
mDatabase.addListenerForSingleValueEvent(registrationListener);}

```

Лістинг А.2 - Метод перевірки введених користувачем даних на LoginActivity

```

loginButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
emailCheck =
emailLogin.getText().toString().trim().replace(".",",");
passwordCheck = passLogin.getText().toString().trim();
if(!emailCheck.isEmpty() && !passwordCheck.isEmpty()){
checkCorrectData();
}else if(emailCheck.isEmpty()){
Toast.makeText(LoginActivity.this, "Input your email",
Toast.LENGTH_SHORT).show();
}else if(passwordCheck.isEmpty()){
Toast.makeText(LoginActivity.this, "Input your Password",
Toast.LENGTH_SHORT).show();}}});
private void checkCorrectData() {
mDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
@Override
public void onDataChange(@NonNull DataSnapshot snapshot) {
if(snapshot.child("users").child(emailCheck).exists()){
User user =
snapshot.child("users").child(emailCheck).getValue(User.class);
if(user==null) {

```

```

Toast.makeText(LoginActivity.this, "no user",
Toast.LENGTH_SHORT).show();
}else {
if (!user.getEmail().equals(emailCheck.replace(", ", "."))) {
Toast.makeText(LoginActivity.this, "Email incorrect",
Toast.LENGTH_SHORT).show();
} else {
if (!user.getPassword().equals(passwordCheck)) {
Toast.makeText(LoginActivity.this, "Password incorrect",
Toast.LENGTH_SHORT).show();} else {
if (user.getRole().equals("User")){
Toast.makeText(LoginActivity.this, "Login Complete",
Toast.LENGTH_SHORT).show();
Intent intent = new Intent(LoginActivity.this, ShopActivity.class);
intent.putExtra("user",user);
startActivity(intent);
finish();
} else if (user.getRole().equals("Admin")) {
Toast.makeText(LoginActivity.this, "Login Admin Complete",
Toast.LENGTH_SHORT).show();
Intent intent = new Intent(LoginActivity.this,
ActivityShopAdmin.class);
intent.putExtra("user",user);
startActivity(intent);
finish();
}else{
Toast.makeText(LoginActivity.this, "Login Complete",
Toast.LENGTH_SHORT).show();
Intent intent = new Intent(LoginActivity.this, ShopActivity.class);
intent.putExtra("user",user);
startActivity(intent);
finish();}}}}}}
@Override
public void onCancelled(@NonNull DatabaseError error) {
Toast.makeText(LoginActivity.this, "Login Incorrect",
Toast.LENGTH_SHORT).show();});}

```

ДОДАТОК Б

Лістинг Б.1 - Adapter для каталогу з продуктами

```
public class ListOfProductAdapter extends
RecyclerView.Adapter<ProductsListViewHolder> {
    Context context;
    List<Product> productsList;
    private ItemClickListener prodItemClickListener;
    private ButtonClickListener buttonClickListener;
    public ListOfProductAdapter(Context context, List<Product>
productsList, ItemClickListener itemClickListener,
ButtonClickListener buttonClickListener) {
        this.context = context;
        this.productsList = productsList;
        this.prodItemClickListener = itemClickListener;
        this.buttonClickListener = buttonClickListener;}
    @NonNull
    @Override
    public ProductsListViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType){
        return new
ProductsListViewHolder(LayoutInflater.from(context).inflate(R.layout
.recycler_view_product, parent, false));}
    @Override
    public void onBindViewHolder(@NonNull ProductsListViewHolder holder,
int position) {
        holder.nameOfProduct.setText(productsList.get(position).getName());
        holder.weightOfProduct.setText(String.valueOf(productsList.get(positi
ion).getWeight()));
        holder.priceOfProduct.setText(String.valueOf(productsList.get(positi
on).getPrice()));
        holder.amountOfProduct.setText(String.valueOf(productsList.get(positi
on).getProductAmount()));
        Picasso.get().load(Uri.parse(productsList.get(position).getImage()))
.into(holder.imageOfProduct);
        holder.favouriteButton.setOnClickListener(new View.OnClickListener()
{@Override
        public void onClick(View v) {
            buttonClickListener.onButtonClick(productsList.get(position));
        }});
        holder.itemView.setOnClickListener(view ->{
            prodItemClickListener.onItemClick(productsList.get(position));});}
    @Override
    public int getItemCount() {
        return productsList.size();}
```

Лістинг Б.2 - Activity для додавання продукту

```
public class AddProductActivity extends AppCompatActivity {
```

```

private EditText addProductName, addProductPrice, addProductWeight,
addProductAmount, addProductAllergen, addProductCategory;
private Button addProduct, closeAddProductActivity;
private ImageView addProductImage;
private DatabaseReference mDatabase;
private StorageReference mStorageRef;
private Uri uploadUri;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_product);
    initiation();
    addProductImage.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {getImageFromSys();});
    addProduct.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String categoryOfProduct = addProductCategory.getText().toString();
            String nameOfProduct = addProductName.getText().toString();
            String allergenOfProduct = addProductAllergen.getText().toString();
            double productPrice =
            Double.parseDouble(addProductPrice.getText().toString());
            double productWeight =
            Double.parseDouble(addProductWeight.getText().toString());
            int productAmount =
            Integer.parseInt(addProductAmount.getText().toString());
            if(categoryOfProduct.isEmpty() || nameOfProduct.isEmpty() || allergenOfP
            roduct.isEmpty() || productPrice==0.0 || productWeight==0.0 || productAmou
            nt==0) {Toast.makeText(AddProductActivity.this, "Incorrect DATA,
            please check your input text", Toast.LENGTH_SHORT).show();} else {
            Product prod = new Product(categoryOfProduct, nameOfProduct,
            allergenOfProduct, productPrice, productWeight, productAmount,
            uploadUri.toString());
            mDatabase.child(categoryOfProduct).child(nameOfProduct).setValue(pro
            d);
            ValueEventListener registrationListener = new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    if (dataSnapshot.exists()) {
                        Toast.makeText(AddProductActivity.this, "Your Product save in DB",
                        Toast.LENGTH_SHORT).show();
                        Intent returnIntent = new Intent(AddProductActivity.this,
                        ActivityShopAdmin.class);
                        returnIntent.putExtra("result", "Some result");
                        setResult(Activity.RESULT_OK, returnIntent);
                        finish();
                    } else {
                        Toast.makeText(AddProductActivity.this, "Your Product doesn't save
                        in DB", Toast.LENGTH_SHORT).show();}
                    }
                @Override
                public void onCancelled(DatabaseError databaseError) {

```

```

Toast.makeText(AddProductActivity.this, "Your Product doesn't save
in DB", Toast.LENGTH_SHORT).show();});
mDatabase.addListenerForSingleValueEvent(registrationListener);}}});
closeAddProductActivity.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View v) {finish();}}});
@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
super.onActivityResult(requestCode, resultCode, data);
if(requestCode==595 && data!=null && data.getData() != null){
if(resultCode==RESULT_OK){addProductImage.setImageURI(data.getData()
);
uploadImageUri();}}}
private void uploadImageUri(){
Bitmap bitmap = ((BitmapDrawable)
addProductImage.getDrawable()).getBitmap();
ByteArrayOutputStream baos =new ByteArrayOutputStream();
bitmap.compress(Bitmap.CompressFormat.PNG,100,baos);
byte[] byteArray = baos.toByteArray();
StorageReference mStRef =
mStorageRef.child(System.currentTimeMillis()+"My Image");
UploadTask up = mStRef.putBytes(byteArray);
Task<Uri> task = up.continueWithTask(new
Continuation<UploadTask.TaskSnapshot, Task<Uri>>() {
@Override
public Task<Uri> then(@NonNull Task<UploadTask.TaskSnapshot> task)
throws Exception {
return mStRef.getDownloadUrl();}}).addOnCompleteListener(new
OnCompleteListener<Uri>() {
@Override
public void onComplete(@NonNull Task<Uri> task) {
uploadUri = task.getResult();}}});
private void getImageFromSys() {
Intent intentChoosePhoto = new Intent();
intentChoosePhoto.setType("image/*");
startActivityForResult(intentChoosePhoto,595);

```

Лістинг Б.3 - Activity з детальною інформацією про продукт

```

Bundle arguments = getIntent().getExtras();
Product productDetail;
if(arguments!=null){
productDetail = (Product)
arguments.getSerializable("productDetail");
userDetail = (User) arguments.getSerializable("user");
nameOfProductDetail.setText(productDetail.getName());
priceOfProductDetail.setText(String.valueOf(productDetail.getPrice()
));
weightOfProductDetail.setText(String.valueOf(productDetail.getWeight
()));

```

```

productAmountDetail.setText(String.valueOf(productDetail.getProductAmount()));
productAllergenDetail.setText(productDetail.getAllergen());
Picasso.get().load(productDetail.getImage()).into(imageOfProduct);
weightOfProd = buyAmount*productDetail.getWeight();
priceOfProd = buyAmount*productDetail.getPrice();
plusAmount.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        buyAmount++;
        if(buyAmount>productDetail.getProductAmount()){
            Toast.makeText(ProductDetailInfActivity.this, "No more products",
            Toast.LENGTH_SHORT).show();}else{
            amountOfBuyDetail.setText(String.valueOf(buyAmount));
            String formattedDouble =
            String.format("%.2f",buyAmount*productDetail.getPrice() );
            String formattedDoubleWeight =
            String.format("%.2f",buyAmount*productDetail.getWeight() );
            weightOfProd = buyAmount*productDetail.getWeight();
            priceOfProd = buyAmount*productDetail.getPrice();
            priceOfProductDetail.setText(formattedDouble);
            weightOfProductDetail.setText(formattedDoubleWeight);}}});
minusAmount.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(buyAmount>0){
            buyAmount--;
            amountOfBuyDetail.setText(String.valueOf(buyAmount));
            String formattedDouble =
            String.format("%.2f",buyAmount*productDetail.getPrice() );
            String formattedDoubleWeight =
            String.format("%.2f",buyAmount*productDetail.getWeight() );
            weightOfProd = buyAmount*productDetail.getWeight();
            priceOfProd = buyAmount*productDetail.getPrice();
            priceOfProductDetail.setText(formattedDouble);
            weightOfProductDetail.setText(formattedDoubleWeight);
        } else{
            Toast.makeText(ProductDetailInfActivity.this, "Impossible
            operation", Toast.LENGTH_SHORT).show();}}});
addToFavouriteProductDetail.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FirebaseDatabase userDBFav=FirebaseDatabase.getInstance();
        DatabaseReference userFavDR =
        userDBFav.getReference().child("users");
        userFavDR.child(userDetail.getEmail().replace(".", ",")).child("favourite")
        .child(productDetail.getName()).setValue(productDetail);}}});
buyProduct.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Product productBuy = new Product();
        FirebaseDatabase userDBFav=FirebaseDatabase.getInstance();

```

```

DatabaseReference userFavDR =
userDBFav.getReference().child("users");
productBuy.setName(productDetail.getName());
productBuy.setProductAmount(productDetail.getProductAmount());
productBuy.setAllergen(productDetail.getAllergen());
productBuy.setCategory(productDetail.getCategory());
productBuy.setImage(productDetail.getImage());
productBuy.setWeight(weightOfProd);
productBuy.setPrice(priceOfProd);
userFavDR.child(userDetail.getEmail().replace(".", ",")).child("shopping cart").
child(productBuy.getName()).setValue(productBuy);
int totalAmount = productDetail.getProductAmount()-buyAmount;
FirebaseDatabase newInfProd=FirebaseDatabase.getInstance();
DatabaseReference newInfProdAmount =
newInfProd.getReference().child("category");
newInfProdAmount.child(productDetail.getCategory()).child(productDetail.getName()).child("productAmount").setValue(totalAmount);
Toast.makeText(ProductDetailInfActivity.this,
productDetail.getCategory(), Toast.LENGTH_SHORT).show();
Toast.makeText(ProductDetailInfActivity.this,
productDetail.getName(), Toast.LENGTH_SHORT).show();
Toast.makeText(ProductDetailInfActivity.this,
String.valueOf(newInfProdAmount.child(productDetail.getCategory()).child(productDetail.getName()).child("productAmount")),
Toast.LENGTH_SHORT).show();
productAmountDetail.setText(String.valueOf(totalAmount));});});

```

ДОДАТОК В

Лістинг В.1 - Activity з корзиною покупок

```

Bundle arguments = getIntent().getExtras();
if(arguments!=null){
    userInf = (User) arguments.getSerializable("user");
    email = userInf.getEmail().replace(".", ",");
    mDatabase =
    FirebaseDatabase.getInstance().getReference().child("users").child(e
    mail).child("shopping cart");}
    initiation();
    shopListProd = new ArrayList<Product>();
    mDatabase.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
    for (DataSnapshot categorySnapshot : dataSnapshot.getChildren()) {
    if(indexOfUse==0){
    shopListProd.add(categorySnapshot.getValue(Product.class));
    totalPrice+=categorySnapshot.getValue(Product.class).getPrice();
    myListShoppingAdapter.notifyDataSetChanged();
    buyAllButton.setText("Total price: "+String.valueOf(totalPrice)+"
    UAH");}}}
    @Override
    public void onCancelled(DatabaseError databaseError) {}));
    myListShoppingAdapter = new
    ListOfShoppingAdapter(getApplicationContext(), shopListProd, new
    ListOfShoppingAdapter.ItemClickListener() {
    @Override
    public void onItemClick(Product product) {
    shopListProd.clear();
    Intent intent = new Intent(ShoppingCart.this,
    ProductDetailInfActivity.class);
    intent.putExtra("productDetail",product);
    startActivity(intent);}}, new
    ListOfShoppingAdapter.ButtonClickListener() {
    @Override
    public void onClick(Product product) {
    shopListProd.remove(product);
    totalPrice-=product.getPrice();
    buyAllButton.setText("Total price:"+String.valueOf(totalPrice));
    mDatabase.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
    for (DataSnapshot categorySnapshot : dataSnapshot.getChildren()) {
    if(product.getName().equals(categorySnapshot.getValue(Product.class)
    .getName())){
    mDatabase.child(categorySnapshot.getKey()).removeValue();
    indexOfUse++;}}}
    @Override
    public void onCancelled(DatabaseError databaseError) {}));
    myListShoppingAdapter.notifyDataSetChanged();});

```

```
productsRecycler.setLayoutManager(new LinearLayoutManager(this));
productsRecycler.setAdapter(myListShoppingAdapter);
```

Лістинг В.2 - Activity для кур'єра

```
Bundle arguments = getIntent().getExtras();
if(arguments!=null){
    userInf = (User) arguments.getSerializable("user");
    role = userInf.getRole();}
driverID = userInf.getEmail().replace(".", "", "");
LogoutDriverButton = Button)findViewById(R.id.driver_logout_button);
SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
LogoutDriverButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        currentLogoutDriverStatus = true;
        finish();
        DisconnectDriver();});}
getAssignedCustomerRequest();}
private void getAssignedCustomerRequest() {
    assignedCustomerRef = FirebaseDatabase.getInstance().getReference()
        .child("Users").child("Drivers").child(driverID).child("CustomerRide
        ID");
    assignedCustomerRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                {customerID = dataSnapshot.getValue().toString();
                getAssignedCustomerPosition();}
            }
            else {customerID = "";}
            if (PickUpMarker!=null){PickUpMarker.remove();}
            if(AssignedCustomerPositionListener!= null)
            {AssignedCustomerPositionRef.removeEventListener(AssignedCustomerPos
            itionListener);}}}
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}));}
private void getAssignedCustomerPosition() {
    AssignedCustomerPositionRef =
    FirebaseDatabase.getInstance().getReference().child("Customers
    Requests").child(customerID).child("1");
    AssignedCustomerPositionListener =
    AssignedCustomerPositionRef.addValueEventListener(new
    ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                List<Object> customerPositionMap = (List<Object>)
                dataSnapshot.getValue();
                double locationLat =
                Double.parseDouble(customerPositionMap.get(0).toString());
```

```

double locationLng =
Double.parseDouble(customerPositionMap.get(1).toString());
LatLng DriverLatLng = new LatLng(locationLat, locationLng);
PickUpMarker = mMap.addMarker(new
MarkerOptions().position(DriverLatLng).title(userInf.getEmail()));
mMap.moveCamera(CameraUpdateFactory.newLatLng(DriverLatLng));
mMap.animateCamera(CameraUpdateFactory.zoomTo(11));}
@Override
public void onCancelled(@NonNull DatabaseError databaseError) {});}
@Override
public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;
buildGoogleApiClient();
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {return;}
mMap.setMyLocationEnabled(true);}
@Override
public void onConnected(@Nullable Bundle bundle) {
locationRequest = new LocationRequest();
locationRequest.setInterval(100000);
locationRequest.setFastestInterval(100000);
locationRequest.setPriority(locationRequest.PRIORITY_HIGH_ACCURACY);
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {return;}
LocationServices.FusedLocationApi.requestLocationUpdates(googleApiCl
ient, locationRequest, this);}
@Override
public void onConnectionSuspended(int i) {}
@Override
public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {}
@Override
public void onLocationChanged(Location location)
{if(getApplicationContext() != null){
lastLocation = location;
LatLng latLng = new LatLng(location.getLatitude(),
location.getLongitude());
mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
mMap.animateCamera(CameraUpdateFactory.zoomTo(20));
//driverID = userInf.getEmail();
DatabaseReference DriverAvalablityRef =
FirebaseDatabase.getInstance().getReference().child("Driver
Available");
GeoFire geoFireAvalablity = new GeoFire(DriverAvalablityRef);

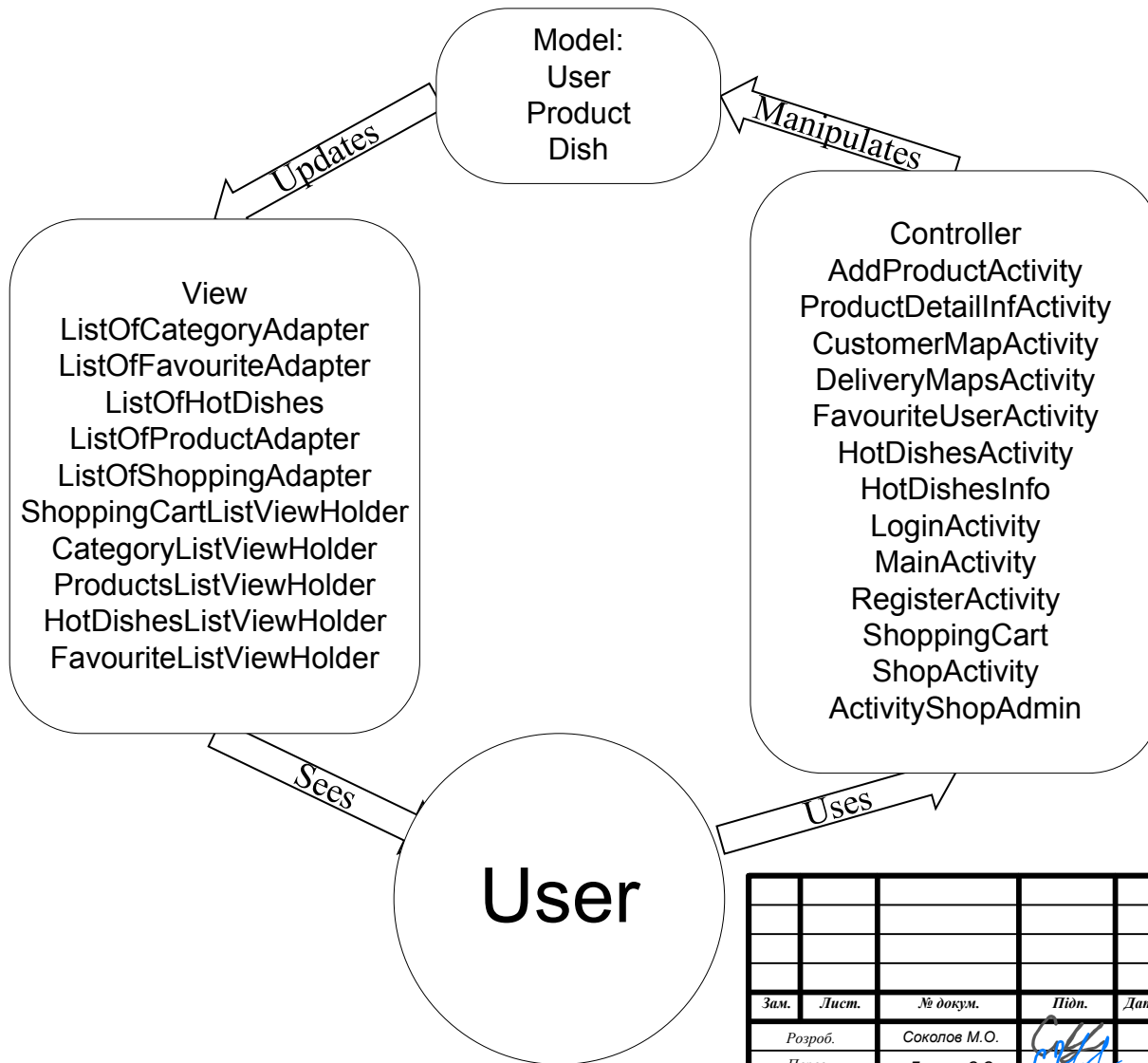
```

```

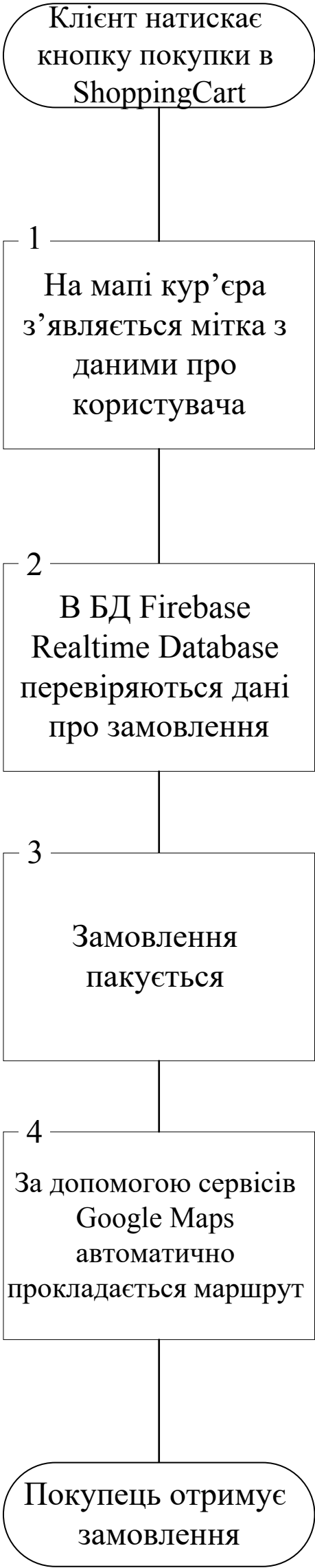
DatabaseReference DriverWorkingRef =
FirebaseDatabase.getInstance().getReference().child("Driver
Working");
GeoFire geoFireWorking = new GeoFire(DriverWorkingRef);
switch (customerID)
{case "":
geoFireWorking.removeLocation(driverID);
geoFireAvalablity.setLocation(driverID, new
GeoLocation(location.getLatitude(), location.getLongitude()));
break;
default:
geoFireAvalablity.removeLocation(driverID);
geoFireWorking.setLocation(driverID, new
GeoLocation(location.getLatitude(), location.getLongitude()));
break;}}}
protected synchronized void buildGoogleApiClient()
{googleApiClient = new GoogleApiClient.Builder(this)
.addConnectionCallbacks(this).addOnConnectionFailedListener(this)
.addApi(LocationServices.API).build();
googleApiClient.connect();
}
@Override
protected void onStop() {
super.onStop();
if (!currentLogoutDriverStatus) {
DisconnectDriver();}}
private void DisconnectDriver(){
DatabaseReference DriverAvalablityRef =
FirebaseDatabase.getInstance().getReference().child("Driver
Available");
GeoFire geoFire = new GeoFire(DriverAvalablityRef);
geoFire.removeLocation(driverID);
}

```

Схема взаємодії між елементами архітектурного паттерну MVC



					13.02070849.51918 ПЛ1		
					Розробка Android-застосунку магазину з доставкою. Схема взаємодії між елементами архітектурного паттерну MVC		
Зам.	Лист.	№ докум.	Підп.	Дата	Лит.	Маса	Масшт.
Розроб.		Соколов М.О.	<i>[Signature]</i>				
Перев.		Грушко С.С.	<i>[Signature]</i>				
Т.контр.					Лист 1	Листів 1	
Н.контр.		Щербак Н.В.	<i>[Signature]</i>		НУ «Запорізька політехніка»		
Затв.		Кудерметов Р.К.	<i>[Signature]</i>		КНТ-519		



					13.02070849.51918 A1											
					Розробка Android-Застосунку магазину з доставко. Алгоритм процесу покупки товару						Лист.		Маса		Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата												
Розроб.		Соколов М.О.														
Перев.		Грушко С.С.														
Т.контр.											Лист 1		Листів 1			
Н.контр.		Щербак Н.В.									НУ «Запорізька політехніка» КНТ-519					
Затв.		Кудерметов Р.К.														

Макет каталогу товарів

Додавання товару

Категорії продукції

Категорії продукції

Перемикання ролі

Додати категорію

Продукт

Продукт

Продукт

Продукт

Каталог

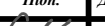
Обране

Кошик

Доставка

Кабінет

Страви

					13.02070849.51918 ПЛ2										
					Розробка Android-Застосунку магазину з доставко. Макет каталогу товарів					Лист.		Маса		Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата											
Розроб.		Соколов М.О.													
Перев.		Грушко С.С.													
Т.контр.															
										Лист 1		Листів 1			
Н.контр.		Щербак Н.В.													
Затв.		Кудерметов Р.К.													
					НУ «Запорізька політехніка» КНТ-519										