ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА МОБІЛЬНОЇ ПЛАТФОРМИ ДЛЯ B2C- ПРОДАЖУ КОСМЕТИЧНОЇ ПРОДУКЦІЇ МОВОЮ DART»**

Виконала: студентка 4 курсу, групи ПД-43

спеціальності

121 Інженерія програмного забезпечення

_____Полтавець Н.В._____
(прізвище, ім‘я, по батькові)

Керівник _____Шевченко С.М._____
(прізвище, ім‘я, по батькові)

Рецензент _____
(прізвище, ім‘я, по батькові)

Київ - 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність – 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення
Негоденко О.В.

"____" _____2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Полтавець Наталії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка мобільної платформи для B2C- продажу косметичної продукції мовою Dart" »
Керівник роботи Шевченко С.М., к.п.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "24" лютого 2023 року №26
2. Строк подання студентом роботи 1.06.2023 року
3. Вихідні дані до роботи:
Інструменти з розробки програмного забезпечення: Adroid Studio SDK, Xcode simulator, MVC, Material Design, Git, Flutter, Dart.
Наукові джерела з питань розробки мобільного додатку з використанням фреймворку Flutter.
Технології та документація мови програмування Dart, фреймворку Flutter та Material Design.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Аналіз предметної області найближчих за функціоналом існуючих додатків до додатку для продажу косметичних засобів.
2. Теоретичні основи для розробки програмного засобу, вибір та обґрунтування технологій та засобів розробки.
3. Розробка мобільного додатку для продажу косметичних засобів.
4. Тестування розробленого додатку.

5. Перелік графічного матеріалу.
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.

3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма класів.
6. Діаграма діяльності (оплата замовлення).
7. Екрані форми

6. Дата видачі завдання 24.02.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз науково-технічної літератури	24.02 – 12.03	
2	Опис предметної області	13.03 – 20.03	
3	Постановка задачі	21.03 – 28.03	
4	Програмна реалізація	29.03 – 30.04	
5	Тестування розробленого ПО	01.05 – 10.05	
6	Вступ, висновки, реферат	11.05 – 20.05	
7	Попередній захист роботи	22.05 – 26.05	
8	Перевірка на плагіат	27.05 – 31.05	
9	Здача роботи в деканат	01.06.	

Студент _____ Полтавець Н.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шевченко С.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: с. 61, 3 табл., 49 рис., 2 дод., 10 джерела.

МОБІЛЬНИЙ ДОДАТОК, FLUTTER, DART, UI, EBAY, TICKETS PARTY, FEEH LA, MVC, ООП, МОБІЛЬНИЙ ДОДАТОК ДЛЯ ПРОДАЖУ КОСМЕТИЧНИХ ЗАСОБІВ

Об`єкт дослідження – процес взаємодії з клієнтами B2C-продажів косметичної продукції.

Предмет дослідження – програмні засоби для забезпечення взаємодії з клієнтами B2C-продажів косметичної продукції.

Мета роботи – покращення процесу взаємодії з клієнтами B2C-продажів косметичної продукції за рахунок використання мобільного додатку мовою Dart.

Методи дослідження – системно-структурні методи, порівняльний аналіз методи передачі, зберігання та обробки інформації, уніфікований процес створення програмного забезпечення.

Дане дослідження присвячене проблемі ефективної взаємодії з клієнтами у процесі онлайн-продажу косметичних засобів. У роботі проведено аналіз аналогів серед існуючих мобільних застосунків: Ebay, Епіцентр, Party Ticket, визначено їх переваги та недоліки, які стали основою для формулювання вимог до нового застосунку та його проєктування. Для розробки програмного забезпечення мобільного додатку вибрано мову Dart та фреймворк Flutter, обґрунтовано їх застосування. Мобільний застосунок для продажу косметичної продукції пройшов тестування, результати – задовільні.

Даний додаток може використовуватися для персональних цілей, з метою освіти в галузі мобільної розробки для ознайомлення та вивчення використання фреймворку Flutter та мови програмування Dart та для інтернет-магазинів для покращення взаємодії з клієнтами у процесі Інтернет-торгівлі.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI - “User Interface” (користувацький інтерфейс)

IDE - Integrated Development Environment (інтегроване середовище розробки)

GCC - “GNU Compiler Collection” і є набором компіляторів

GDB - “GNU Debugger” і є відлагоджувачем для програм

MVC- model-view-controller

ООП - Об'єктно-орієнтоване програмування

ВСТУП

Смартфони невід'ємна частина повсякденного життя в 2023 році. Раніше люди мали різні пристрої з обмеженими можливостями для вирішення своїх проблем. Калькулятор, компас, фотоапарат тощо було об'єднані в невелику коробку, яка відома під назвою смартфоном. І це дало поштовх для розвитку мобільних додатків як інструментів для користувачів. Мобільні додатки стали інструментами для навчання, планування, замовлення, бронювання чи купівлі товарів і послуг. З розповсюдженням мобільних пристроїв розробка мобільних додатків стала ключовим фактором у тому, як компанії та організації охоплюють і взаємодіють зі своїми клієнтами. У результаті мобільні додатки дозволяють магазинам залишатися конкурентоспроможними на торговій арені.

Ринок косметичних засобів розвивається шаленими темпами. Цільова аудиторія косметичних засобів включає в себе як жінок, так і чоловіків різного віку, що впливає на попит магазинів косметичними засобами. І задля того, щоб магазини були конкурентоспроможними на торговій арені зросла потреба в Інтернет-торгівлі за використанням веб-сайтів та мобільних додатків, бо сьогодні покупці віддають перевагу купівлі онлайн через можливість вибору та придбання товару або послуги, не виходячи з будинку в будь-який час. Перед магазинами косметичних товарів виникає задача розробки мобільного додатку.

Все вище викладене підтвердило актуальність даного дослідження і його важливість.

Метою роботи є покращення процесу взаємодії з клієнтами B2C-продажів косметичної продукції за рахунок використання мобільного додатку мовою Dart.

Для досягнення цієї мети в роботі необхідно вирішити такі *завдання*:

1. Здійснити огляд літературних джерел, існуючих засобів розв'язання завдань предметної галузі.

2. Здійснити порівняльний аналіз аналогів програмного забезпечення (Ebay, Party Tickets,Feeh la); визначити функціональні та нефункціональні вимоги на основі отриманих результатів порівняння аналогів.

3. Обґрунтувати вибір технологій для розробки мобільного додатку з продажу косметичних продуктів.

4. Спроєктувати архітектуру системи та інтерфейс додатку для продажу косметичних засобів.

5. Розробити мобільну платформу для B2C-продажу косметичної продукції.

6. Провести тестування розробленої мобільної платформи

Виходячи з цього, **об'єктом** дослідження є процес взаємодії з клієнтами B2C-продажів косметичної продукції, **предметом** дослідження – програмні засоби для забезпечення взаємодії з клієнтами B2C-продажів косметичної продукції

Методи дослідження. Для вирішення вищезгаданих завдань у роботі використано наступні методи: системно-структурні методи, порівняльний аналіз, методи передачі, зберігання та обробки інформації, уніфікований процес створення програмного забезпечення. Для розробки програмного забезпечення використані мова Dart та фреймворк Flutter.

Практична значущість результатів. Розроблений мобільний додаток може використовуватися для персональних цілей, з метою освіти в галузі мобільної розробки для ознайомлення та вивчення використання фреймворку Flutter та мови програмування Dart студентами спеціальності 121 «Інженерія програмного забезпечення» та для інтернет-магазинів для покращення взаємодії з клієнтами у процесі Інтернет-торгівлі.

1 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Аналіз програмного забезпечення дозволяє покращити та спростити розробку мобільних застосунків. В першу чергу, аналізуючи існуючи аналоги розглядається їх інтерфейс та функціональність, а також визначаються їх переваги та недоліки, задля подальшого використання цієї інформації для створення та покращення майбутнього мобільного застосунку. Інтерфейс користувача є одною з основних частин розробки, оскільки саме він впливає на роботу користувача з додатком.

1.1 Ebay

Ebay - це онлайн платформа, яка дозволяє користувачам здійснювати купівлю та продаж товару. Ebay має інтернет-сайт та мобільний застосунок для двох операційних систем, як Android та IOS.

Для авторизації користувачу дає змогу входу завдяки акаунту в Facebook, Google, Apple (доступно тільки на телефонах з операційною системою IOS) та використання логіну та паролю, що є зручною функцією для користувачів.

На Рисунок 1.1.1 зображено екран авторизації в мобільному додатку Ebay.

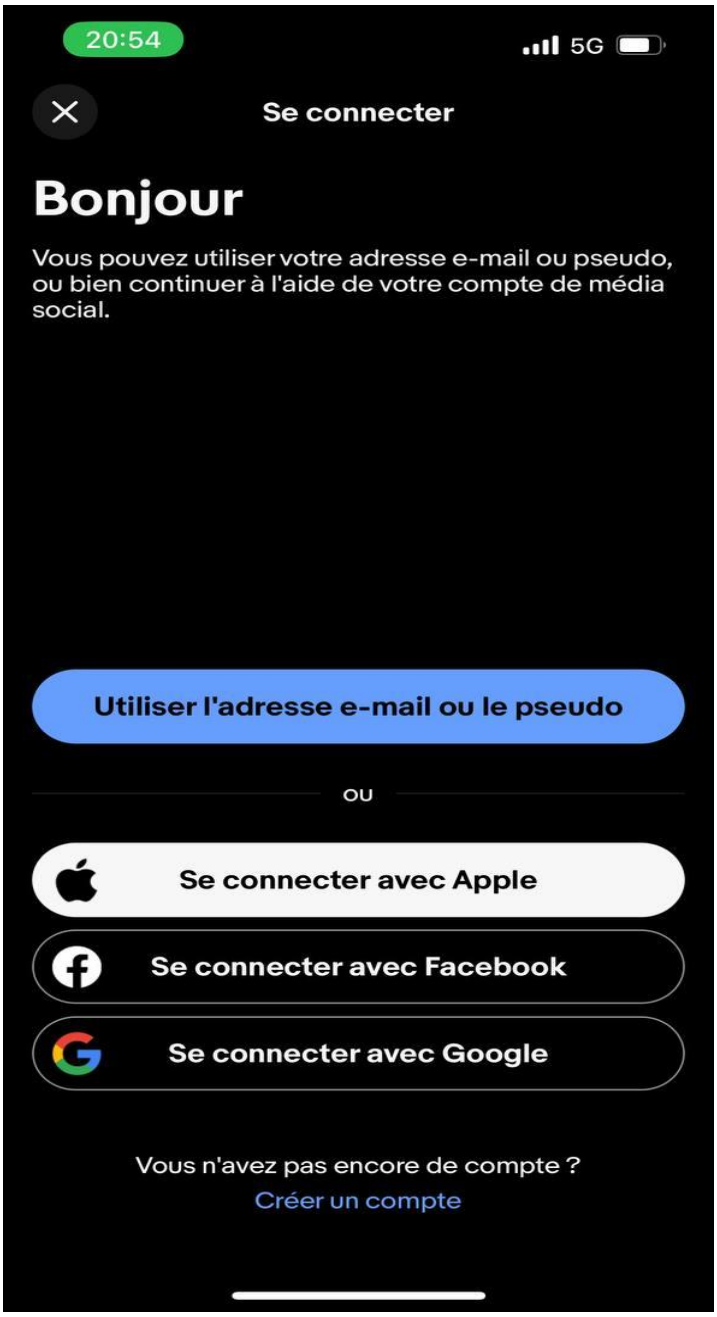


Рисунок 1.1.1 - Екран авторизації в мобільному застосунку Ebay

Головний екран (Рисунок 1.1.2) це перше, що бачить користувач. Одразу звертається увага, що для легшої навігації застосунком використовується нижня панель навігації. Також для зручності користувача кнопка, яка пересуває користувача на сторінку з доданими товарами в кошик представлено окремо в верхньому правому куті. Оскільки мобільний застосунок, який розробляється в процесі даного дослідження, також є магазином, то дані особливості навігації мобільного застосунку Ebay, розглядаються для подальшого використання.

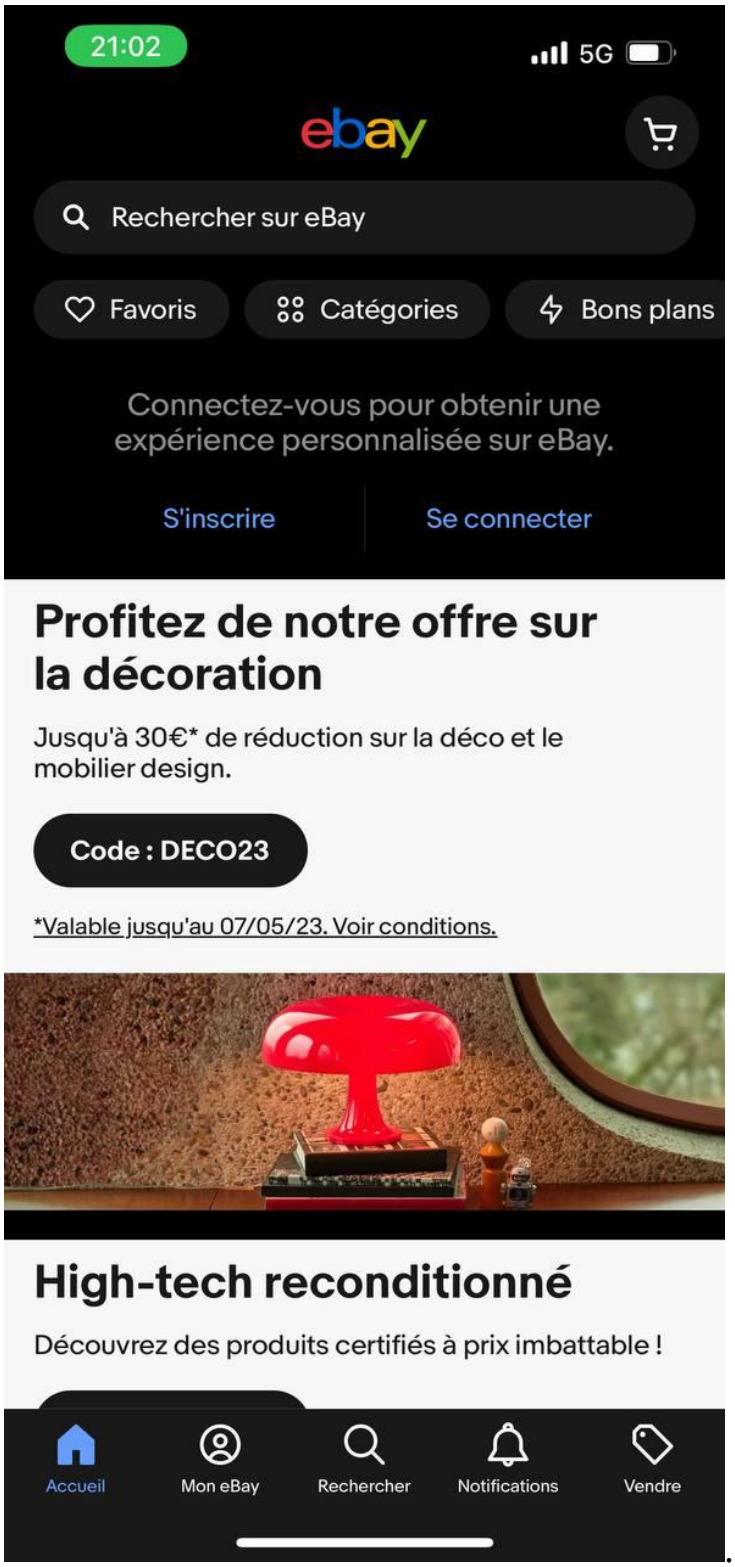


Рисунок 1.1.2 - Головна сторінка мобільного застосунку Ebay

Екран на якому представлені деякі функції мобільного додатку Рисунок 1.1.3). Є можливість зв'язку з покупцем, передивитись збережені товари, авторизація та інші.

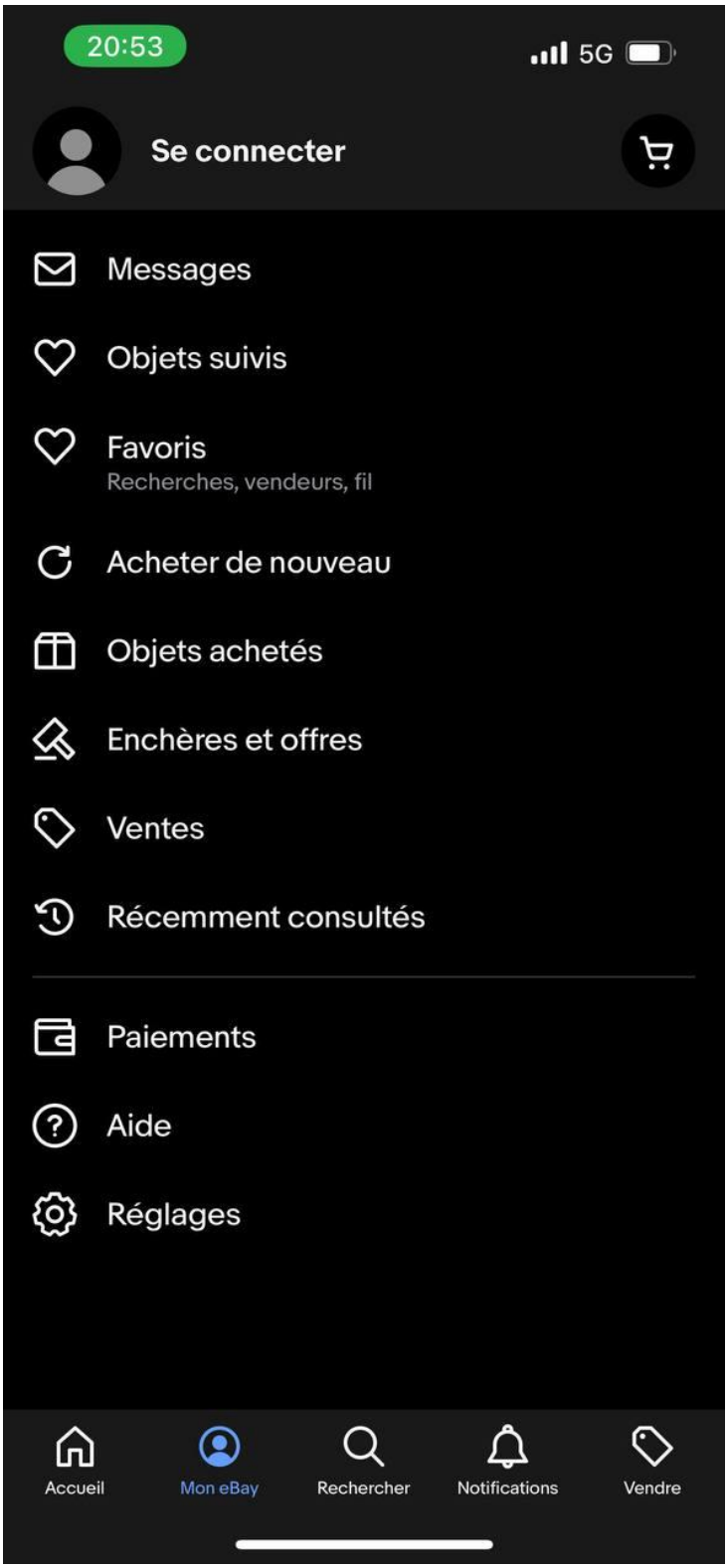
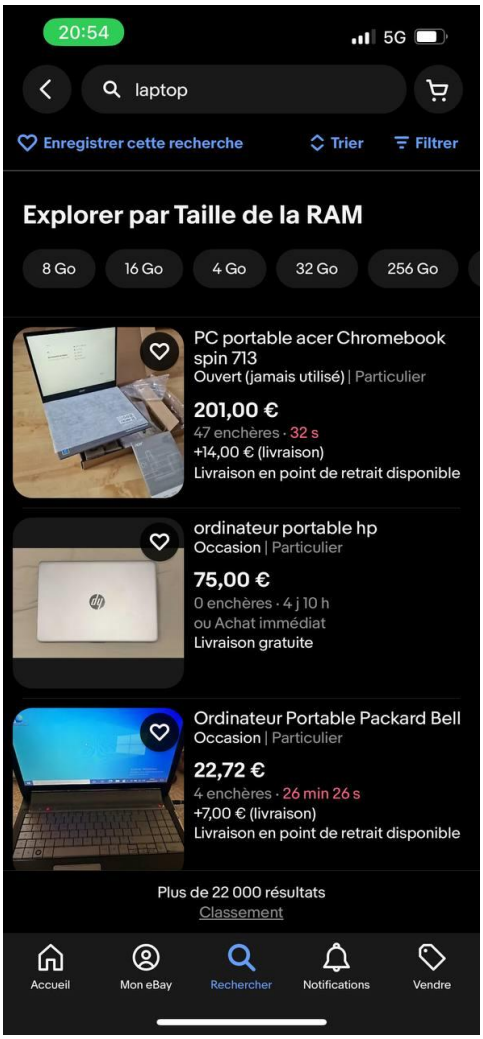
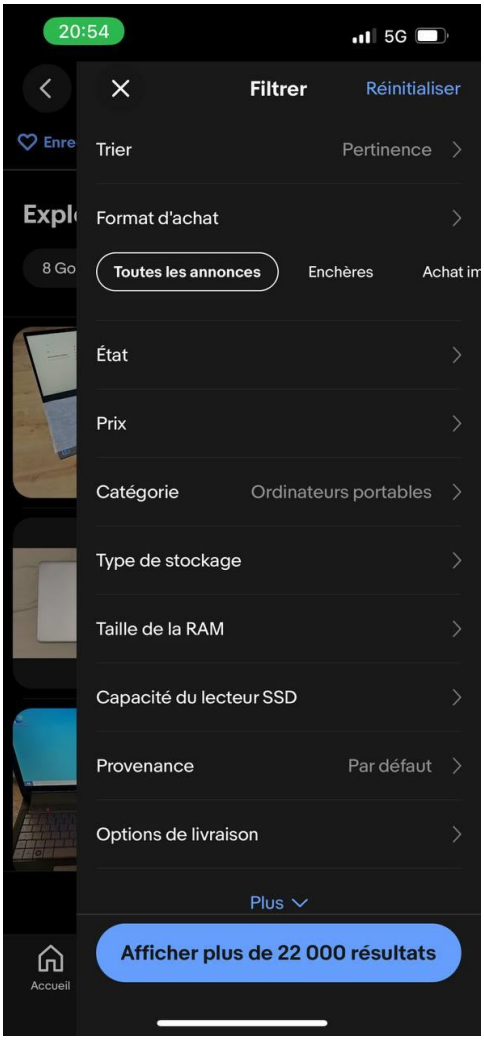


Рисунок 1.1.3 - Екран “Mon eBay” в мобільному додатку Ebay

Для подальшого планування інтерфейсу мобільного застосунку, який буде результатом даного дослідження, можна відміти, як представлено екран з переліком товару(Рисунок 1.1.4) у вигляді списку та можливість його фільтрувати за певними ознаками(Рисунок 1.1.5).





Рисунки 1.1.4 і 1.1.5 - Екран пошуку товару та функція фільтрування товару в мобільному додатку Ebay

Переваги Ebay:

1. Змога фільтрувати товар, за особистими запитами, що прискорює пошук потрібного товару;
2. Функція порівняння ціни на однаковий товар серед різних продавців.
3. Дозволяє користувачам купувати та продавати товари;
4. Доступність на різних мовах та в різних країнах;
5. Зручний та безпечний спосіб оплати через мобільний додаток;
6. Зв'язок з продавцями через додаток.

Недоліки Ebay:

1. Деякі функції можуть працювати повільно або некоректно;
2. Можливість купівлі товару низької якості або несумісного з описом;

3. Можливість покупки тільки після авторизації.

1.2 Feeh La shopping

Feeh.la — це платформа для онлайн-покупок. Магазин продає понад 30 000 продуктів і категорій на вибір, таких як подарунки, канцтовари, електроніка, побутова техніка та навіть домашні тварини, які можливо купити онлайн через їх веб-сайт та мобільний додаток. Для аналізу аналогів взято саме їх мобільний додаток.

При запуску додатку відкривається головний екран(Рисунок 1.1.6). На сторінці представлено пошук товару, пропозиції магазину, каталог товару, і нижче перелік товарів. Тобто користувач одразу зручно може почати пошук товару, який цікавить та дізнатися про цікаві пропозиції магазину. Наявна також нижня панель навігації.

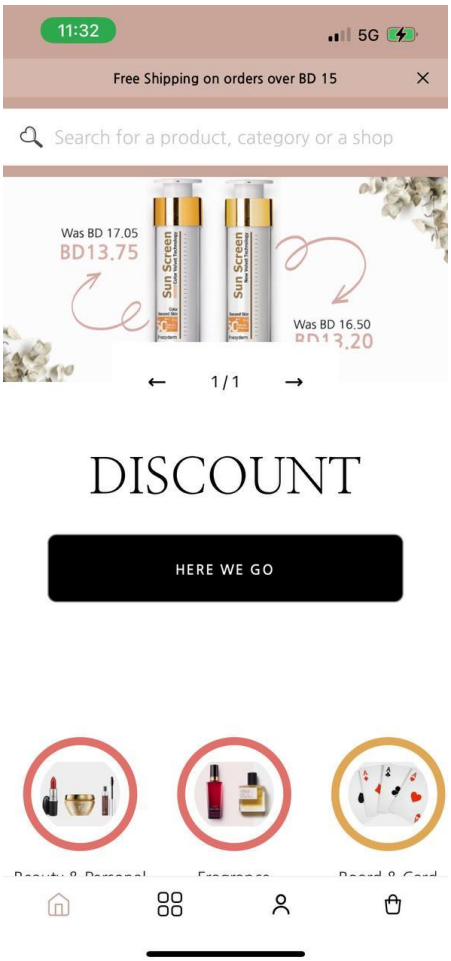


Рисунок 1.1.6 - Головний екран додатку Feeh La shopping

На Рисунок 1.1.7 представлена функція фільтрації товару.

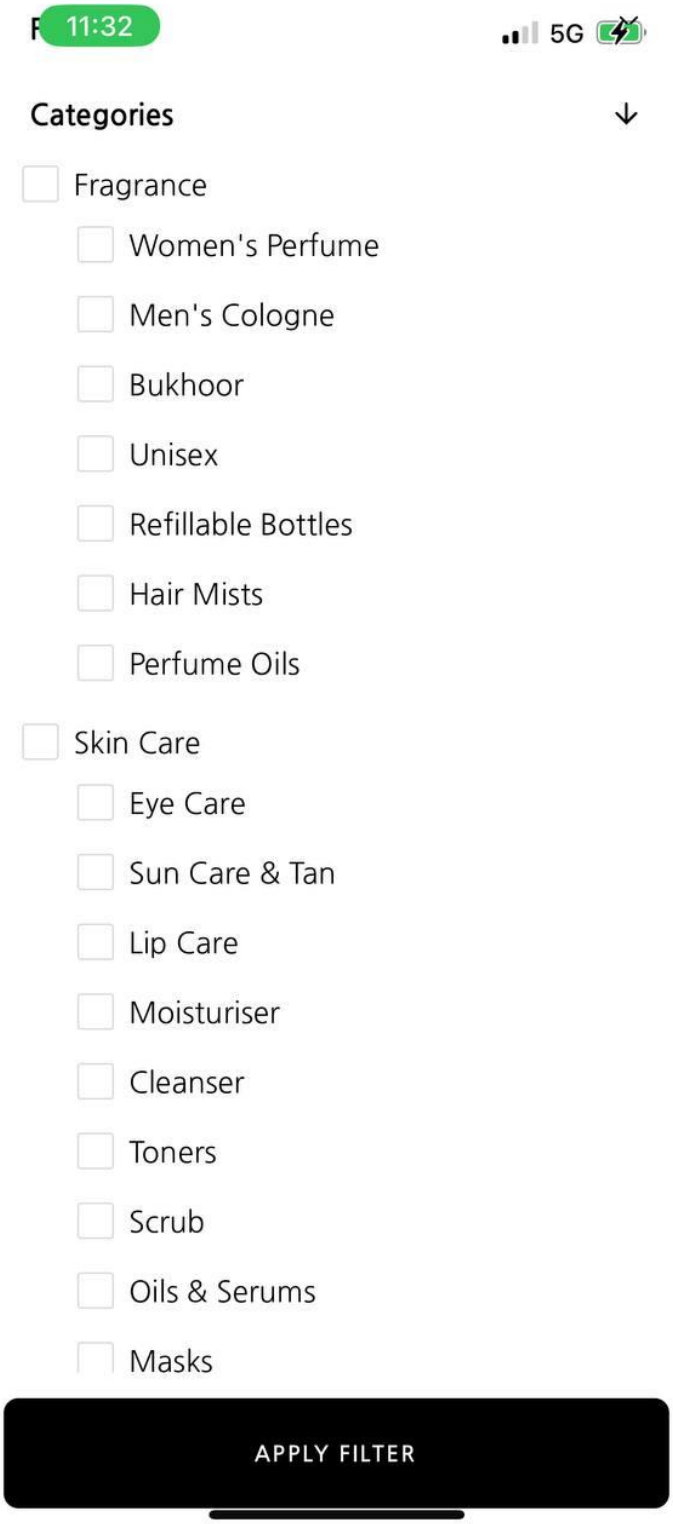


Рисунок 1.1.7 - Функція фільтра в додатку Feeh La shopping

На Рисунок 1.1.8 представлен екран з категоріями товару в додатку Feeh La shopping.

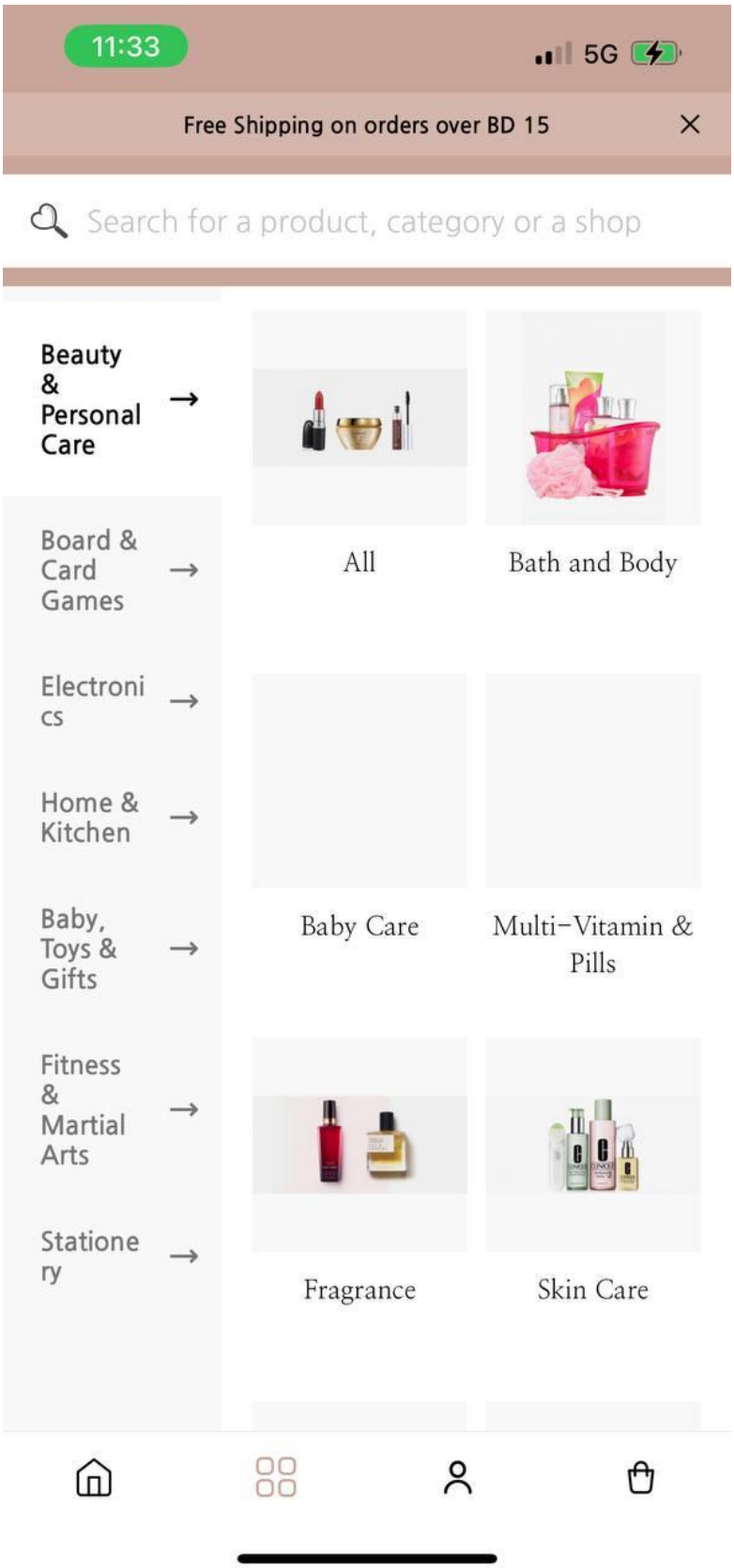


Рисунок 1.1.8 - Екран з категоріями товару в додатку Feeh La shopping

Екран з авторизацією(Рисунок 1.1.9) за допомогою акаунта в Google або Apple. Також представлена контактна інформація магазину, але відсутній зворотній зв'язок з магазином через додаток.

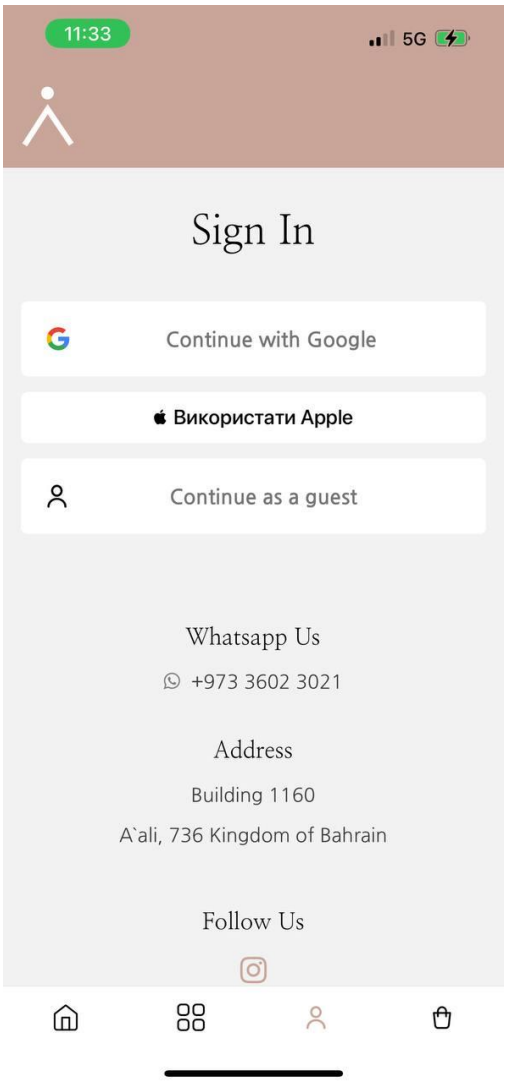


Рисунок 1.1.9 - Екран з авторизацією та контактною інформацією магазина в додатку Feeh La shopping

Переваги Feeh La shopping:

1. Авторизація за допомогою акаунту Google та Apple;
2. Можливість оплати онлайн;
3. Фільтрація товару за особистими запитами;
4. Наявність пошуку товару.

Недоліки Feeh La shopping:

1. Деякі функції можуть працювати повільно або некоректно;
2. Відсутність зворотного зв'язку з магазином через додаток;

3. Можливість покупки тільки після авторизації;
4. Некоректна розмітка об'єктів відносно один одного.

1.3 Party Ticket

Мобільний додаток з купівлі електронних квитків в нічні клуби України – Party Ticket. Створений мобільний додаток дозволяє користувачеві максимально полегшити процедуру вибору і покупки квитків на заходи, відрізняється простотою використання, легкою і візуально зрозумілою навігацією і відмінними показниками оперативності обробки інформації та обміну нею.

При запуску програми відкривається головний екран(Рисунок 1.1.10) з афішами, афіши змінюються скролінгом в бік, з гарною анімацією. Є недолік з розміткою афіши до кнопки купівлі. Якщо афіша велика, перекривається частково кнопка. Також наявна нижня панель навігації, що дає змогу зручно пересуватися між екранами.

Рисунок. 1.1.10 - Головний екран, який висвічується при запуску додатку Party Ticket

В даній програмі товаром є квитки на виступи, тож афіши переліковуються як список товарів. На Рисунок 1.1.11 представлено екран з виведенням афіш у вигляді списку. Але відсутня фільтрація для пошуку виступу за смаком. Оскільки програма продає квитки як на вистави, так і на музичні концерти, потрібна фільтрація за типом вистави. Також музичні виконавці виступають в різному жанрі музики, можна додати фільтрацію за жанром виконавця.

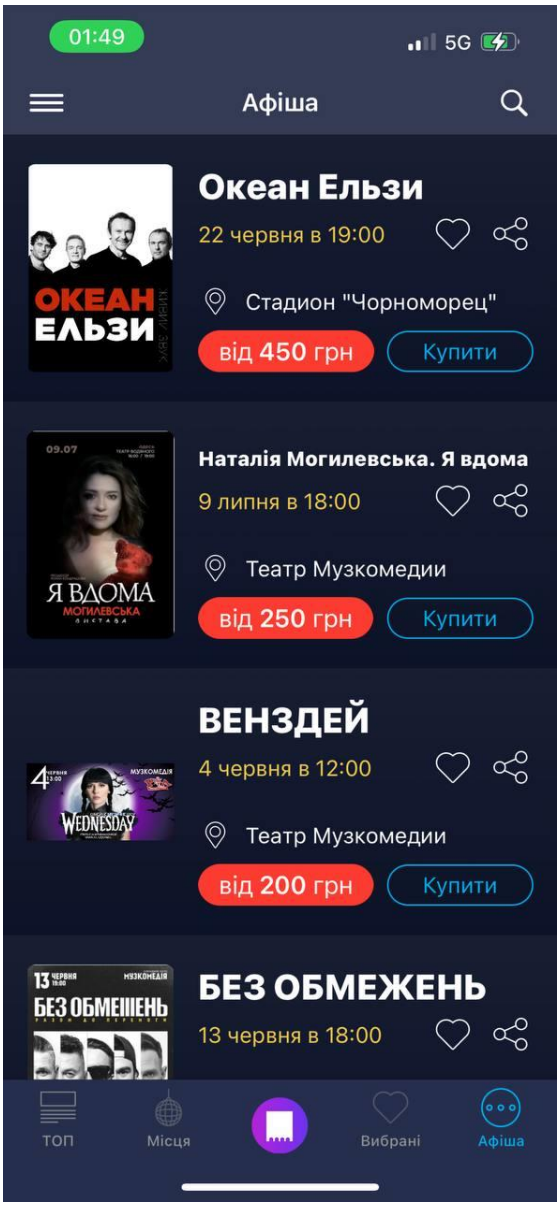


Рисунок 1.1.11 - Екран з афішами виступів в додатку Party Ticket

На Рисунок 1.1.12 представлено екран, який зображує лист місць, де відбуваються події, на котрі можна купити квитки через даний додаток.

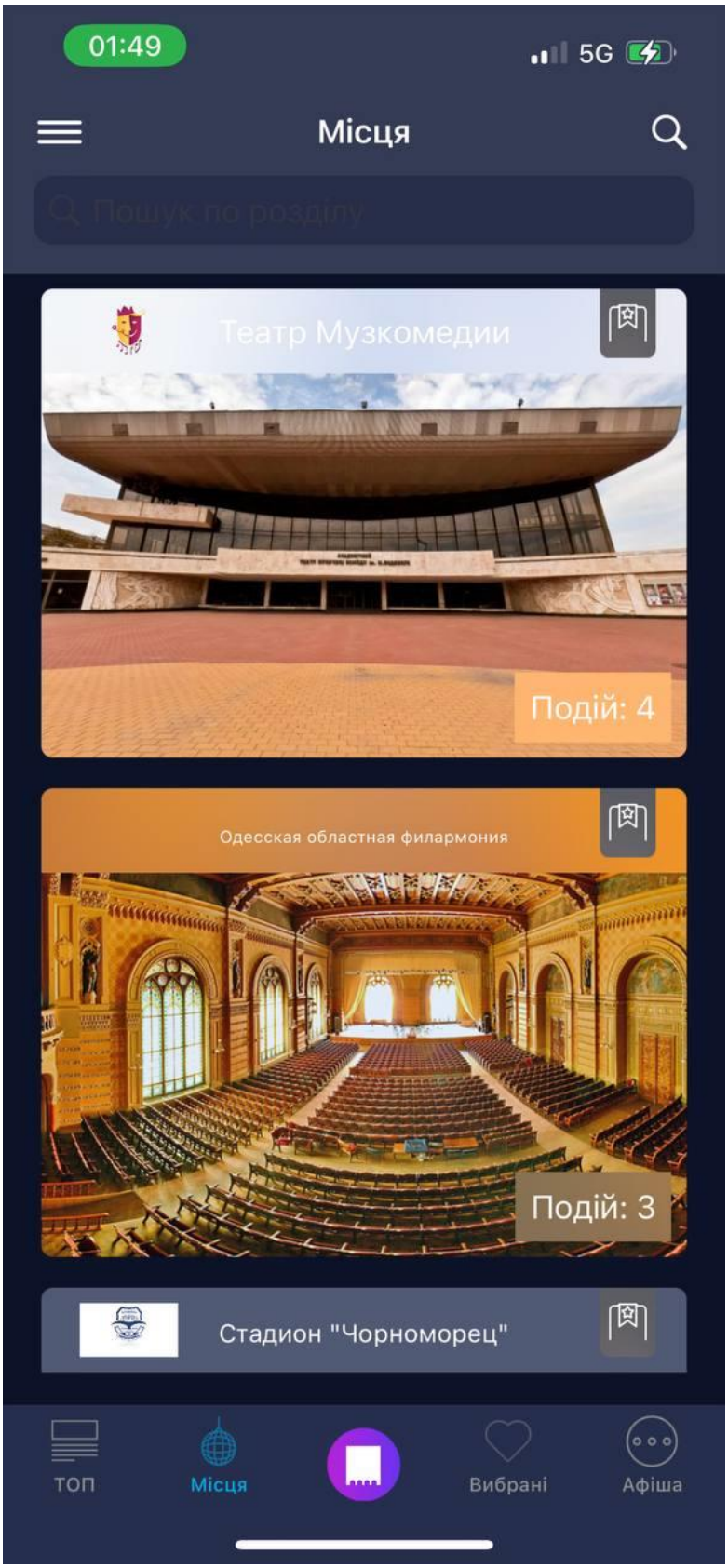


Рисунок 1.1.12 - Екран “місця” в додатку Party Ticket

Окрім нижньої панелі навігації наявна бокова панель(Рисунок 1.1.13), але без авторизації не всі сторінки доступні.

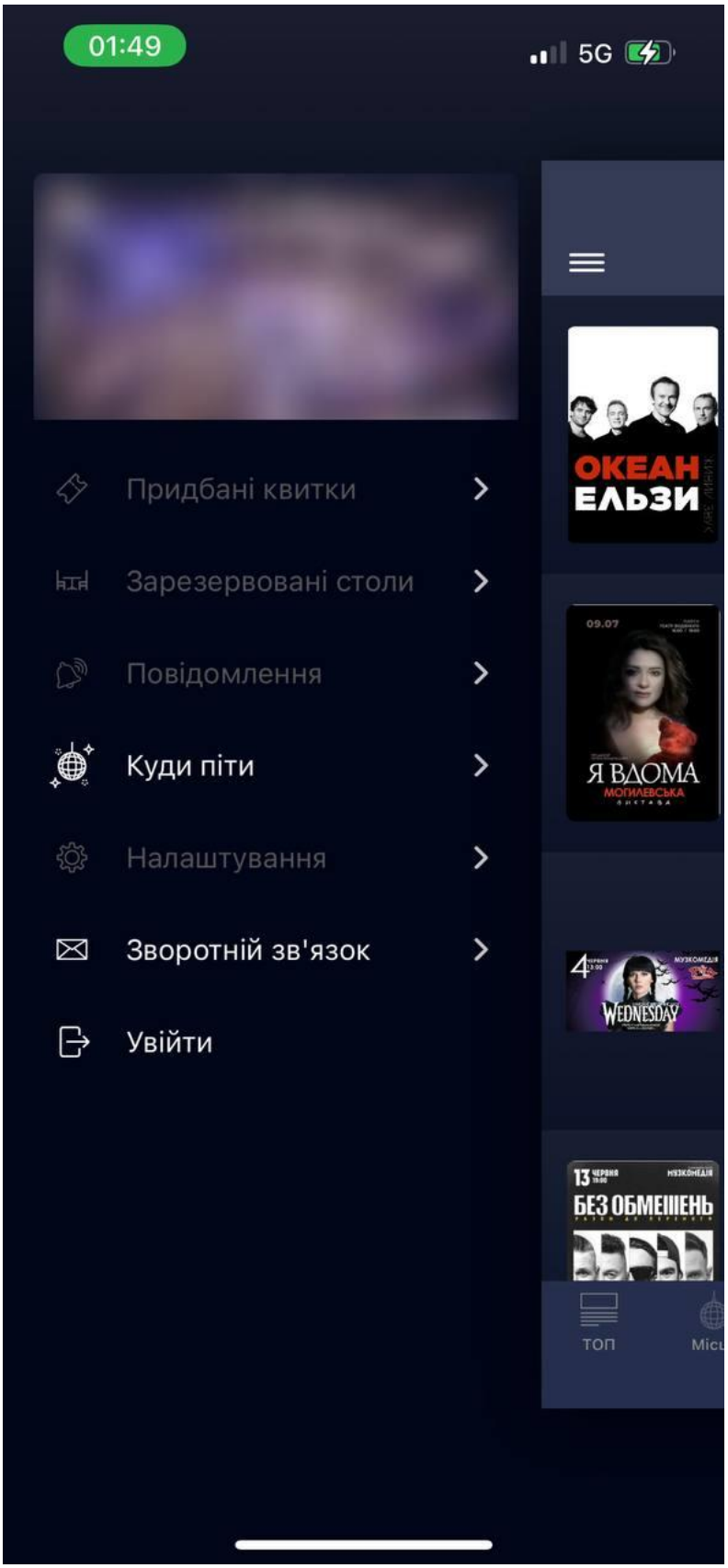


Рисунок 1.1.13- Бокова панель навігації в додатку Party Ticket

В мобільному додатку доступна авторизація та реєстрація(Рисунок 1.1.14) тільки через мобільний телефон.

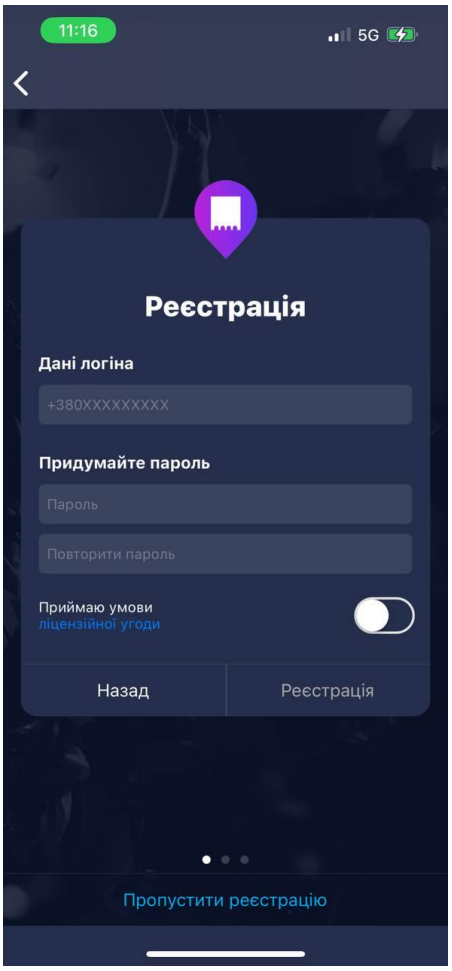


Рисунок 1.1.14 - Сторінка реєстрації в додатку Party Ticket

Переваги Party Ticket:

1. Можливість купляти квитки онлайн;
2. Відстеження виступів в залежності місця проведення;
3. Можливість зворотного зв'язка;
4. Анімація, яка робить інтерфейс привабливішою для користувачів.

Недоліки Party Ticket:

1. Наявність тільки однієї мови, що не дає доступність для користувачів які не розуміють українською;
2. Обов'язковість авторизації для купівлі квитка;
3. Доступність до деяких функцій тільки після авторизації;
4. Некоректна розмітка об'єктів відносно один одного;
5. Відсутність авторизації через акаунти Google або Facebook.

1.4 Висновки

За результатами аналізу аналогів можна зробити висновок, що мобільні додатки мають свої переваги та недоліки. До уваги даних переваг та недоліків можна розробити мобільний додаток, взявши переваги застосунків, та опрацювати недоліки задля запобігання їх в мобільному застосунку. Складемо таблицю(таблиця 1.4.1) для порівняння аналогів, де “+” - це наявність функції, “-” - її відсутність.

Таблиця 1.4.1 - Порівняння аналогів

	Ebay	Feeh La shopping	Party Ticket	Додаток для продажу косметичних засобів
Результат пошуку з використанням фільтру	+	+	-	+
Оплата онлайн	+	+	+	+
Панель навігації	+	+	+	+
Покупка без авторизації	-	-	-	-
Авторизація	+	+	+	+
Підтримка двох платформ операційної системи	+	+	+	+
Можливість додавання товару в кошик	+	+	+	+

2 ВИМОГИ ТА ОЦІНКА ЯКОСТІ СИСТЕМИ

2.1 Функціональні вимоги

Функціональна вимога - це формулювання того, як система повинна поводитися. Він визначає, що повинна робити система, щоб задовольнити потреби або очікування користувача. Функціональні вимоги можна розглядати як функції, які виявляє користувач. [10]

2.1.1 Реєстрація та авторизація

Однією з функції мобільного застосунку для магазина продажу косметичних засобів є реєстрація та авторизація. Для реєстрації в системі користувач повинен заповнити форму з полями для імені, прізвища, адреси електронної пошти та пароля. Для входу в систему користувач повинен ввести адресу електронної пошти та пароль. Якщо вони будуть введені правильно, користувач отримає доступ до свого особистого кабінету.

Таким чином, реєстрація та аутентифікація в мобільних додатках забезпечує безпеку та комфорт користувачаіприяєпідвищеннюефективностіB2Cпродажів косметичної продукції.

2.1.2 Функція додавання продуктів до кошика

Користувач використовує мобільний застосунок для купівлі косметичних засобів онлайн, тож користувачу потрібно мати змогу додати потрібний товар у кошик, задля можливості продовжити шопінг не загубивши товари, які вже були обрани. Користувач може перевірити загальну вартість замовлення. У кошику міститься повна інформація про додані товари та їхню вартість. Після того, як користувач додав до кошика всі необхідні товари, він може перейти до оформлення замовлення, заповнивши свої персональні дані та обравши способи доставки та оплати.

Таким чином, можливість додавання товарів до кошика є важливим елементом мобільних додатків для B2C продажів косметики і допомагає забезпечити зручність та ефективність процесу покупки для користувача.

2.1.3 Функція оформлення замовлення та оплата замовлення

Для оформлення замовлення користувачеві необхідно заповнити форму, вказавши особисту інформацію (ім’я, прізвище, адресу доставки, номер телефону, електронну пошту) та вибрати способи доставки та оплати. Після заповнення форми користувач може перевірити всі деталі, пов’язані з його замовленням, і ввести необхідні виправлення. Після підтвердження замовлення дані про нього будуть збережені в нашій базі. Це забезпечує стабільність і надійність процесу збору та обробки даних.

2.1.4 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) є графічним інструментом моделювання, який відображає взаємодію між акторами (користувачами) та системою у рамках конкретного додатку або системи. Вона дозволяє ідентифікувати основні функціональні можливості системи та стосунки між ними. [6]

Функціональні можливості користувача:

1. Користувач може переглядати товар
2. Додавати косметичні товари в кошик
3. оформлення замовлення
4. Оплата замовлення
5. Регістрація або авторизація

На Рисунок 2.1.4.1 зображено діаграма варіантів використання взаємодії користувача з мобільним додатком для продажу косметичних засобів.

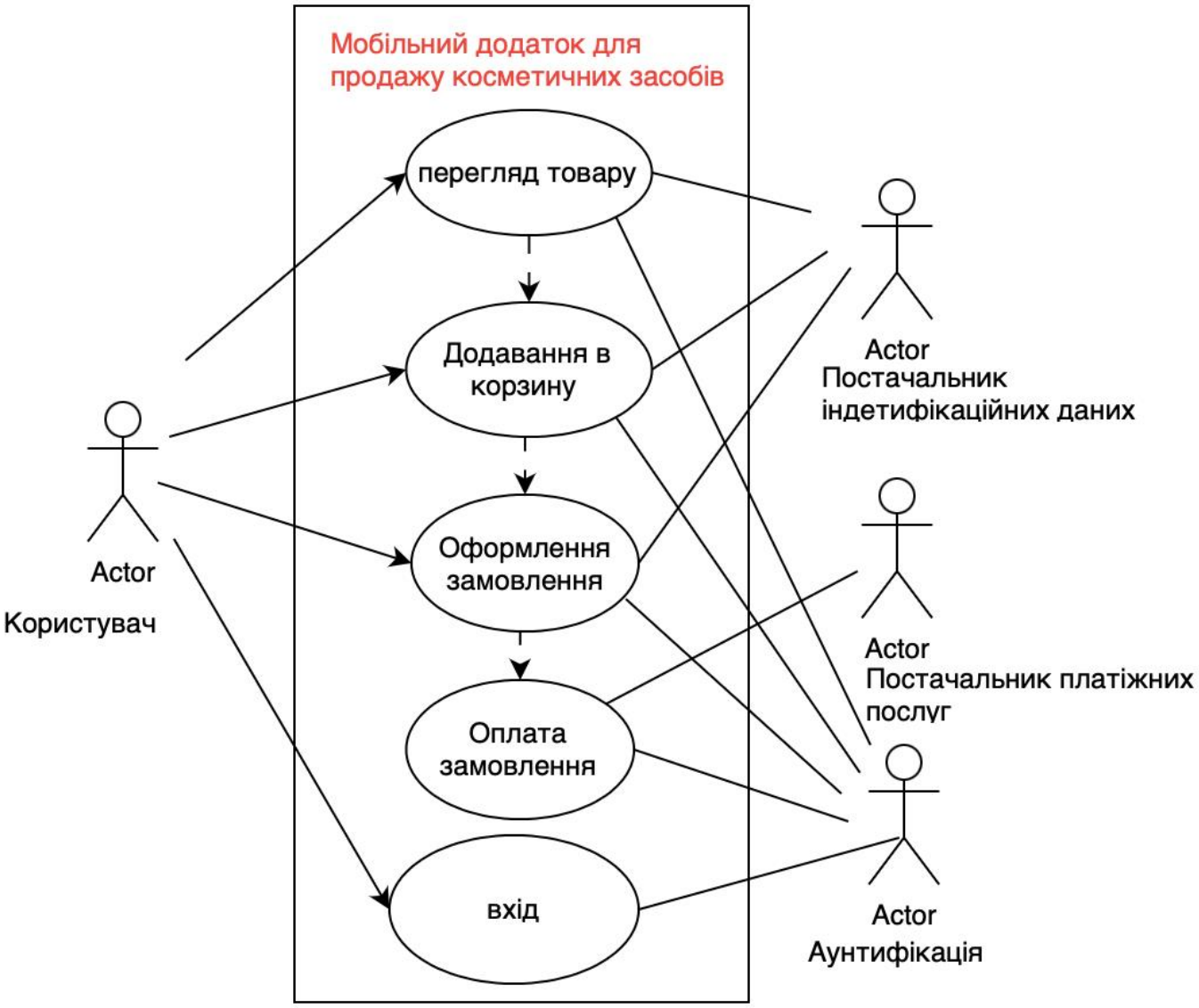


Рисунок 2.1.4.1 - Діаграма варіантів використання - користувач

2.1.5 Діаграма діяльності (Оплата замовлення)

Діаграма діяльності — це графічне зображення послідовності кроків і дій, які відбуваються під час певного процесу чи операції.[6] Дана діаграма показує процес оплати замовлення в додатку для продажу косметичних засобів. Структура процесу оплати замовлення має наступні кроки:

1. Користувач обирає косметичні товари і додає їх до кошика.
2. Користувач переходить до сторінки оформлення замовлення та натискає кнопку "Оплатити".

3. Система перевіряє наявність товарів у кошику та доступність для доставки.

4. Система переадресовує користувача на форму для введення необхідних даних оплати.

5. Користувач вводить необхідну інформацію про платіж (наприклад, номер кредитної карти, CVV-код, адресу доставки тощо) і натискає кнопку "Оплатити".

6. Система передає дані платежу на обробку до платіжного провайдера або банку.

7. Платіжний провайдер або банк перевіряють дані платежу та достатність коштів на рахунку користувача.

8. Система отримує підтвердження оплати від платіжного провайдера або банку.

9. Система оновлює статус замовлення на "Замовлення прийнято".

На Рисунок 2.1.5.1 зображено діаграма діяльності для функції оплати замовлення в мобільному додатку для продажу косметичних засобів.

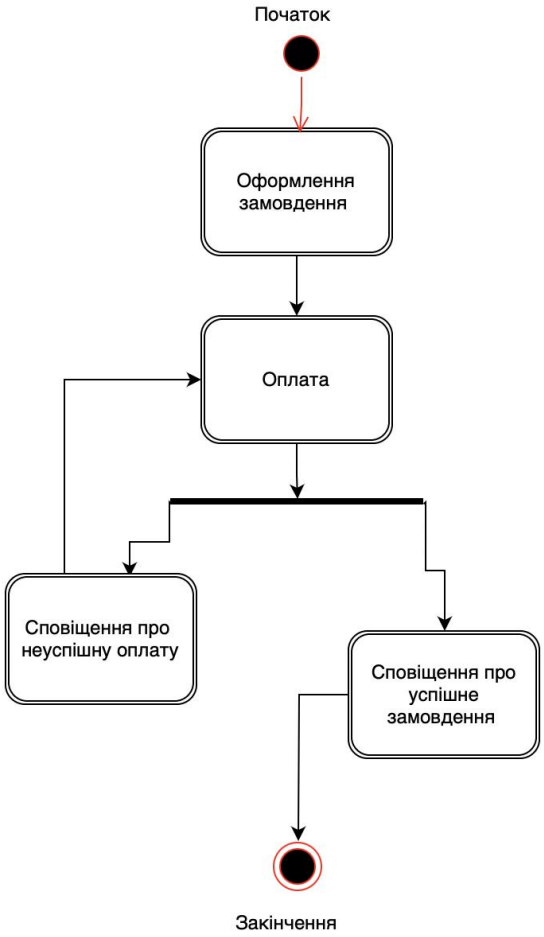


Рисунок 2.1.5.1 - діяльності (Оплата замовлення)

2.1.6 Діаграма класів

Діаграма класів (Class diagram) створена для візуалізації структури класів і взаємозв'язків між ними в програмі. Вона надає графічне представлення класів, їх полів, методів та спільних властивостей.

На Рисунок 2.1.6.1 зображено діаграма класів в мобільному додатку для продажу косметичних засобів.

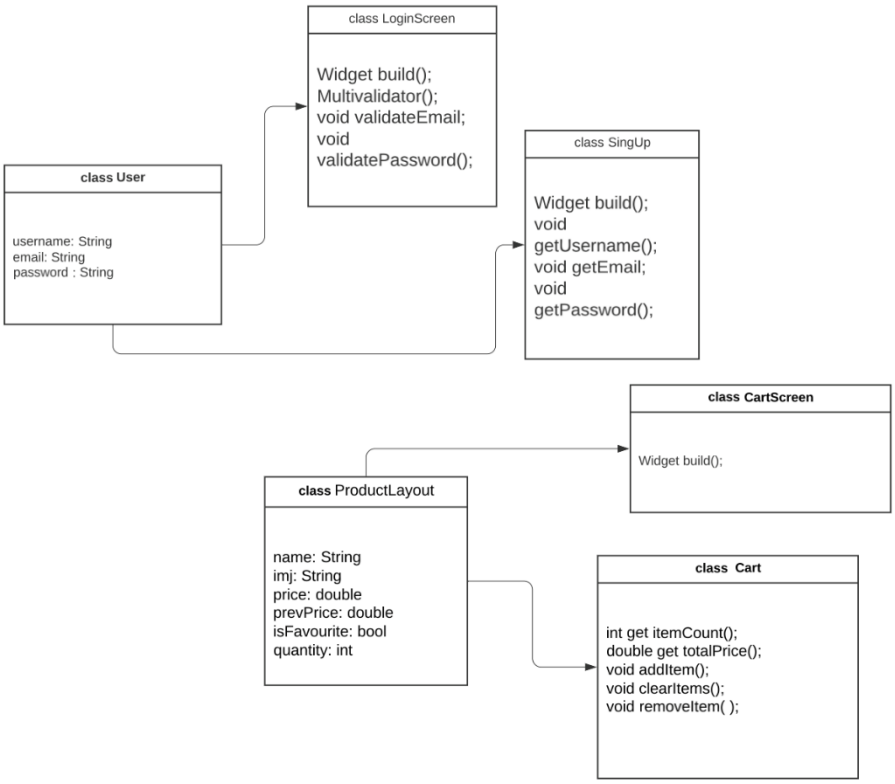


Рисунок 2.1.6.1 - Діаграма класів

2.2 Нефункціональні вимоги

Зосередження лише на функціональних вимогах може бути недостатнім, щоб забезпечити якісний мобільний додаток для мобільної платформи магазину. Тому, крім функціональних вимог, можна також визначити нефункціональні вимоги.

2.2.1 Надійність

Користувач має мати змогу використовувати мобільний додаток в будь-який час доби. Також всі функції мають працювати без затримок та перебоїв. Оскільки це впливає на роботу користувача з мобільним додаток. Мобільний додаток є платформою для купівлі косметичного товару, що впливає на продажі. Конкуренція серед магазинів є високою, особливо на ринку косметичної продукції. Мобільний додаток є один з пунктів, який впливає на підвищення успіху серед конкуренції магазинів.

2.2.2 Швидкодія

Додаток має відповідати на запити користувачів негайно та працювати швидко. Користувачі очікують швидку відповідь на запити, а програми повинні запускатися швидко без непотрібної затримки. Швидкість є ключовою вимогою до додатків мобільного інтернет-магазину, оскільки користувачі очікують, що їхні додатки працюватимуть швидко та ефективно. Якщо ваша програма працює повільно, користувачі можуть бути незадоволені та відмовитися від програми, а це може призвести до втратити потенційних клієнтів.

2.2.3 Безпека

Додаток має забезпечувати безпеку відносно конфіденційності даних користувачів та захисту від кібератак. Додаток містить дані користувачів, а саме їх контакті дані та деякі паспортні дані(ім'я, прізвище, дата народження). Тому забезпечення безпеки щодо конфіденційності даних користувачів є важливим аспектом в розробці.

2.2.4 Доступність та сумісність

На ринку мобільних телефонів найбільш популярні операційні системи це Android та IOS. Тому мобільний додаток має бути доступним на дві операційні системи. Що показує перевагу мови програмування Dart та фреймворку Flutter. Flutter дає змогу писати єдиний код на дві операційні системи(Android та IOS).

2.2.5 Розширюваність

Додаток має бути готовим до майбутнього розширення функціоналу. На сьогоднішній день швидко почала розвиватися штучний інтелект, з оглядом на це

вже існують бібліотеки з штучним інтелектом для використанням в мобільних додатках. Таких як PyTorch Mobile, який можна використовувати для створення моделей рекомендацій на основі даних про косметику та минулі покупки. PyTorch Mobile - це платформа глибокого навчання для навчання та запуску моделей машинного навчання (ML), що прискорює швидкість від дослідження до виробництва. Для інтегрування в Flutter можна за допомогою пакету torch_flutter.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Інструменти для розробки

Існує безліч інструментів для розробки мобільних додатків, які можуть допомогти в створенні високоякісного програмного забезпечення. Для розробки мобільного додатку для продажу косметичних засобів було обрано наступні інструменти:

1. Android Studio;
2. Xcode;
3. Git;
4. Material Design;
5. Dart;
6. Flutter;
7. MVC.

3.1.1 Середовище розробки

Android Studio — це інтегроване середовище розробки (IDE), спеціально розроблене для розробки програм Android. Офіційна IDE для розробки Android, розроблена Google. [8]

Android Studio надає комплексні інструменти та функції, які дозволяють розробникам легко створювати, тестувати та налагоджувати її програми Android. Він пропонує зручний інтерфейс і широкі інструменти для оптимізації процесу розробки.

Android Studio постійно оновлюється новими функціями та вдосконаленнями для підтримки останніх версій платформи Android і практики розробки. Безкоштовне завантаження для операційних систем Windows, macOS і Linux.

На Рисунок 3.1.1.1 зображено вікно інтегрованого середовища розробки - Android Studio.

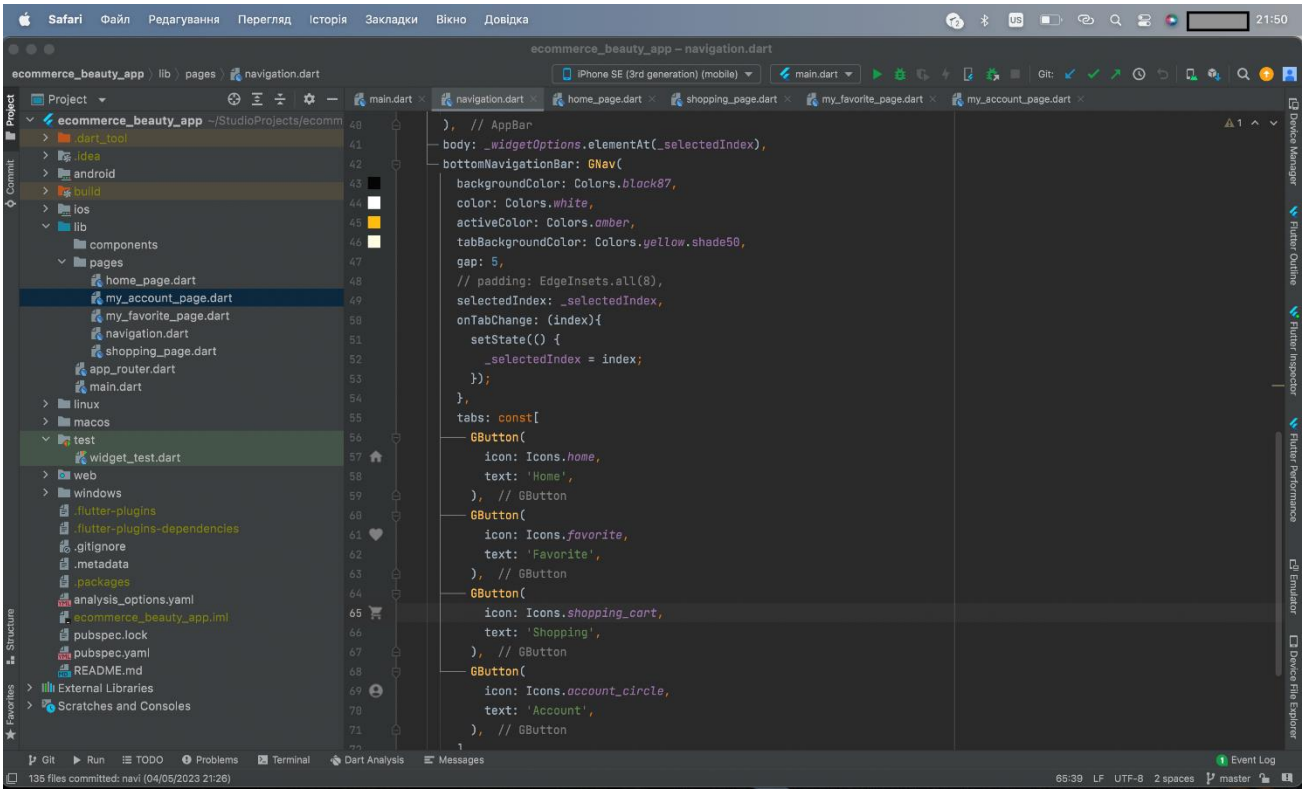


Рисунок 3.1.1.1 - Вікно Android Studio

3.1.2 Xcode IOS Simulator

Xcode — це інтегроване середовище розробки (IDE), розроблене Apple для macOS. Він в основному використовується для розробки програмного забезпечення для платформ Apple, таких як iOS, macOS, watchOS і tvOS. Xcode пропонує широкий набір інструментів, редакторів і утиліт, які спрощують процес розробки для розробників Apple. Він містить такі функції, як редактор вихідного коду з підсвічуванням синтаксису, інструменти налагодження, інструменти розробки інтерфейсу, інструменти аналізу продуктивності та симулятор для тестування програм на різних пристроях. [2]

За допомогою симулятора Xcode IOS Simulator можна запускати програми безпосередньо з ядра macOS, що значно покращує роботу кінцевих користувачів. Зручність цього призвела до розробки незліченних додатків, які регулярно тестуються та працюють на симуляторі.

На Рисунок 3.1.2.1 зображено вікно інтегрованого середовища розробки Xcode.

Рисунок 3.1.2.1 - Вікно Xcode

3.1.3 Мова програмування

Для розробки мобільного застосунку було обрано мову програмування Dart. Дана мова була розроблена компанією Google. Парадигма мови є об’єктна-орієнтована. Тож під час розробки використовувались принципи та фундаментальні поняття ООП: клас, об'єкт, метод, наслідування, інкапсуляція.

Flutter фреймворк, який застосовується, як інструмент для розробки мобільних додатків. Особливістю даного фреймворку є можливість розробки на дві операційні системи(Android та iOS). Інструмент був розроблений мовою програмування Dart. На відміну від програмних фреймворків React і Apache Cordova, архітектура Flutter пропонує адаптивний стиль без використання моста JavaScript, досягаючи рівня продуктивності, який конкурує з його прямим конкурентом React Native.[9]

3.1.4 Git

Git — розподілена система контролю версій, яка широко використовується в розробці програмного забезпечення. Його створив у 2005 році винахідник Linux Лінус Торвальдс. Git дозволяє кільком розробникам співпрацювати над проектом, відстежуючи зміни у файлах і координуючи злиття різних версій. [8]

Git дозволяє кожному розробнику мати повну локальну копію репозиторію проекту, включаючи всю історію. Це дозволяє працювати в автономному режимі та вносити зміни у файли без постійного доступу до центрального сервера. Git відстежує зміни, створюючи серію комітів, які є знімками вашого проекту на певний момент часу.

Розробники можуть створювати гілки в її Git, щоб самостійно працювати над різними функціями та виправляти помилки. Гілки дозволяють паралельну розробку та експериментування без впливу на основну кодову базу. Після внесення змін можна повернутися до головної гілки.

Git надає потужні інструменти для вирішення конфліктів, які можуть виникнути під час об’єднання змін із різних гілок. Він також надає такі функції, як додавання тегів для легкого визначення конкретних моментів в історії проекту та зберігання, що дозволяє розробникам тимчасово зберегти свою роботу та перейти до іншої гілки.

Такі популярні платформи хостингу, як GitHub, GitLab і Bitbucket, пропонують розміщення репозиторію Git і додаткові функції співпраці, такі як відстеження проблем і запити на отримання. Ці платформи полегшують командну співпрацю, перевірку змін коду та керування проектами.

Загалом, Git — це універсальна та широко використовувана система контролю версій, яка допомагає розробникам ефективно співпрацювати, відстежувати зміни у своїй кодовій базі та підтримувати детальну історію розробки проекту.

3.1.5 Material Design

Material Design — це комплексна система дизайну, розроблена Google, щоб забезпечити єдину та узгоджену візуальну мову для розробки інтерфейсів

користувача на різних платформах і пристроях.[5] Мета полягає в тому, щоб створити інтуїтивно зрозумілий, інтерактивний і привабливий досвід для ваших користувачів. Декілька важливих аспектів матеріального дизайну:

1. Material. Мова дизайну натхненна реальними матеріалами та їхніми фізичними властивостями. Використовуйте такі принципи, як висота, світло та тінь, щоб створити відчуття глибини та ієрархії в інтерфейсі користувача. Це допомагає користувачам зрозуміти структуру та зв'язки між різними елементами.
2. Властивість Material. Material Design визначає набір властивостей, які описують поведінку та зовнішній вигляд елементів UI. Ці властивості включають висоту, колір, типографіку, форму, рух тощо. Послідовне використання цих властивостей дозволяє створювати послідовні та візуально привабливі інтерфейси користувача.
3. Адаптивний макет. Material Design наголошує на адаптивних макетах, які адаптуються до різних розмірів і орієнтацій екрана. Заохочуючи використання гнучких сіток і адаптивних контрольних точок, він гарантує, що інтерфейс користувача виглядає та працює добре на різних пристроях, від маленьких мобільних екранів до великих настільних моніторів.
4. Жирна типографіка. Material Design заохочує використання чіткої, розбірливої та жирної типографіки. Надає вказівки щодо вибору правильного шрифту, розміру шрифту та висоти рядка, щоб забезпечити читабельність. Типографіка відіграє важливу роль у передачі ієрархії, важливості та загального тону вашої програми.
5. Яскравий колір. Material Design має яскраву колірну палітру, яку можна використовувати для створення візуально привабливих інтерфейсів користувача. Він надає систему основних і акцентних кольорів, які допомагають створити послідовну візуальну ідентичність вашої програми. Ваш вибір кольорів має бути уважним і відповідати бренду та меті вашої програми.

6. Розумний запит. Рух є невід'ємною частиною матеріального дизайну, привносячи плавність і мету у взаємодію. Це включає в себе анімацію, переходи та жести, які забезпечують зворотній зв'язок, привертають увагу користувача та передають зв'язки між різними елементами. Плавні, цілеспрямовані рухи покращують користувальницький досвід і роблять взаємодію приємнішою.
7. Компоненти та шаблони. Material Design пропонує широкий спектр готових компонентів інтерфейсу користувача та шаблонів, які можна використовувати для створення узгоджених і впізнаваних інтерфейсів. Ці компоненти включають кнопки, картки, списки, ящики навігації, снек-бари тощо. Ці компоненти дозволяють підтримувати звичний користувацький досвід у різних програмах.
8. Доступність. Material Design сприяє інтегративному дизайну та доступності. Він містить вказівки щодо створення інтерфейсів, які можуть використовувати люди з різними здібностями. Це стосується контрасту кольорів, розміру тексту, розміру торкання та підтримки допоміжних технологій. Матеріальний дизайн отримав широке застосування дизайнерами та розробниками завдяки своїй універсальності та орієнтованому на користувача підходу. Він містить набір принципів, інструкцій і ресурсів, які дозволяють створювати візуально привабливі, інтуїтивно зрозумілі та узгоджені інтерфейси користувача на різних платформах. Фреймворк Google Flutter забезпечує чудову підтримку для впровадження Material Design у кросплатформенну розробку програм.

3.2 Архітектурний шаблон програмування MVC

При розробці додатку було обрано архітектурне рішення MVC.MVC розшифровується як Model, View та Controller. Його мета — розділити код і зони відповідальності при розробці програмного забезпечення. Основна мета MVC полягає в тому, щоб відокремити інтерфейс проекту від функціональності та даних, які використовуються програмою. Три компонентами MVC:

1. Model (модель містить дані та логіку, необхідну для маніпулювання даними.)
2. View (представлення стосуються інтерфейсу користувача. Тобто він відображає дані та отримує вхідні дані від користувача);
3. Controller (контролер містить бізнес-логіку. Н. Контроль даних, які відображаються користувачам, обробка введених даних тощо).

Розрізнення між цими трьома основними компонентами проекту допомагає створювати чистий, модульний код, який робить код придатним для повторного використання, і допомагає з паралельною розробкою. Це значно полегшує роботу з проектами, оскільки зміни в одній частині не впливають на інші частини проекту. Тому, щоб тримати їх окремо, потрібен комунікаційний потік, який дозволяє їм взаємодіяти та забезпечувати функціональність проекту.

На рисунку 3.2.1 показано взаємодію призначень MVC. [1]

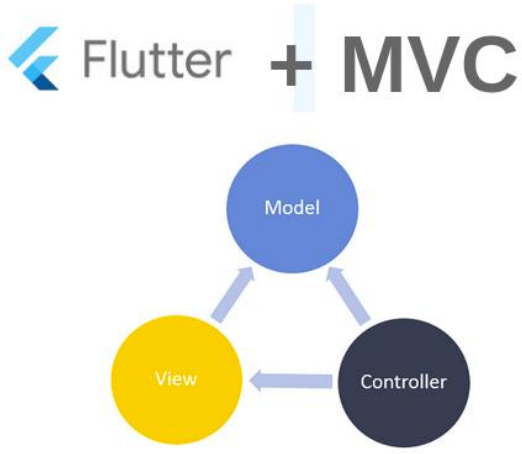


Рисунок 3.2.1 - Взаємодія призначень MVC

4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

При розробці мобільного додатку для продажу косметичних засобів використовувалась офіційна документація про мову програмування Dart та фреймворк Flutter створена компанією Google. [3][4]

4.1 Головна точку входу запуску програми

Метод runApp()(Рисунок 4.1.1) використовується для ініціалізації та запуску програми. Він приймає Widget як аргумент, який є коренем ієрархії віджетів програми. Потім цей віджет передається до FlutterEngine для відтворення на екрані. Метод runApp() викликається у функції main(). main() — це функція, яка є точкою входу. В цьому файлі програма вперше виконується.

В методі runApp() реалізована частинний код навігації мобільного додатку для продажу косметичних засобів, а саме оголошення маршрутів призаченням кожному маршруту “ім’я”. Оскільки мобільний додаток має декілька екранів, то це зумовлює перехід до одного екрану з багатьох частин додатку.

Навігація реалізована за допомогою визначення маршруту з іменем в файлі main.dart і використовується для його навігації.

```
runApp(MultiProvider(
  providers: [
    ChangeNotifierProvider<AuthBlock>.value(value: AuthBlock()),
    ChangeNotifierProvider<Cart>.value(value: Cart()),
    ChangeNotifierProvider<StoreContent>.value(value: StoreContent()),
  ],
  child: MaterialApp(
    localizationsDelegates: [
      AppLocalizations.delegate,
      GlobalMaterialLocalizations.delegate,
      GlobalWidgetsLocalizations.delegate,
      GlobalCupertinoLocalizations.delegate,
    ],
    supportedLocales: [Locale('en'), Locale('ar')],
    debugShowCheckedModeBanner: false,

    theme: FlexColorScheme.light(...).toTheme,  // FlexColorScheme.light
    darkTheme: FlexColorScheme.dark(...).toTheme,  // FlexColorScheme.dark
    themeMode: ThemeMode.light,
    initialRoute: '/',
    routes: <String, WidgetBuilder>{
      '/': (BuildContext context) => LoginScreen(),
      '/forgot': (BuildContext context) => ForgotPassword(),
      '/signup': (BuildContext context) => Signup(),
      '/home': (BuildContext context) => Home(),
      Products.routname: (BuildContext context) => Products(),
      Admin.routename: (BuildContext context) => Admin(),
      EditProducts.routename: (BuildContext context) => EditProducts(),
      Categorise.routeName: (BuildContext context) => Categorise(),
      CartScreen.routeNeme: (BuildContext context) => CartScreen(),
      Checkout.routeName: (BuildContext context) => Checkout(),
```

Рисунок 4.1.1 - функція runApp() в файлі в main.dart

4.2 Навігація в мобільному додатку для продажу косметичних товарів

Задля зручної навігації в додатку було реалізовано бокове меню з переходом на сторінки додатку для продажу косметичної продукції. Реалізація навігації через бокове меню було реалізовано в файлі drawer.dart. Для створення бічного меню використовується віджет AppDrawer(Рисунок 4.2.1). Цей віджет має стан, тому він є StatefulWidget.

_AppDrawerState (Рисунок 4.2.1) реалізує метод build, який повертає віджети для побудови бічного меню. В залежності від значення змінної isAdmin (яка визначає, чи є користувач адміністратором), будуть показані різні елементи меню.

Також використовується Provider для отримання кількості товарів у кошику та ModalRoute для отримання даних про адміністраторський статус користувача.

```
class AppDrawer extends StatefulWidget {
  @override
  _AppDrawerState createState() => _AppDrawerState();
}

class _AppDrawerState extends State<AppDrawer> {
  @override
  Widget build(BuildContext context) {
    final isAdmin = ModalRoute.of(context)!.settings.arguments as bool;
    final cartNumber = Provider.of<Cart>(context).items.length;
    // AuthBlock auth = Provider.of<AuthBlock>(context);
    return isAdmin
        ? Column(...)  // Column
        : Column(...);  // Column
  }
}
```

Рисунок 4.2.1 - код реалізації класів AppDrawer() та _ AppDrawerState()

Основні елементи бічного меню включають зображення, елементи списку, які відповідають домашній сторінці, категоріям, кошику, умовам використання, панелі адміністратора та виходу з облікового запису. Ці елементи надають користувачеві можливість навігації по додатку, перегляду кошика, переходу до адміністраторських функцій тощо. Для реалізації інтерфейсу бічного меню використовувались наступні віджети: Column, Listview, ClipOval, ListTile, Icon, Text та Container. На Рисунок 4.2.2 зображено код реалізації віджету Column() з використанням віджктів Column, Listview, ClipOval, ListTile, Icon, Text та Container.

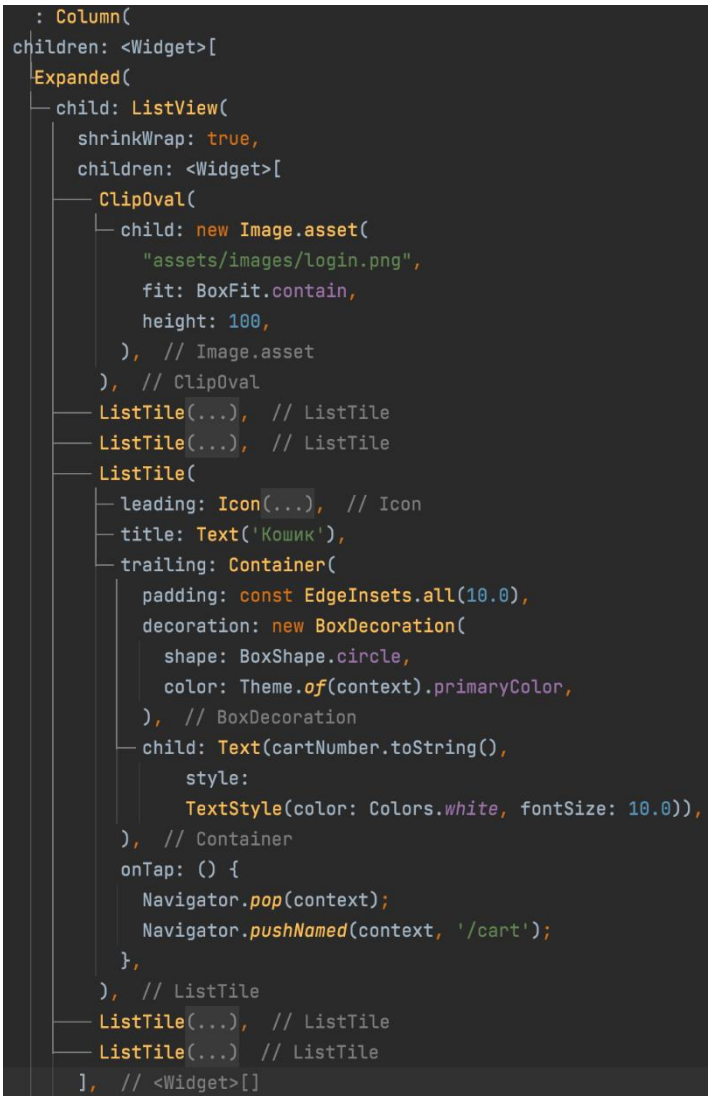


Рисунок 4.2.2 - код реалізації віджету Column() в методі build() в класі _ AppDrawerState() в файлу drawer.dart

4.3 Налаштування залежностей

Для мови програмування Dart потрібні залежності конфігурації. Під час збірки нового проекту з використанням фреймворку Flutter середовище розробки автоматично створює файл pubspec.yaml. Даний пакет містить метадані для визначення його залежностей. Файл pubspec — це файл із кодом YAML під назвою pubspec.yaml, який містить усі метадані пакета, шрифти та файли зображень. В таблиці 4.3.1 представлено бібліотеки, які використовувались під час реалізації проекту.

Таблиця 4.3.1 - Залежності конфігурації

Назва бібліотеки	**Опис**
cupertino_icons	Бібліотека містить набір іконок Cupertino, які використовуються в дизайні iOS.
font_awesome_flutter	Бібліотека містить набір іконки Font Awesome .
google_fonts	Бібліотека шрифтів Google Fonts.
badges	Бібліотека компонентів для створення бейджів (значків).
intl	Бібліотека для локалізації та міжнародного форматування тексту
carousel_slider	Бібліотека створювання карусельних слайдерів для прокрутки контенту
cached_network_image	Бібліотека забезпечує кешування та завантаження зображень з мережі.
flutter_rating_stars	Бібліотека надає компоненти для відображення та вибору рейтингу.
provider	Бібліотека реалізує патерн "постачальник" (provider) для управління станом додатка.
flutter_secure_storage	Бібліотека дозволяє безпечно зберігати та отримувати дані, такі як токени аутентифікації.
fluttertoast	Бібліотека надає можливість виводити сповіщення (тостери)
flex_color_scheme	Бібліотека дозволяє налаштовувати кольорові схеми (themes).
flutter_credit_card	Бібліотека надає компоненти для відображення та редагування кредитних карт.
favorite_button	Бібліотека додає можливість додавати кнопку "Вподобати".
url_launcher	Бібліотека дозволяє відкривати посилання та запускати інші додатки з Flutter додатків.

orm_field_validator	Бібліотека надає зручні засоби для валідації полів форми у Flutter додатках з використанням ORM.
firebase_core	Бібліотека дозволяє інтегрувати Firebase SDK у Flutter додатки та встановлює основні залежності.
firebase_database	Бібліотека забезпечує доступ та взаємодію з Firebase Realtime Database.

Для визначення шрифту вказується в файлі pubspec.yaml(рисунок 22). Ваги шрифтів додані як атрибути до кожного з файлів шрифтів, щоб можна було вказати конкретну вагу шрифту при використанні. Рисунок 4.3.1 показує налаштування використання шрифтів сімейства "Dubai" з різними вагами у додатку.

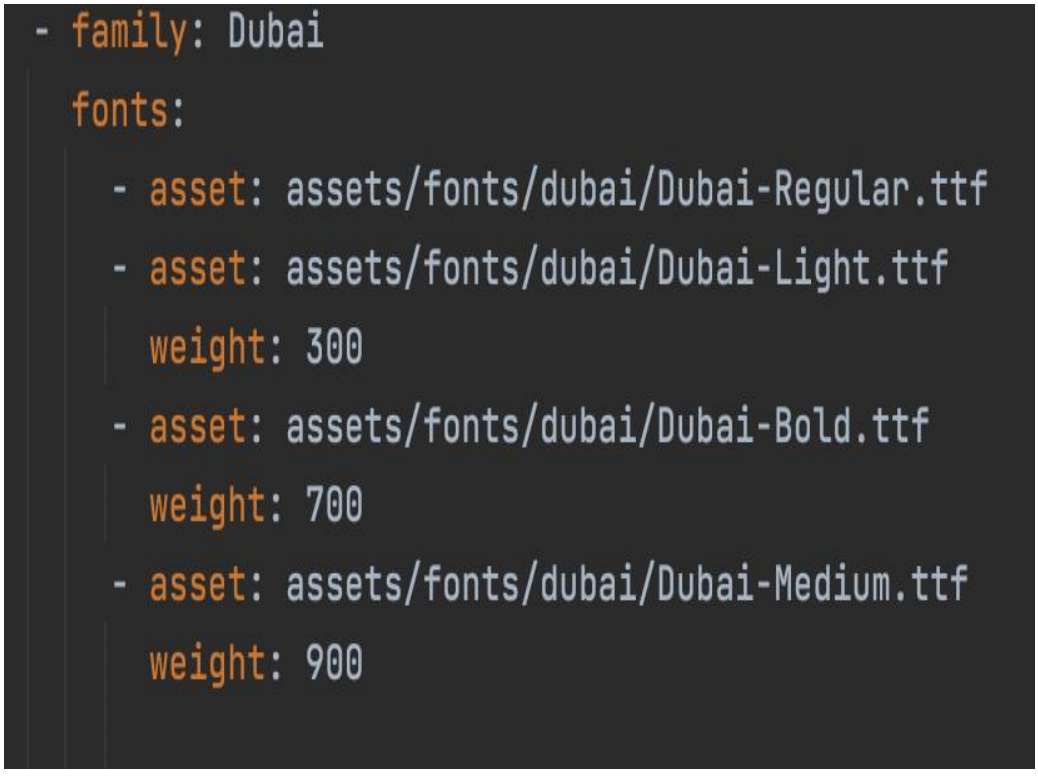


Рисунок 4.3.1 - Вказування шрифта в файлі pubspec.yaml

4.4. Аутеризація

Клас User(Рисунок 4.4.1) визначає модель користувача з основними атрибутами, такими як ім'я користувача, електронна пошта та пароль. Клас також має конструктор(конструктор класу приймає обов'язкові аргументи username, email

та password для створення об'єкту User.) та фабричний метод для створення об'єктів User з вхідних даних у форматі JSON.

```
class User {
  String username;
  String email;
  String password;
  User({required this.username, required this.email, required this.password});
  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      username: json['username'],
      email: json['email'],
      password: json['password']
    );
  }
}
```

Рисунок 4.4.1. - код реалізації класу User() в файлі user.dart

4.4.1 Екран авторизація

Екран авторизації унаслідується від класу StatefulWidget. StatefulWidget є віджетом, метою якого є спостерігати за зміною стану об’єкту з часом. Його побудова здійснюється через два класи. Для реалізації класу було впроваджені наступні крокі:

1. Створено клас, який розширює 'StatefulWidget', який повертає стан у 'createState()'.

2. Створено клас «State» для віджетів, які можуть змінювати свої значення під час виконання.

3. У класі «State» реалізовано метод «build()».

На Рисунок 4.4.1.1 зображено реалізацію класів LoginScreen() та _LoginScreenState().

```dart
class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  GlobalKey<FormState> formkey = GlobalKey<FormState>();
  var isAdmin = false;
  void validate() {
    if (formkey.currentState!.validate()) {
      Navigator.of(context).pushNamed("/home", arguments: isAdmin);
    } else {
      print("не валідно");
    }
  }
  @override
  Widget build(BuildContext context) {...}
}
```

Рисунок 4.4.1.1 - код реалізації класів LoginScreen() та _LoginScreenState()

Екран входу використовується для відображення різних функцій облікового запису, таких як можливість створити новий обліковий запис і можливість відновити пароль. Він також містить кнопку, за допомогою якої можна розпочати процес.

На Рисунок 4.4.1.2 зображення фрагмент коду реалізації інтерфейсу екрану входу.

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.blueGrey ,
    body: SingleChildScrollView(
      child: SizedBox(
        height: 700,
        child: Container(
          padding: const EdgeInsets.fromLTRB(25.0, 25.0, 25.0, 25.0),
          child: Form(
            autovalidateMode: AutovalidateMode.always,
            key: formkey,
            child: Column(...),  // Column
          ),  // Form
        ),  // Container
      ),  // SizedBox
  ));  // SingleChildScrollView, Scaffold
}
```

Рисунок 4.4.1.2 - код методу build() класі _LoginScreenState файлі login-screen.dart

Фрагмент коду на Рисунок 4.4.1.3 зображує частичну реалізацію інтерфейса екрана входу та містить різні поля введення, параметри та кнопки для автентифікації користувача. Інтерфейс користувача побудований на основі Flutter і використовує кілька віджетів Flutter, таких як Scaffold, SingleChildScrollView, TextFormField, GestureDetector і ElevatedButton. Елементи інтерфейсу користувача стилізуються за допомогою різних властивостей, таких як колір, розмір шрифту та вирівнювання.

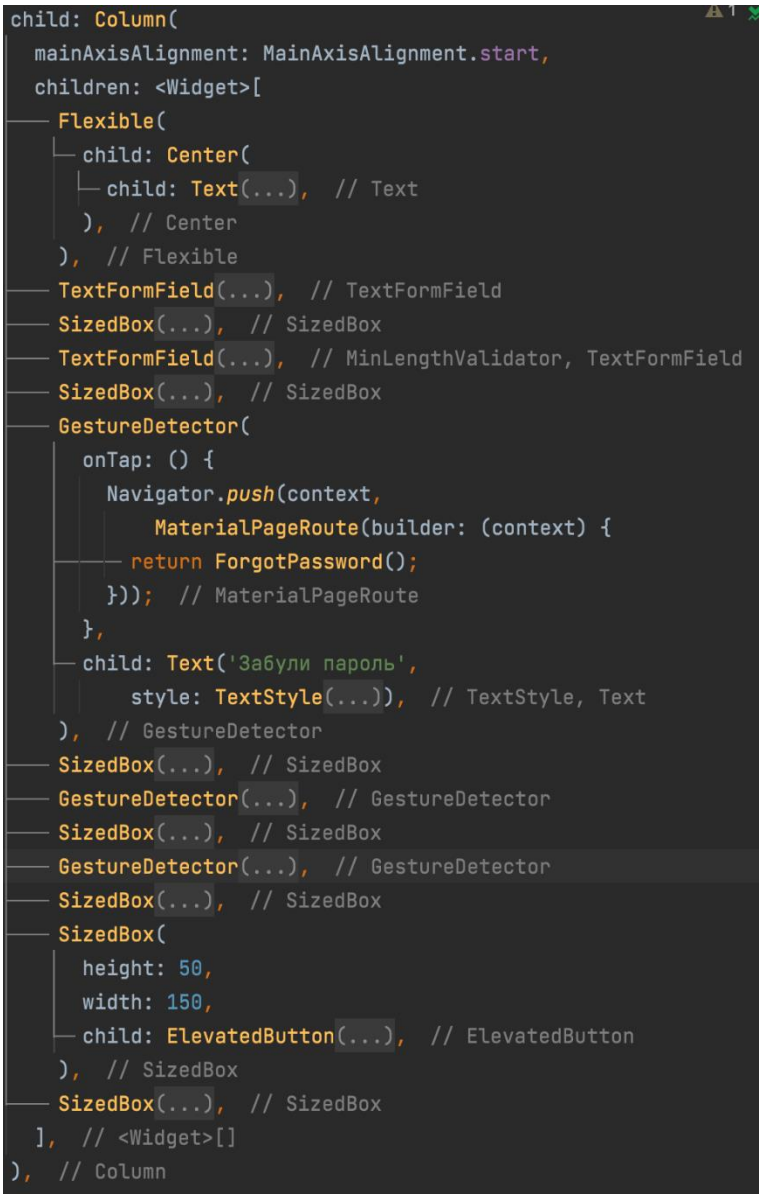


Рисунок 4.4.1.3 - код реаліалізації віджету Column в методі build() в класі _LoginScreenState в файлі login-screen.dart

Віджет TextFormField(Рисунок 4.4.1.4) створює поле введення для збору адрес електронної пошти. Для налаштування зовнішнього вигляду та поведінки передбачено кілька властивостей.

Властивість KeyboardType має значення TextInputType.emailAddress. Це надає користувачеві правильну розкладку клавіатури, включаючи символ «@», для легкого введення електронної пошти.

Властивості оформлення визначають зовнішній вигляд полів введення. Він містить контурну рамку із закругленими кутами, напівпрозорий сірий фон, значок конверта як префікс і текст підказки «email», який відображається в полі введення.

Для властивості Validator встановлено значення MultiValidator, який застосовує кілька правил перевірки до поля введення. RequiredValidator перевіряє, чи поле не пусте, і відображає повідомлення про помилку "Немає бути пусто", якщо воно є. EmailValidator перевіряє, чи введене значення має дійсний формат адреси електронної пошти, інакше відображає повідомлення про помилку "Невалідний емейл".

Загалом віджет TextFormField надає поле введення адреси електронної пошти з візуальним стилем і правилами перевірки, щоб переконатися, що введене значення відповідає вказаним критеріям.

```
TextFormField(
  keyboardType: TextInputType.emailAddress,
  decoration: InputDecoration(
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(16),
    ),  // OutlineInputBorder
    fillColor: Colors.grey[500]!.withOpacity(0.5),
    prefixIcon: Icon(FontAwesomeIcons.envelope),
    hintText: 'Email',
  ),  // InputDecoration
  validator: MultiValidator([
    RequiredValidator(errorText: "Немає бути пусто"),
    EmailValidator(errorText: "Невалідний емейл "),
  ]),  // MultiValidator
),  // TextFormField
```

Рисунок 4.4.1.4 - код реалізації форми запису пошти віджетом TextFormField() в _LoginScreenState файлі login-screen.dart

Віджет ElevatedButton(Рисунок 4.4.1.5) використовується для створення кнопок. Він має властивості, які визначають поведінку, мітку та стиль кнопки.

Властивість onPressed встановлена на функцію перевірки, яка є зворотним викликом, що викликається, коли натискається кнопка. Властивість дочірнього елемента встановлюється на текстовий віджет із текстом «Зайти», який представляє назву кнопки. Властивість style визначає зовнішній вигляд кнопки. Властивість

backgroundColor використовує MaterialStateProperty.all, щоб встановити основний колір фону кнопки, визначений у темі програми. Завдяки цьому кнопка виглядає та відчувається відповідно до решти інтерфейсу користувача.Для властивості Shape встановлено значення RoundedRectangleBorder із радіусом 10 для круглої межі. Це надає кнопкам приємний і сучасний вигляд із закругленими кутами. Загалом цей віджет ElevatedButton відображає кнопку, яка запускає дію під час натискання, і відображає текст «Зайти» як підпис.

```
child: ElevatedButton(
  onPressed: validate,
  child: Text("Зайти"),
  style: ButtonStyle(
    backgroundColor: MaterialStateProperty.all(
        Theme.of(context).primaryColor),
    shape: MaterialStateProperty.all(
      RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10)),
    ),
  ),  // ButtonStyle
),  // ElevatedButton
```

Рисунок 4.4.1.5 - код реалізації кнопки віджетом ElevatedButton() в _LoginScreenState файлі login-screen.dart

На екрані входу користувачі можуть ввести адресу електронної пошти та пароль, відновити пароль або створити новий обліковий запис. Він також забезпечує перевірку форми для полів електронної пошти та пароля. Щоб фрагмент коду працював належним чином, важливо, щоб використовувані віджети та бібліотеки мали необхідні імпорти.

4.4.2 Регестрація

Клас Singup()(Рисунок 4.4.2.1) є віджетом із збереженням стану (StatefulWidget) і відповідає за відображення сторінки реєстрації. Цей клас містить

форму для введення даних користувача, таких як ім’я, адреса електронної пошти, пароль, і кнопку «Зареєструватися» для надсилання даних.

Метод validate() викликається при натисканні кнопки "Реєстрація". Він перевіряє, чи пройшла форма валідацію. Якщо форма валідна, він перенаправляє користувача на домашню сторінку. У іншому випадку виводить повідомлення "не валідний".

```dart
class Signup extends StatefulWidget {
  @override
  State<Signup> createState() => _SignupState();
}

class _SignupState extends State<Signup> {
  GlobalKey<FormState> formkey = GlobalKey<FormState>();
  var isAdmin = false;
  void validate() {
    if (formkey.currentState!.validate()) {
      Navigator.of(context).pushNamed("/home", arguments: isAdmin);
    } else {
      print("не валідний");
    }
  }

  @override
  Widget build(BuildContext context) {...}
}
```

Рисунок 4.4.2.1 - код реалізації класів Signup() та _SignupState() в файлі singup.dart

Метод build(BuildContext context)(Рисунок 4.4.2.2) викликається для побудови та повернення віджета, який буде відображений на екрані. У цьому методі визначається вигляд та структура сторінки реєстрації.

```
@override
Widget build(BuildContext context) {
  Size size = MediaQuery.of(context).size;
  return Scaffold(
    backgroundColor: Colors.blueGrey ,
    body: SingleChildScrollView(
      child: SizedBox(
        height: 800,
        child: Container(
          padding: const EdgeInsets.fromLTRB(25.0, 25.0, 25.0, 25.0),
          child: Column(...),  // Column
        ),  // Container
      ),  // SizedBox
    ),  // SingleChildScrollView
  );  // Scaffold
```

Рисунок 4.4.2.2 - код реалізації методу build() класі _SignupState() в файлі singup.dart

Віджети Column(Рисунок 4.4.2.3) використовуються для створення вертикального макета для впорядкування дочірніх віджетів. Властивість children є списком віджетів.

Віджета SizedBox, який забезпечує порожній простір із висотою

size.width * 0,1.Віджет Stack використовується, щоб скласти кілька дочірніх віджетів один на одного. Він містить віджет Center, який центрує дочірній віджет ClipOval. Віджет ClipOval обрізає свою властивостість child до овалу. Дочірній елемент ClipOval — це віджет BackgroundFilter, який застосовує ефект розмиття до своїх дочірніх елементів. Дочірній елемент BackgroundFilter — це віджет CircleAvatar, який відображає круглий аватар із розмитим фоном. CircleAvatar має віджет Icon із піктограмою FontAwesome «Користувач» білого кольору. Цей набір віджетів створює круглий аватар із розмитим фоном і піктограмою користувача в центрі.

Віджет Positioned впорядковуються в віджет Stack і розміщують піктограми зі стрілками в певних місцях. Він визначає позицію за допомогою властивостей top і left. Дочірній елемент Positioned — це віджет Container.

Кнопку створено за допомогою віджета ElevatedButton. Він має властивість стилю, яка визначає зовнішній вигляд кнопки, наприклад колір і форму фону. На кнопці відображається текст «Реєстрація». Властивість onPressed налаштована на функцію перевірки, яка є функцією зворотного виклику під час натискання кнопки.

```
child: Column(
  children: [
    SizedBox(...),  // SizedBox
    Stack(
      children: [
        Center(
          child: ClipOval(...),  // ClipOval
        ),  // Center
        Positioned(
          top: size.height * 0.08,
          left: size.width * 0.56,
          child: Container(
            height: size.width * 0.1,
            width: size.width * 0.1,
            decoration: BoxDecoration(
              color: Color(0xff5663ff),
              shape: BoxShape.circle,
              border: Border.all(color: Colors.white, width: 2),
            ),  // BoxDecoration
            child: Icon(
              FontAwesomeIcons.arrowUp,
              color: Colors.white,
            ),  // Icon
          ),  // Container
        )  // Positioned
      ],
    ),  // Stack
    SizedBox(...),  // SizedBox
    Form(...),  // Form
  ],
),  // Column
```

Рисунок 4.4.2.3 - код реалізації віджету Column в методі build() в класі _SignupState() в файлі singup.dart

Віджет Form(Рисунок 4.4.2.4) використовуються для створення форм із автоматичною перевіркою. Властивість autovalidateMode має значення

AutovalidateMode.always, щоб увімкнути автоматичну перевірку форми. Властивість ключа встановлюється на унікальний ключ форми, formkey. Форма має віджет Column, який містить кілька віджетів TextFormField.

Кожне TextFormField (рисунок) представляє поле введення у формі та має різноманітні властивості, такі як декорації, типи клавіатури, засоби перевірки тощо, щоб налаштувати його зовнішній вигляд і поведінку. Властивості оформлення використовуються для визначення зовнішнього вигляду полів введення, таких як рамки, кольори заливки, префікси та текст підказки. Властивість KeyboardType визначає тип клавіатури, що відображається в полі введення. Властивість Validator використовується для перевірки введених користувачем даних на відповідність певним правилам. Обов'язкова перевірка полів, перевірка формату електронної пошти та перевірка мінімальної довжини.

```
Form(
  autovalidateMode: AutovalidateMode.always,
  key: formkey,
  child: Column(
    children: [
      TextFormField(...),  // TextFormField
      SizedBox(...),  // SizedBox
      TextFormField(...),  // TextFormField
      SizedBox(...),  // SizedBox
      TextFormField(...),  // MinLengthValidator, TextFormField
      SizedBox(...),  // SizedBox
      TextFormField(...),  // MinLengthValidator, TextFormField
      SizedBox(...),  // SizedBox
      ElevatedButton(...),  // ElevatedButton
      SizedBox(...),  // SizedBox
      Row(...),  // Row
      SizedBox(...),  // SizedBox
    ],
  ),  // Column
)  // Form
```

Рисунок 4.4.2.4 - код реалізації віджету Form() в методі build() в класі _LoginScreenState в файлі login-screen.dart

Створено віджет Row(Рисунок 4.4.2.5), який використовує віджети рядків для впорядкування дочірніх віджетів. Властивість mainAxisAlignment має значення MainAxisAlignment.center для горизонтального центрування дочірніх елементів. Дочірніми елементами рядка є два текстові віджети. Перший приклад відображає текст «Вже є акаунт?». Другий відображає текст «Вхід». Другий текстовий віджет загорнутий у віджет GestureDetector, тому торкання його запускає навігацію до сторінки входу.

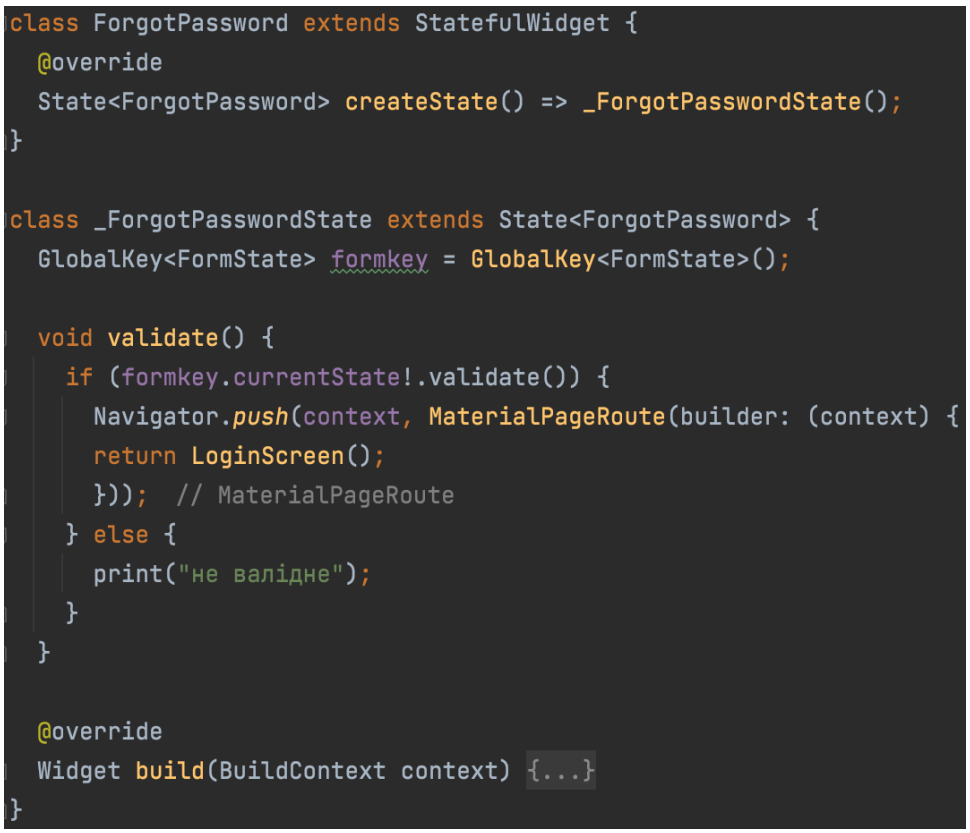
Рисунок 4.4.2.5 - код реалізації віджету Row() в методі build() в класі _LoginScreenState в файлі login-screen.dart

4.4.3 Реалізація функції відновлення доступу до акаунта

Якщо користувач забув пароль від свого акаунта, додаток пропонує фунцію відновлення доступу до акаунта. ForgotPassword

Клас ForgotPassword() наслідується від класу StatefulWidget() і відповідає за відображення сторінки "Забули пароль". Клас містить форму для введення електронної пошти користувача, на яку будуть надіслані інструкції для скидання

пароля. На Рисунок 4.4.3.6 зображено код реалізації класів ForgotPassword() та _ForgotPassword State().



```dart
class ForgotPassword extends StatefulWidget {
  @override
  State<ForgotPassword> createState() => _ForgotPasswordState();
}

class _ForgotPasswordState extends State<ForgotPassword> {
  GlobalKey<FormState> formkey = GlobalKey<FormState>();

  void validate() {
    if (formkey.currentState!.validate()) {
      Navigator.push(context, MaterialPageRoute(builder: (context) {
      return LoginScreen();
      }));  // MaterialPageRoute
    } else {
      print("не валідне");
    }
  }

  @override
  Widget build(BuildContext context) {...}
}
```

Рисунок 4.4.3.6 - код реалізації класів ForgotPassword() та _ForgotPassword State() в файлі forgot-password.dart

Віджет SingleChildScrollView(Рисунок 4.4.3.7) забезпечує прокручування свого дочірнього вмісту в разі, якщо вміст не поміщається на екрані. Вміст розміщений всередині контейнера з встановленими відступами та кольором фону. Контейнер має також декоративне зображення на фоні, яке можна налаштувати. Всередині контейнера розміщений стовпчик з формою для введення електронної пошти та кнопкою "Send" для відправки запиту на скидання пароля. В таблиці 4.4.3.1 представлені атрибути використані в віджеті SingleChildScrollView.

Таблиця 4.4.3.1 - Атрибути віджета SingleChildScrollView

Атрибут віджета SingleChildScrollView	Опис
padding	Відступи всередині контейнера

height	Висота контейнера
color	Колір фону контейнера
decoration	Декорація контейнера
child (Widget)	Дочірній вміст контейнера, який містить форму та кнопку.

```
Widget build(BuildContext context) {
  Size size = MediaQuery.of(context).size;
  return Stack(children: [
    Scaffold(
      appBar: AppBar(...),  // AppBar
      body: SingleChildScrollView(
        child: Container(
          padding: EdgeInsets.all(25.0),
          height: 700,
          color: Colors.blueGrey,
          decoration: BoxDecoration(...),  // DecorationImage, BoxDecoration
          child: Column(
            children: [
              Form(
                autovalidateMode: AutovalidateMode.always,
                key: formkey,
                child: Center(
                  child: Column(
                    children: [
                      SizedBox(...),  // SizedBox
                      Container(
                        width: size.width * 0.8,
                        child: Text(...),  // TextStyle, Text
                      ),  // Container
                      SizedBox(...),  // SizedBox
                      TextFormField(...),  // TextFormField
                      SizedBox(
                        height: 20,
                      ),  // SizedBox
                      SizedBox(
                        height: 50,
                        width: 150,
                        child: ElevatedButton(
                          onPressed: validate,
                          child: Text("Send"),
                          style: ButtonStyle(...),  // ButtonStyle
```

Рисунок 4.4.3.7 - код реалізації методу biuld() в _ForgotPassword State() в файлі forgot-password.dart

4.5 Реалізація функції “кошик” для оформлення замовлення косметичних засобів

Становий віджет CartScreen(Рисунок 4.5.1) відображає екран кошика. Є статичне поле routeName, що містить шлях до цього екрану. _CartScreenState(Рисунок 4.5.1) є приватним станом для CartScreen. В ньому створюється порожній список products, який буде використовуватись для збереження товарів у кошику. Метод build викликається для побудови і повернення віджету. В цьому методі отримується інформація про кошик з Provider, і обчислюється загальна вартість товарів у змінній total.

```dart
class CartScreen extends StatefulWidget {
  static const routeNeme = '/cart';
  @override
  _CartScreenState createState() => _CartScreenState();
}

class _CartScreenState extends State<CartScreen> {
  final List<Map<dynamic, dynamic>> products = [
  ];

  @override
  Widget build(BuildContext context) {
    final cartinfo = Provider.of<Cart>(context);
    var total = cartinfo.totalPrice;
    return Scaffold(
      appBar: AppBar(
        title: Text('Кошик'),
        actions: [
          IconButton(
              tooltip: 'Очистити кошик',
              onPressed: () {
                cartinfo.clearItems();
              },
              icon: Icon(Icons.clear))  // IconButton
        ],
      ),  // AppBar
      body: Column(...));  // Column, Scaffold
  }
}
```

Рисунок 4.5.1 - код реалізації класів CartScreen() та _ CartScreenState() в файлі cart.dart

Наданий фрагмент коду показує структуру віджета Column(Рисунок 4.5.2), який містить кілька вертикально розташованих дочірніх віджетів. Віджет Column створюється для створення вертикального масиву дочірніх віджетів. Column

містить два віджети Padding з різними властивостями. Padding - це віджет із заповненням, який додає відступ навколо дочірнього віджета.

Перший віджет Padding має віджет Container, який містить текстовий віджет. Цей текстовий віджет відображає кількість товарів у кошику, отриману з cartinfo.items.length. Текст форматується за допомогою класу TextStyle, а колір встановлюється на основний колір поточної теми.

Другий віджет Padding містить віджет ButtonTheme. ButtonTheme має віджет ElevatedButton, який запускає дію при натисканні. Для тексту кнопки встановлено значення "Купити" і стилізовано основними кольорами поточної теми. Коли кнопка натиснута, вона перевіряє, чи є в кошику товари, і якщо так, то переходить на сторінку оформлення (Checkout.routeName).

```
body: Column(
  children: <Widget>[
    Padding(
      padding: const EdgeInsets.only(top: 12.0, bottom: 12.0),
      child: Container(
        child: Text(...)),  // TextStyle, Text, Container
    ),  // Padding
    Flexible(...),  // Flexible
    Container(...),  // Padding, Container
    Padding(
      padding: const EdgeInsets.only(...),  // EdgeInsets.only
      child: ButtonTheme(
        buttonColor: Theme.of(context).primaryColor,
        minWidth: double.infinity,
        height: 40.0,
        child: ElevatedButton(
          onPressed: () {
            if (cartinfo.items.length == 0) {
              return;
            } else
              Navigator.of(context).pushNamed(Checkout.routeName);
          },
          child: Text(
            "Купити",
            style: TextStyle(color: Colors.white, fontSize: 16),
          ),  // Text
        ),  // ElevatedButton
      ),  // ButtonTheme
    ),  // Padding
  ],  // <Widget>[]
));  // Column, Scaffold
```

Рисунок 4.5.2 - код реалізації віджету Column в методі build() в класі _CartScreenState() в файлі cart.dart

Віджет Flexible(Рисунок 4.5.3) використовуються для створення гнучких макетів, які розширюються, поки не заповнять доступний простір. Обгортає віджет ListView.builder і дозволяє змінювати його розмір залежно від доступного простору макета. У ListView.builder властивість itemCount встановлюється на довжину cartinfo.items, щоб визначити кількість елементів для відображення в списку. Властивість itemBuilder — це функція зворотного виклику, яка визначає макет і поведінку кожного елемента в списку. Поточний контекст та індекс використовуються як параметри.

Кожен елемент у списку представлено віджетом. Віджет, який можна закрити, дає змогу закривати дочірні віджети, проводячи пальцем. Для ідентифікації кожного елемента потрібен унікальний ключ. Властивість onDismissed налаштована на функцію зворотного виклику, яка надсилається, коли елемент видаляється. Це використовує cartinfo.removeItem для видалення відповідного елемента з cartinfo.items, оновлення статусу та відображення його SnackBar із повідомленням підтвердження. Крім того, виклик products.removeAt(index) видаляє елемент зі списку продуктів. Функція setState використовується для сповіщення Flutter про те, що інтерфейс користувача потребує оновлення.

Дочірнім елементом Dismissible є віджет InkWell, який додає ефект хвилі дотику під час торкання елемента.

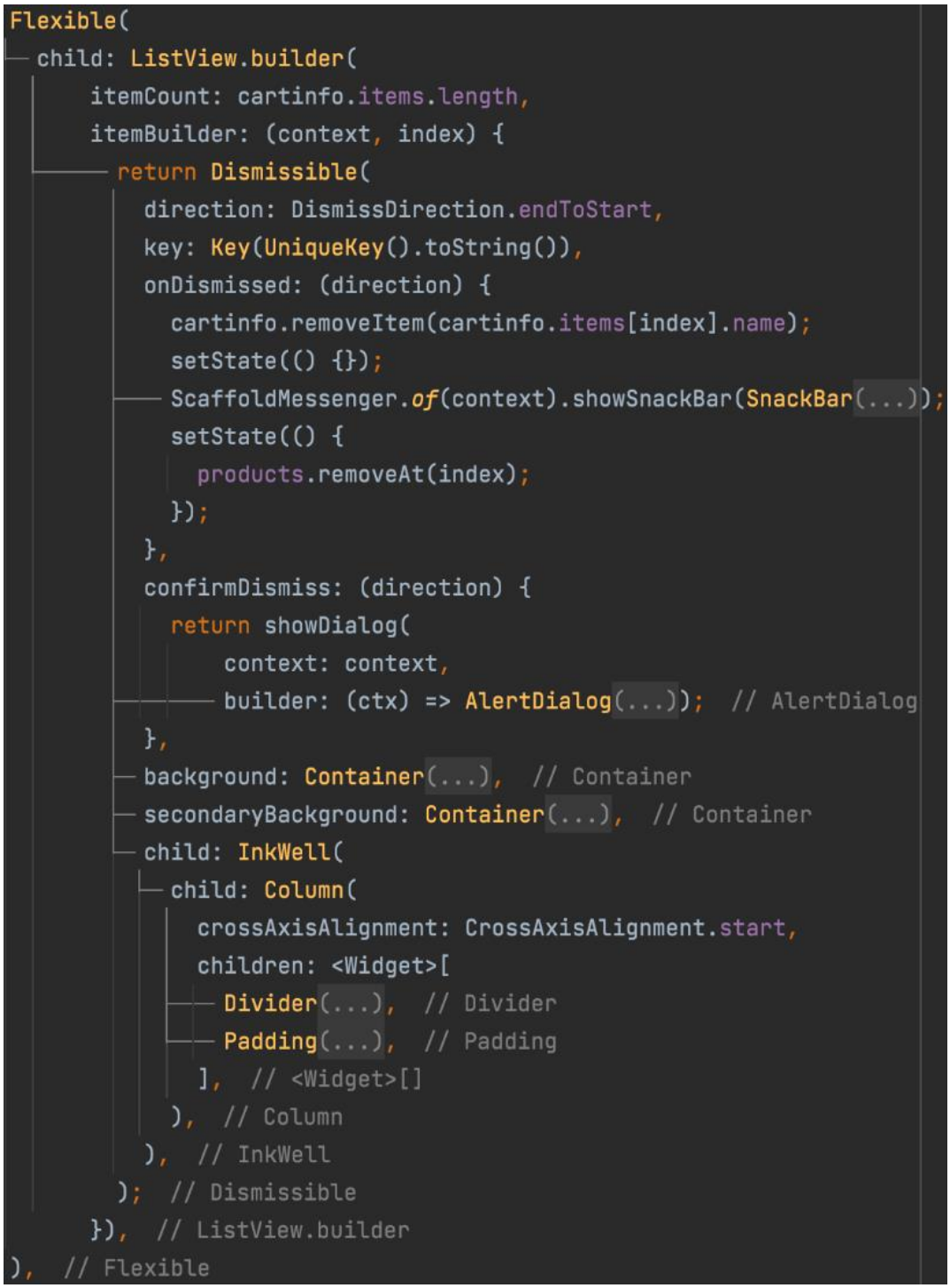


Рисунок 4.5.3 - код реалізації віджету Flexible() в методі build() в класі _ CartScreenState() в файлі cart.dart

Властивість confirmDismiss(Рисунок 4.5.3) налаштована на функцію зворотного виклику, яка реалізує віджет AlertDialog(Рисунок 4.5.4), коли користувач намагається закрити елемент. AlertDialog запитує підтвердження видалення елемента. Якщо користувач вибирає «Так», елемент віддаляється, інакше він залишається в списку.

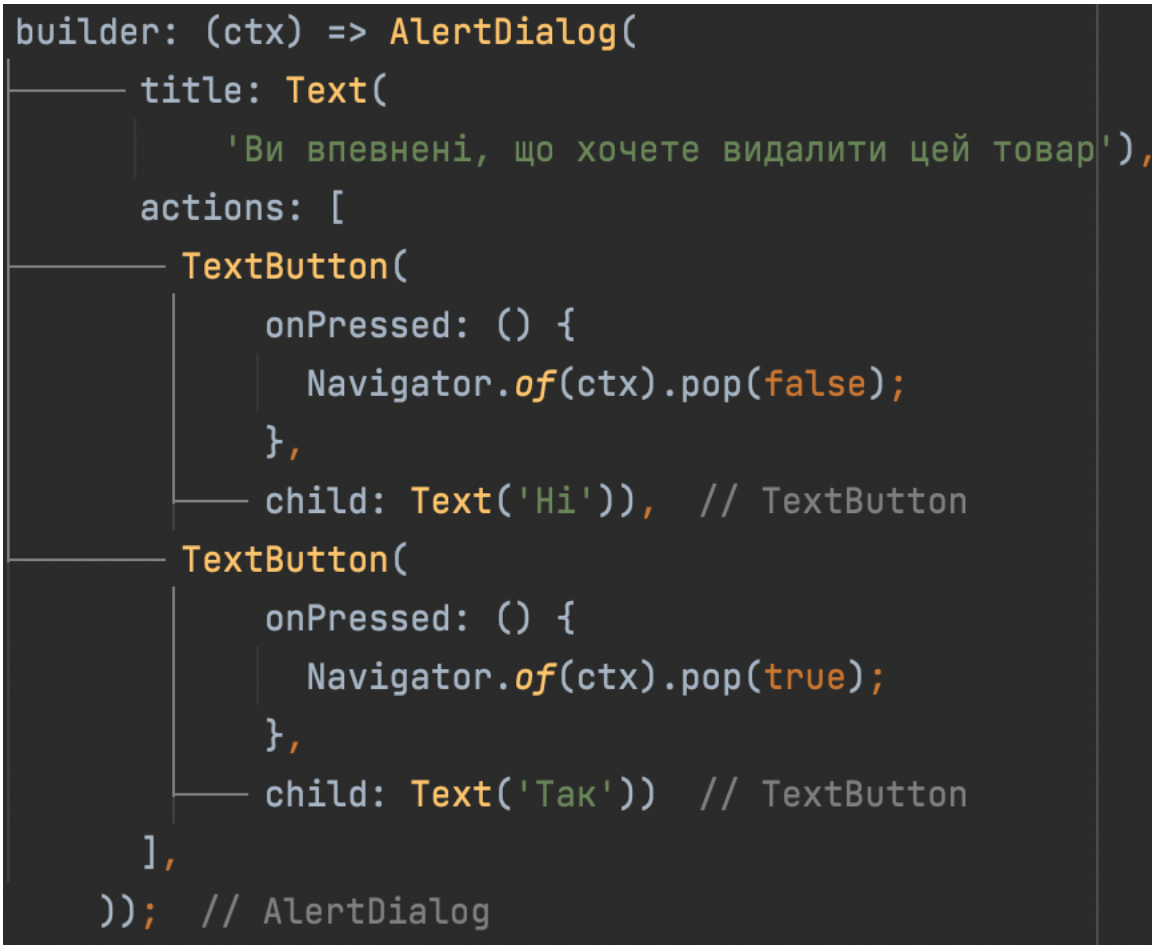


Рисунок 4.5.4 - код реалізації віджету AlertDialog в методі build() в класі _CartScreenState() в файлі cart.dart

Клас Cart(Рисунок 4.5.5) включає приватну змінну _items, яка містить список товарів у кошику. Реалізовано наступні методи:

- addItem - метод для додавання товару до кошика. Якщо товар з таким ім'ям вже присутній, збільшується його кількість на 1. Якщо товару немає в кошику, додається з початковою кількістю 1.
- clearItems - метод для очищення кошика, видаляючи всі товари.
- removeItem - метод для видалення товару з кошика за заданим ідентифікатором (ім'ям). Якщо товар знайдено і успішно видалено, сповіщаються слухачі про зміну даних.

Цей клас також використовує механізм ChangeNotifier, щоб повідомляти про будь-які зміни в даних кошика, що дозволяє оновлювати інтерфейс користувача, якщо відображення кошика залежить від його стану.

```dart
class Cart with ChangeNotifier {
  // Map<String, ProductLayout>
  List<ProductLayout> _items =
      [];
  List<ProductLayout> get items {
    return [..._items];
  }
  int get itemCount {
    return _items.length;
  }
  double get totalPrice {
    var total = 0.0;
    _items.forEach((element) {
      total += element.price * element.quantity;
    });
    return total;
  }
  void addItem(String name, String image, double price, int quantity,
      double prev, bool isFav) {
    quantity <= 1
        ? _items.add(ProductLayout(name, image, price, prev, isFav, 1))
        : _items // i may delete this condition not yet sure though
            .add(ProductLayout(name, image, price, prev, isFav, quantity++));
  }
  void clearItems() {
    _items = [];
    notifyListeners();
  }
  void removeItem(String id) {
    //the id will be the item name at this point
    // _items.removeWhere((element) => element.name == id);
    var x = _items.firstWhere((element) => element.name == id);
    _items.remove(x);

    notifyListeners();
```

Рисунок 4.5.5 - код реалізації класу Card() в файлі cartprovider.dart

4.6 Реалізація функції оплати в додатку для продажу косметичних засобів

На Рисунок 4.6.1 зображено реалізації класів _CheckoutState() та Checkout(). _CheckoutState() є приватним станом для Checkout. В ньому оголошуються змінні, що представляють дані про картку (номер, термін дії, ім'я власника, CVV-код).

Метод build() викликається для побудови і повернення віджету. В цьому методі отримується інформація про кошик з Provider.

```dart
class Checkout extends StatefulWidget {
  static const routeName = '/checkout';
  @override
  State<StatefulWidget> createState() {
    return CheckoutState();
  }
}

class CheckoutState extends State<Checkout> {
  String cardNumber = '';
  String expiryDate = '';
  String cardHolderName = '';
  String cvvCode = '';
  String cvvValidationMessage = 'Please input a valid CVV';
  bool isCvvFocused = false;
  bool useGlassMorphism = false;
  bool useBackgroundImage = false;
  OutlineInputBorder? border;
  final GlobalKey<FormState> formKey = GlobalKey<FormState>();

  @override
  void initState() {...}
  @override
  Widget build(BuildContext context) {...}

  void onCreditCardModelChange(CreditCardModel? creditCardModel) {...}
}
```

Рисунок 4.6.1 - код реалізації класів Checkout() та _ CheckoutState() в файлі checkout.dart

На рисунку 4.6.2 наданий фрагмент коду показує структуру віджета Column, що містить кілька вертикально розташованих дочірніх віджетів.

Рисунок 4.6.2 - код реалізації віджету Column в методі build() в класі _ CheckoutState() в файлі checkout.dart

Віджет CreditCardWidget()(Рисунок 4.6.3) відображає віджет кредитної картки, який відображає номер картки, термін дії, ім'я власника та CVV-код. Властивість showBackView використовується для визначення того, чи показувати зворотний бік картки. Залишаються інші властивості, які визначають оформлення та поведінку віджету.

```
CreditCardWidget(
  cardNumber: cardNumber,
  expiryDate: expiryDate,
  cardHolderName: cardHolderName,
  cvvCode: cvvCode,
  showBackView: isCvvFocused,
  obscureCardNumber: true,
  obscureCardCvv: true,
  isHolderNameVisible: true,
  cardBgColor: Colors.black,
  backgroundImage:
      useBackgroundImage ? 'assets/card_bg.png' : null,
  isSwipeGestureEnabled: true,
  onCreditCardWidgetChange: (CreditCardBrand creditCardBrand) {},
  customCardTypeIcons: <CustomCardTypeIcon>[
    CustomCardTypeIcon(...),  // CustomCardTypeIcon
  ],  // <CustomCardTypeIcon>[]
),  // CreditCardWidget
```

Рисунок 4.6.3 - код реалізації віджету CreditCardWidget() в методі build() в класі _CheckoutState() в файлі checkout.dart

Віджет CreditCardForm(Рисунок 4.6.4) — це форма для збору інформації про кредитну картку та її відображення в інтерфейсі користувача. Віджет CreditCardForm створюється різними властивостями та значеннями для налаштування його поведінки та зовнішнього вигляду. Властивістості віджету CreditCardForm:

1. formKey — це унікальний ключ, який використовується для ідентифікації та перевірки форми. Зазвичай це екземпляр GlobalKey.
2. obscureCvv і obscureNumber є логічними значеннями, які визначають, чи CVV (значення перевірки картки) і номер картки закриті або приховані з міркувань безпеки.
3. CardNumber, cvvCode, cardHolderName і expiryDate — це змінні, які містять поточні значення номера картки, CVV, імені власника картки та терміну дії відповідно.

4. isHolderNameVisible, isCardNumberVisible і isExpiryDateVisible — це логічні значення, які контролюють видимість полів імені власника картки, номера картки та терміну дії відповідно.
5. themeColor і textColor визначають кольори, що використовуються для теми форми та тексту відповідно. Цей код використовує Colors.blue для кольору теми та Colors.white для кольору тексту.
6. CardNumberDecoration, expireDateDecoration, cvvCodeDecoration і cardHolderDecoration — це об’єкти InputDecoration, які визначають вигляд кожного поля форми. Кожен InputDecoration містить «labelText» для мітки поля, «hintText» для тексту заповнювача, «focusedBorder» і «enabledBorder» для стилю межі, коли поле виділено або активне, і «hintStyle» і «hintStyle» для поля. Містить такі властивості, як "labelStyle". Стилі інформаційного тексту та підписів.
7. onCreditCardModelChange — це функція зворотного виклику, яка запускається щоразу, коли змінюється модель кредитної картки.

Загалом, віджет CreditCardForm забезпечує зручний спосіб збору інформації про кредитну картку та відображення її в зручній для використання формі. Він обробляє перевірку введення та форматування, а також дозволяє налаштовувати зовнішній вигляд вашої форми за допомогою різних властивостей і параметрів стилю.

```
CreditCardForm(
  formKey: formKey,
  obscureCvv: true,
  obscureNumber: true,
  cardNumber: cardNumber,
  cvvCode: cvvCode,
  isHolderNameVisible: true,
  isCardNumberVisible: true,
  isExpiryDateVisible: true,
  cardHolderName: cardHolderName,
  expiryDate: expiryDate,
  themeColor: Colors.blue,
  textColor: Colors.white,
  cardNumberDecoration: InputDecoration(
    labelText: 'Номер картки',
    hintText: 'XXXX XXXX XXXX XXXX',
    hintStyle: const TextStyle(color: Colors.white),
    labelStyle: const TextStyle(color: Colors.white),
    focusedBorder: border,
    enabledBorder: border,
  ),  // InputDecoration
  expiryDateDecoration: InputDecoration(...),  // InputDecoration
  cvvCodeDecoration: InputDecoration(...),  // InputDecoration
  cardHolderDecoration: InputDecoration(...),  // InputDecoration
  onCreditCardModelChange: onCreditCardModelChange,
),  // CreditCardForm
```

Рисунок 4.6.4 - код реалізації віджету CreditCardForm() в методі build() в класі _CheckoutState() в файлі checkout.dart

На рис. 4.6.5 зображено фрагмент коду, який демонструє використання віджета ElevatedButton. Ця кнопка використовується для запуску дії при натисканні користувачем. Віджет ElevatedButton створюється різними властивостями, які налаштовують його вигляд і поведінку. Властивість style встановлюється за допомогою методу ElevatedButton.styleFrom і дозволяє налаштувати візуальний стиль кнопки. Він складається з властивості форми, яка використовує його RoundedRectangleBorder із borderRadius 8,0 для округлення кутів кнопки. Основна властивість встановлює колір фону кнопки, використовуючи основний колір, визначений у поточній темі контексту.

Властивість дочірнього елемента встановлюється на віджет Container, який обгортає віджет TextButton. Віджет-контейнер використовується для додавання рамки навколо TextButton. TextButton відображає підпис кнопки, встановлений як текст "Оплатити". Властивості стилю текстового віджета використовуються для налаштування зовнішнього вигляду тексту, наприклад колір, сімейство шрифтів, розмір шрифту та упаковка. Властивість onPressed налаштована на функцію зворотного виклику, яка запускається під час натискання кнопки. Зворотний виклик має умовний оператор, який використовує метод validate(), щоб перевірити, чи форма, пов’язана з formKey, зараз у дійсному стані. Якщо форма дійсна, виконується код всередині блоку if. В іншому випадку, якщо форма недійсна, код всередині блоку else виконується. Якщо кнопку натиснуто, а форма дійсна, для відображення AlertDialog викликається функція showDialog. У цьому діалоговому вікні з’явиться повідомлення про те, що ваше замовлення прийнято. Він також використовує Navigator.pushReplacementNamed для переходу на головний екран після натискання кнопки «ОК».

Таким чином, віджет ElevatedButton використовується для створення стилізованих кнопок із налаштованим виглядом.

Рисунок 4.6.5 - код реалізації віджету ElevatedButton() в методі build() в класі _CheckoutState() в файлі checkout.dart

Метод onCreditCardModelChange()(Рисунок 4.6.6) оновлює стан змінних, що представляють дані про кредитну картку, коли дані вводяться або змінюються у віджеті форми карти. Це викликається при зміні моделі кредитної картки.

```
void onCreditCardModelChange(CreditCardModel? creditCardModel) {
  setState(() {
    cardNumber = creditCardModel!.cardNumber;
    expiryDate = creditCardModel.expiryDate;
    cardHolderName = creditCardModel.cardHolderName;
    cvvCode = creditCardModel.cvvCode;
    isCvvFocused = creditCardModel.isCvvFocused;
  });
}
```

Рисунок 4.6.6 - код реалізації метода onCreditCardModelChange() в класі _ CheckoutState() в файлі checkout.dart

4.7 Ручне тестування

Ручне тестування проекту Flutter включає процес перевірки функціональності, якості та коректності програмного забезпечення шляхом ручного виконання різноманітних операцій із додатком.

Основною метою ручного тестування проекту Flutter є виявлення помилок, винятків і проблем, які можуть виникнути під час роботи програми.

Під час ручного тестування було перевірено наступні аспекти роботи додатку:

1. Функціональність. Було перевірено, що всі функції програми працюють належним чином. Перевірка включає роботи різних кнопок, полів введення, навігації між екранами та інших елементів інтерфейсу.

2. Візуальний дизайн. Перевірено коректність відображення інтерфейсу програми. Елементи інтерфейсу правильно вирівняні.

3. Сумісність. Flutter дозволяє писати єдиний код для двох операційних систем(Android та iOS). Було перевірено коректну роботу додатку для продажи косметичних засобів на двох версіях операційної системи.

Висновки

Дана робота була спрямована на проектування та реалізацію мобільної платформи для B2C- продажу косметичної продукції мовою Dart.

1. Аргументовано актуальність розробленого мобільного додатку на основі аналізу наукової та практичної літератури з розробки програмних продуктів.

2. Шляхом аналізу близьких за функціоналом продуктів (Ebay, Ticket Party, Feeh la) досліджено та встановлено переваги та недоліки трьох аналогів, на основі чого сформовано та описано функціональні та нефункціональні вимоги до розробленого продукту.

3. Обґрунтовано вибір засобів розробки мобільної платформи для B2C-продажу косметичної продукції мовою Dart: середовище розробки Android Studio, мову програмування Dart на платформі Flutter; інструмент для керування версіями Git.

4. Спроектовано архітектуру системи та розроблено мобільну платформу для продажу косметичної продукції; для розробки програмного забезпечення було використано технологію MVC, за допомогою якої було реалізовано інтерфейс користувача; для розробки були використано шаблони проектування та сучасні принципи ООП.

5. Здійснено тестування розробленої платформи для B2C-продажу косметичної продукції мовою Dart, отримані задовільні результати.

Перелік посилань

1. Design Patterns in Flutter- Part 1(MVC) [Електронний ресурс]: [Веб-сайт]. Режим доступу до ресурсу: https://medium.flutterdevs.com/design-patterns-in-flutter-part-1-c32a3ddb00e2
2. Advanced Features Of Xcode Simulator: Improve Your IOS App Development Experience [Електронний ресурс]: [Веб-сайт]. Режим доступу до ресурсу: https://geniusee.com/single-blog/xcode-simulators-advanced-features#what_is_the_xcode_ios_simulator
3. Документація про Flutter [Електронний ресурс]: [Веб-сайт]. Режим доступу до ресурсу: https://docs.flutter.dev
4. Документація про Dart [Електронний ресурс]: [Веб-сайт]. Режим доступу до ресурсу: https://dart.dev/guides
5. Матеріальний дизайн - система рекомендацій, компонентів і інструментів, які підтримують найкращі методи розробки інтерфейсу користувача [Електронний ресурс]: [Веб-сайт]. Режим доступу до ресурсу: https://m3.material.io/develop/flutter
6. Діаграми [Електронний ресурс]: [Веб-сайт]. Режим доступу до ресурсу: http://um.co.ua/1/1-5/1-53871.html
7. Android Studio [Електронний ресурс]: [Веб-сайт]. Режим доступу: https://www.javatpoint.com/android-studio
8. What is Git [Електронний ресурс]: [Веб-сайт]. Режим доступу: https://www.atlassian.com/git/tutorials/what-is-git
9. Why Flutter is the most popular cross-platform mobile SDK [Електронний ресурс]: [Веб-сайт]. Режим доступу до ресурсу: https://stackoverflow.blog/2022/02/21/why-flutter-is-the-most-popular-cross-platform-mobile-sdk/

10. Що таке функціональні вимоги: приклади, визначення, повний посібник [Веб-сайт]. Режим доступу до ресурсу: https://visuresolutions.com/uk/blog/functional-requirements/

ДОДАТОК А

Екрани застосунку

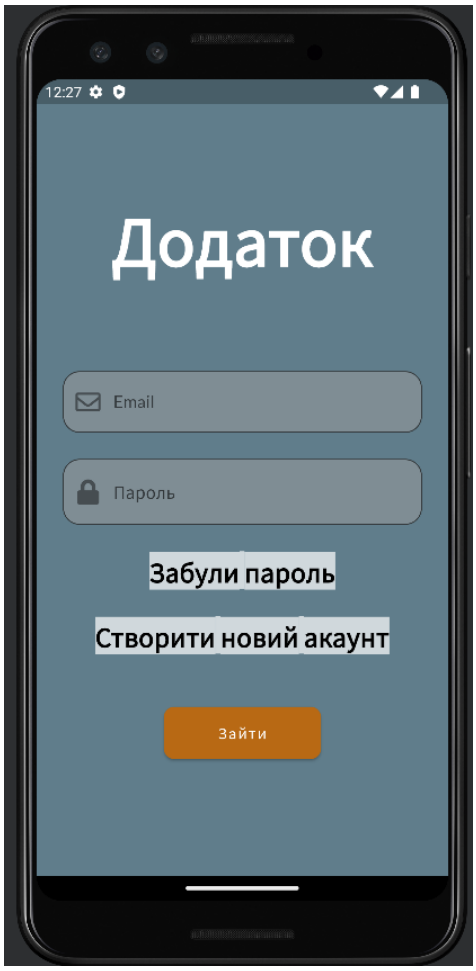


Рисунок 1- Екран авторизації

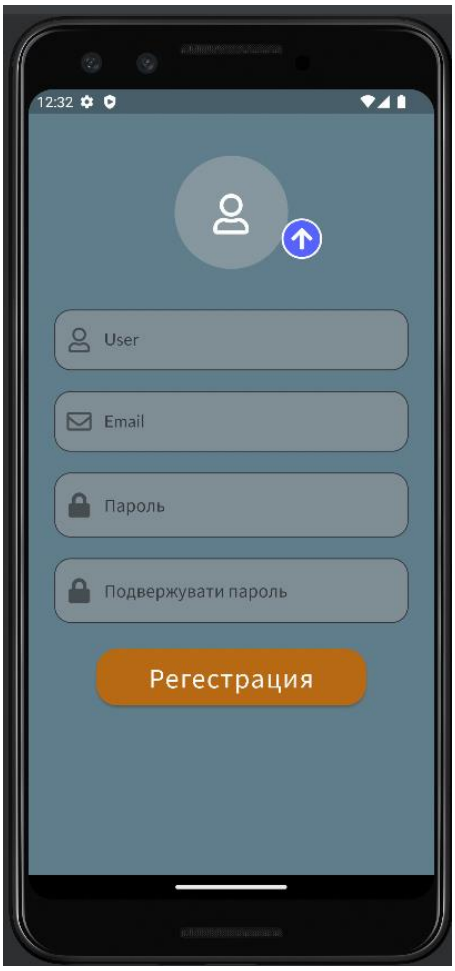


Рисунок 2 - Екран регестрації

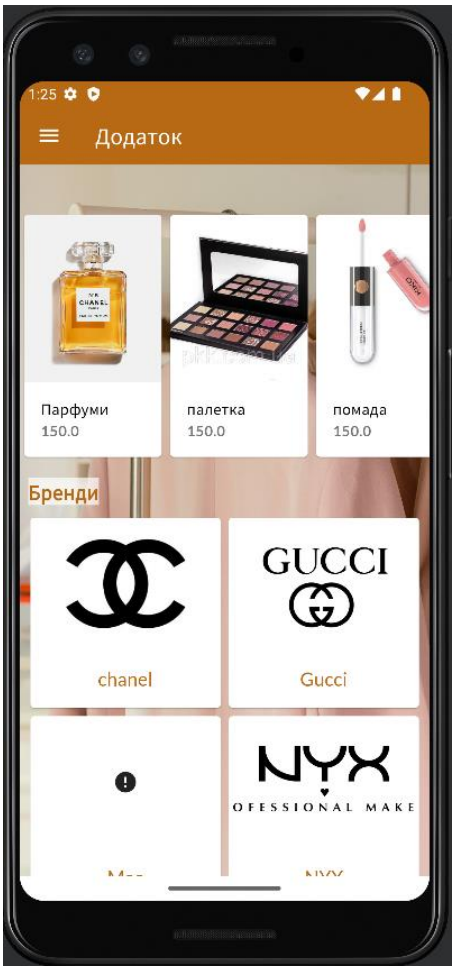


Рисунок 3 - Екран головної сторінки

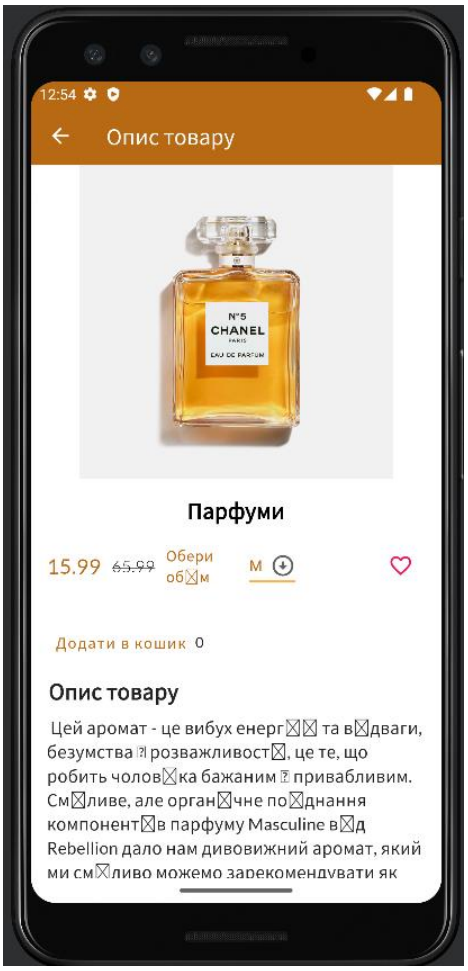


Рисунок 4 - Екран опису косметичного товару

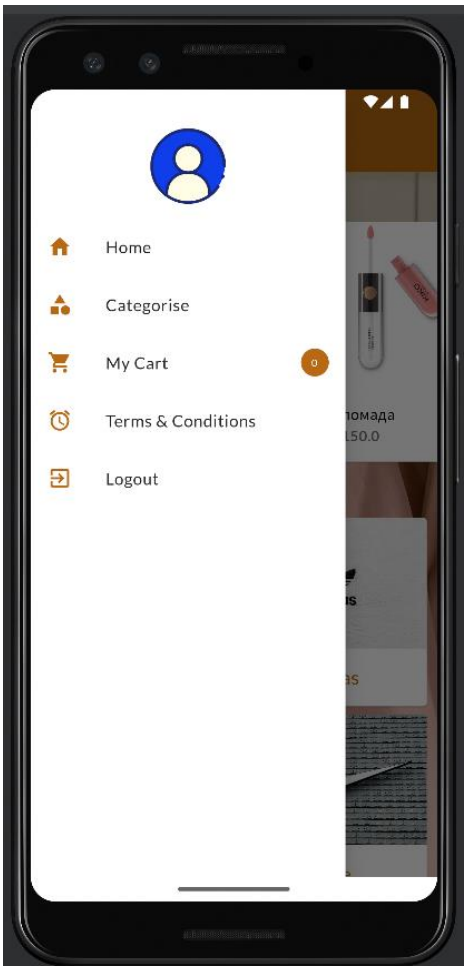


Рисунок 5 - Бокове меню в додатку для продажу косметичних засобів

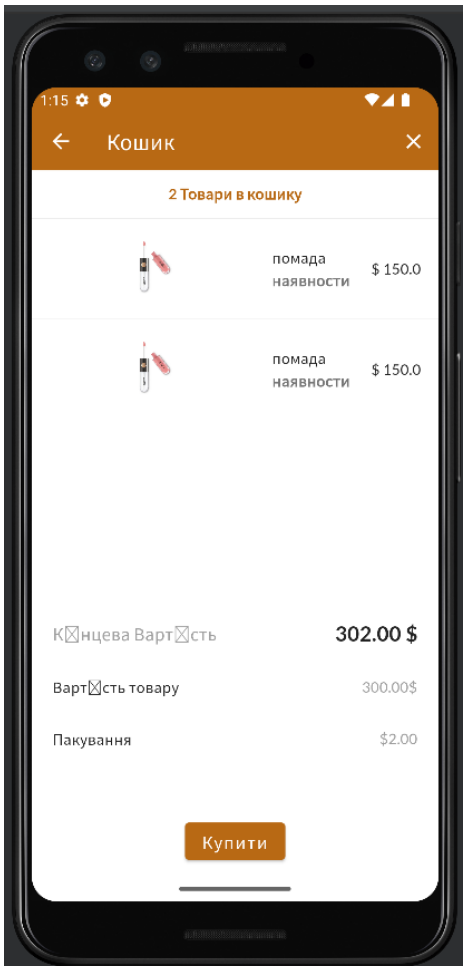


Рисунок 6 - Оформлення замовлення

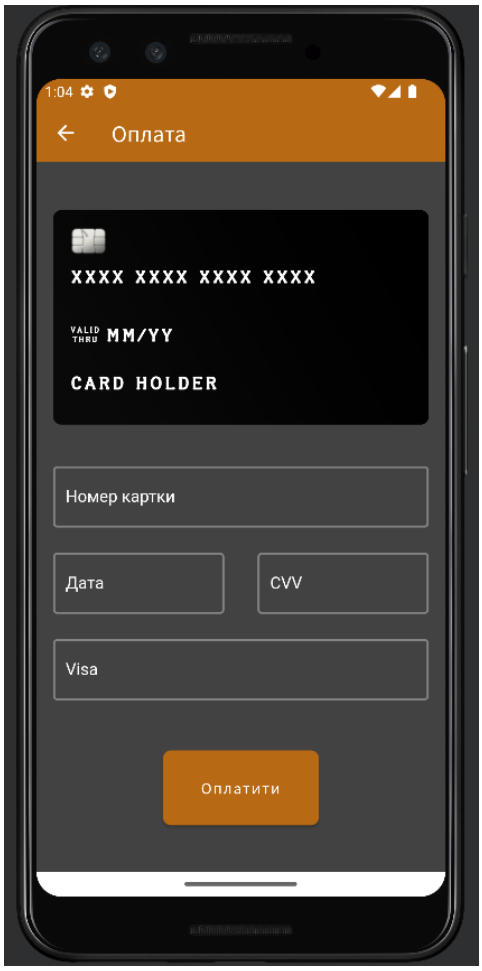


Рисунок 7 - Екран оплати

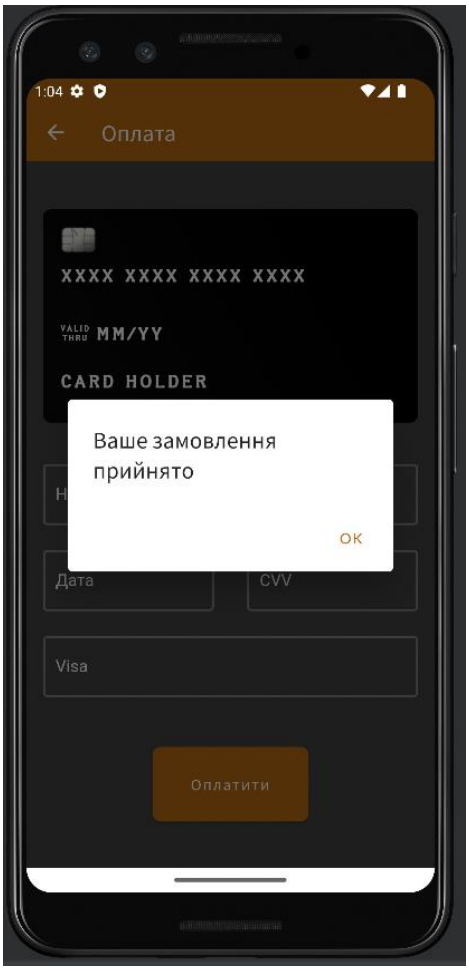


Рисунок 8 - Сповіщення про успішне завершення замовлення

ДОДАТОК В

Демонстраційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка мобільної платформи для B2C-продажу косметичної продукції мовою Dart

Виконала студентка 4 курсу
групи ПД 43

Полтавець Наталія Віталіївна
Керівник роботи

к.п.н, доцент Шевченко Світлана Миколаївна

Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - покращення процесу взаємодії з клієнтами B2C-продажів косметичної продукції за рахунок використання мобільного додатку мовою Dart
- **Об'єкт дослідження -** процес взаємодії з клієнтами B2C-продажів косметичної продукції
- **Предмет дослідження -** програмні засоби для забезпечення взаємодії з клієнтами B2C-продажів косметичної продукції

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Здійснити огляд літературних джерел, існуючих засобів розв'язання завдань предметної галузі.
2. Здійснити порівняльний аналіз аналогів програмного забезпечення (Ebay, Party Tickets,Feeh la); визначити функціональні та нефункціональні вимоги на основі отриманих результатів порівняння аналогів.
3. Обґрунтувати вибір технологій для розробки мобільного додатку з продажу косметичних продуктів.
4. Спроектувати архітектуру системи та інтерфейс додатку для продажу косметичних засобів.
5. Розробити мобільну платформу для B2C-продажу косметичної продукції.
6. Провести тестування розробленої мобільної платформи

АНАЛІЗ АНАЛОГІВ

	Ebay	Feeh La shopping	Party Ticket	Додаток для продажу косметичних засобів
Результат пошуку з використанням фільтру	+	+	-	+
Оплата онлайн	+	+	+	+
Панель навігації	+	+	+	+
Покупка без авторизації	-	-	-	-
Авторизація	+	+	+	+
Підтримка двох платформ операційної системи	+	+	+	+
Можливість додавання товару в кошик	+	+	+	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

- Реєстрація та авторизація
- Функція додавання косметичних продуктів до кошика
- Функція оформлення замовлення та оплата замовлення в додатку для продажу косметичних засобів
- Функція збереження сподобавшись косметичних продуктів

Нефункціональні вимоги:

- Надійність(всі функції мають працювати без затримок та перебоїв)
- Швидкодія(додаток має відповідати на запити користувачів негайно та працювати швидко)
- Безпека (забезпечувати безпеку відносно конфіденційності даних користувачів)
- Доступність та сумісність (мобільний додаток має бути доступним на дві операційні системи Android та IOS)
- Розширюваність(можливість майбутнього розширення функціоналу)

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Android Studio - середа розробки

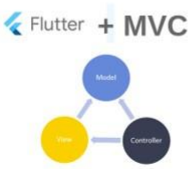
Flutter- фреймворк

Dart- мова програмування

Material Design - принцип дизайну інтерфейсу

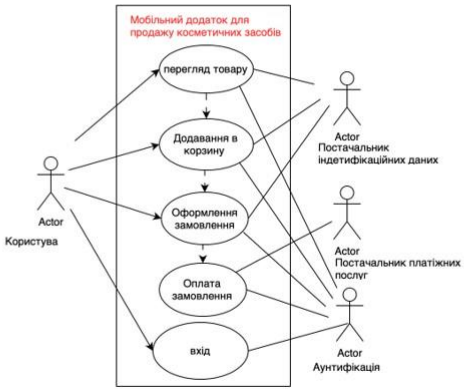
Git- система керування версі ями

Xcode simulator- імулятор середовищ

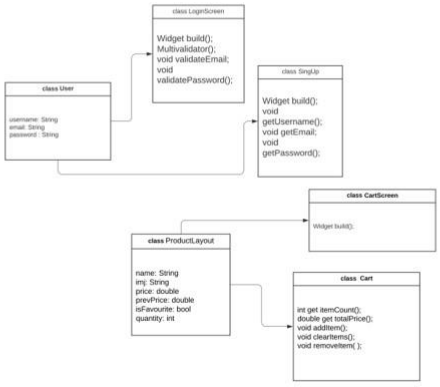


MVC - архітектурний шаблон

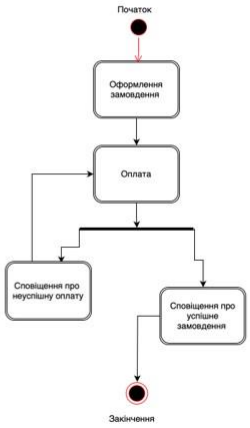
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



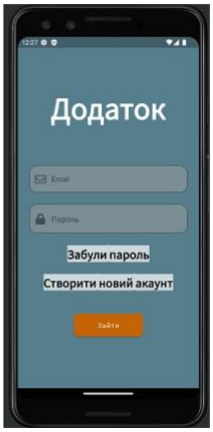
ДІАГРАМА КЛАСІВ



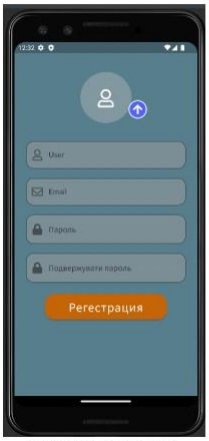
ДІАГРАМА ДІЯЛЬНОСТІ(ОПЛАТА ЗАМОВЛЕННЯ)



ЕКРАННІ ФОРМИ



Екран авторизації



Екран реєстрації

ЕКРАННІ ФОРМИ

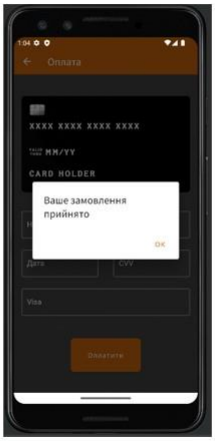
Головний екран

Екран опису товару

ЕКРАННІ ФОРМИ

Екран кошику

Екран оплати



Сповіщення про завершення замовлення

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Полтавець Н.В. Застосування фреймворка флатер для розробки мобільних застосунків. Науково-технічна конференція «Застосування програмного забезпечення в інфокомунікаційних технологіях», 22.04.22. Збірник тез. – К.: ДУТ, 2022. – С. 109 – 111.
- Полтавець Н.В. Штучний інтелект як інструмент для мобільної розробки. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інфокомунікаційних технологіях», 20.04.23. Збірник тез. – К.: ДУТ, 2023. – С. 109 – 110.

ВИСНОВКИ

1. Аргументовано актуальність розробленого мобільного додатку на основі аналізу наукової та практичної літератури з розробки програмних продуктів.
2. Шляхом аналізу близьких за функціоналом продуктів (Ebay, Ticket Party, Feeh Ia) досліджено та встановлено переваги та недоліки трьох аналогів, на основі чого сформовано та описано функціональні та нефункціональні вимоги до розробленого продукту.
3. Обґрунтовано вибір засобів розробки мобільної платформи для B2C-продажу косметичної продукції мовою Dart: середовище розробки Android Studio, мову програмування Dart на платформі Flutter; інструмент для керування версіями Git.
4. Спроектовано архітектуру системи та розроблено мобільну платформу для продажу косметичної продукції; для розробки програмного забезпечення було використано технологію MVC, за допомогою якої було реалізовано інтерфейс користувача; для розробки були використано шаблони проектування та сучасні принципи ООП.
5. Здійснено тестування розробленої платформи для B2C-продажу косметичної продукції мовою Dart, отримані задовільні результати.