Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження класичних та no-code методів
розробки мобільних застосунків
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-5

Шуляк М.А.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Лановий О.Ф.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Шуляку Микиті Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження класичних та no-code методів розробки мобільних застосунків»

Затверджена наказом по університету від 29.03.2024р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи огляд наукової та патентної літератури, аналіз предметної галузі, постановка задачі, розробка програмної системи, планування та опис проведених експериментальних досліджень

4. Перелік питань, що потрібно опрацювати в роботі дослідити класичні та no-code методи для розробки мобільних додатків, визначити ефективність написання мобільних додатків, використовуючи різні підходи та технології, створити методології вибору технології для написання мобільного проекту в залежності від функціоналу

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	23.01 – 14.02.24	*виконано*
2	Аналіз та вибір методів для дослідження	15.02 – 24.02.24	*виконано*
3	Аналіз та моделювання предметної області	17.02 – 28.02.24	*виконано*
4	Планування експериментів	25.02 – 28.02.24	*виконано*
5	Програмна реалізація кожного з обраних для дослідження додатка	25.02 – 01.04.24	*виконано*
6	Експериментальні дослідження	02.04 – 20.04.24	*виконано*
7	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.04 – 23.04.24	*виконано*
8	Написання та оформлення статті та тез доповіді	17.04 – 23.04.24	*виконано*
9	Підготовка пояснювальної записки	01.05 – 01.06.24	*виконано*
10	Підготовка презентації та доповіді	26.05 – 01.06.24	*виконано*
11	Нормоконтроль	01.06 – 08.06.24	*виконано*
12	Рецензування	08.06 – 13.06.24	*виконано*
13	Занесення диплома в електронний архів	15.06.2024	*виконано*
14	Попередній захист	15.06.2024	*виконано*
15	Допуск до захисту у зав. кафедри	16.06.2024	*виконано*

Дата видачі завдання 29 березня 2024р.

Студент (ка) _____ Шуляк М.А.
(підпис)

Керівник роботи _____ доц. Лановий О.Ф.

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 59 с., 17 рис., 2 табл., 6 додатків, 9 джерел.

ЕФЕКТИВНІСТЬ РОЗРОБКИ, МОБІЛЬНА РОЗРОБКА, NO-CODE, LOW-CODE, FLUTTER, FLUTTERFLOW

Робота присвячена дослідженню продуктивності розробки з використанням no-code технологій у порівнянні з традиційними методами програмування. Об’єктом дослідження є процес розробки мобільних застосунків. Метою є аналіз продуктивності no-code платформ та визначення її впливу на розробку. Методами дослідження є вимірювання, аналіз продуктивності розробки, прогнозування часових витрат на розробку за допомогою no-code та кодових платформ.

У результаті було виведено слабкі та сильні сторони no-code розробки. Робота доповнює існуючі дослідження у сфері продуктивності програмної інженерії та розвитку мобільних технологій. Результати роботи рекомендується використовувати для оптимізації процесу вибору методів розробки мобільних додатків, підвищення ефективності та зниження часових витрат на реалізацію проектів. Робота має значимість для розробки програмного забезпечення, оскільки надає сучасний огляд інструментів, які можуть підвищити продуктивність.

Пропозиції щодо розвитку об'єкта дослідження включають дослідження мобільних додатків з іншим функціоналом, що робить доцільним продовження досліджень у цій сфері.

DEVELOPMENT EFFICIENCY, MOBILE DEVELOPMENT, NO-CODE, LOW-CODE, FLUTTER, FLUTTERFLOW

The work is devoted to researching the productivity of development using no-code technologies in comparison with traditional programming methods. The object of research is the process of developing mobile applications. The goal is to analyze the

performance of no-code platforms and determine its impact on development. Research methods include measurement, analysis of development productivity, forecasting of development time costs using no-code and code platforms.

As a result, the weaknesses and strengths of no-code development were deduced. The work complements existing research in the field of software engineering productivity and mobile technology development. The results of the work are recommended to be used to optimize the process of choosing mobile application development methods, increase efficiency, and reduce time spent on project implementation. The work has implications for software development as it provides a state-of-the-art overview of tools that can improve productivity.

Proposals for the development of the research object include the study of mobile applications with other functionality, which makes it reasonable to continue research in this area.

Я, Шуляк Микита Андрійович, студент гр.ІПЗм-22-5, здобувач вищої освіти на другому (магістерському) рівні, кафедра програмної інженерії, заявляю: представлена моя робота, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

OTP – One-time password

iOS – iPhone operating system

IDE – Integrated development environment

ВСТУП

Розробка програмного забезпечення є складним та багатогранним процесом, що потребує ретельного вибору технологій та методів. Результат розробки залежить від багатьох факторів.

Одним з важливих етапів процесу дизайну є аналіз тенденцій користувачів, для їх виявлення також необхідно зрозуміти, що робить сайт зручним і зрозумілим для будь-якого користувача [1].

Важливим аспектом є і вибір технологій для розробки мобільних застосунків, що впливає на вартість, продуктивність та швидкість розробки. У сучасному світі, де мобільні додатки стали невід'ємною частиною нашого життя, ефективність їх створення набуває особливої значущості.

Оцінка програмного забезпечення на етапі проектування має високе практичне застосування, оскільки дозволяє провести оцінку програмного забезпечення до його початку, що в свою чергу дозволяє врахувати більшість можливих ризиків проекту та етапів розробки. У свою чергу, це дозволяє порівняти, які витрати необхідні і яка майбутня вартість проекту [2].

З огляду на постійне зростання ринку мобільних застосунків та збільшення конкуренції, IT-компанії шукають шляхи зниження витрат та підвищення продуктивності.

Важливо оптимізувати процесс розробки програмного забезпечення для зниження витрат та підвищення продуктивності. Використання кросплатформних технологій дозволяє швидше вивести на додаток на ринок, оскільки код не потрібно писати з нуля окремо для різних платформ [3]. Водночас, no-code платформи обіцяють ще більше скорочення витрат часу за рахунок спрощення процесу розробки.

Вибір між класичним методом та методами розробки з низьким вмістом коду є критичним для успішної розробки мобільних застосунків. Кожен з цих підходів має свої переваги та недоліки, і їхнє дослідження допоможе визначити оптимальні умови для їх використання. Таким чином, актуальність теми полягає у

необхідності створення методології вибору технології для написання мобільного проекту в залежності від функціоналу, який необхідно реалізувати.

Метою даного дослідження є визначення часових витрат на розробку мобільних додатків, використовуючи різні підходи, а також розробка математичних формул для оцінки вартості написання проекту з точки зору часу при використанні кожного з досліджених підходів.

Завдання дослідження включають:

- огляд та аналіз існуючих методів розробки мобільних застосунків;
- розробка тестових проектів для кожного з методів (класичний, no-code);
- порівняння часових витрат на розробку кожного проекту;
- аналіз отриманих результатів та виведення математичних формул для оцінки вартості написання проекту.

Об’єктом дослідження є процес розробки мобільних застосунків. Предмет дослідження є методи розробки мобільних застосунків (класичний та no-code), часові витрати на їх розробку.

Для досягнення поставлених завдань у дослідженні використані такі методи дослідження:

- порівняльний аналіз для оцінки ефективності різних підходів до розробки;
- емпіричні методи для вимірювання часових витрат на розробку;
- математичне моделювання для виведення формул оцінки вартості розробки.

Це дозволить удосконалити існуючим знанням о технологіях з низьким вмістом коду.

1 ОГЛЯД НАУКОВОЇ ТА ПАТЕНТНОЇ ЛІТЕРАТУРИ

1.1 Аналіз предметної галузі

1.1.1 Класичні методи розробки

Основні платформи для мобільних додатків — це Android та iOS, які забезпечують величезний ринок для розробників. Класичні мобільні застосунки можуть виконувати різноманітні завдання, починаючи від ігор та розважальних застосунків до корисних інструментів, бізнес-застосунків, соціальних мереж та інших корисних сервісів.

Класичні методи розробки мобільних застосунків охоплюють широкий спектр підходів і інструментів, які використовуються для створення програмного забезпечення для мобільних пристроїв. Загальна характеристика таких методів розробки залежить від конкретних вимог проекту, доступних ресурсів, ступіню підтримці різних платформ та інших факторів. Розробники вибирають той підхід, який найкраще відповідає їхнім потребам і умовам проекту. Існує декілька основних підходів до розробки класичного мобільного застосунку, ці методи обираються у залежності від потреб проекту.

Нативні методи розробки передбачають створення додатків безпосередньо для конкретної операційної системи з використанням її власних інструментів та мов програмування. Для Android це зазвичай Java або Kotlin, а для iOS — Objective-C або Swift. Нативні додатки мають високий рівень продуктивності та інтеграції з апаратним забезпеченням пристрою, що забезпечує високу швидкість і зручність для користувачів. Хоча в сучасних додатках все більше переваги віддається новим мовам програмування, а саме Kotlin та Swift, знання Java та Objective-C все ще необхідно для підтримки старих систем та додатків. Також слід зазначити що при розробці використовуються такі інструменти як Xcode для iOS та Android Studio для Android. Ці потужні IDE включають у себе емулятори девайсів, можливості до розробки застосунків, та багато інших корисних функцій.

Кросплатформні методи розробки дозволяють створювати мобільні додатки, які працюють на декількох платформах одночасно, використовуючи одну кодову базу. Це значно скорочує часові та фінансові витрати на розробку,

оскільки не потрібно писати окремий код для кожної платформи. До основних кросплатформних інструментів належать: Flutter, React Native, Xamaric, Ionic, Kotlin Multiplatform. Можливе застосування комбінованої розробки, наприклад із використанням Flutter, та написанням модулів на нативній мові для розширення функціоналу Flutter.

Проте, кросплатформні додатки можуть вимагати більше ресурсів центрального процесора порівняно з нативними додатками, що може призвести до зниження продуктивності [4]. Хоча ця проблема поступово зникає завдяки зростаючій потужності сучасних мобільних пристроїв.

Ще однією відмінністю є різні вектори розвитку нативних та кросплатформних технологій. Нативні технології зосереджені на створенні додатків для конкретної екосистеми. Наприклад, розробка для Android може включати додатки для розумних годинників на базі Wear OS, тоді як додатки для iOS можна легко імпортувати на десктопні MacOS. Кросплатформні технології, в свою чергу, більше сфокусовані на підтримці нових платформ, таких як Linux, MacOS, Windows та Web.

1.1.2 Методи розробки з низьким вмістом коду

No-code та low-code платформи стали популярними інструментами для розробки мобільних додатків, дозволяючи користувачам без програмістських навичок створювати функціональні додатки. Ці платформи значно скорочують час і витрати на розробку. Загалом експерименти з використанням технологій з низьким вмістом коду показали, no-code та low-code технології мають значно вищу продуктивність, ніж технології на основі коду, коливаючись від приблизно триразового до десятикратного підвищення продуктивності [5].

Хоча no-code технології мають значно вищу продуктивність, це не замінить інші способи створення програмного забезпечення, оскільки no-code код важко використовувати, коли складність рішення зростає [6].

No-code платформи дозволяють створювати додатки без написання коду. Вони використовують інтуїтивні інтерфейси з функцією "перетягування", що

дозволяє швидко та легко створювати додатки, використовуючи готові компоненти. Low-code платформи включають в себе функціонал no-code платформ, але також вимагають мінімальних знань кодування і пропонують більшу гнучкість для кастомізації додатків або створення складнішого функціоналу.

Загальна характеристика для будь-якого no-code конструктору буде наступна:

- надає інтерфейс для для розробки, які часто базуються на перетягуванні та розміщенні елементів у сітці розробки;
- користувачі можуть будувати застосунки, складаючи різні компоненти та логіку, не пишучи код;
- no-code платформи надають готові компоненти, які можна використовувати для створення різних функціональностей (форми, таблиці, графіки тощо);
- вбудовані інструменти автоматизації спрощують створення бізнес-логіки та взаємодії між компонентами без необхідності кодування;
- полегшена інтеграція із зовнішніми сервісами, наприклад вбудована автентифікація, або інтеграція із послугами, такими як Google, або.

Головними перевагами no-code рішень є швидкість розробки та доступність для нетехнічних користувачів. Low-code рішення додають можливість кастомізації. Це все надає можливість швидкого прототипування додатків, що зменшує час та собівартість розробки.

До недоліків методів розробки з низьким вмістом коду відноситься обмежена кастомізація та проблеми з продуктивністю таких додатків. Такі методи можуть не підходити для проектів, що потребують високого рівня кастомізації або спеціалізованих функцій. Інколи такі додатки можуть вимагати більше ресурсів, що впливає на продуктивність. Виявити та виправити такі помилки може бути неможливим через обмежений доступ до коду додатку.

1.2 Виявлення проблем та актуалізація рішень

Згідно з проведеним аналізом наукової літератури, класичні методи розробки мобільних додатків, такі як нативні та кросплатформні підходи, мають свої переваги, включаючи високу продуктивність та гнучкість у реалізації складних функцій. Проте, ці методи вимагають значних часових та фінансових ресурсів.

З іншого боку, no-code та low-code платформи пропонують швидший та дешевший підхід до розробки. Однак, головним недоліком цих платформ є обмежена кастомізація та продуктивність, що може бути критичним для складних додатків, які вимагають специфічних функцій та інтеграцій. Це обмежує можливості розробників і може вимагати додаткових ресурсів для обхідних рішень або переходу на більш гнучкі платформи.

Тож головною проблемою є невизначеність часових витрат на розробку різного функціоналу мобільних проектів за допомогою no-code та low-code методів у порівнянні з класичними методами. Хоча загальні часові витрати на розробку з використанням no-code та low-code платформ можуть бути нижчими, ніж при використанні класичних методів, відсутність глибоких знань обох підходів ускладнює точну оцінку часу, необхідного для реалізації конкретних функцій.

Для вирішення зазначеної проблем необхідно здійснити комплексний аналіз обох підходів – класичного та no-code методу розробки мобільних додатків. Аналіз має включати порівняння часових витрат на реалізацію різних функціоналів, враховуючи продуктивність та обмеження кожного підходу. Зокрема, важливо оцінити, які типи додатків та функцій найкраще підходять для no-code платформ, а які вимагають використання класичних методів.

Дослідження повинно надати більш чітке розуміння ефективності використання методів розробки з низьким вмістом коду у порівнянні з класичними методами розробки, що дозволить приймати обґрунтовані рішення щодо вибору підходу реалізації мобільних проектів.

1.3 Постановка задачі

Для вирішення проблеми визначення часових витрат на розробку мобільних додатків з використанням класичних та no-code методів, у ході даного дослідження будуть проведені емпіричні вимірювання та порівняльний аналіз. Основна задача полягає у створенні двох ідентичних мобільних застосунків за допомогою кожного з досліджуваних методів, щоб визначити, який з підходів є більш ефективним для різних типів функціоналу.

У ході дослідження будуть створені два ідентичні мобільні застосунки. Один із застосунків буде розроблено за допомогою класичного методу. Другий застосунок буде створений за домогою no-code платформи. Обидва застосунки будуть включатимуть однаковий функціонал, який можна поділити наступним чином:

- базовий функціонал: аутентифікація користувача, кешування даних;
- нетиповий функціонал: реалізація складної бізнес логіки, кастомізація додатку.

Перед розробкой додаток буде проаналізований та розподілений на менші функціональні частини. Під час розробки обох застосунків будуть вимірюватися часові витрати на кожну таку функціональну частину.

На основі отриманих даних буде проведено порівняльний аналіз часових витрат між класичним та no-code методами розробки. Будуть виявлені переваги та недоліки кожного підходу для різних типів функціоналу.

На основі зібраних даних та результатів порівняльного аналізу будуть розроблені математичні функції, які дозволять моделювати часові витрати на розробку інших мобільних проектів з використанням кожного з підходів.

Здійсниться оцінка ефективності кожного методу розробки з точки зору часу, витраченого на реалізацію різних функцій мобільних застосунків. Будуть виявлені ті області, де no-code методи можуть замінити класичні методи розробки без значних втрат у якості та продуктивності.

2 ПЛАНУВАННЯ ЕКСПЕРИМЕНТАЛЬНОЇ ЧАСТИНИ ДОСЛІДЖЕННЯ

2.1 Визначення об’єкта, предмета та мети дослідження

Об'єктом дослідження є процес розробки мобільних застосунків. Цей процес охоплює всі етапи розробки, включаючи проектування, програмування, тестування та розгортання мобільних додатків на різних платформах. Аналіз процесу розробки дозволяє вивчити вплив різних методів та інструментів на ефективність створення мобільних застосунків.

Предметом дослідження є методи розробки мобільних застосунків, зокрема класичні методи та no-code методи. Дослідження предмету включає аналіз часових витрат на розробку додатків за допомогою кожного з цих методів, а також вивчення їхніх переваг і недоліків.

Метою даного дослідження є визначення часових витрат на розробку мобільних додатків за допомогою різних підходів, а також розробка математичних формул для оцінки вартості написання проекту з точки зору часу при використанні кожного з досліджених підходів. Дослідження буде зосереджене на порівнянні класичних методів розробки з no-code методами, що дозволить виявити їхні переваги та недоліки, а також розробити рекомендації щодо вибору оптимального методу для різних типів проектів.

Зокрема, у ході дослідження будуть створені два ідентичні мобільні застосунки за допомогою класичного та no-code методів. Шляхом емпіричного вимірювання часових витрат на розробку та порівняльного аналізу будуть розроблені математичні функції, які дозволять моделювати часові витрати на інші мобільні проекти. Мобільні застосунки будуть включати як базовий функціонал (аутентифікація, кешування), так і більш складні функції, що використовують складну бізнес-логіку. Це дозволить визначити, який метод розробки є більш ефективним для реалізації конкретного функціоналу.

2.2 Визначення тестового мобільного застосунку

Зразком для тестування був обраний мобільний застосунок електронної комерції. Застосунок матиме широкий спектр функціоналу, від простих сторінок авторизації до складної фільтрації товарів за багатьма категоріями. Додаток буде підключений до хмарного сховища з базою даних, яка містить інформацію про товари в каталозі, історію покупок, переглядів та профілі користувачів. Дизайн додатку містить унікальні візуальні елементи, що також додає складності у його реалізації. Для певних функцій додатку необхідно реалізувати бізнес логіку. Таким чином, кожна функція може містити деякі з перелічених аспектів, а саме: підключення до хмарного середовища, складні візуальні елементи, бізнес логіку. Кожна функція включає від нуля до трьох аспектів, таким чином можна розподілити фунціонал за складністю та прорахувати як кожен аспект впливає на швидкість реалізації функції.

Для проведення дослідження було обрано мобільний застосунок електронної комерції як зразок для тестування. Цей вибір обумовлений широким спектром функціоналу, який включає як базові, так і складні компоненти, що дозволяє детально проаналізувати ефективність класичних та no-code методів розробки. Додаток можна розділити на наступний функціонал:

- початкова сторінка;
- авторизація та реєстрація за номером мобільного телефону;
- навігація додатком;
- сторінка з головними продуктами;
- сторінка категорій товарів;
- фільтрація продуктів;
- сторінка користувача;
- улюблені продукти;
- кошик з обраними продуктами.

Для реалізації кожної функції додатка необхідно враховувати кілька аспектів, які є невід’ємною частиною функціоналу.

Аспект підключення до хмарного середовища. Це включає налаштування серверної частини, взаємодію з базою даних та забезпечення зберігання даних користувачів;

Аспект складних візуальних елементів. Включає дизайн інтерфейсу користувача, який включає кастомні елементи та анімації.

Аспект бізнес логіки. Це реалізація алгоритмів, які керують процесами в додатку, такими як фільтрація товарів.

Кожна функція додатка може включати від нуля до трьох аспектів, що дозволяє розподілити функціонал за складністю. Це дає можливість більш точно оцінити, як кожен аспект впливає на швидкість реалізації функції.

На рисунку 2.1 продемонстровано екрани для авторизації у додатку. Аспектом цієї функції є підключення до хмарного середовища.

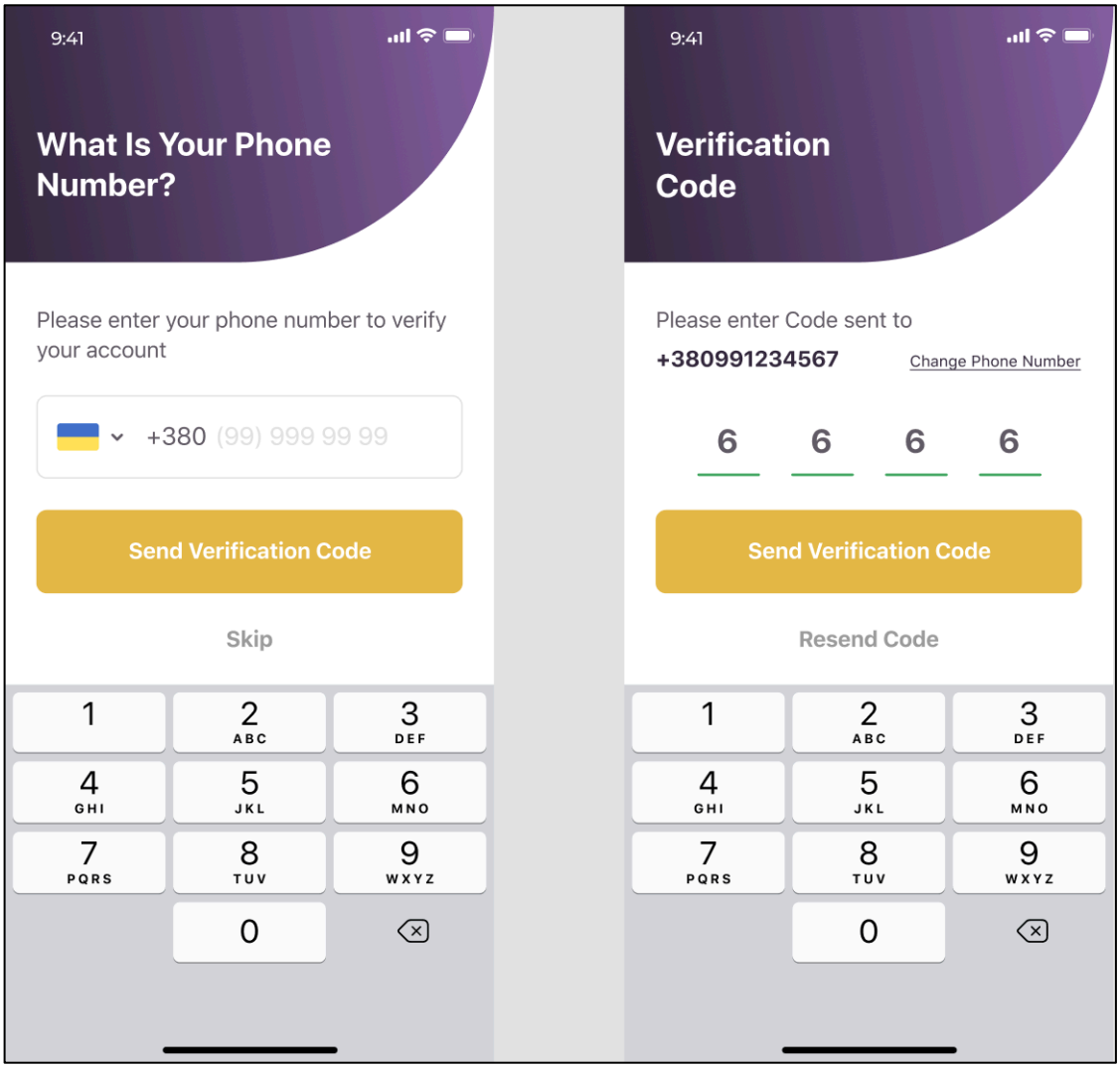

Рисунок 2.1 – Авторизації у додатку

На рисунку 2.2 продемонстровано екрани для фільтрація. Аспектом цієї функції є бізнес логіка та підключення до хмарного середовища.

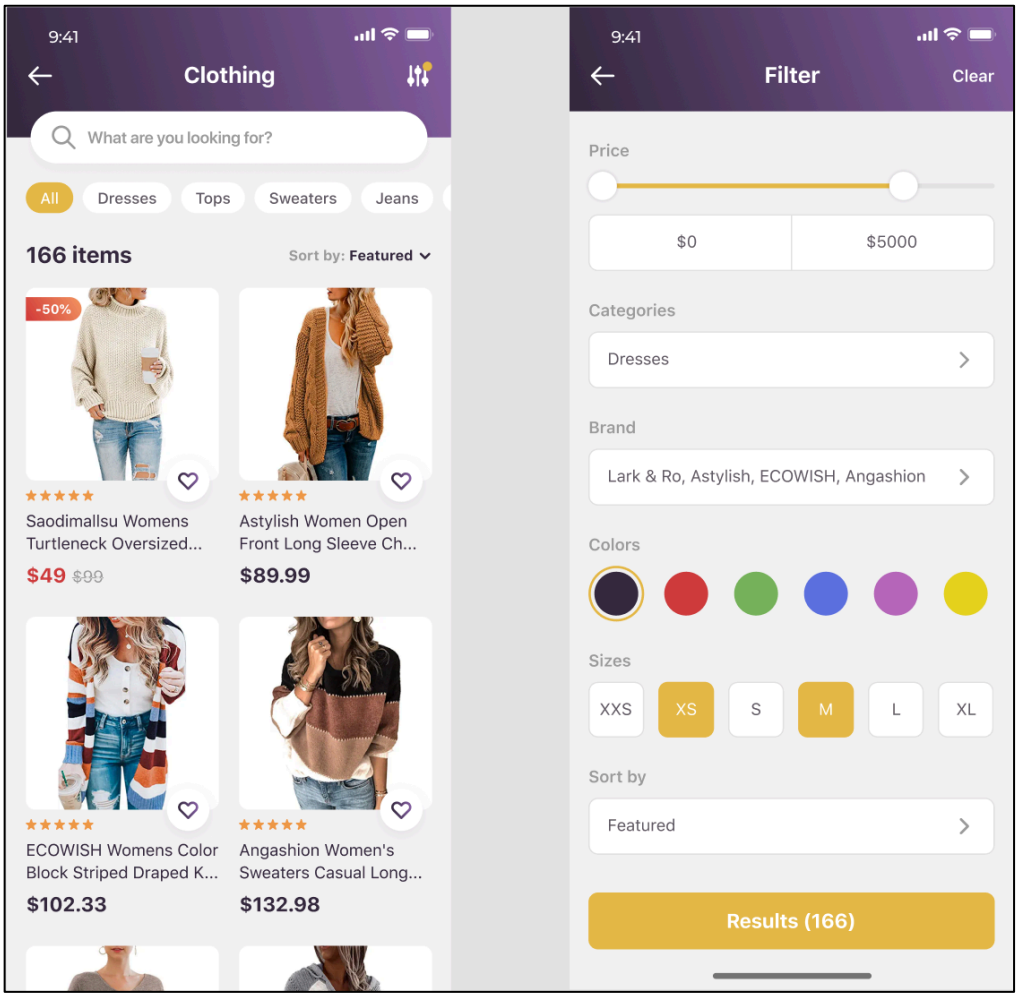

Рисунок 2.2 – Фільтрація у додатку

На рисунку 2.3 продемонстровано компонент для навігації у додатку. Для цього компоненту був розроблений нетиповий дизайн з використанням анімацій. Аспектом цієї функції є складний візуальний елемент.

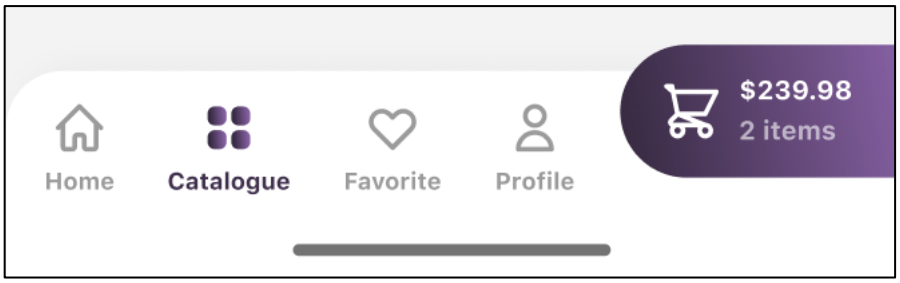

Рисунок 2.3 – Компонент навігації у додатку

Таким чином у додатку присутні різні за складністю та реалізацію функції, що робить можливим більш детально дослідити вплив кожного з методів.

2.3 Вибір технології для проведення дослідження

Вибір технологій для проведення дослідження є однією з фундаментальних її частин. Одночасно потрібно враховувати сучасний попит ринку і його тенденції, щоб дослідження зберігало свою актуальність у найближчий час.

Для визначення найпопулярніших фреймворків для розробки мобільних застосунків було проаналізовано статистику популярних технологій на платформі Stack Overflow. Згідно зі статистикою Stack Overflow, до списку найпопулярніших фреймворків потрапили:

- Flutter;
- React Native;
- SwiftUI;
- Android Jetpack Compose;
- .NET MAUI;
- Cordova;
- Ionic.

На рисунку 2.4 продемонстровано графік трендів мобільних фреймворків.

Проаналізувавши графік трендів, можна зробити висновок, що найпопулярнішим фреймворком на даний момент є Flutter. Його популярність зростала до 2022 року і з того часу залишається стабільною на рівні 3% від усіх опублікованих запитань за останній місяць на Stack Overflow. Завдяки своєму стабільному інтересу та відриву в півтора відсотка від найближчого за попитом фреймворку React Native, можна сказати, що тренд буде збережений найближчим часом.

Оскільки Flutter є найбільш популярним фреймворком за статистикою Stack Overflow, він був обраний як основний інструмент для класичної розробки мобільних застосунків. Це рішення обґрунтоване тим, що Flutter забезпечує

високу продуктивність, широкий функціонал та активну спільноту розробників, що значно спрощує процес створення якісних мобільних додатків.

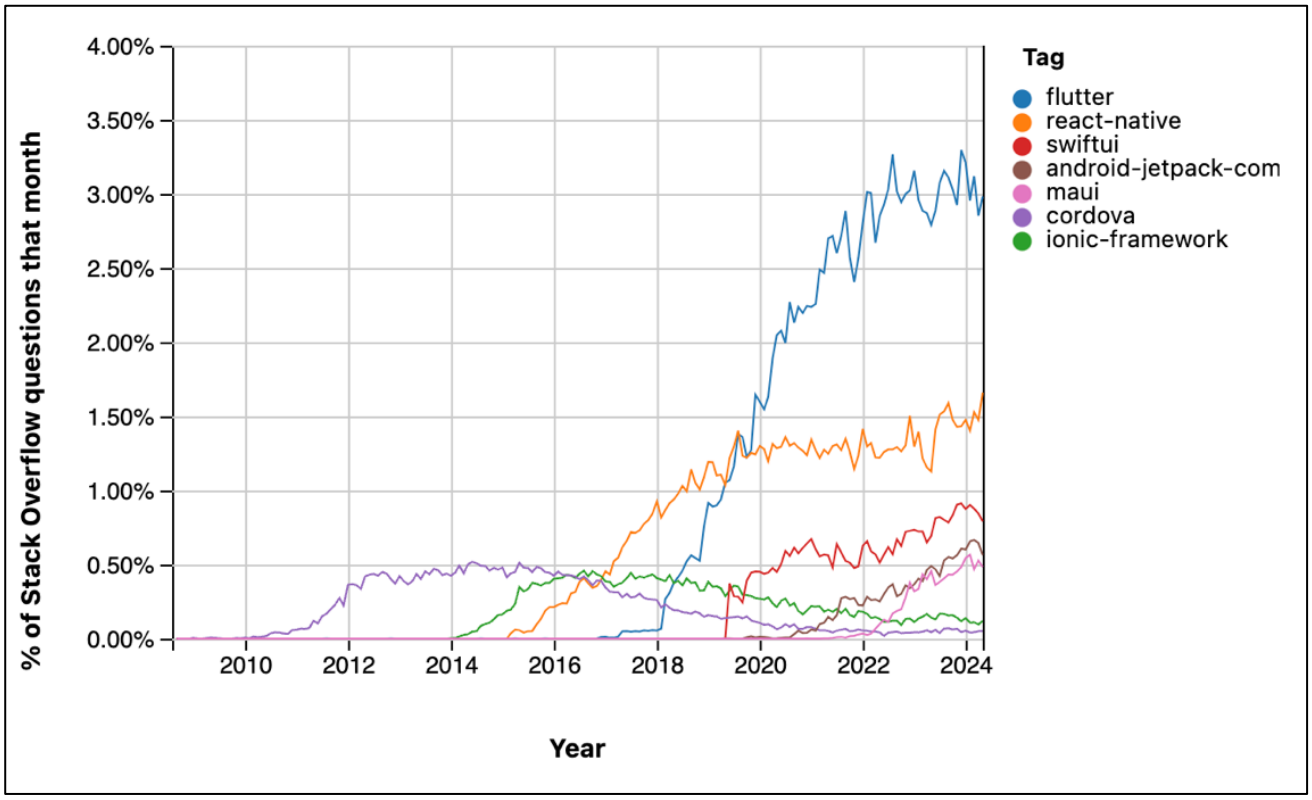

Рисунок 2.4 – Графік трендів мобільних фреймворків

Для no-code методу була обрана платформа FlutterFlow. Ця платформа дозволяє створювати додатки, використовуючи той же фреймворк Flutter, що й традиційний метод розробки. FlutterFlow забезпечує інтуїтивно зрозумілий інтерфейс, що дозволяє швидко і ефективно розробляти мобільні застосунки без необхідності глибоких знань у програмуванні. Використання FlutterFlow забезпечує технічну сумісність та аналогічність з класичним методом розробки, що дозволяє провести коректне порівняння між двома підходами.

Обидва підходи мають свої унікальні особливості, що дозволяє зробити обґрунтовані висновки та вибрати найоптимальніший метод для конкретних завдань і проектів.

3 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

3.1 Опис технології

3.1.1 Класичний метод розробки з Flutter

Фреймворк Flutter використовується для розробки мобільних, веб та десктопних застосунків. Він дозволяє створювати додатки з кросплатформні додатки, використовуючи єдину кодову базу. Flutter використовує мову програмування Dart, яка є об'єктно-орієнтованої та компільованою мовою програмування, що забезпечує високу продуктивність та швидкий час запуску застосунків.

Flutter складається з двох основних частин: Flutter SDK та Flutter Framework. SDK включає бібліотеки, інструменти та API для створення додатків. Framework надає набір компонентів, які використовуються для створення інтерфейсу користувача. Також Flutter має великий каталог пакетів та бібліотек, які можна використовувати для розширення функціональності додатків.

3.1.2 No-code метод розробки з FlutterFlow

FlutterFlow — це візуальне середовище розробки для створення нативних мобільних і веб-додатків. Платформа дозволяє створювати додатки без написання коду, використовуючи інтуїтивний інтерфейс з функцією перетягування. FlutterFlow використовує Dart як основну мову програмування, оскільки платформа базується на Flutter.

Платформа надає візуальний інтерфейс для створення та налаштування додатків. Це включає всі основні компоненти для створення мобільних додатків:

- візуальний редактор для створення інтерфейсу додатка;
- інструмент для налаштування бізнес логіки додатка;
- інструмент для інтеграції з API.

На рисунку 3.1 продемонстровано візуальний редактор для створення інтерфейсу додатка.

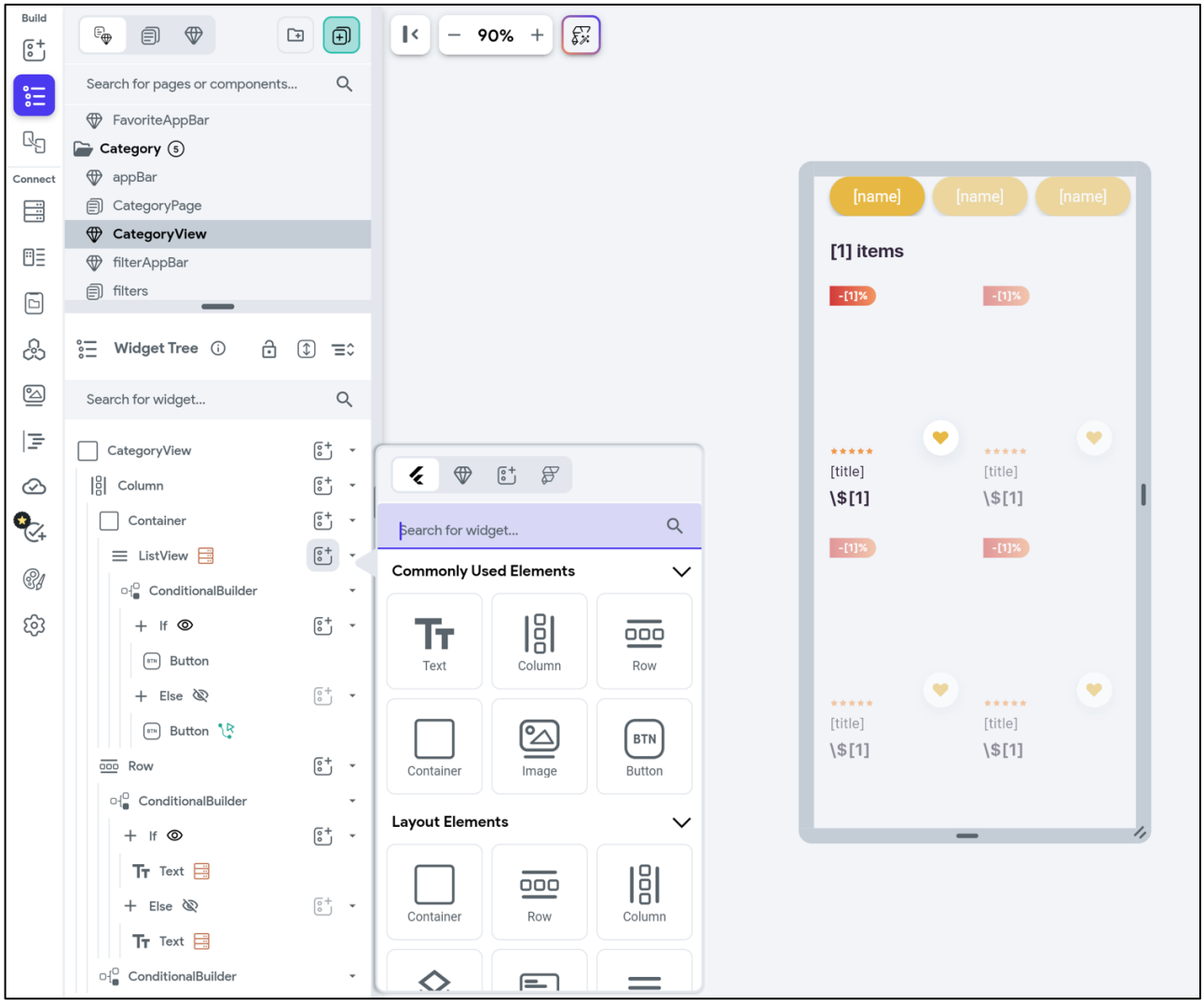

Рисунок 3.1 – Візуальний редактор для створення інтерфейсу додатка

Кожному окремому елементу інтерфейсу можна надати управління над дією. Взаємодія з елементами інтерфейсу запускає бізнес логіку додатку у дію. Взаємодія може бути прямою й опосередковано, у вигаді натискань або зміни життєвого циклу елементу. На рисунку 3.2 продемонстровано інструмент для налаштування бізнес логіки додатка.

Кожен візуальний елемент складається з даних. Тому кожному елементу можна надати можливість завантаження даних для відображення. На рисунку 3.3 продемонстровано інструмент для інтеграції з API.

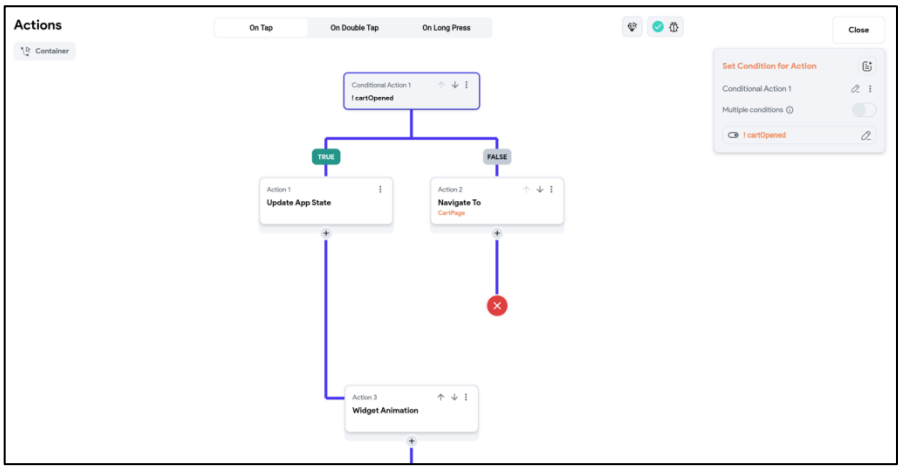

Рисунок 3.2 – Налаштування бізнес логіки додатка

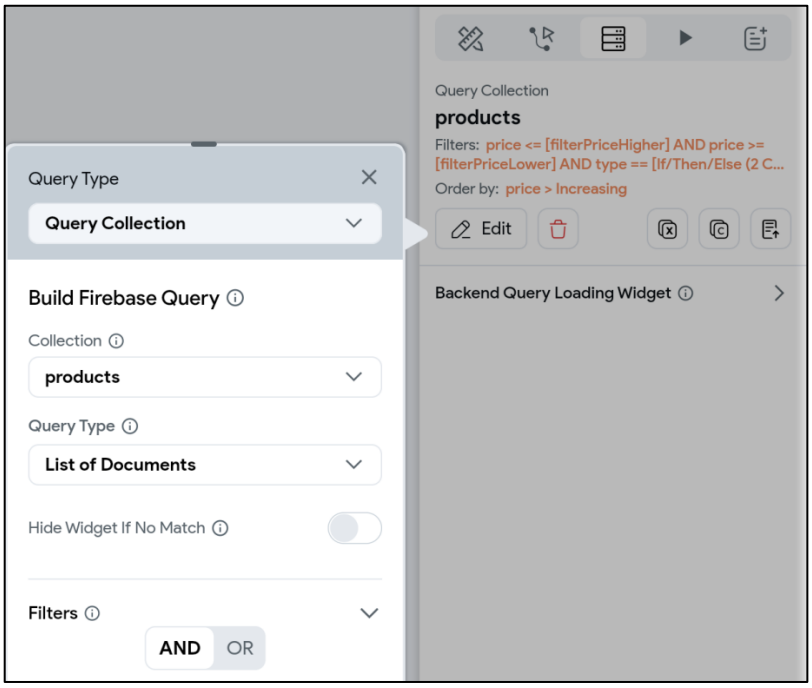

Рисунок 3.3 – Інструмент для інтеграції з API

Таким чином, FlutterFlow включає в себе всі основні інструменти для розробки мобільних застосунків, які знадобляться для проведення дослідження.

3.2 UML-проектування мобільного додатку

3.2.1 Взаємодія користувача з мобільним додаток

Для більш явного уявлення взаємодії з додатком була створена Use Case діаграма. Оскільки розроблені додатки ідентичні за своїм функціоналом, діаграма

застосовна для обох додатків. На рисунку 3.4 продемонстровано Use Case діаграму взаємодії з додатком.

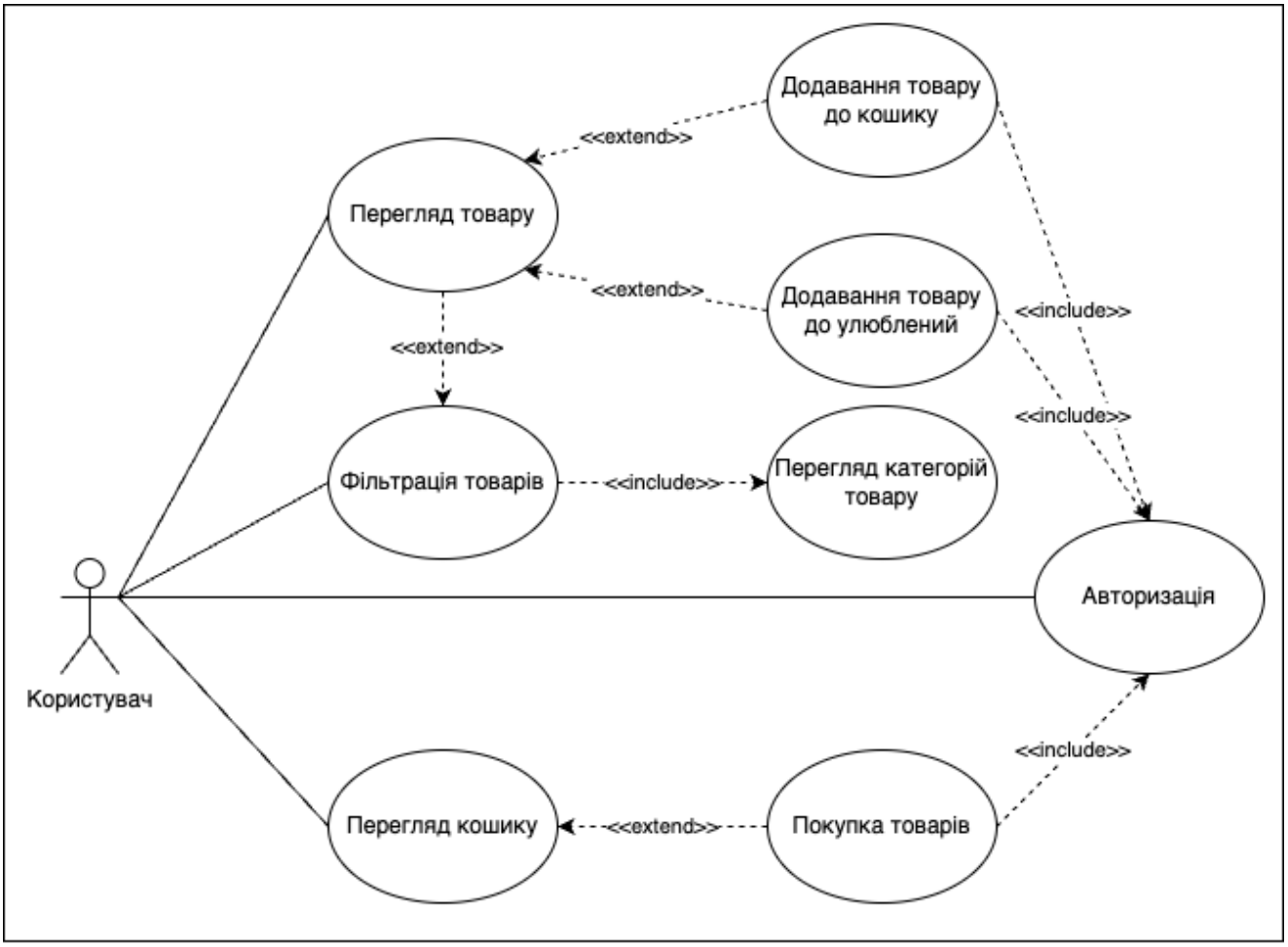

Рисунок 3.4 – Use Case діаграма взаємодії з додатком

На діаграмі відображені основні функції, доступні користувачу:

- перегляд деталей товару;
- фільтрація товарів;
- перегляд кошику з товарами;
- авторизація.

Користувач може переглядати деталі товарів, доступних у додатку. Взаємодіючі з товаром, користувач може додавати його до кошику для подальшої покупки та до списку улюблених для збереження на майбутнє.

Фільтруючи товари, користувач проходить через вибір категорії товарів, щоб значно звузити область пошуку. Після цього користувач може продовжити дію переглядом деталей товару.

Користувач може переглядати вміст кошику, де відображаються всі додані товари. У кошику користувач може здійснити покупку товарів, включених до кошику.

Для додавання товарів до кошику, списку улюблених та для здійснення покупок користувач повинен авторизуватись.

3.2.2 Діаграма станів додатку

Для більш явного уявлення станів додатку на рисунку 3.5 продемонстровано діаграму станів додатку.

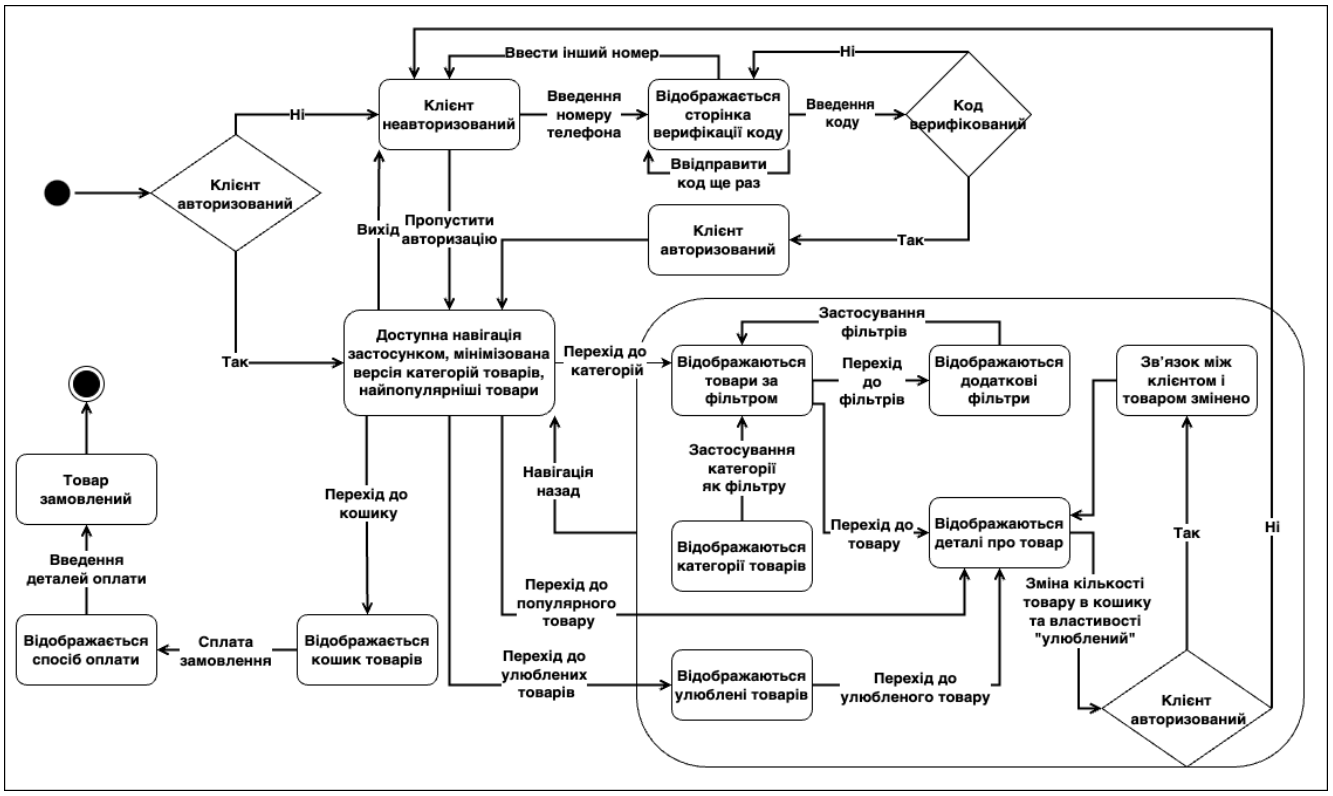

Рисунок 3.5 – Діаграма станів додатку

Початковим станом додатку є перевірка авторизації користувача. Якщо користувач авторизований, додаток переходить до основної частини взаємодії (основного стану). Якщо ні, користувач потрапляє на сторінку авторизації.

У стані не авторизованого користувач вводить номер телефону, після чого переходить до введення коду авторизації. Користувач може перейти до основного стану пропустивши авторизацію, але товари будуть відображатися без можливості

взаємодії. Користувач вводить код авторизації та переходить до основного стану, якщо код дійсний. Якщо ні, користувач залишається у стані введення коду. Користувач має можливість повернутись до стану не авторизованого. Користувач може відправити код авторизації ще раз.

У основному стані користувач взаємодіє з основним функціоналом додатку. Користувач може повернутись до стану не авторизованого. Переходячи звідси до наступних станів, окрім стану замоленого товару та неавторизованого, користувач завжди може повернутись до основного стану.

Переходячи з основного стану до стану категорій товарів, користувач обирає категорію товарів. Після вибору категорії, вона застосовується як фільтр, і користувач переходить до стану фільтрації товарів. Користувач може перейти до стану налаштування фільтрів та застосувати їх для подальшої фільтрації товарів, повернувшись до стану фільтрації товарів. Звідси користувач може перейти до стану перегляду деталей товару.

З основного стану користувач може перейти до стану перегляду списку улюблених товарів. Користувач може перейти до стану перегляду деталей товару зі списку улюблених.

У стані перегляду деталей товару користувач змінює кількість товару в кошику та додає товар до списку улюблених. Якщо користувач не авторизований, додаток переходить до стану авторизації. Якщо авторизований, зміни зберігаються, і стан повертається до відображення деталей товару.

Переходячи з основного стану до стану кошику користувач має можливість змінити вміст кошику та перейти до стану відображення способу оплати. Після вибору способу оплати стан переходить до замовлення товару і взаємодія з додатком завершується.

3.3 Огляд розробленого мобільного додатку

Розроблені мобільні додатки були створені для платформ Android та iOS, використовуючи класичний метод розробки з фреймворком Flutter та no-code

платформу FlutterFlow. Обидва додатка інтегровані з хмарним сховищем Firesore та беруть і зберігають свої дані в хмарній NoSQL базі даних Firestore.

Загальна кількість екранів у додатках становить 12. На рисунку 3.6 продемонстровано загальну структуру додатку.

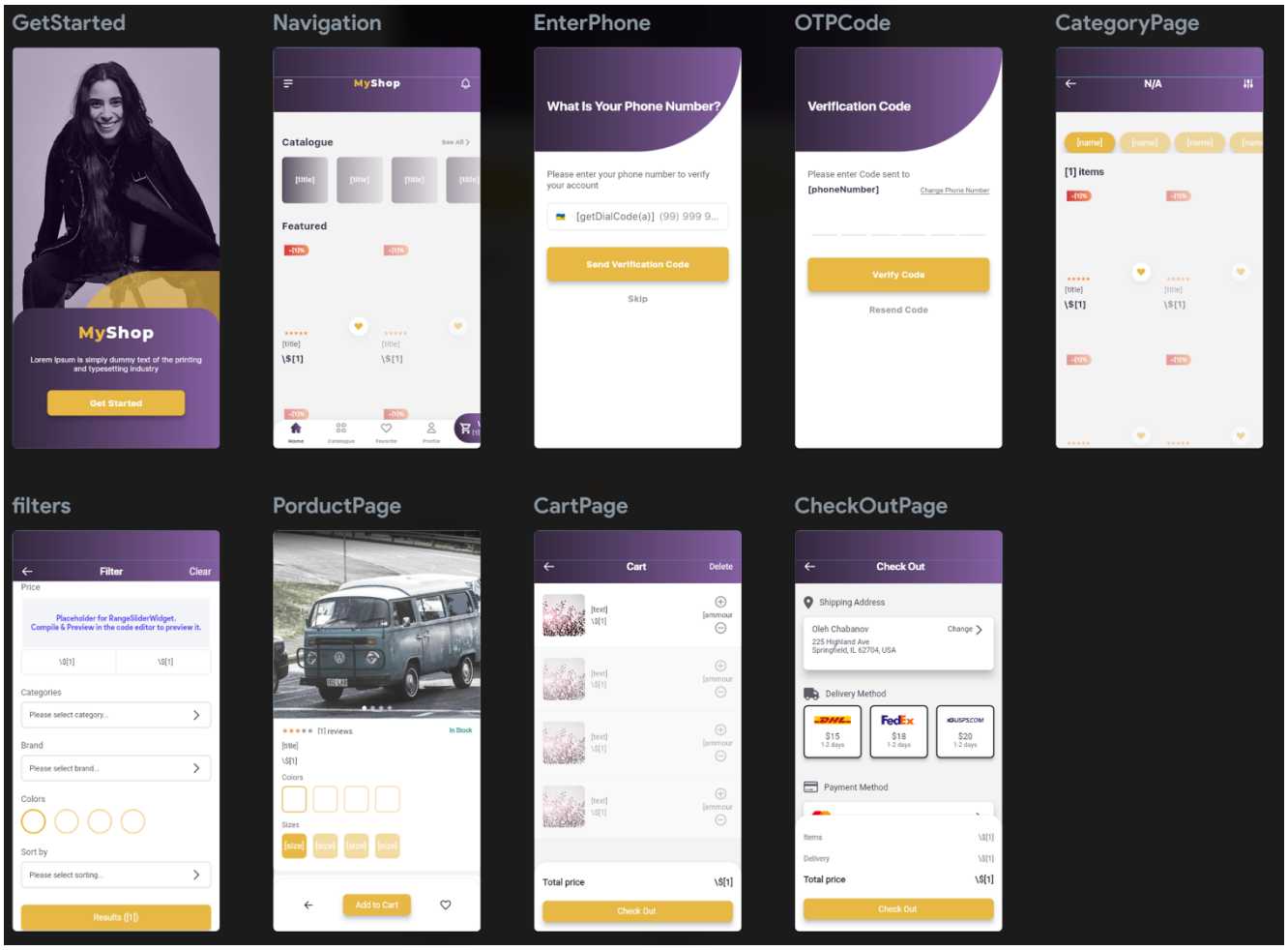

Рисунок 3.6 – Загальна структура додатку у редакторі FlutterFlow

Користувач починає з роботу з додатком з авторизації. Функціонал авторизації підтримує такі дії:

– введення номеру телефону для отримання OTP коду;
– запит на повторну відправку коду;
– авторизація через введення OTP коду.

Користувач вводить свій мобільний номер для отримання коду авторизації та вводить код, отриманий на мобільний номер, для завершення процесу авторизації. На рисунку 3.7 продемонстровано процес авторизації.

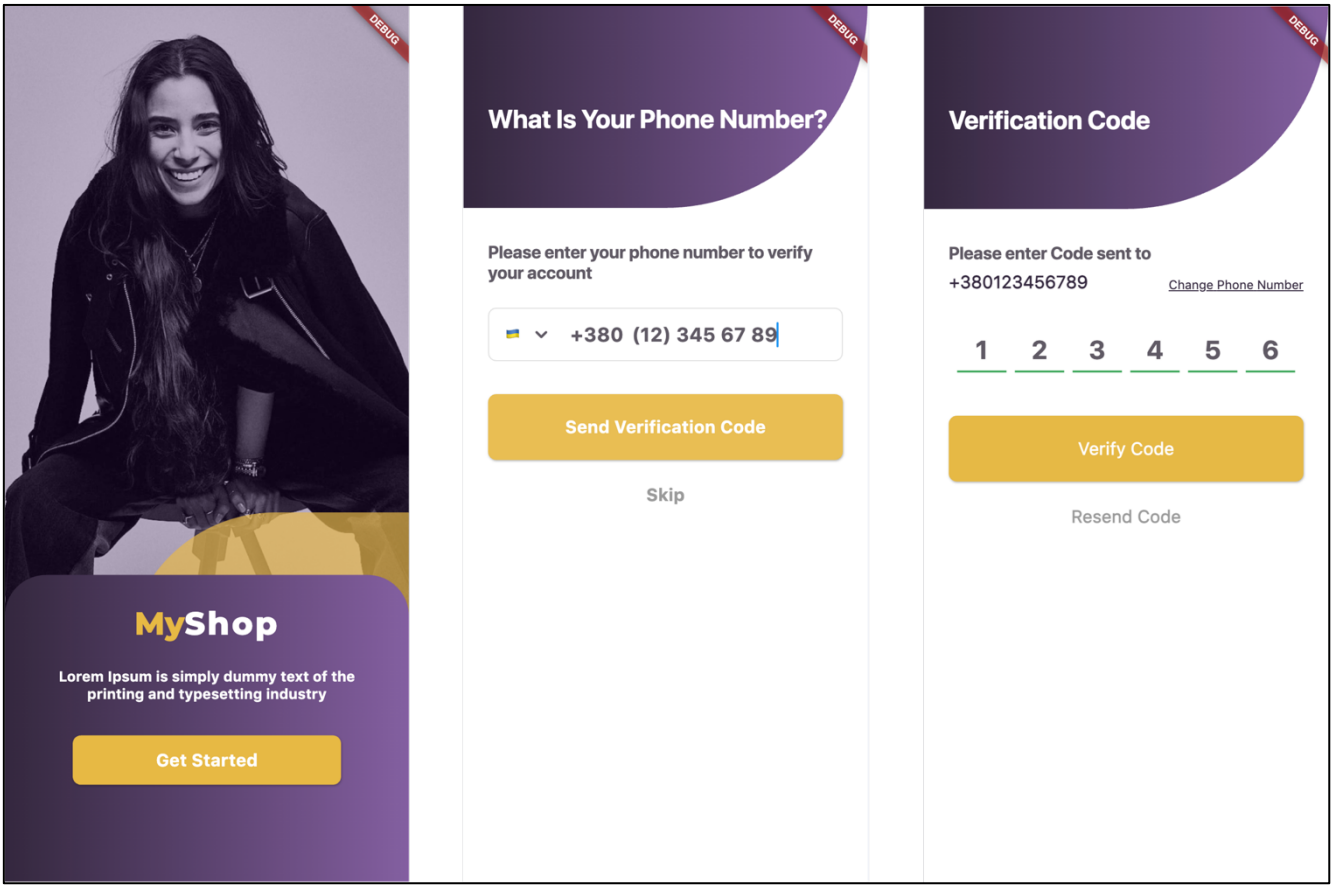

Рисунок 3.7 – Процес авторизації

Після авторизації користувач потрапляє на головну сторінку з популярними товарами. Головна сторінка надає користувачу такий функціонал:

– мінімізований перегляд каталогу;
– перегляд популярних товарів;
– навігація до всіх основних функцій додатку.

Користувач може переглянути детальну інформацію про обраний товар, включаючи опис, ціну та відгуки, натиснувши на елемент товару. Такий функціонал зберігається для елементів товару і для всіх подальших екранів На рисунку 3.8 продемонстровано головну сторінку та сторінку з деталями товару.

На сторінці товару користувач може взаємодіяти з товаром:

– переглянути детальну інформацію про товар;
– додати товар до кошику;
– додати товар до улюблених.

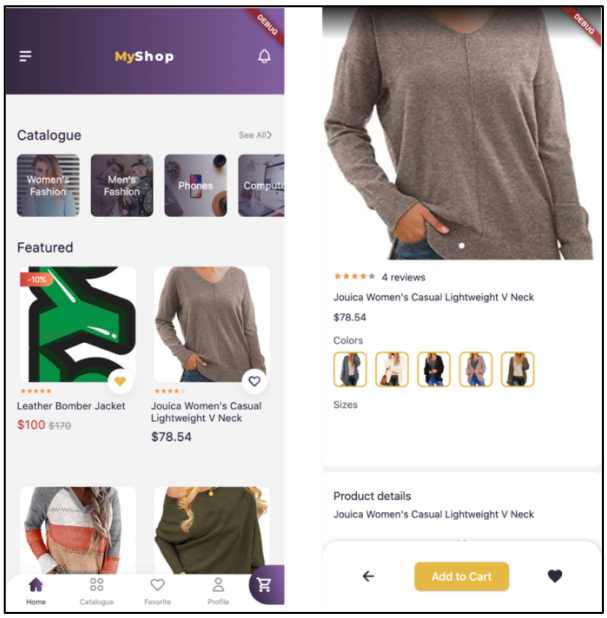

Рисунок 3.8 – Головна сторінка та деталі товару

На рисунку 3.9 продемонстровано категорії товарів. Переходячи до каталогу, користувач може взаємодіяти з наступним функціоналом:

– переглянути категорії;

– переглянути підкатегорії;

– обрати категорію та підкатегорію.

Обравши підкатегорію, вона застосовується як фільтр та користувач попадає на сторінку фільтрованих товарів.

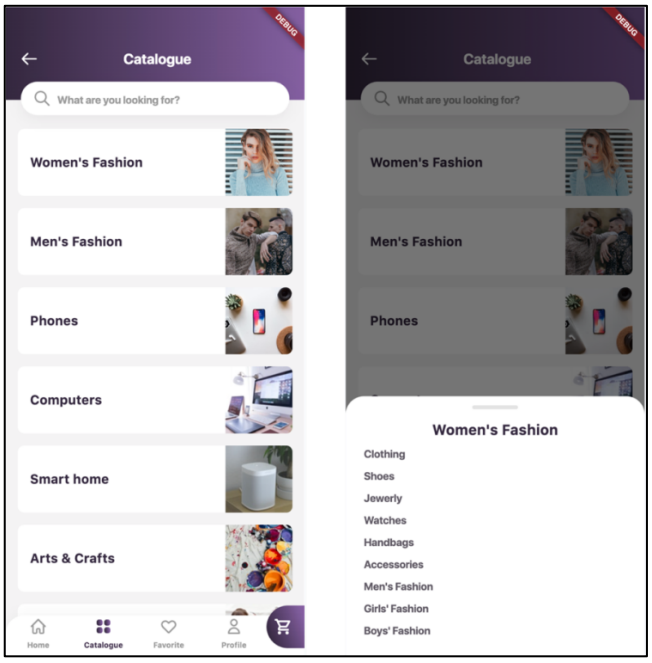

Рисунок 3.9 – Категорії товарів

Екран фільтрованих товарів відображає товари, що відповідають обраним фільтрам. Звідси доступна сторінка з додатковими фільтрами для точного пошуку товарів. На рисунку 3.10 продемонстровано процес фільтрації товарів.

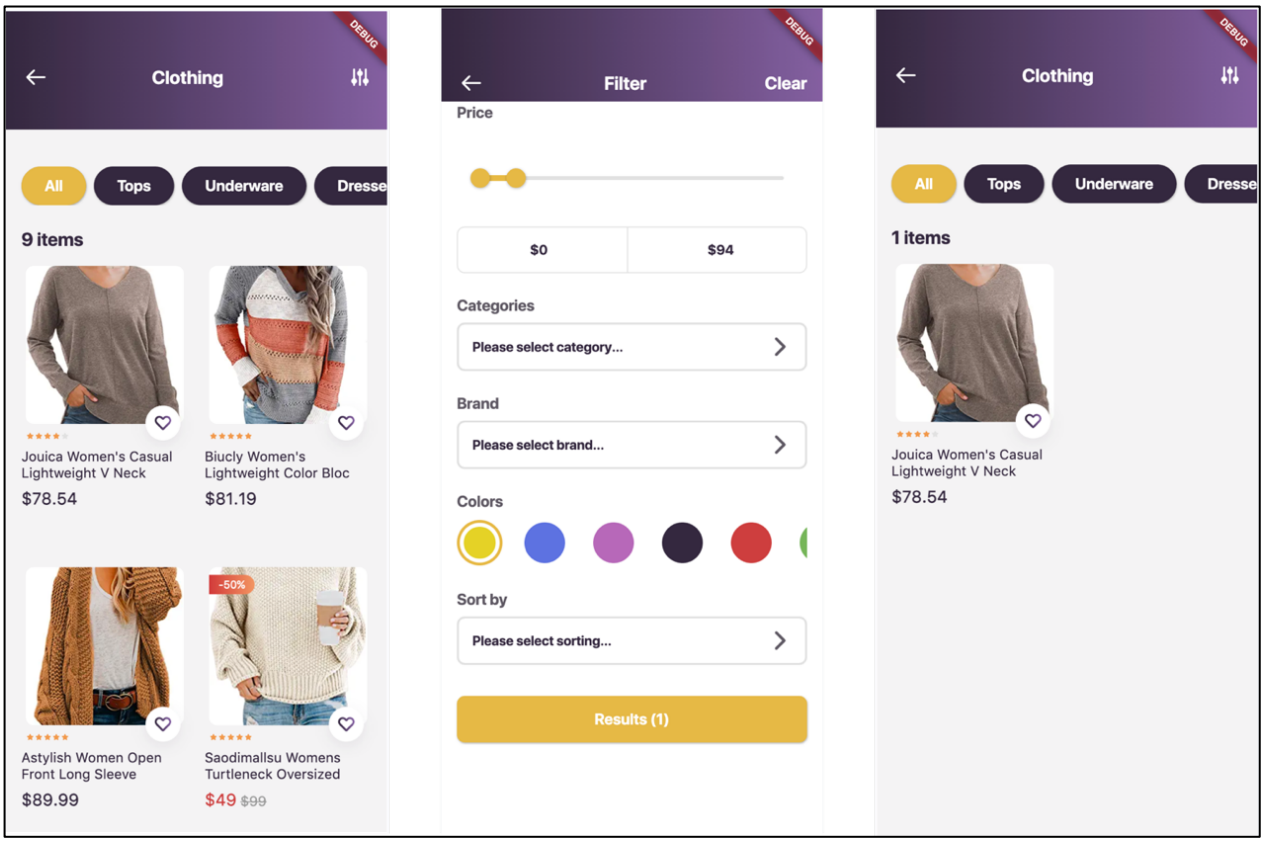

Рисунок 3.10 – Фільтр товарів

Товари можна сортувати за ціною. Додатково доступні наступні функції фільтрації:

– від мінімально обраної ціни;
– до максимальної обраної ціни;
– за категоріями;
– за брендами;
– за кольором.

Виходячи з кількості внутрішнього функціоналу, який необхідно було реалізувати для фільтрації, можна зазначити, що він є найважчою для реалізації частиною додатка. Це добре впливає на кристувацький досвід, але потребує

написання великого об’єму коду. З кодом бізнес логіки функції фільтрації можна ознайомитись у додатку Д.

Перейшовши до сторінки улюблених товарів, можна побачити відсортований список улюблених товарів користувача. Сортувати список можна за ціною. На рисунку 3.11 продемонстровано сторінку улюблених товарів.

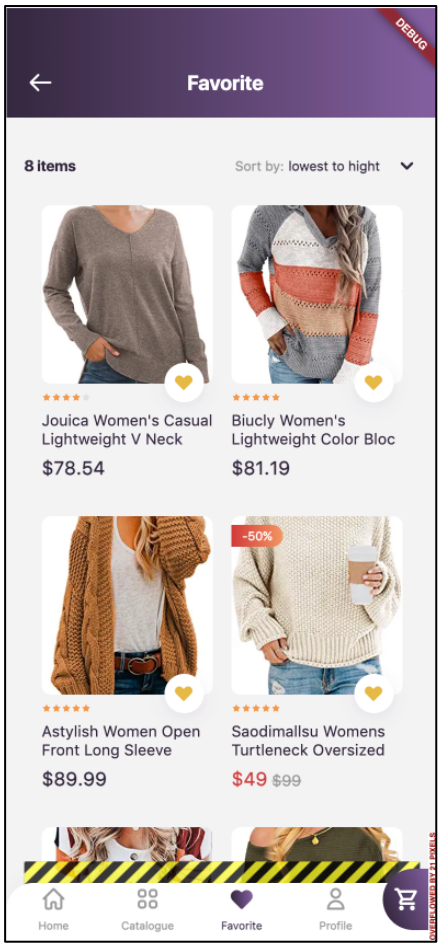

Рисунок 3.11 – Улюблені товари

Користувач може додавати товари до улюблених і до кошику. Товари, додані до кошику, відображаються на відповідній сторінці. Перейшовши до кошику, користувач може:

– видалити вміст кошику;
– змінити кількість товарів у кошику.

Після перегляду або модифікації кількості товарів у кошику, користувач обирає спосіб оплати для завершення замовлення. На рисунку 3.12 продемонстровано процес покупки.

Перейшовши до профілю, користувач може вийти з свого акаунту. На рисунку 3.13 продемонстровано вихід з профілю.

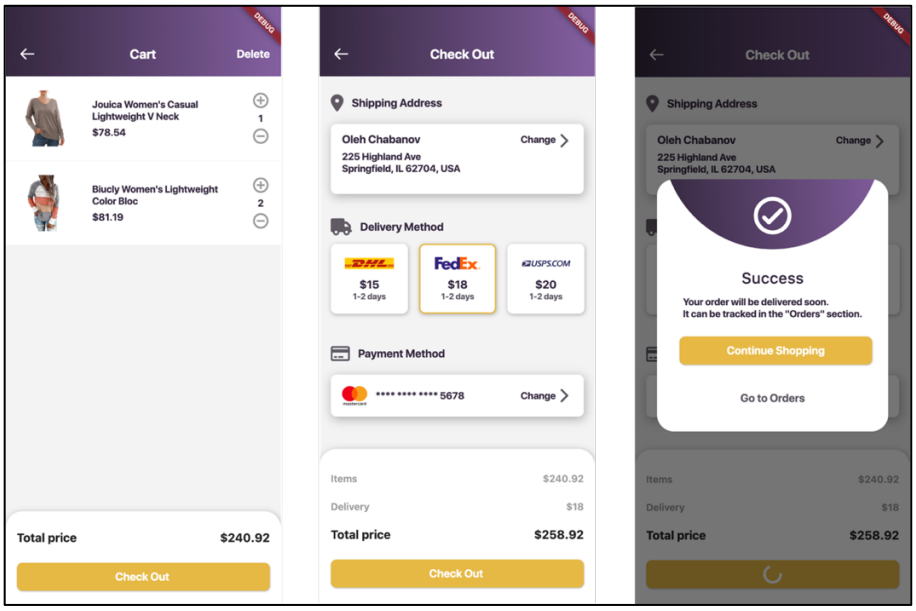

Рисунок 3.12 – Покупка товарів

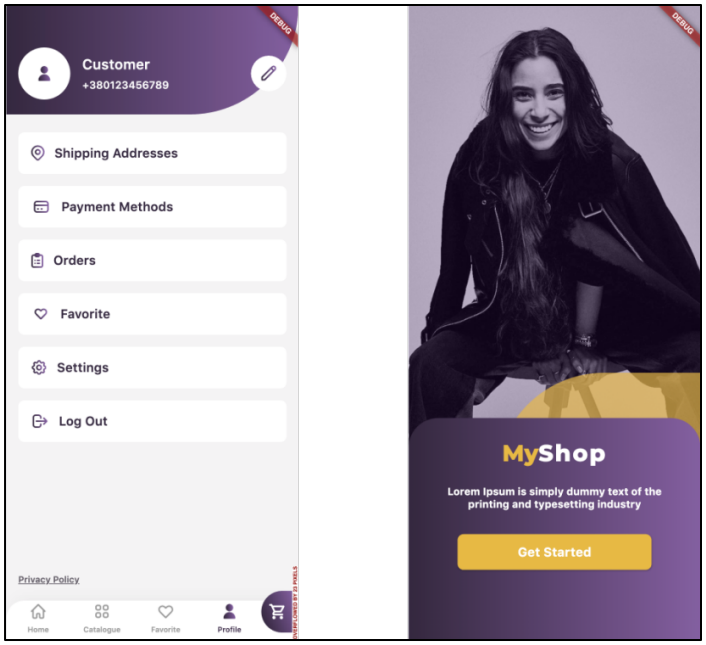

Рисунок 3.13 – Профіль

Після виходу з профілю користувач перенаправляється до авторизації на початкову сторінку.

4 ОПИС ПРОВЕДЕНИХ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Налаштування експериментів

Для проведення експерименту було визначено основні кроки, які необхідно виконати, та методи вимірювання часу, витраченого на реалізацію кожної функції мобільного додатку.

Перш за все були виключені всі додаткові фактори, що змінюють час реалізації функцій та не відносяться до мобільної розробки. Проектування мобільного додатку було розроблено заздалегідь і не включено в час реалізації, оскільки функціонал обох додатків є ідентичним, за виключенням використовуваних технологій. Реалізація хмарного сховища не враховувалася в час розробки, оскільки ця частина є ідентичною для обох додатків і не відноситься до мобільної розробки. Проектування інтерфейсу користувача також не було включено в час реалізації, оскільки було виконано заздалегідь для забезпечення однаковості обох додатків.

Спочатку додаток був реалізований з використанням класичного методу розробки за допомогою фреймворку Flutter. На кожному етапі розробки проводилися вимірювання часу, витраченого на реалізацію кожної окремої функції.

Після завершення класичної розробки, той самий додаток був реалізований за допомогою no-code платформи FlutterFlow. Аналогічно, на кожному етапі розробки проводилися вимірювання часу, витраченого на реалізацію кожної функції.

Кожна функція додатку може мати до трьох властивостей:

- інтеграція з серверними API для отримання та відправки даних;
- впровадження логіки, яка керує поведінкою додатку, таких як фільтрація товарів тощо;
- розробка складних візуальних елементів та інтерфейсів, таких як кастомні компоненти, анімації.

Процедура вимірювання часу виконується наступним чином:

- запуск таймера перед початком роботи над кожною функцією;
- виконання всіх необхідних кроків для реалізації функції (підключення до API, впровадження бізнес логіки, створення інтерфейсу);
- зупинка таймера після завершення роботи над функцією;
- запис часу, витраченого на реалізацію кожної функції, в спеціальну таблицю для подальшого аналізу.

Зібрані дані будуть використані для порівняння часових витрат на реалізацію функцій за допомогою класичних та no-code методів.

4.2 Результати проведених експериментів

Для аналізу часових витрат ознайомимося з даними, зібраними під час дослідження. На таблиці 4.1 наведено функціонал додатку, час на його реалізацію обома методами та властивості

Таблиця 4.1 – Витрати часу на реалізацію функціоналу

	Властивість				
Назва функціоналу	Присутнє підключення до API	Присутня бізнес логіка	Присутній складний інтерфейс	Класичний метод, години	No-code метод, години
Початкова сторінка	Ні	Ні	Ні	2,82	2,11
Аутентифікація за номером телефона	Так	Ні	Ні	12,32	4,37
Елемент навігації застосунком	Ні	Ні	Так	6,33	7,54
Головна сторінка	Так	Ні	Так	5,2	7,34
Сторінка категорій	Так	Ні	Ні	8,9	3,26
Фільтрація товарів	Так	Так	Ні	26,04	29,48
Деталі продукту	Так	Ні	Ні	16,98	9,68

Кінець таблиці 4.1

	Властивість				
Назва функціоналу	Присутнє підключення до API	Присутня бізнес логіка	Присутній складний інтерфейс	Класичний метод, години	No-code метод, години
Профіль користувача	Ні	Ні	Ні	5,97	2,29
Сторінка з улюбленими товарами	Так	Так	Ні	6,39	6,22
Кошик	Так	Ні	Ні	8,12	4,76
Пупка товарів	Так	Ні	Так	5	7,95

Для визначення впливу кожної властивості обчислимо коефіцієнти кожної з властивостей, використовуючи відношення часу виконання функцій No-code методом до часу виконання класичним методом.

Для цього необхідно:

– обчислити середній час виконання функцій класичним методом для кожної властивості;
– обчислити середній час виконання функцій No-code методом для кожної властивості;
– визначити коефіцієнт як відношення середнього часу класичного методу до середнього часу no-code методу.

Так було отримано коефіцієнти для:

– функцій без властивостей;
– функцій із присутнім підключенням API.

Звідси знаходимо що, коефіцієнт для функцій з підключенням API становить приблизно 0,476, а базовий коефіцієнт для функцій без додаткових властивостей становить приблизно 0,501.

Це означає, що функції без додаткових властивостей або тільки з властивістю підключення до API виконуються за допомогою no-code методу вдвічі швидше, ніж класичним методом.

Знаючи ці коефіцієнти і час виконання функцій, розраховано коефіцієнт для бізнес логіки. Було використано функції, які містять бізнес логіку, і обчислено, як час їх виконання співвідноситься з іншими властивостями. Отже, знаючи коефіцієнт підключенням API отримано коефіцієнт 2,21 з урахуванням функції фільтрації товарів та сторінки з улюбленими товарами, де представлена властивість бізнес логіки. Таким же чином був розрахований коефіцієнт для складного інтерфейсу, використовуючи функції з даної властивістю відповідно. Коефіцієнт для функцій зі складним інтерфейсом становить приблизно 2,172. Тож функції зі складним інтерфейсом або бізнес логікою виконуються за допомогою no-code методу на вдвічі довше, ніж класичним методом.

Виходячи з обчислений коефіцієнтів, ефективність no-code методу збільшується при реалізації функцій без додаткових вимог або з вимогою підключення до API. Вимоги щодо бізнес логіки та складного інтерфейсу можуть збільшити час створення функцій порівняно з класичним методом.

Визначившись із коефіцієнтами і за умови відомості про час виконання функції класичним методом та її властивостями, можна розрахувати створення такої функції no-code методом за формулою 4.1:

$$T = X * \alpha * \beta * \gamma * \delta \quad (4.1)$$

де X – час виконання функції класичним методом розробки,

α– базовий коефіцієнт якщо функція не має властивостей, і 1 в іншому випадку,

β– коефіцієнт властивості API, якщо у функції присутня така властивість, інакше 1,

γ– коефіцієнт властивості бізнес логіки, якщо у функції присутня така властивість, інакше 1,

δ– коефіцієнт властивості складного інтерфейсу, якщо у функції присутня така властивість, інакше 1.

Враховуючи зазначену формулу та коефіцієнти, було розраховано очікуваний час на реалізацію функцій no-code методом. В таблиці 4.2 відображено результати розрахунків.

Таблиця 4.2 – Очікуваний час на розробку функцій no-code методом

Назва функціоналу	Класичний метод, години	No-code метод, години	No-code метод, очікуваний результат, години
Початкова сторінка	2,82	2,11	1,41
Аутентифікація за номером телефона	12,32	4,37	5,86
Елемент навігації застосунком	6,33	7,54	13,75
Головна сторінка	5,2	7,34	5,38
Сторінка категорій	8,9	3,26	4,24
Фільтрація товарів	26,04	29,48	27,39
Деталі продукту	16,98	9,68	8,08
Профіль користувача	5,97	2,29	2,99
Сторінка з улюбленими товарами	6,39	6,22	6,72
Кошик	8,12	4,76	3,87
Пупка товарів	5	7,95	5,17

На основі проведеного порівняльного та емпіричного аналізу можна зробити висновок, що no-code метод є швидшим для функцій без складного інтерфейсу та бізнес логіки, а також для функцій з підключенням до API. Однак, для функцій зі складним інтерфейсом класичний метод є більш ефективним.

Розроблені математичні моделі дозволяють прогнозувати часові витрати на розробку мобільних додатків, враховуючи різні властивості функцій.

4.3 Шляхи подальшого розвитку дослідження

Вже зараз результати дослідження можуть бути використані на практиці для вибору оптимальних методів розробки мобільних додатків у залежності від їх функціоналу. Також ці результати можуть бути основою для подальших досліджень і розробок у галузі програмної інженерії. Але подальший розвиток дослідження вимагає збору більшої кількості даних, використання точніших методів аналізу, а також включення нових технологій для порівняння.

Проведене дослідження базується на обмеженій кількості функцій і технологій. Для отримання більш точних і узагальнених результатів необхідно збільшити вибірку даних. Це може включати аналіз додаткових мобільних застосунків з різним функціоналом і складністю. Застосування більш складних статистичних методів та моделей машинного навчання для аналізу часових витрат на розробку теж дозволить краще зрозуміти вплив різних факторів на ефективність різних методів розробки.

Сфера розробки мобільних додатків швидко розвивається, і нові технології з'являються регулярно. Необхідно періодично оновлювати дані та аналіз, щоб включати нові інструменти та методи розробки. Це дозволить забезпечити актуальність дослідження та його результатів.

ВИСНОВКИ

Було проведено комплексне дослідження ефективності різних методів розробки мобільних додатків, зокрема класичних методів і no-code підходів. Проаналізовані отримані дані та розроблені математичні моделі для прогнозування часових витрат на розробку.

No-code методи показали себе ефективними для функцій без складного інтерфейсу та бізнес логіки, а також для функцій з підключенням до API. З іншого боку, класичні методи виявилися більш ефективними для функцій зі складним інтерфейсом та бізнес логікою.

Було розроблено математичні моделі для оцінки часу розробки мобільних додатків на основі властивостей функцій. Ці моделі дозволяють прогнозувати часові витрати на no-code розробку.

Отримані результати мають високу практичну значущість. Розроблені моделі можуть бути використані розробниками та менеджерами проектів для більш точного планування часових витрат на розробку мобільних додатків, а також для вибору оптимального методу розробки в залежності від конкретних вимог проекту. Використання результатів дослідження може значно підвищити ефективність процесу розробки, знизити витрати на розробку та підтримку додатків, а також покращити якість кінцевого продукту.

Таким чином, дане дослідження робить вагомий внесок у розуміння процесів розробки мобільних застосунків та допомагає знаходити оптимальні рішення для різних умов і вимог проектів.

Дослідження демонструє високу ефективність використання як класичних методів розробки, так і no-code платформ, залежно від конкретних умов та вимог проекту. Застосування результатів даного дослідження у практичній діяльності дозволить більш точно планувати та вибірати відповідні інструментів розробки в залежності від потреб.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Iryna Gruzdo, Iryna Kyrychenko, Glib Tereshchenko та Oleksandr Shanidze, “Analysis of Models Usability Methods Used on Design Stage to Increase Site Optimization”, CEUR Workshop Proceedings, 2023, 3403, pp. 387–409. URL: https://ceur-ws.org/Vol-3403/paper31.pdf (дата звернення: 02.04.2024).

2. Iryna Gruzdo, Iryna Kyrychenko, Glib Tereshchenko та Nadiya Shanidze, “Metrics applicable for evaluating software at the design stage” // 5th International Conference on Computational Linguistics and In-telligent Systems (COLINS-2021), Kharkiv, Ukraine, April 22-23, 2021. – CEUR Workshop Proceedings, 2021, 2870, Volume I, pp. 916-936. URL: https://ceur-ws.org/Vol-2870/paper69.pdf (дата звернення: 03.04.2024).

3. Rutuja Kurale and Kumkum Bala, " A Comparative Study of Flutter With Other Cross-Platform Mobile Application Development.", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.9, Issue 12, pp.a368-a372, December 2021. URL: http://www.ijcrt.org/papers/IJCRT2112036.pdf (дата звернення: 12.04.2024).

4. Masaad Alsaid, Mohamed Abdal Mohsin, Tarig Mohamed Ahmed, Sadeeq Jan, Fazal Qudus Khan, Mohammad, and Amjad Ullah Khattak. 2021. “A Comparative Analysis of Mobile Application Development Approaches: Mobile Application Development Approaches”, Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences 58 (1):35-45. URL: https://doi.org/10.53560/PPASA(58-1)717 (дата звернення: 20.04.2024).

5. Varajão, João, António Trigo, and Miguel Almeida. 2023. "Low-Code Development Productivity: 'Is Winter Coming' for Code-Based Technologies?", Queue 21, no. 5 (September/October 2023): 40-60. URL: https://doi.org/10.1145/3631183 (дата звернення: 01.05.2024).

6. Yan Zhaohang. “The Impacts of Low/No-Code Development on Digital Transformation and Software Development.”, ArXiv abs/2112.14073 (2021). URL: https://arxiv.org/pdf/2112.14073 (дата звернення: 04.05.2024).

7. Stack Overflow Insights. URL: https://insights.stackoverflow.com/trends?tags=flutter%2Creact-native%2Cionic-framework%2Cswiftui%2Ccordova%2Candroid-jetpack-compose%2Cmaui (дата звернення: 12.05.2024).

8. Flutter documentation. URL: https://docs.flutter.dev/ (дата звернення: 15.05.2024).

9. FlutterFlow Docs. URL: https://docs.flutterflow.io/ (дата звернення: 24.05.2024).