Київський національний торговельно-економічний університет

Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка мобільного додатку для інтернет-магазину «FastShop» на платформі Android»

Студента 4 курсу, 7 групи,
спеціальності 121 «Інженерія програмного забезпечення» _____ Колошка Олександра Володимировича
підпис студента

Науковий керівник
кандидат технічних наук,
професор _____ Пашорін Валерій Іванович
підпис керівника

Гарант освітньої програми
кандидат технічних наук,
доцент _____ Цензура Микола Олександрович
підпис керівника

КИЇВ – 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 Інженерія програмного забезпечення

Затверджую
Зав. кафедри інженерії
програмного забезпечення
та кібербезпеки
Криворучко О.В.
"20" жовтня 2020 р.

Завдання
на випускну кваліфікаційну роботу студентові

Колошку Олександру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи «Розробка мобільного додатку для інтернет-магазину «FastShop» на платформі Android

Затверджена наказом ректора від "30" жовтня 2020 р. № 3225

2. Строк здачі студентом закінченої роботи

3. Цільова установка та вихідні дані до роботи

Мета роботи створення мобільного додатку для функціонування інтернет-магазину

Об'єкт дослідження процес розробки мобільних додатків

Предмет дослідження методи та алгоритми створення мобільних додатків

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант	Підпис, дата	
№	(прізвище, ініціали)	Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ СКЛАДОВІ ФУНКЦІОНУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ТА РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

1.1. Аналіз предметної області

1.2. Що необхідно для створення та функціонування інтернет-магазину

1.3. Особливості створення мобільних додатків для інтернет-магазинів

1.4. Висновки до розділу 1

РОЗДІЛ 2. АНАЛІЗ ПРЕДМЕТА ДОСЛІДЖЕННЯ, ОПИС ЙОГО ПАРАМЕТРІВ ТА ХАРАКТЕРИСТИК

2.1. Постановка задачі

2.2. Проектування програмного забезпечення

2.3. Моделювання бази даних

2.4. Висновки до розділу 2

РОЗДІЛ 3. ОПИС ЗАПРОПОНОВАНОГО ТЕХНІЧНОГО РІШЕНЯ

3.1. Програмні засоби котрі використані при розробці продукту

3.2. Розробка серверної частини програми

3.3. Розробка клієнтського додатку Android

3.4. Висновки до розділу 1

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

1. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	*21.09.2020*	*21.09.2020*
2.	*Вступ та перелік літературних джерел*	*14.12.2020*	*14.12.2020*
3.	*Розділ 1. «Теоретичні складові функціонування інтернет-магазину та розробки мобільних додатків»*	*19.02.2021*	*19.02.2021*
4.	*Розділ 2. «Аналіз предмета дослідження опис його параметрів та характеристик»*	*05.03.2021*	*05.03.2021*
5.	*Розділ 3. «Опис запропонованого технічного рішення»*	*10.04.2021*	*10.04.2021*
6.	*Висновки*	*24.04.2021*	*24.04.2021*
7.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*	*14.05.2021*	*14.05.2021*
8.	*Підготовка автореферату та презентації доповіді*	*17.05.2021*	*17.05.2021*
9.	*Попередній захист випускної кваліфікаційної роботи*	*18.05.2021 – 21.05.2021*	*21.05.2021*
10.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*01.06.2021*	*01.06.2021*
11.	*Здача прошитої випускної кваліфікаційної роботи на кафедру*	*02.06.2021*	*02.06.2021*
12.	*Публічний захист випускної кваліфікаційної роботи*		

2. Дата видачі завдання «20» жовтня 2020 р.

3. Науковий керівник випускної кваліфікаційної роботи Пашорін В.І.
(прізвище, ініціали, підпис)

4. Гарант освітньої програми Цензура М. О.
(прізвище, ініціали, підпис)

5. Завдання прийняв до виконання студент Колошко О.В.
(прізвище, ініціали, підпис)

6. Відгук керівника випускної кваліфікаційної роботи

Науковий керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

Відмітка про попередній
захист _____

(ПІБ, підпис, дата)

7. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Колошко О.В.

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Цензура М. О.

(прізвище, ініціали, підпис)

Завідувач кафедри Криворучко О. В.

(підпис, прізвище, ініціали)

«____»_____ 2021 р.

Анотація

Дипломна робота на тему «Розробка мобільного додатку для інтернет-магазину «FastShop» на платформі Android» містить 50 сторінок, 14 рисунків та 2 таблиці. Перелік посилань налічує 15 найменувань.

Метою випускної кваліфікаційної роботи є створення додатку для функціонування мобільного інтернет-магазину.

Об’єкт дослідження: процес розробки мобільних додатків.

Предмет дослідження: методи та алгоритми створення мобільних додатків.

В першому розділі «Теоретичні складові функціонування інтернет-магазину та розробки мобільних додатків» розкривається організаційна сутність задачі і коло задач, які повинна виконувати програма.

У другому розділі «Аналіз предмета дослідження, опис його параметрів та характеристик» міститься багаторівневе представлення розробленого програмного забезпечення, що надає можливість осягнути представлену роботу.

Третій розділ «Розробка клієнтського додатку» включає загальний перелік отриманих при виконанні дипломної роботи результатів. У ньому описується міра реалізації зобов’язань та інших вимог, визначених технічним завданням.

У висновках проаналізовано створене програмне забезпечення, визначена ступінь відповідності поставленої задачі та виконаної роботи.

ANNOTATION

Thesis on the topic "Development of a mobile application for the online store "FastShop" on the Android platform" contains 50 pages, 14 figures and 2 tables. The list of links includes 15 names.

The purpose of the final qualifying work is to create an application for the operation of a mobile online store.

Object of research: the process of developing mobile applications.

Subject of research: methods and algorithms for creating mobile applications.

The first section "Theoretical components of the online store and the development of mobile applications" reveals the organizational essence of the task and the range of tasks that must be performed by the program.

The second section "Analysis of the subject of research, description of its parameters and characteristics" contains a multi-level presentation of the developed software, which provides an opportunity to comprehend the presented work.

The third section "Development of the client application" includes the general list of the results received at performance of the diploma work. It describes the extent to which the obligations and other requirements specified in the terms of reference have been met.

In the conclusions the created software is analyzed, the degree of conformity of the set task and the performed work is define.

ЗМІСТ

					КНТЕУ 121 07-09.БР		
Зм.	*Аркуш*	*№ докум*	*Підпис*	*Дата*	*Розробка мобільного додатку для інтернет-магазину «FastShop» на платформі Android*	*Стадія*	Аркуш	Аркушів
Зав. кафедри		Криворучко О.В.				Зміст	2	50
Керівник		Пашорін В.І.				Факультет інформаційних технологій, 4 курс, 7 група		
Гарант		Цензура М.О.						
Розроб.		Колошко О.В.			ЗМІСТ			

ВСТУП

Сьогодні, за часів постійної метушні і швидкого ритму життя, люди мають все менше часу купувати товари в звичних магазинах. До того ж інколи, щоб знайти необхідний товар, потрібно обійти декілька точок продажу. На зміну традиційним магазинам стали приходити інтернет-магазини, реалізовані через веб-сайт, проте наразі не у всіх людей є комп'ютер для доступу до цих сайтів. У той же час у кожної людини вже є особистий мобільний пристрій. Для вирішення цієї проблеми вирішено розробити мобільний додаток інтернет-магазину FastShop на платформі Android.

Мета дипломної роботи полягає у розробці програмного забезпечення Android-додатку, яке допоможе швидко знайти та замовити необхідний товар з мобільного пристрою.

Основними функціями такого ПЗ будуть:

- реєстрація нового облікового запису;

- перегляд каталогу товарів;

- формування корзини замовлення;

- оформлення замовлення.

Актуальність теми дипломної роботи пов'язана зі стрімким ростом популярності інтернет-магазинів в Україні.

Об'єкт дослідження: процес розробки мобільних додатків.

Предмет дослідження: методи та алгоритми створення мобільних додатків.

РОЗДІЛ 1

ТЕОРЕТИЧНІ СКЛАДОВІ ФУНКЦІОНУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ТА РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

1.1 Аналіз предметної області

Для початку необхідно дати визначення дефініції «інтернет-магазин». У загальному випадку - це веб-ресурс, який займається продажами товарів та послуг в глобальній мережі. Завдяки подібному сайту, середньостатистичний користувач має можливість створити замовлення, вибрати спосіб оплати, вказати особисті дані і в результаті отримати бажаний товар.

Для того, щоб інформаційна система могла визначити особу користувача, йому необхідно вказувати свої дані. Відразу після оформлення замовлення клієнт, отримує можливість виконати оплату. Інформація про користувача, як правило, зберігається в базі даних сайту для того, щоб в подальшому, він мав можливість створювати замовлення в найкоротші терміни. Крім того, компанії, що створюють інтернет-магазини на довгострокову перспективу, повинні містити функціонал, пов'язаний з відстеженням відправки товару, і детальну історію покупок.

Сучасні інтернет-магазини можуть створюватися, як «з нуля» , так і за допомогою систем управління контентом, більш відомих, як CMS. Наприклад, WordPress і OpenCart містять модулі, що дозволяють додавати, редагувати всілякі товари. Крім того, для поліпшення якості інтерфейсу, можуть використовуватися плагіни, створені безпосередньо користувачами системи.

Великі компанії використовують спеціально створені для них інформаційні системи. У свою чергу, малі і середні магазини акцентують свою увагу на програмному забезпеченні, що знаходиться у вільному доступі. Яскравим прикладом такого ПЗ є osCommerce.

Варто відзначити, що CMS Інтернет-Магазину може бути представлена, як коробковий продукт, безпосередньо пов'язаним з конкретним хостинг-майданчиком. Крім того, він може розроблятися в рамках веб-студії, яка виконує всі необхідні дії по просуванню і адаптації в глобальній мережі.

Інтерфейс інтернет-магазину, призначений для адміністраторів ресурсу, повинен знаходитися на невидимій для гостей частині сайту. Така частина також називається бек-офісом. Вона включає дані для торгового, податкового та бухгалтерського обліку. Більшість сучасних інтернет-магазинів інтегровані з спеціалізованими системами обліку.

На сьогоднішній день, найбільш популярними є два види інтернет-магазинів, а саме:

1) Інтернет-магазин, як додатковий майданчик для реально існуючої компанії. Такий ресурс функціонує заради продажу товару безпосередньо зі складу. Як правило, на офіційному сайті компанії вказують нижчу ціну, порівнюючи з наземними пунктами продажів.

2) Інтернет-магазин, який займається перепродажем товарів. Актуальний напрямок для українського ринку, адже в межах країни можна продати товар з націнкою в 400-500 відсотків. Як правило, продаж такого типу може бути актуальним не тільки всередині країни, а й на міжнародній арені. В останньому випадку, інтернет-магазин отримує прибуток за рахунок комісій, що оплачують продавці за додавання власного товару на сайт. Тобто, в сучасному світі інтернет-магазин може бути гарантом угоди між сторонами (певним продавцем і його цільовою аудиторією). Для фільтрації продавців і постачальників, на таких

платформах використовується система «репутація». В її рамках, певний покупець може відправити скаргу представникам сайту на постачальника, тим самим отримавши можливість повернути гроші, або отримати інший товар взамін.

Інтернет-магазини можуть відрізнятися і в контексті способу продажу:

1) Сайти з фіксованою ціною товару, де замовник може додатково, за власним бажанням, оплатити доставку товару до необхідного місця. Багато великих інтернет-магазинів втрачають значний трафік через те, що не дивлячись на низьку вартість товарів, публікують завищену ціну на доставку, тим самим мотивуючи покупців шукати інший ресурс для покупки.

2) Сайти, що містять систему аукціону. На кожен доданий в систему товар оголошується аукціон, в якому можуть взяти участь всі бажаючі користувачі. Продавець на такому сайті може виставити початкову вартість, а також розмір оплати, в рамках якого продаж відбудеться без торгів. Інтернет-магазини подібного типу менш розвинені на українському ринку, але в подальшому можна очікувати, що отримавши досвід західних колег, вітчизняні підприємці перейдуть і на таку модель.

Головна перевага інтернет-магазинів полягає в не прив'язаності до конкретної точки продажу. Адже, по причині того, що їх власникам немає необхідності платити значні гроші на оренду приміщень, вони можуть скорочувати вартість на товари. Як підсумок, більшість товарів можна придбати за ціною в 1,5-2 рази нижче, ніж в великих мережах магазинів.

1.2 Що необхідно для створення та функціонування Інтернет-магазину

Розвиток онлайн-продажів в Україні в 2013-2014 рр. спонукало невеликі роздрібні магазини відкрити свої точки продажів в інтернеті. Однак багато хто з них до 2019-2020 рр. закрився, так як не всі роздрібні торговці змогли пристосуватися до нових правил.

Все пов’язано з особливостями функціонування наземних магазинів, та магазинів в глобальній мережі. Ключова особливість традиційного магазину - його вітрини і викладка речей на полицях. Особливість інтернет-магазину - головна сторінка, де опиняється користувач при переході по посиланню, та безпосередньо уся необхідна інформація про товар.

Рішення про покупку відвідувач приймає, виходячи з опису продукції, фото, виду самого інтернет-магазину. Головна відмінність інтернет-магазину від звичайної точки роздрібного продажу в неможливості подивитися на товар вживу до його отримання.

Саме тому багато споживачів ще не готові купувати, наприклад, одяг на сайтах. Вони бояться не вгадати з розміром або реальним відтінком. Покупки через Інтернет швидкопсувних продуктів, наприклад, фруктів і овочів, також пов'язані з ризиком: зображення на сторінці може не збігтися з реальним якістю.

На новий рівень функціонування інтернет-магазинів вийшло після початку пандемії COVID-19. Унаслідок карантину, більшість магазинів не мало можливість продавати свої товари.

Єдиним виходом для них був перехід в онлайн, внаслідок чого вони намагалися вибудувати взаємодію з цільовою аудиторією. Компанії, що робили подібний перехід раніше конкурентів, отримували необхідні переваги, в тому числі і додаткову цільову аудиторію.

Не останню роль у роботі сучасного інтернет-магазину відіграє наявність додаткового програмного забезпечення, що дозволяє взаємодіяти з цільовою аудиторією, за допомогою мобільних пристроїв. Адаптація роботи офіційного сайту під основні мобільні операційні системи, дасть можливість клієнтам купувати товари безпосередньо через смартфони, або планшети.

Це значно скорочує час на покупку, тим самим збільшуючи прибуток компанії. Також, важливо враховувати наявність інтуїтивно зрозумілого інтерфейсу, що дозволяє взаємодіяти з системою найбільшій кількості потенційних клієнтів компанії.

Кількість мобільних додатків для підтримки діяльності інтернет-магазинів в Україні, скоріш за все, буде збільшуватися і надалі. Завдяки досвіду карантинних обмежень, компанії отримають загальне розуміння про перевагу функціонування бізнесу в глобальній мережі, тим самим приділяючи цьому напрямку більше уваги.

Як інтернет-магазини працюють над вирішенням проблеми неясного для споживача якості? По-перше, вони пропонують безкоштовно повернути в магазин річ, якщо вона не підійшла за розміром, кольором або виду. По-друге, влаштовують шоу-руми, в яких потенційний покупець може побачити товар, щоб визначитися з покупкою. Якщо відкрити точку продажів нереально, в інтернет-магазині публікуються додаткові фотографії виробів і їх 3d-моделі. Наприклад, такий спосіб візуального просування впливає на продажу взуття.

Далі, розглянемо дії, що необхідно виконати для відкриття інтернет-магазину:

1. Вибрати нішу. Для цього треба зрозуміти, яку проблему споживача можна вирішити, відкривши свій онлайн-бізнес. Інтернет-магазини, що працюють по дропшиппінгу (безпосередньо від постачальника), можуть тримати ціни близькі до оптових, тобто вирішувати проблему покупок якісних товарів за низькими цінами. Визначення ніші дозволить вибрати групу товарів для продажу.

2. Провести аналіз конкурентів. Таким чином обчислюються їх сильні і слабкі сторони, як їх можна обійти в боротьбі за покупця.

3. Знайти постачальників: в Інтернеті через пошукові системи, в каталогах постачальників, на ярмарках і виставках, на ресурсах конкурентів, серед місцевих виробників. Їх можна поділити на дві групи, в залежності від того, як працює інтернет-магазин з постачальниками: дропшіппінг-постачальники (для торгівлі без складу, відправляють річ безпосередньо покупцеві, отримуючи гроші вже за фактом продажу); оптові постачальники (відвантаження зі складу після передоплати). Всім потенційним партнерам слід зателефонувати або написати, щоб запитати прайс-лист і умови роботи.

4. Виходячи з наявних оптових цін і цінових пропозицій конкурентів, можна приблизно підрахувати можливу рентабельність бізнесу в цій сфері.

5. Вибрати доменне ім'я, та систему для розробки інтернет-магазину.

6. Розробити сайт, підготувати дизайн, розмістити фотографії та супутню інформацію.

7. Орендувати хостинг, розмістити сайт на хостингу.

8. Зареєструвати ПП або ТОВ і вибрати систему оподаткування.

9. Визначитися зі способами доставки, з якими працює інтернет-магазин. Замовлення можна надсилати за допомогою:

- кур'єра (дорого, але надійно);
- пошти (відносно дешево, не дуже надійно, зате в будь-яку населену точку України);
- через пункти видачі замовлень (можна домовитися з іншими традиційними торговими точками, використовуючи їх площі для цієї мети);
- Дропшиппінг (річ відправляє постачальник, завдання власника інтернет-магазину - тільки прийняти замовлення).

10. Визначитися зі способом оплати (готівковий і безготівковий розрахунок з кур'єром, готівковий і безготівковий розрахунок безпосередньо в межах компанії, безготівковий на сайті).

1.3 Особливості створення мобільних додатків інтернет-магазинів

Інтернет-магазини можуть створити ще один інструмент для продажу своїх товарів і послуг. Якщо покупець уже використовує смартфон або планшет для пошуку і вивчення товарів, то бажано надати йому можливість за їх допомогою виконати необхідні дії для покупки.

Мобільний додаток для інтернет-магазину дає можливість компанії підвищити продажі, створити ще один канал залучення покупців і отримати ефективний інструмент для збільшення повторної взаємодії з цільовою аудиторією.

На сьогодні вагомий відсоток користувачів регулярно здійснює покупки через мобільні пристрої. Виходячи з цього, розробка та впровадження програмного забезпечення для мобільних операційних систем, надасть компанії можливості для:

- Збільшення продажів і зростання аудиторії;
- Збільшення лояльності клієнтів;
- Зростання повторних продажів;
- Інформування клієнтів про пропозиції та акції компанії;
- Оптимізація внутрішніх бізнес-процесів.

У свою чергу, клієнт отримує мобільну версію інтернет-магазину, яка може приймати і обробляти замовлення з будь-яких мобільних пристроїв. Безсумнівна перевага додатків для покупок - налаштовані сервіси і структура звичних інтерфейсів типового інтернет-магазину.

До важливих функцій, що значно збільшують зацікавленість потенційних покупців, потрібно віднести:

- Каталог з можливістю пошуку;
- "Розумний" фільтр пошуку по товарах і опцій;
- Різні види прийому платежів;
- Детальний опис товару;
- Спілкування з клієнтами за допомогою push-повідомлень;
- Історія замовлень і переглядів;
- Облік геолокації.

Створення мобільного застосування для інтернет-магазину повинно починатися з бізнес-експертизи. Розробник повинен провести аналіз наявних програм, вивчити відгуки, кількість версій, дати останніх оновлень.

Далі, на прототипі потрібно провести моделювання усіх можливих для користувача сценаріїв. Ключовою дією на цьому етапі є опис нетипових функцій і розробка макетів сторінок.

Такий підхід дозволяє визначити найбільш зручний формат програми та вибрати краще рішення ще до початку роботи над дизайном і розробкою програми.

Важливо ретельно опрацьовувати формат сторінки товарів, макет кошика, картки товарів і сторінки категорії. Особливу увагу бажано приділяти дизайну - він повинен бути зрозумілим користувачам.

Загальний процес роботи над мобільним додатком виглядає наступним чином:

- Ідея продукту і бізнес-експертиза

Представники компанії вирішують, що їм потрібне розширення наявного програмного забезпечення, а тому починають втілювати в життя ідею мобільного додатка, веб-ресурсу для того, щоб автоматизувати бізнес-процеси. Спеціалісти компанії повинні провести бізнес-експертизу, провести консультації з адміністрацією для того, щоб допрацювати ідейну складову, враховуючи усі наявні потреби.

- Оцінка проекту

Компанія повинна усвідомлювати, яку суму вона готова витратити на створення інтернет-магазину. Після того, як цей момент вирішений, необхідно зробити комерційну пропозицію розробникам.

- Створення прототипу

Розуміючи, чого чекають від програми користувачі, програмний відділ повинен приступити до створення прототипів майбутнього інтернет-магазину. Грамотно вибудуваний інтерфейс - запорука того, що користувач швидко зорієнтується, як додаток допоможе вирішити його проблему. Представники компанії, в свою чергу, отримують можливість подивитися функціонал майбутньої програми без програмної частини.

- Дизайн продукту

Потрібно розробити дизайн всіх екранів і станів елементів. Для збільшення попиту з боку цільової аудиторії, бажано дбати про те, щоб графічні рішення були зрозумілими і зручними. Кінцевий покупець отримує дизайн, який виділяється серед конкурентів.

- Документальна частина розробки

Цей етап напряму залежить від того, чи є в компанії окремий програмний відділ, або потрібно додатково шукати розробників.

•Планування програмного коду

Згідно з планом на розробку інформаційної систему, визначається, який саме функціонал буде наявний в кінцевому інтернет-магазині. Бажано, щоб додаток мав інтерфейс, адаптований до левової частини сучасних мобільних пристроїв.

•Просування додатку після розробки

Важливо, щоб кінцевий результат розробки, був у подальшому прорекламований за допомогою SEO та SMM-технології. Для цього, потрібно, щоб більша частина постійних клієнтів, у мінімальні терміни дізналася про створення нового програмного забезпечення компанії.

Далі, наведемо усі етапи взаємодії покупця з мобільним додатком інтернет-магазину:

1. Завантажує, інсталює та відвідує додаток магазину;
2. Шукає і вибирає товар;
3. Поміщає його в кошик;
4. Доповнює кошик іншими товарами; Оформляє замовлення;
5. Співробітник зв'язується з покупцем для підтвердження замовлення;
6. Відбувається оплата (оплачувати можна і після оформлення, але тоді власнику магазину треба бути впевненим, що все є в наявності);
7. Замовлення передається в службу доставки (виняток: Інтернет-магазини, що працюють по дропшипінгу, вони не займаються доставкою).

Всі вимоги до мобільного додатку інтернет-магазину зводяться до розробки зручного ресурсу для покупців. Загальний список виглядає так:

1. Зручність навігації по мобільному додатку. Важливо, щоб потенційні покупці мали можливість виконати усі необхідні дії за мінімальні відрізки часу;

2. Наявність логічного і зрозумілого з першого погляду каталогу. Сегментація за категоріями і класифікація за ознаками;

3. Зручність пошуку. Фільтри для пошуку груп;

4. Історія переглядів і збереження кошику, коли користувач покинув інтернет-магазин;

5. Статистика перегляду кожного товару окремо і, при високому попиті, кількість замовлень цієї речі;

6. Блок «з цим купують», «підійде по стилю». Рекомендації по схожим товарним групам;

7. Збільшення попиту буде мати місце, коли в межах мобільного додатку будуть міститься посилання на сторінки компанії в соціальних мережах;

8. Використання форму взаємодії між покупцем та адміністрацією ресурсу. Магазин може використовувати онлайн-чат з менеджером, форму заявки при відсутності речі на складі, кнопку «передзвонити», а також спливаючий банер зі знижкою при покиданні додатку;

9. Впровадження форми для відгуків; Наявність ретельно продуманої сторінки з умовами доставки. На ній повинні бути вказані терміни, ціни та інші умови;

10. Функціонал, що дозволяє проводити акції та розіграші призів, впроваджувати дисконтні програми та інші способи залучення і утримання замовників;

11. Можливість покупки без заповнення форми реєстрації.

Так як працює інтернет-магазин для збільшення прибутку, а не збору контактів, не варто нехтувати тими покупцями, які не хочуть залишати дані про

себе, крім номера телефону (для зв'язку). Зате покупцям, які заповнили форму реєстрації, можна пообіцяти додаткові бонуси.

Також важлива підтримка декількох способів оплати (на сайті і готівкою при отриманні). Продажі збільшаться, якщо в інтернет-магазині будуть приймати не тільки оплату банківськими картами, але і цифровими грошима.

Цільову аудиторію розроблений додаток може отримати чотирма способами:

1. Покупець сам зайшов в інтернет-магазин для покупки певного товару;

2. Покупець знайшов додаток інтернет-магазину в глобальній мережі, або на ресурсах партнерів компанії;

3. Покупець зіткнувся з рекламним оголошенням і перейшов по ньому;

4. Покупець прийшов до додатку після дзвінка менеджера з продажу.

Якщо клієнт вже використовує додаток інтернет-магазину, то можна відразу запропонувати йому оплатити товар, якщо він в наявності або може бути замовлений магазином у постачальника. Замовлення, зазвичай, підтверджується (поштою або по смс). З легкістю вести повноцінний складський облік допоможе система для інтернет-магазинів від вітчизняних розробників. Така система надасть можливості максимально автоматизувати процес інвентаризації та формування заявок на поставку товару, в один клік роздрукувати весь пакет супровідних документів на замовлення, в тому числі чек, накладну і рахунок-фактуру.

В додатку інтернет-магазину, що не має власного приміщення для зберігання, приходять такі ж клієнти, як і в інтернет-магазин зі складом. І ймовірно, вони навіть не знають, у якого магазина є склад, у якого немає, так як процес покупки не відрізняється. Розглянемо докладно, як працює додаток інтернет-магазин без складу, з моменту оплати клієнтом товару. Головна відмінність схеми в процесі поставки.

1. Резерв товару у постачальника після отримання грошей від клієнта;

2. Оплата послуг компанії-постачальника;

3. Компанія-постачальник віддає товар представнику інтернет-магазину (кур'єру або перевізнику);

4. Доставка замовлення; Клієнт підтверджує, що товар отриманий;

5. Покупцеві відправляються бонуси на наступну покупку і інші пост продажного акції.

Ця схема не відрізняється складністю і зменшує обов'язки інтернет-магазину. Власник такої торговельної точки тільки організовує інтернет-магазин, приймає замовлення і передає їх партнерам. Товар забирається зі складу постачальника і відправляється покупцеві. Плюс такої схеми в тому, що для торгівлі не потрібен значний стартовий капітал на закупівлю товару. Так як працює інтернет-магазин з постачальниками безпосередньо, зникає необхідність у моніторингу споживчого попиту і наявності власного приміщення. Можна вести бізнес прямо з дому.

1.4 Висновки до розділу 1

В ході виконання першого розділу проаналізовано актуальність функціонування інтернет-магазинів в Україні. Завдяки створенню та просуванню подібних ресурсів, користувач з будь-якої точки країни може придбати необхідний матеріал за адекватною ціною.

Розглянуто особливості функціонування інтернет-магазинів та мобільних додатків. Проаналізовано шляхи створення інтернет-магазину, основні вимоги до нього, ризики на які піддається власник.

РОЗДІЛ 2

АНАЛІЗ ПРЕДМЕТА ДОСЛІДЖЕННЯ, ОПИС ЙОГО ПАРАМЕТРІВ ТА ХАРАКТЕРИСТИК

2.1 Постановка задачі

Потрібно розробити мобільний додаток інтернет-магазину «FastShop» під управлінням операційної системи Android здатний проводити реєстрацію нових користувачів, відображати відомості про товари, приймати замовлення від клієнтів.

Розроблюваний додаток повинен містити таку основну інформацію:

- Інформація про користувачів;
- Інформація про товари;
- Інформація про категорії товарів;
- Інформація про замовлення;
- Інформація про доставку;
- Інформація про оплату.

Основні вимоги до клієнтського додатку «FastShop»:

1. Зручність інтерфейсу. Інтерфейс програми повинен бути інтуїтивно зрозумілий користувачеві та забезпечувати зручність роботи з програмою.
2. Задля того, що повноцінно використовувати додаток, користувач має пройти процедуру реєстрації. Лише після цього він зможе повноцінно використовувати функціонал програмного продукту.
3. Додаток повинен швидко реагувати на взаємодію з користувачем.

4. Додаток потрібно розробляти використовуючи модульну структуру, щоб у майбутньому швидко додати нові функціональні можливості, коли ми захочемо підвищити його складність.

5. Додаток має проводити перевірку даних, введених користувачем. Номер телефону має відповідати звичному стандарту, пароль повинен містити не менше 8 символів та бути регістрозалежним, в той час як логін – ні.

Вимоги до складу та параметрів технічних засобів:

Так як розробляється Android-додаток, то вимоги до складу та параметрів технічних засобів полягають у версії операційної системи Android та розмірі вільної системної пам’яті:

- Версія Android 5.0 та вище;
- Вільної пам’яті у системній пам’яті – не менше 20 мб;
- Доступ до мережі інтернет.

Для функціонування програми не потрібно жодного додаткового стороннього програмного забезпечення.

Програмне забезпечення повинно використовувати мінімум апаратних ресурсів користувача, тому вся основна інформація про користувачів та товари має зберігатися на окремому сервері у базі даних та завантажуватись за необхідністю, з урахуванням всіх вибраних фільтрів.

Вище описаний ідеальний варіант вимог для розробки мобільного додатку «FastShop».

2.2 Проектування програмного забезпечення

Під час проектування програмного забезпечення беруть до уваги в основному зовнішні атрибути, тобто докладний опис того, як буде застосовуватись майбутній продукт з точки зору користувача, не торкаючись

питання практичної реалізації. На цій стадії уточнюються і даються в більш формалізованому вигляді вимоги до виробу, визначаються його функції та проектується взаємодія з користувачем, розроблюються структури вхідних даних та вихідної інформації.

Програмне забезпечення розробляється для користувача, якому необхідно знайти та дізнатись докладну інформацію про товар, його ціну та характеристики, при необхідності – сформувати корзину товарів та оформити замовлення. Позначимо актора системи як «Користувач». Приведемо короткий опис акторів у таблиці 2.1.

Таблиця 2.1.

Актори розроблюваного ПЗ

Актор	Короткий опис
Користувач	Особа, котра реєструється, переглядає товари, оформлює замовлення з ОС Android.
Адміністратор	Особа, котра додає нові або коригує існуючі товари, приймає замовлення від Користувача.

Розроблюване програмне забезпечення має певні функції, які в мові UML відображаються у вигляді варіантів використання. Виявлені варіанти використання зведені до таблиці 2.2.

Таблиця 2.2.

Виявлені варіанти використання

Код	Основний актор	Найменування	Формулювання
A1	Користувач	Реєстрація в додатку	Проходження процедури реєстрації шляхом введення

			персональних даних
A2	Користувач	Вхід до особистого кабінету	Пройти процедуру входу шляхом введення логіну та паролю
A3	Користувач	Перегляд інформації про товар	Відкрити детальну інформацію про вибраний товар
A4	Користувач	Додати товар до корзини	Формування корзини товарів, що бажає придбати Користувач
A5	Користувач	Підтвердження замовлення	Введення даних про оплату, доставку та остаточне підтвердження замовлення
B1	Адміністратор	Керування каталогом товарів	Додавання нових, редагування та видалення існуючих товарів з каталогу
B2	Адміністратор	Підтвердження замовлення	Перевірка замовлення Користувача та повідомлення про його статус

За виявленими варіантами використання побудовано діаграму варіантів використання, яка представлена на рисунку 2.1.

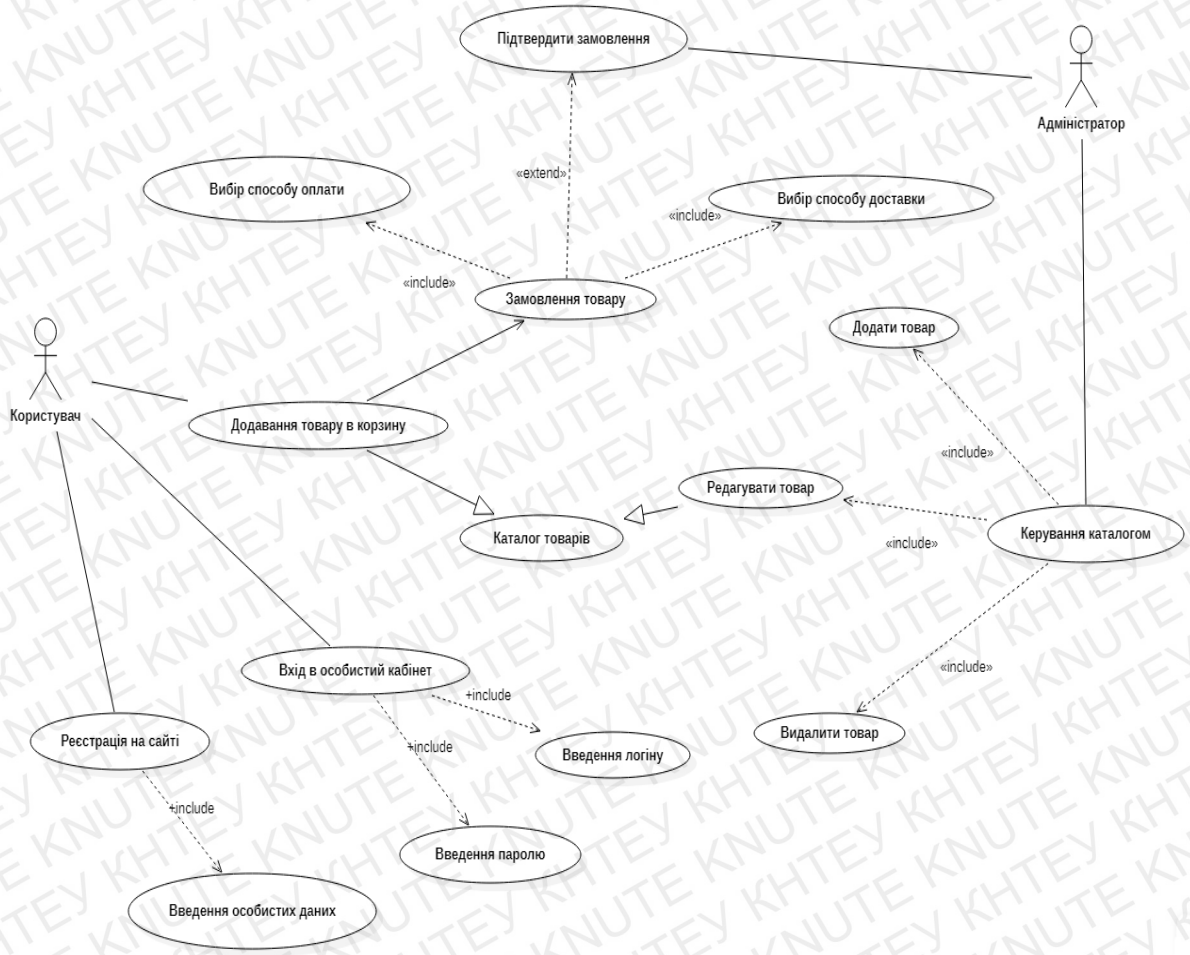

Рисунок 2.1. Діаграма варіантів використання ПЗ

2.1.2 Специфікації варіантів використання

Зробимо детальний опис виявлених варіантів використання, які описані у таблиці 2.2.

1) Специфікація варіанту використання реєстрації нового користувача.

Мета в контексті: Користувач створює новий обліковий запис для роботи з додатком.

Область застосування: поточна система.

Передумови: Користувач перейшов на сторінку «Реєстрація».

Успіх Кінцевий стан: В системі успішно зареєстровано нового користувача та йому надана сесія для роботи в додатку.

Умова, при якій мета не досягнута: відсутній доступ до інтернету, сервер не відповів вчасно.

Первинний Актор: Користувач

Trigger: Натискання користувачем кнопки «Зареєструватись».

ОСНОВНИЙ УСПІШНИЙ СЦЕНАРІЙ

1. Система просить користувача ввести дані для реєстрації.

2. Користувач вводить свої ПІБ, email, номер телефону, пароль та дату народження у спеціально відведені поля.

3. Система перевіряє чи такого користувача ще не зареєстровано та правильність вводу даних.

4. Система додає нового користувача до БД.

5. Система повідомляє користувача про успішну реєстрацію та надає йому доступ до додатку.

РОЗШИРЕННЯ

a) Користувач ввів дані, що не відповідають масці для введення:

1. Кнопка «Зареєструватись» недоступна та з’являється сповіщення про помилку в полях.

b) Користувач ввів email або номер телефону, що вже зареєстрований у системі та натиснув кнопку «Зареєструватись».

1. Система перевірила чи на дані вже була реєстрація.

2. Користувач отримав сповіщення, що на введені дані реєстрація неможлива.

c) Користувач втратив інтернет-з’єднання.

1. Користувач отримав сповіщення, що у нього немає доступу в інтернет.

d) Сервер з БД «завис» або не відповів вчасно.

1. Користувач отримав сповіщення, що система наразі недоступна та прохання спробувати пізніше.

2. Адміністратор перезапустив систему або виправив порушення у її роботі.

ВІДПОВІДНА ІНФОРМАЦІЯ

Пріоритет: Середній

Цільова ефективність: Реєстрація за 2 хвилини

Частота: 25 / день

2) Специфікація варіанту використання додавання товару до корзини

Мета в контексті: Користувач додає товар до корзини для подальшого замовлення

Область застосування: поточна система.

Рівень: Користувацький.

Передумови: Користувач переглядав сторінку з товаром.

Успіх Кінцевий стан: В корзину додано одну одиницю вибраного товару.

Умова, при якій мета не досягнута: Товару немає в наявності, сталася помилка на сервері, користувач втратив інтернет-з’єднання.

Первинний Актор: Користувач

Trigger: Натискання користувачем кнопки «Додати до корзини».

ОСНОВНИЙ УСПІШНИЙ СЦЕНАРІЙ

1. Користувач перейшов на сторінку з товаром, ознайомився з його описом та вирішив замовити.

2. Користувач натиснув кнопку «Додати до корзини».

3. Система перевіряє доступність товару для замовлення.

4. Товар успішно додано до корзини.

РОЗШИРЕННЯ

а) Товару немає в наявності

1. Користувач отримав сповіщення, що покупка товару неможлива

б) Користувач втратив інтернет-з’єднання.

1. Користувач отримав сповіщення, що у нього немає доступу в інтернет.

в) Сервер з БД «завис» або не відповів вчасно.

1. Користувач отримав сповіщення, що система наразі недоступна та прохання спробувати пізніше.

2.Адміністратор перезапустив систему або виправив порушення у її роботі.

ВІДПОВІДНА ІНФОРМАЦІЯ

Пріоритет: Високий

Цільова ефективність: Додання товару до корзини за 30 секунд

Частота: 250 / день

3) Специфікація варіанту використання оформлення замовлення

Мета в контексті: Користувач купує товар

Область застосування: поточна система.

Рівень: Користувацький.

Передумови: Користувач вибрав товари до корзини.

Успіх Кінцевий стан: Користувач успішно замовив товар.

Умова, при якій мета не досягнута: Товару немає в наявності, сталася помилка на сервері, користувач втратив інтернет-з’єднання.

Первинний Актор: Користувач

Trigger: Підтвердження замовлення.

ОСНОВНИЙ УСПІШНИЙ СЦЕНАРІЙ

1. Користувач переходить до корзини та перевіряє, при необхідності корегує замовлення.
2. Користувач вибирає спосіб доставки та оплати, вводить необхідні дані для оформлення замовлення.
3. Користувач натискає кнопку «Підтвердження замовлення».
4. Успішно створено нове замовлення

РОЗШИРЕННЯ

а) Товару немає в наявності

1. Користувач отримав сповіщення, що покупка товару неможлива

б) Користувач втратив інтернет-з’єднання.

1. Користувач отримав сповіщення, що у нього немає доступу в інтернет.

в) Користувач ввів некоректні дані

1. Користувач отримав сповіщення, що дані невірні.

2. Кнопка «Підтвердження замовлення» недоступна

ВІДПОВІДНА ІНФОРМАЦІЯ

Пріоритет: Високий

Цільова ефективність: Замовлення товару за 3 хвилини

Частота: 100 / день

4) Специфікація варіанту використання керування каталогом

Мета в контексті: Адміністратор керує каталогом магазину

Область застосування: поточна система.

Рівень: Адміністратор.

Передумови: Необхідно змінити каталог.

Успіх Кінцевий стан: Додано/змінено/видалено товар.

Умова, при якій мета не досягнута: неправильно оформлено товар, сталася мережева помилка.

Первинний Актор: Адміністратор

Trigger: Натиснуто кнопку зміна.

ОСНОВНИЙ УСПІШНИЙ СЦЕНАРІЙ

1. Адміністратор вибирає яку дію необхідно виконати з каталогом.
2. Заповнення необхідних даних.
3. Адміністратор натискає кнопку «Зміна».
4. Успішно додано/змінено/видалено товар.

РОЗШИРЕННЯ

а) Неправильно оформлений товар

1. Повідомлення про неправильну дію.

б) Сталася мережева помилка.

1. Сповіщення про помилку.

2. Відправлення звіту про помилку.

ВІДПОВІДНА ІНФОРМАЦІЯ

Пріоритет: Високий

Цільова ефективність: Добавляння, зміна чи видалення товарів за 1 хвилину.

Частота: 80 / день

Вище наведено текстовий опис основних варіантів використання програмного забезпечення «FastShop» з погляду користувача та адміністратора.

Для того, що краще показати принцип роботи додатку також використовуються діаграми діяльності, котрі дозволяють моделювати послідовності бізнес-процесів або дій, реалізованих методами класів. На рисунку 2.2 показано діаграму діяльності для варіанту використання А5 – підтвердження

замовлення. При створенні замовлення ще раз перевіряється наявність необхідного товару, якщо його немає – замовлення закривається, в іншому варіанті користувач має вибрати спосіб доставки та оплати та здійснити оплату, після чого відбувається передача товару та завершення замовлення.

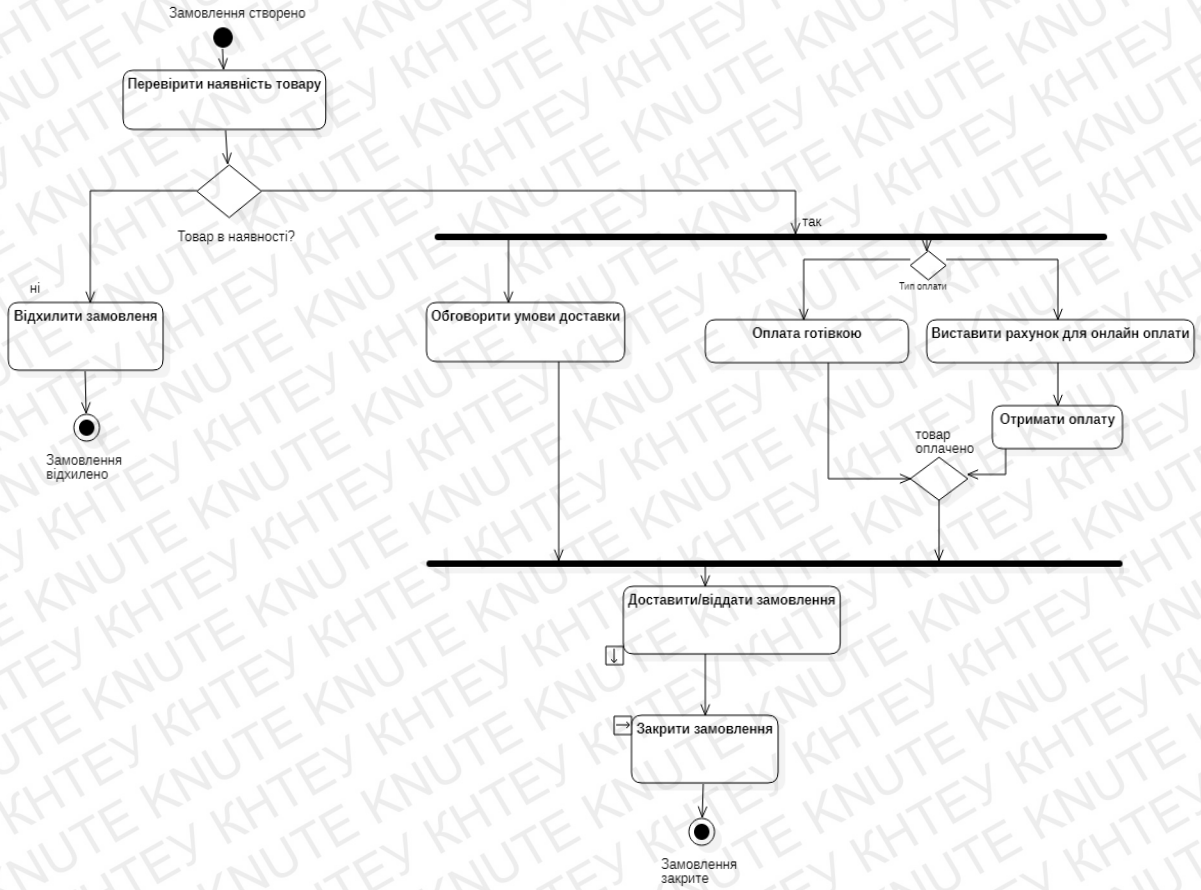

Рисунок 2.2. Діаграма діяльності варіанту використання «Підтвердження замовлення»

2.3 Моделювання бази даних

Задля того, щоб правильно і легко побудувати базу даних програмного забезпечення необхідно визначити основні сутності предметної області.

На мові інфологічного моделювання основні сутності предметної області «Інтернет-магазин» можуть бути подані в такому вигляді:

– Користувач (логін, пароль, ПІБ, адреса, країна, телефон);

– Категорії (назва, опис);

– Статус (назва);

– Товар (назва товару, ціна, кількість, опис, картинка, категорія, вага);

– Замовлення (номер замовлення, користувач, дата замовлення, адреса доставки, статус);

– Деталі замовлення (номер замовлення, товар, кількість).

На основі цих даних була створена логічна схема бази даних.

Логічна схема – модель бази даних, виражена в поняттях моделі даних. Цим відрізняється від концептуальної моделі, яка описує семантику предметної області без вказівки технології (конкретних методів реалізації), і від фізичної моделі, яка описує конкретні фізичні механізми, що застосовуються для зберігання даних в накопичувачах. На рисунку 2.2 зображено логічну модель даних.

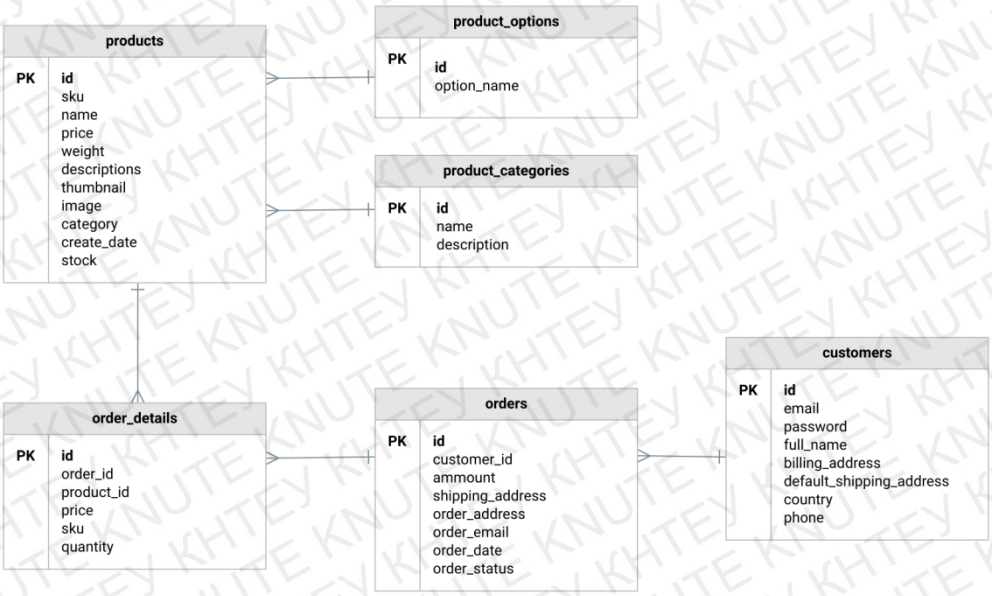

Рис. 2.3. Логічна модель предметної області

Оскільки для функціонування інтернет-магазину необхідний постійне оновлення актуальних товарів та цін, розроблюваний продукт являється клієнт-серверним додатком. В нашому випадку клієнтом є сама програма на мобільному пристрої, а сервер розташований на віддаленому веб-хостингу. Принцип роботи наступний:

- При вчинені якоїсь дії, клієнт надсилає HTTP GET/POST запит з параметрами на сервер;
- Сервер приймає запит і запускає необхідний сценарій реалізації;
- Сценарій обробляє отримані параметри від клієнта і формує запит до бази даних;
- База даних повертає дані, що були необхідні до серверу, а той в свою чергу відправляє їх назад в відповідь клієнту у вигляді масиву JSON.

Принцип роботи клієнт-серверного додатку представлений на рисунку 2.4.

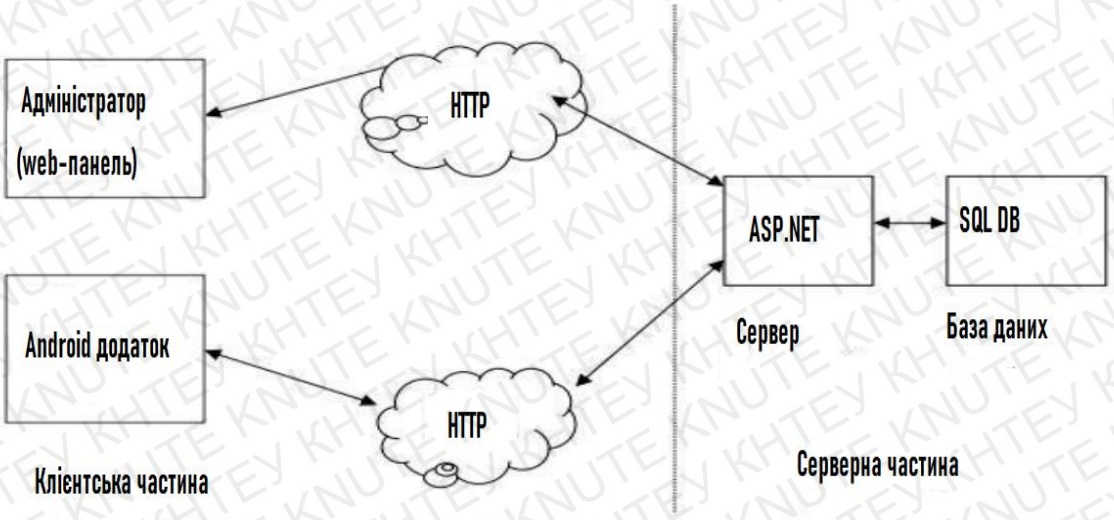

Рис. 2.4. Принцип роботи клієнт-серверного додатку

2.4 Висновки до розділу 2

В ході написання цього розділу було виконано проектування майбутньої системи за допомогою мови уніфікованого моделювання UML. Розглянуто основні сценарії роботи програми, визначено можливі варіанти помилок, що можуть виникнути в результаті її роботи. Також була змодельована майбутня база даних мобільного додатку, розроблені зв’язки між таблицями.

РОЗДІЛ 3

ОПИС ЗАПРОПОНОВАНОГО ТЕХНІЧНОГО РІШЕННЯ

1.1 Програмні засоби котрі використані при розробці продукту

Для розробки мобільних додатку FastShop була використана мова об’єктно-орієнтованого програмування C# та мова розмітки XML. Синтаксис C# близький до Java та C++. Мова має сувору статичну типізацію, підтримує поліморфізм, перезавантаження операторів та атрибути. Перейнявши багато від своїх попередників, C# виключає деякі функції, котрі були проблематичними при розробці програм, наприклад множинне успадкування класів з C++. Мова розмітки XML використовується у конфігураційних файлах додатків та при створені інтерфейсу, а за допомогою мови C# реалізується фунціонал.

В якості середовища для розробки використано Microsoft Visual Studio 2019. Дана IDE поєднує в собі швидкий редактор коду в парі з потужним дебагером, підтримку багатьох мов програмування, легке встановлення додаткових модулів.

Для створення безпосередньо клієнтської частини Android додатку використано Xamarin. Це фреймворк для кросплатформленої розробки мобільних додатків з використанням мови програмування C#. Він складається з декількох частин:

- Xamarin.Android — бібліотека класів для C#, що надає розробнику доступ до Android SDK [1];
- Xamarin.IOS — бібліотека класів для C#, що надає розробнику доступ до iOS SDK;
- Компілятори для iOS и Android;

- IDE Xamarin Studio;
- Плагін для Visual Studio.

Щоб створити працюючий сервер було використано ASP.NET – платформу розробки веб-додатків, в склад якої входять: веб-сервіси, програмна інфраструктура і модель програмування від компанії Майкрософт.

Для створення і тестування інтерфейсу прикладного програмування (API) використано такі програмні рішення як Postman та Fiddler.

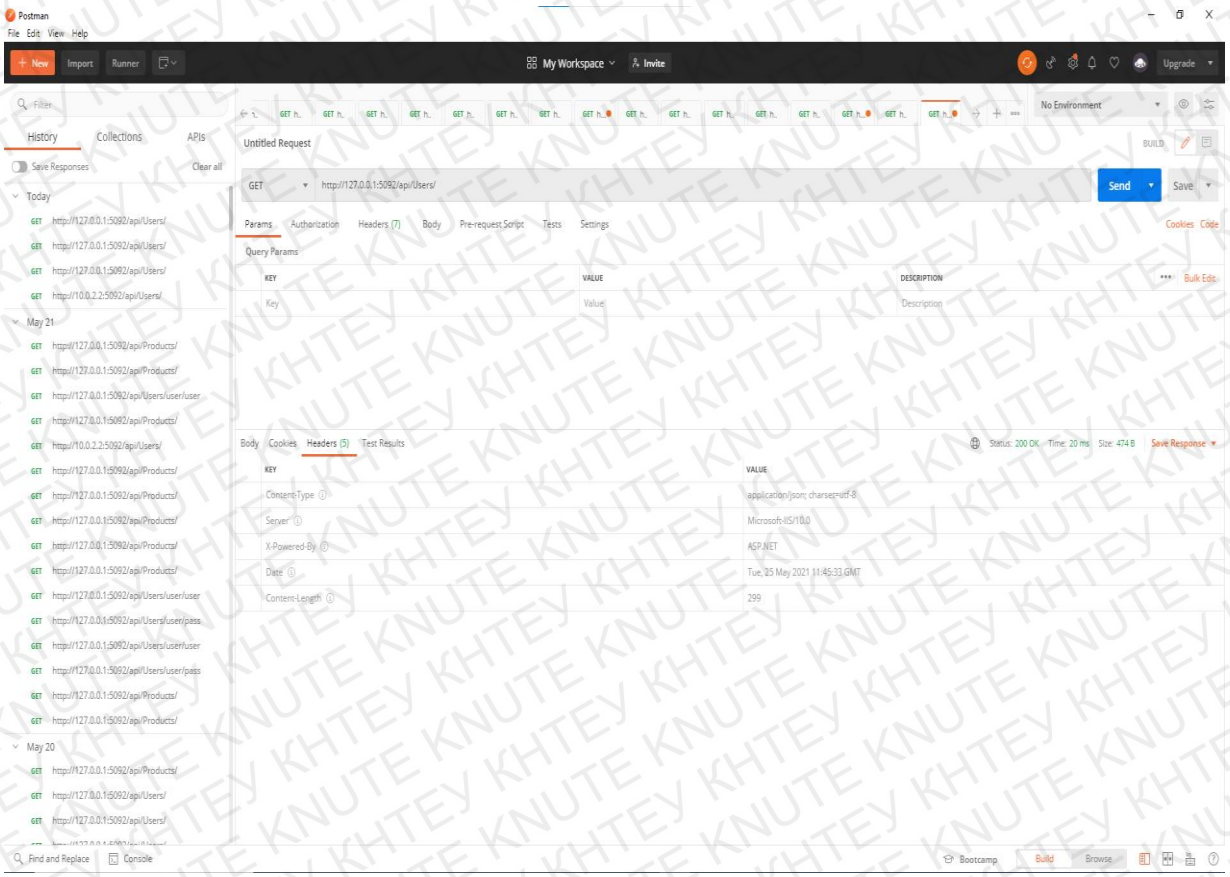

Рис. 3.1. Інтерфейс Postman

Для роботи з базою даних, створенням запитів було використано структуру об’єктно-реляційного відображення (ORM) Entity Framework.

1.2 Розробка серверної частини програми

Серверна частина розроблюваного мобільного додатку FastShop включає в себе базу даних та ASP.NET служби, котра слугує проміжною ланкою між клієнтом і базою даних та виконує роль обробки запитів і відповідей. Для

створення бази даних використовується підхід code first, при якому головним стає код, а не база. Для того, щоб створити базу даних використовуючи Entity Framework необхідно створити класи, які будуть виконувати роль таблиць, а в них оголосити об'єкти, котрі будуть виступати у вигляді полів. Для прикладу таблиця Товари буде виглядати як показано на рисунку 3.2.

```
namespace Server.Models
{
    Ссылок: 15
    public class Product
    {
        Ссылок: 3
        public int ID { get; set; }
        Ссылок: 5
        public string Name { get; set; }
        Ссылок: 5
        public int Price { get; set; }
        Ссылок: 5
        public int Weight { get; set; }
        Ссылок: 5
        public string Description { get; set; }
        Ссылок: 5
        public string Image { get; set; }
        Ссылок: 5
        public int CategoryID { get; set; }
        Ссылок: 0
        public Category category { get; set; }
    }
}
```

Рис. 3.2. Вигляд класу Product.cs

Після того, як усі класи були створені, необхідно створити клас контексту (рис. 3.3), який необхідний для ініціалізації бази даних. Він необхідний, щоб у майбутньому легко отримувати доступ до даних, оперувати значеннями полів, створювати запити.

```
using Microsoft.EntityFrameworkCore;
using Server.Models;

namespace Server.Data
{
    Ссылок: 21
    public class ProductsContext : DbContext
    {
        Ссылок: 0
        public ProductsContext(DbContextOptions<ProductsContext> options) : base(options)
        {

        }

        Ссылок: 8
        public DbSet<Category> Categories { get; set; }
        Ссылок: 6
        public DbSet<Order> Orders { get; set; }
        Ссылок: 6
        public DbSet<OrderDetails> OrderDetails { get; set; }
        Ссылок: 7
        public DbSet<Product> Products { get; set; }
        Ссылок: 7
        public DbSet<Status> Statuses { get; set; }
        Ссылок: 8
        public DbSet<User> Users { get; set; }
        Ссылок: 0
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Category>().ToTable("Category");
            modelBuilder.Entity<Order>().ToTable("Order");
            modelBuilder.Entity<OrderDetails>().ToTable("OrderDetails");
            modelBuilder.Entity<Product>().ToTable("Product");
            modelBuilder.Entity<Status>().ToTable("Status");
            modelBuilder.Entity<User>().ToTable("User");
        }
    }
}

```

Рис. 3.3. Вигляд класу ProductsContext.cs

Для того, що створити працююче API для сервера, необхідно скористатися центральною ланкою в архітектурі ASP.NET – контролером. При отримані запиту система маршрутизації вибирає його обробки потрібний контролер і передає йому дані запиту. Контролер обробляє ці дані і відправляє назад результат обробки.

В ASP.NET Core MVC контролер являє собою звичайний клас на мові C#, який унаслідується від абстрактного базового класу Microsoft.AspNetCore.Mvc.Controller. Для прикладу у розроблюваному сервері контролер для керування товарами буде виглядати наступним чином:

```
namespace Server.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProductsController : ControllerBase
    {
        private readonly ProductsContext _context;
        public ProductsController(ProductsContext context)
        {
            _context = context;
        }
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Product>>> GetProducts()
        {
            return await _context.Products.ToListAsync();
        }
        [HttpGet("{id}")]
        public async Task<ActionResult<Product>> GetProduct(int id)
        {
            var product = await _context.Products.FindAsync(id);

            if (product == null)
            {
                return NotFound();
            }
            return product;
        }
```

```
[HttpPut("{id}")]
public async Task<IActionResult> PutProduct(int id, Product product)
{
    if (id != product.ID)
    {
        return BadRequest();
    }

    _context.Entry(product).State = EntityState.Modified;

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!ProductExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return NoContent();
}
[HttpPost]
public async Task<ActionResult<Product>> PostProduct(Product product)
{
    _context.Products.Add(product);
    await _context.SaveChangesAsync();
    return CreatedAtAction("GetProduct", new { id = product.ID }, product);
}
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteProduct(int id)
{
    var product = await _context.Products.FindAsync(id);
    if (product == null)
```

```
            {
                return NotFound();
            }
            _context.Products.Remove(product);
            await _context.SaveChangesAsync();
            return NoContent();
        }
        private bool ProductExists(int id)
        {
            return _context.Products.Any(e => e.ID == id);
        }
    }
}
```

Атрибут Route вказує шлях виконання запиту, що допомагає системі розпізнати яку дію необхідно виконати. Щоб використовуючи контролер отримати дані товарів необхідно виконати Get запит на адресу http://ip:port/api/Products/. Приклад тестування запиту програмним додатком Postman показано на рисунку 3.4. Задня того, щоб отримати інформацію лише про один товар необхідно додати номер товару до адреси запиту. За допомогою Put запиту можна змінити вже існуючий товар, Post – додає новий, а Delete видаляє необхідний товар.

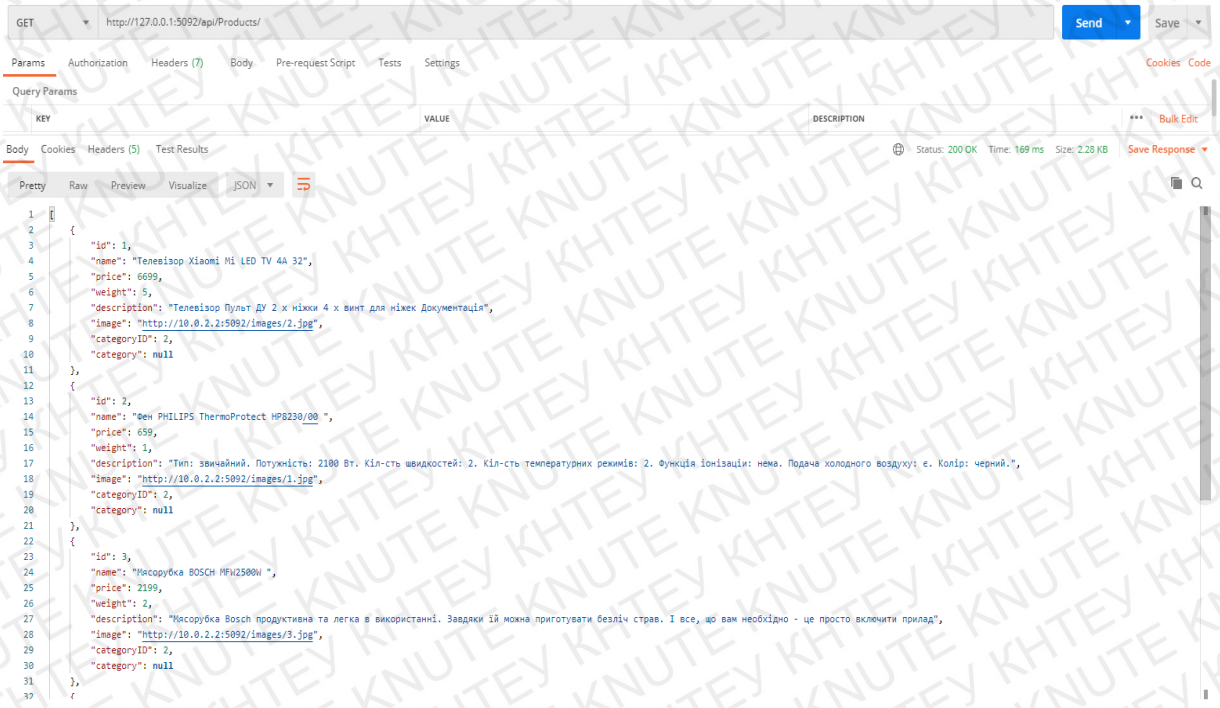

Рис. 3.4. Запит для отримання товарів

Аналогічно були створені контролери для керування користувачами, категоріями, замовленнями. Після того, як було створено базу даних та налаштовано сервер для обміну даних можна переходити до розробки клієнтського додатку.

1.3 Розробка клієнтського додатку Android

Розроблюваний мобільний додаток інтернет-магазину FastShop містить п’ять основних функціональних сторінок.

1. Сторінка авторизації (рис. 3.5). При відкритті додатку користувач відразу потрапляє на сторінку авторизації, де його просять ввести свої дані для входу. Це пара логін-пароль який він використовував при реєстрації. Для підтвердження входу

необхідно натиснути кнопку Вхід. Якщо користувач ще не створив особистий кабінет, то він може зареєструватися натиснувши відповідну кнопку і перейти на іншу функціональну сторінку.

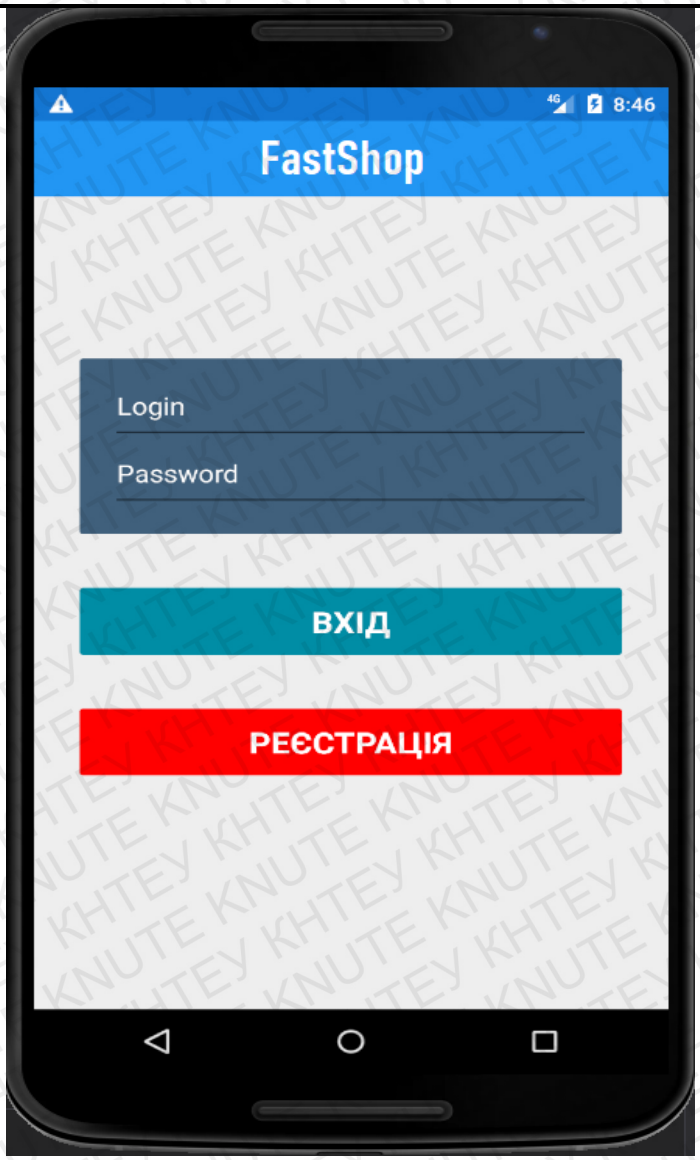

Рис. 3.5. Сторінка авторизації

2. Сторінка реєстрації (рис 3.6). Використовуючи цей екран користувач може створити новий обліковий запис в системі. Щоб це зробити він має ввести своє ім'я, логін, пароль, адресу, телефон та країну та натиснути кнопку Зареєструватися. Після чого буде відправлено запит на сервер і якщо дані введено коректно і логін є унікальним, відбудеться процедура створення нового облікового запису.

Рис. 3.6. Сторінка реєстрації

3. Каталог товарів (рис. 3.7). На даній сторінці показується інформація про наявні товари в інтернет-магазині FastShop. Відображається фото товару, його назва, опис, категорія, ціна. Список виконано у вигляді стрічки, для того щоб показати товари нижчі по списку, необхідно проскролити донизу.

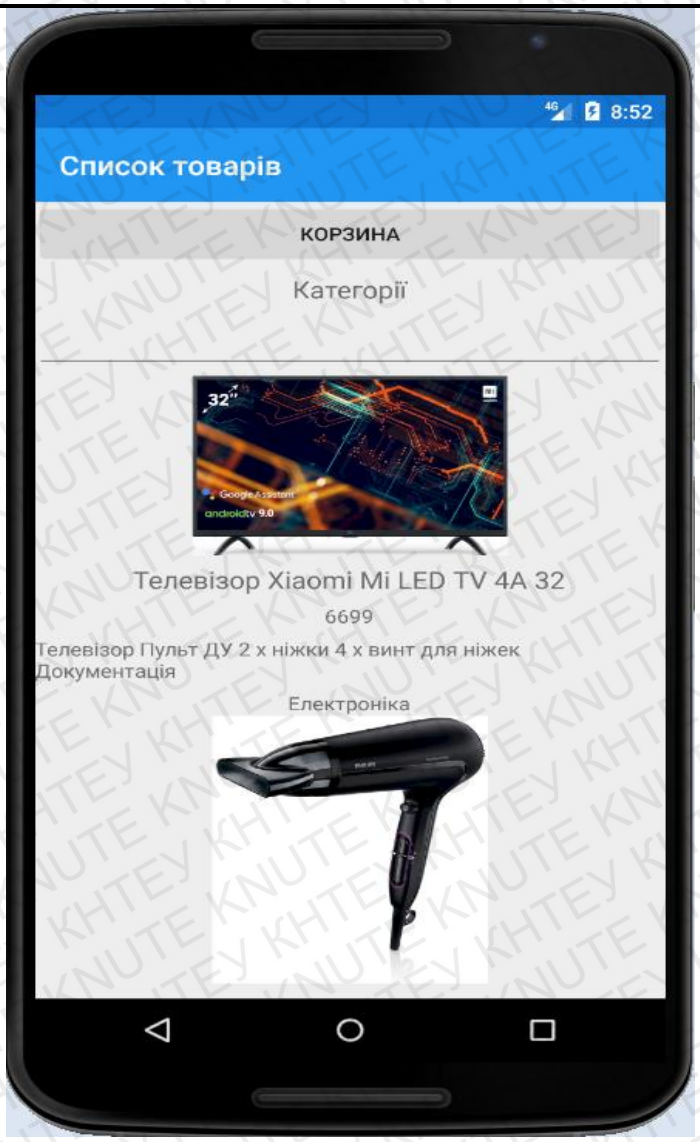

Рис. 3.7. Каталог товарів

4. Сторінка товару (рис. 3.8). Перехід на дану сторінку відбувається при натисканні на товар з каталогу товарів. На ній показується більш детальна інформація про продукт. Користувач ознайомлюється з нею і якщо його все влаштовує, він додає товар до корзини шляхом натискання на відповідну кнопку.

Рис. 3.8. Сторінка товару

5. Корзина товарів (рис. 3.9). Користувач переходить на корзину товарів коли обрав товари для покупки. На цій сторінці відображаються товари, що були обрані до покупки, після цього користувач вводить підтверджує своє замовлення або корегує його.

Рис. 3.9. Корзина товарів

Для розробки кожної інформаційної сторінки у Xamarin.Forms передбачено два файли xaml та cs. Перший виконується у вигляді xml розмітки та відповідає за візуальне оформлення форми, а другий за функціональні можливості. Для прикладу xaml файл для сторінки входу виглядає наступним чином:

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Products.Pages.LoginPage"
             BackgroundImage="logo.jpg">
  <ContentPage.Content>
    <StackLayout Orientation="Vertical" Padding="30" Spacing="40">
      <BoxView HeightRequest="10"/>
      <Image HorizontalOptions="Center" WidthRequest="300" Source="maco.jpg"/>
```

```
            <Frame BackgroundColor="#BF043055" HasShadow="False">
                <StackLayout Orientation="Vertical" Spacing="10">
                    <Entry x:Name="Login" Text="{Binding Login}" Placeholder="Login"
                           PlaceholderColor="White" HeightRequest="40"
                           TextColor="White"/>
                    <Entry x:Name="Password" Text="{Binding Password}" Placeholder="Password"
                           PlaceholderColor="White" HeightRequest="40"
                           IsPassword="True"
                           TextColor="White"/>
                </StackLayout>
            </Frame>
            <Button Command="{Binding SubmitCommand}" Text="Вхід" TextColor="White"
                    FontAttributes="Bold" FontSize="Large" HorizontalOptions="FillAndExpand"
                    BackgroundColor="#088da5" />
            <Button Command="{Binding RegisterCommand}" Text="Реєстрація"
TextColor="White"
                    FontAttributes="Bold" FontSize="Large" HorizontalOptions="FillAndExpand"
                    BackgroundColor="Red" />
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

В той же час для програмування cs файлу використовується мова C#, прив’язка подій відбувається до натискань кнопок, або безпосередньо до виклику форми (рис. 3.10).

```
namespace Products.Pages
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    Ссылок: 5
    public partial class LoginPage : ContentPage
    {
        ссылка: 1
        public LoginPage()
        {
            var vm = new LoginViewModel()
            {
                Navigation = this.Navigation
            };
            this.BindingContext = vm;
            vm.DisplayInvalidLoginPrompt += () => DisplayAlert("Помилка", "Неправильний логін чи пароль, перевірте дані", "OK");
            InitializeComponent();

            Login.Completed += (object sender, EventArgs e) =>
            {
                Password.Focus();
            };

            Password.Completed += (object sender, EventArgs e) =>
            {
                vm.SubmitCommand.Execute(null);
            };
        }
    }
}
```

Рис. 3.10. Прив’язка подій

Аналогічно була розроблена візуальна та функціональна частини у інших сторінках мобільного додатку FastShop. В результаті чого створено працюючий Android застосунок.

3.4 Висновки до розділу 3

В даному розділі було розроблено серверну частину та клієнтський додаток інтернет-магазину FastShop. Було налагоджено спілкування між клієнтом та сервером шляхом HTTP запитів, розроблено запити до бази даних. Програма має зручний, не перевантажений елементами керування інтерфейс, з яким розбереться навіть недосвідчений користувач. У додатку реалізовано всі функціональні вимоги, що задовольняють потреби користувача у швидкому доступу до товарів. Програму було розроблено на мові програмування C#, з використанням технологій Xamarin, ASP.NET та Entity Framework, середовище програмування Visual Studio 2019, додаток призначена для використання на пристроях з операційною системою Android 5.0 і вище.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Мета дипломної роботи полягала у розробці мобільного додатку для інтернет-магазину FastShop на платформі Android. Для досягнення поставлених цілей в дипломній роботі було реалізовано наступні завдання:

- проаналізовано існуюче програмне забезпечення для мобільних пристроїв;
- сформульовано вимоги до програмного забезпечення Android-додатку;
- виконано технічне проектування;
- виконано робоче проектування;

Для реалізації програмного забезпечення використовувалось середовище розробки Microsoft Visual Studio 2019 в поєднанні з технологією Xamarin.

Розроблене програмне забезпечення задовольняє вимогам, поставленим в технічному завданні.

Можна додати, що даний продукт має можливість подальшого розширення, його можна доповнювати додатковими функціями, що зробить вищевказану систему гнучкою та в певній мірі більш універсальною.

Для аналізу сучасних методів та технологій проектування програмного забезпечення, був проведений огляд літературних та інтернет-джерел. У результаті дослідження були описані необхідні технології для створення програмного забезпечення та розроблено модулі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка крос-платформних додатків для мобільних пристроїв [Електронний ресурс]. – URL: http://eprints.zu.edu.ua/22019/1/MakovskyiAPSI2016.pdf
2. Get started with Xamarin in Visual Studio [Електронний ресурс] – URL: https://stormpath.com/blog/tutorial-get-started-with-xamarin-in-visual-studio
3. Розробка мобільних додатків. [Електронний ресурс] – URL: http://kiev.itstep.org/razrabotka-mobilnyh-prilozhenij-pod-android
4. Ложковський А. Г. Теорія масового обслуговування в телекомунікаціях / А. Г. Ложковський. – Одеса: ОНАЗ ім. О.С. Попова, 2010. – 112 с.
5. How to Build CRUD REST APIs with ASP.NET Core 3.1 and Entity Framework Core, Create JWT Tokens, and Secure APIs [Електронний ресурс]. – URL: http://sd.blackball.lv/articles/read/17552
6. Creating page templates for Xamarin Forms [Електронний ресурс]. – URL: https://medium.com/@HoussemDellai/creating-page-templates-for-xamarin-forms-ec8cbc336d25
7. Google для мобільних пристроїв [Електронний ресурс]. – URL:: http://www.google.com.ua/intl/ALL_ua/mobile/android/
8. Загальна структура мови UML [Електронний ресурс]. – URL: https://studfile.net/preview/3541374/page:35/
9. Шилдт Г. C# 4.0. Повна документація: пер. з англ. / Г. Шилдт. : «Вільямс», 2011. – 608 с.
10. Фрімен А. Entity Framework Core 2 для ASP.NET Core MVC для професіоналів: пер. з англ. / А. Фрімен. Діалектика, 2019 – 624с.

					КНТЕУ 121 07-09.БР		
Зм.	*Аркуш*	*№ докум*	*Підпис*	*Дата*			
Зав. кафедри		Криворучко О.В.			*Розробка мобільного додатку для інтернет-магазину «FastShop» на платформі Android*	*Стадія*	Аркуш	Аркушів
Керівник		Пашорін В.І.				СД	49	50
Гарант		Цензура М.О.				Факультет інформаційних технологій, 4 курс, 7 група		
Розроб.		Колошко О.В.			Список використаних джерел			

11. Основні поняття Баз даних [Електронний ресурс]. – URL: http://www.lessons-tva.info/edu/e-inf2/m2t4.html
12. Актуальні питання становлення інтернет-магазинів в Україні [Електронний ресурс]. – URL: https://conferences.vntu.edu.ua/index.php/mn/mn2019/paper/viewFile/6174/5151
13. Тренди у створенні інтернет-магазинів [Електронний ресурс]. – URL: https://www.plerdy.com/ua/blog/ecommerce-website-creation-trends-2019/
14. Мельникова О. М. Смартфони на Android / О. М. Мельникова. : «Ексмо», 2013. – 305 с.
15. Амелін К. С. Введення в розробку додатків для мобільних платформ. / К. С. Амелін, О. Н. Граничин, В. И. Кіяєв. – М. : «ВВМ», 2011. – 507 с.

ДОДАТКИ

Додаток А

Сервер інтернет-магазину

А) Клас CategoriesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Server.Data;
using Server.Models;
namespace Server.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class CategoriesController : ControllerBase
  {
    private readonly ProductsContext _context;

    public CategoriesController(ProductsContext context)
    {
      _context = context;
    }
    [HttpGet]
    public async Task<ActionResult<IEnumerable<Category>>> GetCategories()
    {
      return await _context.Categories.ToListAsync();
    }
    [HttpGet("{id}")]
    public async Task<ActionResult<Category>> GetCategory(int id)
    {
      var category = await _context.Categories.FindAsync(id);

      if (category == null)
```

```
    {
        return NotFound();
    }
    return category;
}
[HttpPut("{id}")]
public async Task<IActionResult> PutCategory(int id, Category category)
{
    if (id != category.ID)
    {
        return BadRequest();
    }
    _context.Entry(category).State = EntityState.Modified;
    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!CategoryExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return NoContent();
}
[HttpPost]
public async Task<ActionResult<Category>> PostCategory(Category category)
{
    _context.Categories.Add(category);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetCategory", new { id = category.ID }, category);
}
```

```
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteCategory(int id)
        {
            var category = await _context.Categories.FindAsync(id);
            if (category == null)
            {
                return NotFound();
            }
            _context.Categories.Remove(category);
            await _context.SaveChangesAsync();
            return NoContent();
        }
        private bool CategoryExists(int id)
        {
            return _context.Categories.Any(e => e.ID == id);
        }
    }
}
```

Б) Клас OrderDetailsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Server.Data;
using Server.Models;

namespace Server.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrderDetailsController : ControllerBase
    {
        private readonly ProductsContext _context;
        public OrderDetailsController(ProductsContext context)
        {
```

```
    _context = context;
}
[HttpGet]
public async Task<ActionResult<IEnumerable<OrderDetails>>> GetOrderDetails()
{
    return await _context.OrderDetails.ToListAsync();
}
[HttpGet("{id}")]
public async Task<ActionResult<OrderDetails>> GetOrderDetails(int id)
{
    var orderDetails = await _context.OrderDetails.FindAsync(id);
    if (orderDetails == null)
    {
        return NotFound();
    }
    return orderDetails;
}
[HttpPut("{id}")]
public async Task<IActionResult> PutOrderDetails(int id, OrderDetails orderDetails)
{
    if (id != orderDetails.ID)
    {
        return BadRequest();
    }
    _context.Entry(orderDetails).State = EntityState.Modified;
    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!OrderDetailsExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
```

```
            }
        }
        return NoContent();
    }
    [HttpPost]
    public async Task<ActionResult<OrderDetails>> PostOrderDetails(OrderDetails orderDetails)
    {
        _context.OrderDetails.Add(orderDetails);
        await _context.SaveChangesAsync();
        return CreatedAtAction("GetOrderDetails", new { id = orderDetails.ID }, orderDetails);
    }
    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteOrderDetails(int id)
    {
        var orderDetails = await _context.OrderDetails.FindAsync(id);
        if (orderDetails == null)
        {
            return NotFound();
        }
        _context.OrderDetails.Remove(orderDetails);
        await _context.SaveChangesAsync();
        return NoContent();
    }
    private bool OrderDetailsExists(int id)
    {
        return _context.OrderDetails.Any(e => e.ID == id);
    }
  }
}
```

В) Клас OrdersController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Server.Data;
```

```
using Server.Models;

namespace Server.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class OrdersController : ControllerBase
  {
    private readonly ProductsContext _context;
    public OrdersController(ProductsContext context)
    {
      _context = context;
    }
    [HttpGet]
    public async Task<ActionResult<IEnumerable<Order>>> GetOrders()
    {
      return await _context.Orders.ToListAsync();
    }
    [HttpGet("{id}")]
    public async Task<ActionResult<Order>> GetOrder(int id)
    {
      var order = await _context.Orders.FindAsync(id);

      if (order == null)
      {
        return NotFound();
      }
      return order;
    }
    [HttpPut]
    public async Task<IActionResult> PutOrder(Order order)
    {
      _context.Entry(order).State = EntityState.Modified;
      try
      {
        await _context.SaveChangesAsync();
      }
      catch (DbUpdateConcurrencyException)
```

```
        {
            return NotFound();
        }
        return NoContent();
    }
    [HttpPost]
    public async Task<ActionResult<Order>> PostOrder(Order order)
    {
        _context.Orders.Add(order);
        await _context.SaveChangesAsync();
        return CreatedAtAction("GetOrder", new { id = order.ID }, order);
    }
    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteOrder(int id)
    {
        var order = await _context.Orders.FindAsync(id);
        if (order == null)
        {
            return NotFound();
        }
        _context.Orders.Remove(order);
        await _context.SaveChangesAsync();
        return NoContent();
    }
    private bool OrderExists(int id)
    {
        return _context.Orders.Any(e => e.ID == id);
    }
  }
}
```

Г) Клас ProductsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
```

```
using Server.Data;
using Server.Models;

namespace Server.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class ProductsController : ControllerBase
  {
    private readonly ProductsContext _context;

    public ProductsController(ProductsContext context)
    {
      _context = context;
    }
    [HttpGet]
    public async Task<ActionResult<IEnumerable<Product>>> GetProducts()
    {
      return await _context.Products.ToListAsync();
    }
    [HttpGet("{id}")]
    public async Task<ActionResult<Product>> GetProduct(int id)
    {
      var product = await _context.Products.FindAsync(id);
      if (product == null)
      {
        return NotFound();
      }
      return product;
    }
    [HttpPut("{id}")]
    public async Task<IActionResult> PutProduct(int id, Product product)
    {
      if (id != product.ID)
      {
        return BadRequest();
      }
      _context.Entry(product).State = EntityState.Modified;
```

```
    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!ProductExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return NoContent();
}
[HttpPost]
public async Task<ActionResult<Product>> PostProduct(Product product)
{
    _context.Products.Add(product);
    await _context.SaveChangesAsync();
    return CreatedAtAction("GetProduct", new { id = product.ID }, product);
}
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteProduct(int id)
{
    var product = await _context.Products.FindAsync(id);
    if (product == null)
    {
        return NotFound();
    }
    _context.Products.Remove(product);
    await _context.SaveChangesAsync();
    return NoContent();
}
private bool ProductExists(int id)
{
```

```
        return _context.Products.Any(e => e.ID == id);
    }
  }
}
```

Д) Клас UsersController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Server.Data;
using Server.Models;
namespace Server.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class UsersController : ControllerBase
  {
    private readonly ProductsContext _context;
    public UsersController(ProductsContext context)
    {
      _context = context;
    }
    [HttpGet]
    public async Task<ActionResult<IEnumerable<User>>> GetUsers()
    {
      return await _context.Users.ToListAsync();
    }
    [HttpGet("{id}")]
    public async Task<ActionResult<User>> GetUser(int id)
    {
      var user = await _context.Users.FindAsync(id);
      if (user == null)
      {
        return NotFound();
      }
      return user;
```

```
    }
    [HttpGet("{login}/{password}")]
    public async Task<ActionResult<User>> GetUser(string login, string password)
    {
        var user = await _context.Users.Where(x=>x.Login.Equals(login) &&
x.Password.Equals(password)).FirstOrDefaultAsync();

        if (user == null)
        {
            return NotFound();
        }
        return user;
    }
    [HttpPut("{id}")]
    public async Task<IActionResult> PutUser(int id, User user)
    {
        if (id != user.ID)
        {
            return BadRequest();
        }
        _context.Entry(user).State = EntityState.Modified;
        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!UserExists(id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return NoContent();
    }
    [HttpPost]
```

```
    public async Task<ActionResult<User>> PostUser(User user)
    {
      _context.Users.Add(user);
      await _context.SaveChangesAsync();
      return CreatedAtAction("GetUser", new { id = user.ID }, user);
    }
    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteUser(int id)
    {
      var user = await _context.Users.FindAsync(id);
      if (user == null)
      {
        return NotFound();
      }
      _context.Users.Remove(user);
      await _context.SaveChangesAsync();
      return NoContent();
    }
    private bool UserExists(int id)
    {
      return _context.Users.Any(e => e.ID == id);
    }
  }
}
```

Клієнтський додаток

А) Клас ProductService.cs

```
using FastShop.Models;
using System.Collections.Generic;
using System.Net;
using System.Net.Http;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
namespace FastShop.Services
{
   class ProductService
   {
      const string Url = $"http://{baseUrl}/api/Products/";
      JsonSerializerOptions options = new JsonSerializerOptions
      {
         PropertyNameCaseInsensitive = true,
      };
      private HttpClient GetClient()
      {
         HttpClient client = new HttpClient();
         client.DefaultRequestHeaders.Add("Accept", "application/json");
         return client;
      }
      public async Task<IEnumerable<Product>> Get()
      {
         HttpClient client = GetClient();
         string result = await client.GetStringAsync(Url);
         return JsonSerializer.Deserialize<IEnumerable<Product>>(result, options);
      }
   }
}
```

Б) Клас Basketpage.xaml.cs

```
using FastShop.ViewsModel;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace FastShop.Pages
```

```
{
  [XamlCompilation(XamlCompilationOptions.Compile)]
  public partial class BasketPage : ContentPage
  {
    BasketViewModel ViewModel;
    public BasketPage()
    {
      InitializeComponent();
      ViewModel = new BasketViewModel()
      {
        Navigation = this.Navigation
      };
      BindingContext = ViewModel;
    }
    protected override async void OnAppearing()
    {
      await ViewModel.GetOrderDetails();
      base.OnAppearing();
    }
  }
}
```

В) Клас Basketpage.xaml

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="FastShop.Pages.BasketPage" Title="Корзина">
  <StackLayout>
    <ListView x:Name="productsList" ItemsSource="{Binding OrderDetails}" HasUnevenRows="True">
      <ListView.ItemTemplate>
        <DataTemplate>
          <ViewCell>
            <ViewCell.View>
              <StackLayout>
                <Image Source="{Binding ProductImage}" />
                <Label Text="{Binding ProductName}" FontSize="Medium" HorizontalOptions="Center"/>
                <Label Text="{Binding ProductPrice}" FontSize="Small" HorizontalOptions="Center"/>
                <Label Text="{Binding ProuctDescription}" FontSize="Small"
HorizontalOptions="Center"/>
              </StackLayout>
```

```
            </ViewCell.View>
          </ViewCell>
        </DataTemplate>
      </ListView.ItemTemplate>
    </ListView>
    <StackLayout IsVisible="{Binding IsBusy}"
          HorizontalOptions="Center" VerticalOptions="CenterAndExpand" Padding="20">
      <Label Text="Завантаження даних..." TextColor="Gray" HorizontalOptions="Center" />
      <ActivityIndicator IsRunning="{Binding IsBusy}" Color="Red" >
      </ActivityIndicator>
    </StackLayout>
    <StackLayout>
      <Label Text="Адреса доставки" HorizontalOptions="Center" />
      <Entry x:Name="AddressEntry" Text="{Binding SellAddress}" Placeholder="Login"
             PlaceholderColor="White" HeightRequest="40"
             />

      <Button Text="Купити" Command="{Binding BuyCommand}" />
    </StackLayout>
  </StackLayout>
</ContentPage>
```

Г) Клас LoginPage.xaml

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class=" FastShop.Pages.LoginPage"
        BackgroundImage="logo.jpg">
  <ContentPage.Content>
    <StackLayout Orientation="Vertical" Padding="30" Spacing="40">
      <BoxView HeightRequest="10"/>
      <Image HorizontalOptions="Center" WidthRequest="300" Source="maco.jpg"/>
      <Frame BackgroundColor="#BF043055" HasShadow="False">
        <StackLayout Orientation="Vertical" Spacing="10">
          <Entry x:Name="Login" Text="{Binding Login}" Placeholder="Login"
                 PlaceholderColor="White" HeightRequest="40"
                 TextColor="White"/>
          <Entry x:Name="Password" Text="{Binding Password}" Placeholder="Password"
                 PlaceholderColor="White" HeightRequest="40"
                 IsPassword="True"
```

```
                TextColor="White"/>
        </StackLayout>
    </Frame>
    <Button Command="{Binding SubmitCommand}" Text="Вхід" TextColor="White"
            FontAttributes="Bold" FontSize="Large" HorizontalOptions="FillAndExpand"
            BackgroundColor="#088da5" />
    <Button Command="{Binding RegisterCommand}" Text="Реєстрація" TextColor="White"
            FontAttributes="Bold" FontSize="Large" HorizontalOptions="FillAndExpand"
            BackgroundColor="Red" />
  </StackLayout>
 </ContentPage.Content>
</ContentPage>
```

Д) Клас LoginPage.xaml.cs

```
using FastShop.ViewModels;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace Products.Pages
{
 [XamlCompilation(XamlCompilationOptions.Compile)]
 public partial class LoginPage : ContentPage
 {
  public LoginPage()
  {
   var vm = new LoginViewModel()
   {
    Navigation = this.Navigation
   };
   this.BindingContext = vm;
   vm.DisplayInvalidLoginPrompt += () => DisplayAlert("Помилка", "Неправильний логін чи пароль, перевірте дані", "OK");
   InitializeComponent();

   Login.Completed += (object sender, EventArgs e) =>
   {
    Password.Focus();
   };
   Password.Completed += (object sender, EventArgs e) =>
```

```
        {
            vm.SubmitCommand.Execute(null);
        };
    }
  }
}
```

Е) Клас RegisterPage.xaml

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class=" FastShop.Pages.RegisterPage" Title="Реєстрація">
  <StackLayout>
    <StackLayout>
      <Label Text="Ім'я" />
      <Entry Text="{Binding Path=Model.Name}" FontSize="Medium" />
      <Label Text="Логін" />
      <Entry Text="{Binding Path=Model.Login}" FontSize="Medium" />
      <Label Text="Пароль" />
      <Entry Text="{Binding Path=Model.Password}" FontSize="Medium" />
      <Label Text="Адреса" />
      <Entry Text="{Binding Path=Model.Address}" FontSize="Medium" />
      <Label Text="Телефон" />
      <Entry Text="{Binding Path=Model.Phone}" FontSize="Medium" />
      <Label Text="Країна" />
      <Entry Text="{Binding Path=Model.Country}" FontSize="Medium" />
    </StackLayout>
    <StackLayout Orientation="Horizontal" HorizontalOptions="CenterAndExpand">
      <Button Text="Зареєструватися" Command="{Binding ViewModel.RegisterCommand}"
CommandParameter="{Binding Model}" />
    </StackLayout>
  </StackLayout>
</ContentPage>
```

Є) Клас RegisterPage.xaml.cs

```
using Products.Models;
using FastShop.ViewsModel;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
namespace FastShop.Pages
{
```

```
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RegisterPage : ContentPage
    {
        public User Model { get; private set; }
        public RegisterViewModel ViewModel { get; private set; }
        public RegisterPage()
        {
            InitializeComponent();
            Model = new User { };
            ViewModel = new RegisterViewModel(Navigation)
            {
                Navigation = this.Navigation
            };
            this.BindingContext = this;
        }
    }
}
```

Ж) Клас ProductPage.xaml

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class=" FastShop.Pages.ProductPage" Title="Список товарів">
    <StackLayout>
        <StackLayout>
            <Button Text="Корзина" Command="{Binding ShowBasketCommand}"
CommandParameter="{Binding Model}"/>
            <Label x:Name="header" Text="Категорії" FontSize="Medium" HorizontalOptions="Center"/>
            <Picker x:Name="picker" SelectedIndexChanged="picker_SelectedIndexChanged"
ItemsSource="{Binding Categories}" ItemDisplayBinding="{Binding Name}"/>
        </StackLayout>
        <ListView x:Name="productsList" ItemsSource="{Binding Products}"
                  SelectedItem="{Binding SelectedProduct, Mode=TwoWay}" HasUnevenRows="True">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <ViewCell.View>
                            <StackLayout>
                                <Image Source="{Binding Image}" />
                                <Label Text="{Binding Name}" FontSize="Medium" HorizontalOptions="Center"/>
                                <Label Text="{Binding Price}" FontSize="Small" HorizontalOptions="Center"/>
```

```
                    <Label Text="{Binding Description}" FontSize="Small" HorizontalOptions="Center"/>
                    <Label Text="{Binding CategoryName}" FontSize="Small" HorizontalOptions="Center"/>
                </StackLayout>
              </ViewCell.View>
            </ViewCell>
          </DataTemplate>
        </ListView.ItemTemplate>
      </ListView>
      <StackLayout IsVisible="{Binding IsBusy}"
                   HorizontalOptions="Center" VerticalOptions="CenterAndExpand" Padding="20">
        <Label Text="Завантаження даних..." TextColor="Gray" HorizontalOptions="Center" />
        <ActivityIndicator IsRunning="{Binding IsBusy}" Color="Red" >
        </ActivityIndicator>
      </StackLayout>
    </StackLayout>
</ContentPage>
```

3) Клас ProductPage.xaml.cs

```
using FastShop.ViewsModel;
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace FastShop.Pages
{
  [XamlCompilation(XamlCompilationOptions.Compile)]
  public partial class ProductPage : ContentPage
  {
    ProductViewModel ViewModel;
    public ProductPage()
    {
      InitializeComponent();
      ViewModel = new ProductViewModel()
      {
        Navigation = this.Navigation
      };
      BindingContext = ViewModel;
    }
    protected override async void OnAppearing()
```

```
        {
            await ViewModel.GetCategories();
            await ViewModel.GetProducts();
            base.OnAppearing();
        }
        void picker_SelectedIndexChanged(object sender, EventArgs e)
        {
            header.Text = "Ви вибрали: " + picker.Items[picker.SelectedIndex];
            ViewModel.GetProducts(picker.Items[picker.SelectedIndex]);
        }
    }
}
```