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 архітектурних рішень крос-платформного програмного забезпечення для створення сучасних мобільних застосунків
(тема)

Виконав:
Здобувач 2 року навчання
групи ІПЗм-23-1

Анастасія ЧЕРВІНСЬКА
(Власне ім'я, ПРІЗВИЩЕ)

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

Керівник доц. Ірина АФАНАСЬЄВА
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
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)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Червінська Анастасія Любомирівна
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження архітектурних рішень крос-платформного програмного забезпечення для створення сучасних мобільних застосунків»
Затверджена наказом по університету від 25.04. 2025р. № 290 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 05.06.2025
3. Вихідні дані до роботи теоретичні дослідження, практичне дослідження, відкриті джерела, офіційна документація фреймворків Flutter, React Native, MAUI.
4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз предметної галузі і постановка задачі, огляд та аналіз літературних джерел з дослідження, дослідження теоретичне, дослідження практичне

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	10.10.2024	*виконано*
2	Аналіз предметної галузі і постановка задачі	21.01.2025	*виконано*
3	Огляд й аналіз літературних, наукових джерел	20.02.2025	*виконано*
4	Теоретичне дослідження	21.03.2025	*виконано*
5	Підготовка до апробації результатів дослідження. Публікація матеріалів	18.04.2025	*виконано*
6	Практичне дослідження	14.05.2025	*виконано*
7	Підготовка пояснювальної записки	15.05.2025	*виконано*
8	Підготовка презентації та доповіді	29.05.2025	*виконано*
9	Перевірка на плагіат	30.05.2025	*виконано*
10	Нормоконтроль	04.06.2025	*виконано*
11	Рецензування	06.06.2025	*виконано*
12	Попередній захист	07.06.2025	*виконано*
13	Занесення диплома в електронний архів	10.06.2025	*виконано*
14	Допуск до захисту у зав. кафедри	11.06.2025	*виконано*

Дата видачі завдання 8 жовтня 2024р.

Студент (ка) _____ Анастасія ЧЕРВІНСЬКА
(підпис)

Керівник роботи _____ доц. Ірина АФАНАСЬЄВА
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 76 с., 37 рис., 1 табл., 17 джерел.

АРХІТЕКТУРНІ РІШЕННЯ, КРОС-ПЛАТФОРМА, МОБІЛЬНІ ЗАСТОСУНКИ, ФРЕЙМВОРК, ANDROID, FLUTTER, IOS, MAUI , REACT NATIVE.

Об’єктом дослідження є крос-платформні фреймворки для написання мобільних застосунків.

Метою роботи є проведення дослідження сучасних крос-платформних фреймворків, які використовуються в індустрії мобільних застосунків, зробити аналіз їх переваг та недоліків, та запропонувати вдосконалення стратегії розробки.

Методами розробки та проектування є аналіз проблемної області дослідження, вибір Flutter, MAUI, React Native для проведення дослідження шляхом порівняння переваг та недоліків.

Наукова новизна полягає в систематизованому підході до порівняння крос-платформних фреймворків із врахуванням усіх технічних характеристик. Запропоновано критерії оцінки, які дозволяють об’єктивно обрати найпродуктивніший фреймворк.

У процесі дослідження проаналізовано конструктивні, технологічні й техніко-експлуатаційні характеристики та показники Flutter, MAUI, React Native, включаючи архітектуру, принцип роботи, основні компоненти, середовище розробки, продуктивність та зручність у використані.

У результаті кваліфікаційної роботи розроблено три мобільні застосунки, по одному для кожного з досліджуваних крос-платформних фреймворків: Flutter, MAUI, React Native.

Результати роботи можуть бути використані під час вибору фреймворку для розробки мобільного застосунку, а також у навчальному процесі для демонстрації відмінностей фреймворків.

Розроблені рекомендації можуть використовуватися у сфері IT, а саме у команді розробників мобільних застосунків, а також у освітніх установах.

ARCHITECTURAL SOLUTIONS, CROSS-PLATFORM, MOBILE APPLICATIONS, FRAMEWORK, ANDROID, FLUTTER, IOS, MAUI, REACT NATIVE.

The object of research is cross-platform frameworks for writing mobile applications.

The purpose of the work is to conduct a study of modern cross-platform frameworks used in the mobile application industry, to analyze their advantages and disadvantages, and to propose improvements in the development strategy.

Methods of development and design are the analysis of the problem area of research, the choice of Flutter, MAUI, React Native to conduct research by comparing the advantages and disadvantages.

The scientific novelty lies in a systematic approach to comparing cross-platform frameworks, considering both technical characteristics and practical performance. Evaluation criteria are proposed that allow an objective selection of the most productive framework.

The study analyzes the design, technological and operational characteristics and indicators of Flutter, MAUI, React Native, including architecture, principle of operation, main components, development environment, performance and usability.

As a result of the qualification work, three mobile applications were developed, one for each of the studied.

The results can be used when choosing a framework for developing a mobile application, as well as in the educational process to demonstrate the differences between frameworks.

The developed recommendations can be used in the IT sector, namely in the mobile application development team, as well as in educational institutions.

Умови публікації звіту: заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElAr KhNURE.

Завідувачу кафедри
ПІ_____
(скорочена назва кафедри)
проф. Кирилу СМЕЛЯКОВУ_
(вчене звання, сласне ім'я, прізвище)

ЗАЯВА
щодо самостійності виконання кваліфікаційної роботи та можливості її публікації (та/або публікації анотації кваліфікаційної роботи) в електронному архіві відкритогодоступу ElAr KhNURE

Я, _Червінська Анастасія Любомирівна_____
(прізвище, ім'я, по батькові)

здобувач вищої освіти на другому (магістерському) рівні вищої освіти академічної групи_ІПЗм-23-1_____
кафедра_____ програмної інженерії_____,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему _Дослідження архітектурних рішень крос-платформного програмного забезпечення для створення сучасних мобільних застосунків_____,
(назва роботи)

що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в репозиторії "ElArKhNURE". Погоджуюся з авторським договором, відповідно до Положення про репозиторій ХНУРЕ "ElArKhNURE". Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з вимогами академічної доброчесності, згідно з якими виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

Дата Підпис

ЗМІСТ

ВСТУП

Сьогодні смартфони є у кожного, і це вже далеко не просто модний гаджет. Смартфон – це невід'ємна частина нашого життя. Він дозволяє спілкуватися з друзями та рідними в будь-якій точці світу, відправляти повідомлення і швидко знаходити потрібну інформацію.

Але це не лише про спілкування, це гарний інструмент для розвитку. Через смартфон можна дізнаватися нові речі, слухати аудіокниги та подкасти, дивитися відеоуроки та дізнаватися про різні культури. Також вони допомагають підтримувати здоровий спосіб життя, дотримуватись режиму харчування або водного балансу. Після встановлення такого виду застосунків можна отримати інструмент для фітнесу або щоденник для харчування.

Сучасні мобільні застосунки мають багато можливостей. Від бронювання квитків до онлайн-банкінгу чи онлайн знайомств. Лише кілька дотиків можуть легко вирішити проблему, яка раніше займала години. Та вирішити проблему за декілька хвилин.

Ці технології роблять наше життя простіше, допомагають працювати та навчатись віддалено. Ці пристрої стали невід'ємною частиною нашого повсякденного життя.

У сучасному світі мобільні застосунки надають багато можливостей, роблять наше повсякденне життя більш комфортним. Важливо, що успіх мобільних застосунків пов'язаний з якістю їх програмного забезпечення, а також швидкістю їх розробки та оновлення для пристосування к постійним змінам сучасних реалій.

Сьогодні перед розробниками постає велика кількість технологій, які можливо використати для розробки мобільних застосунків. Вибираючи технологію, розробнику необхідно враховувати попередній досвід, функціональність майбутнього програмного забезпечення і терміни реалізації проекту.

Завдяки швидкому розвитку технологій і мінливим вимогам багато розробників обирають кросплатформені рішення. Для цього є багато причин:

швидка реалізація для різних платформ, легка підтримка завдяки загальній кодової базі та економія часу.

Для розробника під час вибору технології для створення застосунку важливим є баланс між якістю програмного забезпечення та швидкістю розробки. Хороший мобільний застосунок повинен відповідати усім функціональним вимогам, працювати без збоїв на різних пристроях і операційних системах. Саме тому розробники вибирають такі технології, які дозволяють забезпечити надійність та швидкість.

Останнім часом кросплатформні рішення привертають до себе все більше уваги. Адже вони не лише скорочують час розробки, але й допомагають створювати продукти, які виглядають і працюють однаково добре на будь-якому пристрої. Саме на це варто звертати увагу в досліджені.

Отже, основною метою дослідження є:

- вивчення сучасних методів та технологій, які використовуються в індустрії мобільних застосунків;
- аналіз переваг та недоліків фреймворків;
- вдосконалення стратегії розробки крос-платформних застосунків;
- дослідження аспектів дизайну інтерфейсу, архітектури програмного забезпечення, оптимізації продуктивності та взаємодії з користувачем.

В результаті роботи планується розробити набір практичних рекомендацій для розробників, які сприятимуть покращенню якості та продуктивності мобільних застосунків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

У наш час, коли усі в повсякденному житті користуються мобільними додатками, важливо знати, які технології найкраще відповідають вимогам ринку і користувачів. Технічні рішення прямо впливають на функціональність, продуктивність і користувальницький досвід [1]. Вони є основою до успіху мобільного застосунку.

Все частіше розробники вибирають крос-платформні рішення, замість нативної розробки [2]. Для цього існує ряд причин:

- крос-платформні фреймворки дозволяють розробникам використовувати один і той самий код для різних платформ, що дозволяє економити час та ресурси, в порівняні з написанням окремих нативних застосунків для Android та iOS;
- розробка для обох платформ одночасно дозволяє швидше випустити продукт на ринок, зменшуючи час від ідеї до реалізації;
- собівартість розробки значно нижче, що дозволяє вкласти більше грошей в маркетинг і залучити більше користувачів.

У сучасному світі розробки мобільних застосунків виникає питання про найбільш оптимальне та ефективне крос-платформне рішення.

Вибір ускладняється тим, що на ринку представлено багато різних доступних технологій. Розглянемо найпопулярніші: Flutter, React Native та Xamarin.

React Native – це крос-платформний фреймворк з відкритим вихідним кодом для розробки мобільних застосунків під платформи iOS та Android, який створив Facebook у 2015 році. Він використовує Java Script як мову програмування. React Native має такі переваги:

- функція живого перезавантаження React Native дозволяє бачити та працювати зі змінами в режимі реального часу; розробник може зробити виправлення в коді під час завантаження програми, і це буде відображено в застосунку з автоматичним перезавантаженням;

- React Native має великі можливості рендерингу і використовує компонентний підхід, який дозволяє легко створювати як прості, так і складні конструкції інтерфейсу;
- можливість багаторазово використовувати код React Native, що допомагає заощадити витрати на розробку;
- React Native пропонує багато різних плагінів, включаючи нативні та модулі на основі Java Script;
- React Native має ком'юніті більш ніж 100 000 [3] активних учасників, яке зможе надати експертну підтримку;
- інженери Facebook постійно оновлюють та поліпшують фреймворк.

За допомогою цього фреймворку були створені такі відомі проекти, як Instagram, Bloomberg та Pinterest.

Flutter – це набір інструментів та фреймворк з відкритим вихідним кодом для мобільних застосунків для Android і iOS, веб-додатків і десктопних додатків для Windows, macOS і Linux з використанням мови програмування Dart, розроблений Google.

Flutter користується популярністю через:

- функція «hot reload» дозволяє бачити оновлення в режимі реального часу без перевантаження програми;
- Flutter застосунки працюють на рівні з нативними мобільними додатками;
- має зростаючу спільноту, що може допомогти з проблемами;
- має зрозумілу документацію;
- є багато відео YouTube, доступних для тих, хто хоче почати вивчати Flutter або поліпшити свої навички;
- Flutter легко дозволяє налаштовувати навіть дуже складний дизайн інтерфейсу та надає багато вбудованих можливостей для різних анімацій;
- Flutter представляє інструменти, які спрощують процес тестування.

Є кілька досить великих компаній, які активно використовують цей фреймворк: Google Ads, eBay Motors, SpaceX Go.

Xamarin – це платформа з відкритим вихідним кодом, призначена для створення сучасних застосунків для iOS, Android і Windows з .NET. Використовує мову програмування C#.

На жаль, з 1 травня 2024 року Microsoft офіційно перестав підтримувати цей фреймворк. Останніми версіями, які підтримувалися Xamarin, були Android API 34 та Xcode 15 SDK (iOS 17). Також варто звернути увагу, що з квітня 2025 року Apple вимагає, щоб усі застосунки подані у App Store підтримували iOS SDK 18. Як було вже зазначено Xamarin не підтримує цю версію SDK. Можемо зробити висновок, що усі застосунки написані на Xamarin доведеться переписувати і не варто використовувати цей фреймворк для нових застосунків. Отже, Xamarin втратив свою актуальність. Microsoft рекомендує розробникам використовувати Multi-platform App UI (MAUI) замість Xamarin.

MAUI дозволяє створювати кросплатформені мобільні та десктопні застосунки з використанням однієї кодової бази для всіх цільових пристроїв. Він також використовує мову програмування C# та розмітку XAML. .NET Multi-platform UI – це молодий та перспективний фрейворк, тому розглянемо його у досліджені.

Преваги використання MAUI:

- код можна використовувати для розробки додатків на декількох платформах; кодування програми базується на бібліотеках C# та .NET;
- MAUI пропонує багато API і плагінів і підтримує крос-платформну розробку застосунків, що дозволяє розробникам насолоджуватися взаємодією на нативному рівні з апаратним забезпеченням пристрою;
- - функція «hot reload» дозволяє бачити зміни в коді без перевантаження програми;
- активна підтримка Microsoft, регулярні оновлення та розширення функціоналу;
- оптимізований рендеринг UI-елементів, завдяки «Handlers» замість «Renderers», якщо порівнювати з Xamarin.Forms.

На цій платформі було написано декілька великих застосунків: Azure App, Berichtenbox і DigiD.

З такої великою кількістю різних технологій виникає потреба у досліджені цих технологій та їх архітектурних рішень.

Актуальність вибору найкращої технології розробки крос-платформних мобільних застосунків основана на швидкому розвитку індустрії мобільних технологій. Щороку кількість пристроїв і платформ зростає, і це створює попит на ефективні та швидкі способи створення програмного забезпечення.

Спроба розробити три мобільні застосунки на платформах Flutter, React Native і MAUI дозволить порівняти результати, оцінити їх ефективність, швидкість і зручність використання. Таке дослідження допоможе визначити переваги, недоліки та обмеження кожної технології.

Крім того, в дослідженнях буде акцентована увага на архітектурні рішення і їх вплив на продуктивність, адаптивність і стабільність мобільних застосунків.

Аналізуючи технічні аспекти, дослідження будуть зосереджені на визначенні найкращих практик для розробки сучасних, ефективних і конкурентоспроможних мобільних застосунків.

2 ОГЛЯД Й АНАЛІЗ ЛІТЕРАТУРНИХ, НАУКОВИХ ДЖЕРЕЛ

2.1 Аналіз досліджень та наукових статей

Дослідження методів створення сервісно-орієнтованих та крос-платформених додатків за допомогою Flutter з серверною частиною на Firebase [4] спрямоване на аналіз Flutter та Firebase в якості технологій розробки програмних систем. Обрана тема є актуальною через стрімкий розвитком крос-платформної розробки та її постійні зміни та вдосконалення. Дослідження включає в себе визначення переваг та недоліків використання Flutter і Firebase в поєднанні, а також їх якісних характеристик при використані для невеликих та масштабних проєктів.

Основною задачею дослідження є перевірка теорії про те, що використання інтеграції Flutter і Firebase може спростити процес розробки крос-платформного клієнтського застосунку. Під час дослідження були використані аналітичні та експериментальні методи, було реалізовано та протестовано архітектури Flutter застосунку.

Результати аналізу показали, що процес розробки на Flutter був дуже швидким та ефективним, система дозволяє реалізувати, будь-яку задумку без великих зусиль. Реалізація будь-якого інтерфейсу також швидка завдяки існуючим у фреймворку механізмам. Дане дослідження дало змогу зрозуміти, що Flutter - сучасна технологія, яка має багато можливостей та зручна для використання.

У досліджені ефективності фреймворку Flutter при розробці високо динамічних мобільних додатків зроблена оцінка його ефективності фреймворку Flutter в порівнянні з іншими технологіями розробки мобільних додатків [5]. У цій роботі ставилося завдання вивчити можливості та обмеження фреймворку Flutter, порівняти його з іншими технологіями та надати практичні рекомендації. У досліджені було створено соціальні мережі на фреймворках Flutter та React Native, з аналізом продуктивності, можливостей та швидкодії. У результаті були виділено такі переваги обох мов:

- Flutter: вища продуктивність, краща швидкодія, більші можливості налаштування;

- React Native: гнучкість у використанні ресурсів, добре відома спільнота розробників.

Використання Flutter визначається як кращий фреймворк ніж React Native, оскільки він має вищу продуктивність, якісну документацію та більше підходить для масштабованих систем.

Дослідження ефективності фреймворку React Native при розробці мобільних додатків [6] включає аналіз інструментів, детальний огляд React Native та розробку демонстраційного застосунку. У експерименті було зроблено порівняння нативного, гібридного та кросплатформенного підходів, з фокусом на React Native. Експеримент продемонстрував оптимальну продуктивність та якість додатка, що використовує React Native. Це дослідження підтвердило те, що кросплатформенні рішення, зокрема React Native, можуть бути оптимальним вибором для економічної, ефективної та якісної мобільної розробки.

Розробка програмного забезпечення для дослідження ефективності засобу Flutter при розробці мобільних додатків [7] спрямоване на вдосконалення розробки мобільних додатків для Android та iOS за допомогою фреймворку Flutter. За актуальністю стоїть зростання популярності мобільних пристроїв у бізнесі. Робота вирішує проблему складності розробки для різних платформ, надаючи практичні рекомендації та прототип бібліотеки для оцінки ефективності Flutter. Це може бути корисним у подальшому досліджені. У експерименті розробки одного додатку з використанням Flutter і іншого за допомогою React Native виявлено, що технології та принципи, які використовує фреймворк Flutter, переважають над його найближчим конкурентом, React Native, у багатьох ключових аспектах розробки мобільних додатків. Для поліпшення рішення проблеми, пов'язаної з вибором між Flutter та React Native, пропонується використовувати Flutter як більш перспективний, продуктивний, та універсальний фреймворк для кросплатформенної мобільної розробки. Також, розроблено прототип плагіну для визначення швидкодії розроблених додатків, що може покращити процес розробки.

Дослідження можливостей .NET MAUI [8] розглядає актуальність розвитку індустрії мобільних пристроїв, обумовлену жорсткою конкуренцією та стрімкими

темпами змін. Зазначається, що для виробників є важливим впровадження інновацій у програмне забезпечення, оскільки зростаючий функціонал пристроїв породжує попит на різноманітні програмні рішення. У зв'язку з цим, дослідження спрямоване на дослідження можливостей технології .NET MAUI, враховуючи різноманіття операційних систем та короткі терміни розробки. Мета включає створити функціональний інтерфейс за допомогою .NET MAUI, що працюватиме на мобільних та настільних платформах. Досліджуються процеси розробки, використання патерну MVVM, а також інтеграція з сучасними API. У результаті розробки застосунку були зроблені наступні висновки:

- висока продуктивність та ефективне використання ресурсів;
- надійні та стабільні додатки на основі MAUI;
- зменшення часу розробки та спрощення управління проектом.

У представлених дослідженнях проаналізовано фреймворки Flutter, React Native і MAUI для кросплатформної розробки мобільних додатків.

Можемо зробити висновки, що Flutter демонструє високу продуктивність, багато можливостей кастомізації та якісну документацію.

React Native видідяється гнучкістю у використанні ресурсів і широкою спільнотою розробників. Він показав себе як хороший інструмент для мобільної розробки, однак у порівнянні з Flutter має певні обмеження щодо продуктивності.

MAUI серед переваг має сучасність, перспективність і високу продуктивність. Завдяки інтеграції з екосистемою Microsoft і використанню патерну MVVM, MAUI підходить для масштабних проектів.

Загалом, кожен із розглянутих фреймворків має свої сильні сторони.

2.2 Аналіз та огляд сайтів та форумів з відкритих джерел

Процес розробки на Flutter дуже швидкий і ефективний, система з віджетами дозволяє реалізувати будь-яку задумку без великих зусиль, реалізація анімацій теж дуже швидка завдяки вбудованим у фреймворк механізмам. Тобто якщо потрібно реалізувати нестандартні інтерфейси і анімації Flutter дозволяє це зробити у рекордні строки. З іншого боку така легкість може спричинити проблеми якщо

розробник не достатньо досвідчений та до кінця не розуміє принципів чистого коду та архітектури [5].

Flutter є сучаснішим за фреймворк React Native, і може похвалитися вищою продуктивністю, можливістю налаштування та кращою документацією [6]. Було встановлено, що найвищу якість продукту має нативний підхід, однак і найбільшу вартість розробки, оскільки вимагає утримання двох окремих команд розробки.

Гібридний застосунок має меншу вартість розробки, однак і нижчу якість продукту. Найоптимальнішим варіантом виявився кросплатформенний застосунок, який поєднує в собі сильні сторони обох підходів, а саме порівняно низьку вартість розробки та високу якість розробленого рішення [7]. В свою чергу, React Native продовжує випереджати Flutter за наступними параметрами:

- поточна кількість розробників, які здатні розробляти програму цією технологією більша, ніж на Flutter; Використання більш поширеної мови програмування (JavaScript замість Dart);
- деякі частини логіки можна перевикористовувати у вебдодатках, які написані за допомогою бібліотеки React.js. Все ще більше вакансій на ринку, ніж у Flutter [9].

У свою чергу MAUI на ринку з 2022 року і це робить його однією з найсучасніших технологій, і він вже отримав визнання серед розробників. Кожен MAUI-контрол обробляється через Handlers, які напряму викликають відповідні нативні API. Це дозволяє розробникам працювати з платформою на «рідному» рівні, чого не завжди можна досягти в React Native чи Flutter.

Завдяки роботі кожного елемента з нативним API MAUI забезпечує високу продуктивність застосунків. У порівнянні, React Native використовує JavaScript-бридж для зв’язку з нативними модулями, що може створювати затримки. Flutter хоч і має високу продуктивність, але залежить від власного движка рендерингу, який може займати більше ресурсів.

3 ПОСТАНОВКА ЗАДАЧІ

3.1 Методологія експерименту

Flutter, React Native та MAUI обрані для проведення експерименту, оскільки вони є одними з найбільш популярних та використовуваних крос-платформних рішень. Експеримент буде базуватися на створенні прототипів застосунків для різних технологій та їх подальшому тестуванні. Об'єктом експерименту є розробка мобільного застосунку з простим UI та з клієнт-серверною взаємодією за допомогою кожного з обраних фреймворків.

Етапи проведення експерименту:

- створення базового проєкту для кожного фреймворку, що включає в себе роботу з сервером та інтерфейс застосунку;
- розробка однакового набору функціональності для кожного фреймворку, зокрема робота з графічним інтерфейсом, взаємодія з користувачем, робота з сервером тощо;
- фіксація часу, необхідного для виконання кожного етапу розробки, включаючи стартову конфігурацію проекту, вибір імплементації, тестування та налагодження;
- вимірювання частоти виникнення помилок, продуктивності та виконання функцій у реальному часі;
- розрахунок вартості проекту для кожного фреймворку, враховуючи витрати на розробку, тестування та підтримку.

Проаналізувавши методи тестування сучасних застосунків [10], можна здійснити ґрунтовну оцінку результатів мобільних застосунків шляхом застосування таких методів аналізу:

- використання статистичних методів для порівняння отриманих результатів та визначення статистично значущих різниць між фреймворками;
- зіставлення показників ефективності кожного фреймворку для визначення того, який із них найбільш оптимальний для даного проекту;

- формулювання загального висновку щодо ефективності та придатності кожного фреймворку для розробки крос-платформних мобільних застосунків.

Використання цієї методології дозволить систематично та об'єктивно провести експеримент та забезпечити надійні результати для подальшого аналізу та порівняння крос-платформних фреймворків.

3.2 Специфікація програмного забезпечення

Основні вимоги до програмного забезпечення:

- три мобільні застосунки, які написані на останніх стабільних версіях вибраних фреймворків;
- використання засобів для точного вимірювання часу, витраченого на розробку та виконання функцій для кожного фреймворку;
- реалізація функціоналу для вимірювання продуктивності, такого як частота виникнення помилок та швидкість виконання;
- забезпечення можливості автоматичного збору та обробки статистичних даних під час експерименту для подальшого аналізу.

Вимоги до експериментального забезпечення:

- забезпечення наявності зручного інтерфейсу для створення та конфігурації проектів для кожного фреймворку;
- можливість інтеграції з популярними системами контролю версій для забезпечення зручного ведення розробки.

Вимоги до інтерфейсу застосунків:

- розробка інтуїтивного інтерфейсу для зручного користування програмним забезпеченням;
- наявність засобів для візуалізації отриманих результатів експерименту для зручного аналізу;
- забезпечення можливості зберігання та вивантаження отриманих даних для подальшого використання;
- використання простих базових елементів інтерфейсу.

Варто звернути увагу на те, що розробка під iOS можлива тільки на macOS девайсі. У даному досліджені буде описана інсталяція тільки на macOS, для інших операційних систем неможливо встановити деякі компоненти, які відповідно надають можливість розробляти та запускати застосунки на iOS девайсах.

3.3 План-програма експерименту

План експерименту з дослідження кросплатформного програмного забезпечення складається з шести етапів.

На першому етапі буде налаштоване середовища розробки та інструментів для вимірювання часу та продуктивності.

На другому етапі створимо однаковий набір функціональності для кожного фреймворку: зручний графічний інтерфейс, взаємодію з користувачем та роботу з сервером.

На третьому етапі буде проведено автоматизовані тести. Вони необхідні для перевірки функціональності, а також для збору та аналізу статистичних даних про час розробки та виконання функцій.

На четвертому етапі буде виконане вимірювання, як часто виникають помилки, а також визначено швидкість виконання функцій.

На п’ятому етапі буде оформлено та представлено результати експерименту у графічному вигляді, а також підготовлено звіт з висновками.

На останньому етапі проведена оцінка ефективності та придатності кожного фреймворку і формулювання загальних висновків та рекомендацій.

Цей план-програма забезпечить послідовне проведення експерименту. Він гарантує об'єктивність та достовірність отриманих результатів для подальшого аналізу та порівняння крос-платформних фреймворків.

4 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ

4.1 Визначення методології теоретичних досліджень

Основні кроки методологічного підходу включають детальний аналіз наукових публікацій, статей та документації кожного з фреймворків для більш глибокого розуміння їхньою специфіки та можливих обмежень. Також проводиться огляд існуючих концепцій кросплатформеного програмного забезпечення з метою визначення основних архітектурних принципів та рішень. Дослідження охопить оцінку якості роботи, масштабованості та продуктивності системи архітектур. Також зможемо врахувати можливості стабільності у більшості критичних ситуацій. Окрім цього буде проведено аналіз потенційних обмежень та переваг роботи з іншими мовами програмування і інструментами для розробки.

Методологія також буде враховувати дослідження сучасних тенденцій у галузі кросплатформної розробки, а також вивчення найкращих практик. Важливою деталлю є практичне застосування отриманих теоретичних знань на практиці. Це допоможе оцінити їх вплив на реальні кросплатформні проєкти програмного забезпечення.

У ході проведення дослідження можуть виникнути деякі обмеження. Дуже важливо враховувати їх при аналізі та розгляді отриманих результатів. Нижче наведені потенційні обмеження:

- детальне дослідження всіх можливих архітектурних аспектів є важким завданням через обмежені ресурси та обсяг роботи;
- швидкий розвиток технологій може призвести до змін функцій та характеристик фреймворків;
- дослідження орієнтоване на конкретні сценарії використання та типи мобільних застосунків; результати можуть бути залежними від контексту і можуть не враховувати всі можливі варіанти використання.

4.2 Математичне моделювання

Щоб ефективно порівнянти крос-платформні фреймворків Flutter, React Native та MAUI, а також визначити їхню ефективність, буде використано метод формалізації на основі наступних критеріїв.

- час розробки;
- продуктивність;
- вартість проекту;
- якість архітектури.

Визначимо час, необхідний для розробки конкретних функцій у кожному фреймворку та відобразимо його у вигляді математичних формул 4.1:

$$T = T_F + T_R + T_M \tag{4.1}$$

де T_F – час розробки для Flutter,

T_R – час розробки для React Native,

T_M – час розробки для MAUI.

Врахуємо частоту виникнення помилок та швидкість виконання функцій для кожного фреймворку та представимо їх у вигляді числових показників 4.2:

$$P = P_F + P_R + P_M \tag{4.2}$$

де P_F – продуктивність для Flutter,

P_R – продуктивність для React Native,

P_M – продуктивність для MAUI.

Оцінимо вартість розробки та підтримки проекту для кожного фреймворку і визначимо залежність від обсягу та складності проєкту за допомогою формули 4.3:

$$C = C_F + C_R + C_M \tag{4.3}$$

де C_F – вартість проекту для Flutter,

C_R – вартість проекту для React Native,

C_M – вартість проекту для MAUI.

Врахуємо якість архітектури, тобто її масштабованість, стабільність та продуктивність. Вимірюється від 0 до 1, де 1 - найвища якість. Формула 4.4 для оцінки якості архітектури:

$$Q = Q_F + Q_R + Q_M \tag{4.4}$$

де Q_F – якість архітектури для Flutter,

Q_R – якість архітектури для React Native,

Q_M – якість архітектури для MAUI.

Введемо коефіцієнти вагомості для кожного критерію. Тоді, загальний показник ефективності E визначений за формулою 4.5:

$$E = W_T \times T + W_P \times P + W_C \times C + W_Q \times Q \tag{4.5}$$

де W_T – для вагомості часу розробки;

T – час розробки;

W_P – для вагомості продуктивності;

P – продуктивність;

W_C – для вагомості вартості проекту;

C – вартість проекту;

W_Q – для вагомості якості архітектури;

Q – якість архітектури.

Цей підхід дозволяє створити математичну модель. Вона враховує основні моменти вибору крос-платформних фреймворків та дає можливість об'єктивно порівнювати їх на основі конкретних параметрів, які можна виміряти.

4.3 Дослідження найкращих практик у Flutter

Flutter надає можливість створення застосунків з використанням архітектурних патернів, таких як BLoC (Business Logic Component), MVVM (Model-View-ViewModel) та Redux.

Найбільш рекомендованим для крос-платформного розроблення є BLoC через такі переваги:

- Bloc легко поєднується з іншими бібліотеками Flutter;
- забезпечення чіткого розділення бізнес-логіки та інтерфейсу;
- можливість повторного використання коду;
- кожен стан, через який проходить Bloc, можна зафіксувати та проаналізувати;
- масштабованість для великих проєктів.

BLoC реалізує патерн «потоків» (streams), який дозволяє розділяти події та стани, що спрощує управління складними сценаріями. Ключовою особливістю є підтримка реактивного програмування, яке забезпечує автоматичне оновлення інтерфейсу у відповідь на зміни стану, як намальовано на рис. 4.1.

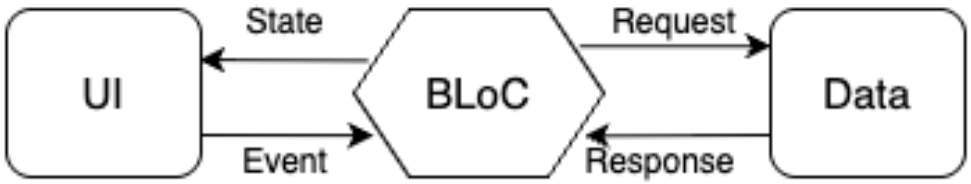

Рисунок 4.1 – Схема роботи BLoC (рисунок виконаний самостійно)

Події (Events) визначають, що має відбутися у додатку. Наприклад, користувач натискає кнопку, і це генерує подію IncrementCounter. Події повідомляють BLoC про необхідність виконати певну дію.

Стан відображає поточний статус додатку, наприклад: завантаження даних, успішне отримання даних, помилка. Кожна зміна стану оновлює інтерфейс користувача.

BLoC (Бізнес-логіка) - це центральний компонент, який отримує події, обробляє їх і виводить нові стани. Основна функція BLoC — метод mapEventToState, який визначає, як подія перетворюється у стан.

Потоки використовуються для передачі подій та станів між компонентами. Події надходять у BLoC через метод add(Event). Стан передається назад у UI через stream.

Такий підхід покращує продуктивність, знижує ризик помилок та полегшує підтримку проєкту.

Інтеграцію із RESTful API краще робити за допомогою бібліотеки Retrofit. Retrofit. Він має такі переваги:

- зручна підтримка HTTP-запитів (GET, POST, PUT, DELETE) через анотації;
- автоматична десеріалізація даних у моделі;
- висока гнучкість та розширюваність за рахунок інтерсепторів та конвертерів.

Retrofit побудований на бібліотеці Dio.

Анотації використовуються для опису методів API, наприклад @GET, @POST, @PUT.

У бібліотеці використовує пакет build_runner для створення клієнтського коду.

Також, він підтримує серіалізацію/десеріалізацію даних у форматі JSON через бібліотеки, такі як json_serializable.

Retrofit виграє над іншими підходами завдяки чіткій типізації, автоматизації обробки запитів і можливості роботи з великими наборами даних.

Flutter дозволяє використовувати Keys для оптимізації перемальовки компонентів. Наприклад, використання GlobalKey забезпечує доступ до стану віджету без необхідності повної перерисовки.

Flutter забезпечує єдиний кодовий базис для Android і iOS, що дозволяє дотримуватись рекомендацій Material Design і Cupertino. Вони забезпечують платформо-специфічний користувацький інтерфейс. Material Design створений для

забезпечення єдиного сучасного дизайну на пристроях Android, з акцентом на кольорові палітри, тіні та анімації. Він включає в себе готові компоненти, такі як FloatingActionButton, SnackBar, і BottomNavigationBar. Cupertino, у свою чергу, створений для пристроїв iOS, надаючи користувачам досвід, що відповідає Apple's Human Interface Guidelines. Компоненти, такі як CupertinoButton, CupertinoNavigationBar, та інші, дозволяють створювати інтерфейси, які виглядають і працюють як нативні iOS-додатки.

Ось так можна виміряти час відмальовки інтерфейсу для подальшого аналізу:

```
@override
  Widget build(BuildContext context) {
    final stopwatch = Stopwatch()..start();

    final app = MaterialApp.router(
      debugShowCheckedModeBanner: false,
      routeInformationParser: appAutoRouter.defaultRouteParser(),
      routeInformationProvider: appAutoRouter.routeInfoProvider(),
      routerDelegate: appAutoRouter.delegate(),
    );

    stopwatch.stop();
    log('Build time: ${stopwatch.elapsedMilliseconds} ms');
    return app;
  }
```

Для вимірювання часу виконання HTTP-запитів можна використовувати інтерсептори:

```
final dio = Dio();
dio.interceptors.add(InterceptorsWrapper(
  onRequest: (options, handler) {
    options.extra['startTime'] = DateTime.now();
    return handler.next(options);
  },
  onResponse: (response, handler) {
    final startTime = response.requestOptions.extra['startTime'];
    final duration = DateTime.now().difference(startTime);
    log('Request time: ${duration.inMilliseconds} ms');
    return handler.next(response);
  },
));
await dio.get('https://request/get');
```

Щоб визначити час відмальовки даних можна використати WidgetsBinding.instance.addPostFrameCallback для вимірювання часу від початку до кінця рендеру.

Також, був проведений аналіз та створений список найкращих практик Flutter:

- використання Material Design і Cupertino;
- уникання непотрібного setState();
- віддавання переваги одному віджету на файл;
- використання SizedBox замість контейнерів;
- утілізування потоків лише за необхідності;
- використання константних конструкторів.

4.4 Дослідження найкращих практик у React Native

React Native підтримує велику кількість архітектурних патернів, таких як Flux, Redux, та MVVM [11]. Найпоширенішим є Redux, який забезпечує централізоване управління станом та спрощує взаємодію між компонентами. Redux дозволяє зберігати стан у єдиному контейнері (store), що робить його ідеальним для складних програм (див. рис. 4.2).

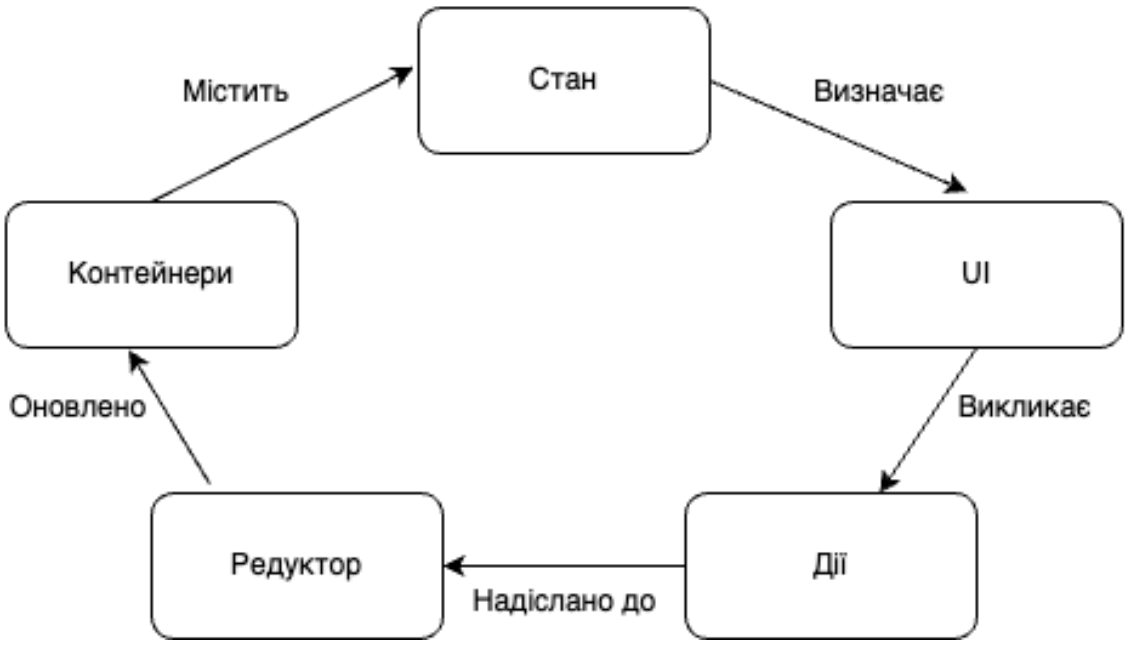

Рисунок 4.2 – Схема роботи Redux (рисунок виконаний самостійно)

Store (Сховище) – це глобальний об'єкт, який зберігає весь стан додатка. У Redux є тільки одне сховище для всієї програми.

Actions (Дії описують, що сталося в додатку. Це звичайні об'єкти з полями type (назва дії) і, за потреби, payload (додаткові дані).

Reducers (Ред'юсери) – це функції, які приймають поточний стан і дію, обробляють її і повертають новий стан.

Dispatch - метод, що відправляє дії до сховища.

Selectors (Селектори) – це функції, які дозволяють отримати потрібні дані зі стану.

Основні переваги Redux:

- централізоване сховище даних;
- передбачуваність стану завдяки чітко визначеним правилам (reducers);
- зручність тестування логіки додатку.

Для інтеграції з API у React Native зазвичай використовується Axios, який підтримує всі стандартні HTTP-запити (GET, POST, PUT, DELETE). Крім цього, Axios має такі переваги:

- легка настройка інтерсепторів для обробки запитів та відповідей;
- вбудована підтримка таймаутів та обробки помилок;
- можливість роботи з токенами для авторизації.

Axios інтегрується з Redux для управління станом і дозволяє оптимізувати мережеву взаємодію через кешування запитів.

Axios у React Native працює як посередник між додатком і сервером для обміну даними через HTTP-протокол. Коли ви хочете надіслати запит, Axios спершу формує конфігурацію, яка включає всі необхідні параметри, як-от адресу, метод запиту, заголовки чи дані. Ця конфігурація проходить через спеціальні перехоплювачі, які можуть змінити або доповнити її, наприклад, додати токен аутентифікації.

Після цього Axios використовує низькорівневі інструменти, такі як XMLHttpRequest у браузері або http у Node.js, для реального виконання запиту. Відправлений запит надходить на сервер, який обробляє його і повертає відповідь.

Ця відповідь теж проходить через перехоплювачі, де можна перевірити статус або виділити корисні дані, після чого повертається у вигляді промісу. Якщо виникає помилка, її можна обробити залежно від типу: це може бути таймаут, проблема з мережею або відповідь сервера із статусом помилки.

React Native використовує Virtual DOM для мінімізації відмальовки компонентів. Динамічне оновлення інтерфейсу забезпечується механізмом дифінгу, який знаходить відмінності між поточним та попереднім станом і оновлює лише необхідні частини.

React Native підтримує використання платформо-орієнтованих компонентів для створення інтуїтивних зрозумілих інтерфейсів. Наприклад:

- для Android використання компонентів Material Design, таких як TouchableNativeFeedback або BottomNavigation;
- для iOS використання компонентів Cupertino, таких як ActionSheetIOS або DatePickerIOS.

Ось так можна виміряти час відмальовки інтерфейсу для подальшого аналізу:

```
const start = performance.now();
AppRegistry.registerComponent('Main', () => {
  const end = performance.now();
  console.log(`Build time: ${end - start} ms`);
  return App;
});
```

Для вимірювання часу запитів у React Native можна застосовувати Axios із використанням інтерсепторів:

```
const instance = axios.create();

instance.interceptors.request.use((config) => {
  config.metadata = { startTime: new Date() };
  return config;
});
instance.interceptors.response.use((response) => {
  const duration = new Date() - response.config.metadata.startTime;
  console.log(`Request duration: ${duration} ms`);
  return response;
});
instance.get('https://request/get');
```

Вимірювання часу рендерингу здійснюється через requestAnimationFrame:

```
useEffect(() => {
  const start = performance.now();
  requestAnimationFrame(() => {
    const end = performance.now();
    console.log(`Render time: ${end - start} ms`);
});
```

React Native також надає інструменти для моніторингу продуктивності, такі як Flipper. Використання мемоізації (React.memo) та оптимізація списків через FlatList або SectionList дозволяють значно покращити продуктивність.

Також, був проведений аналіз та створений список найкращих практик React Native:

- використання React Native CLI;
- тримати бізнес-логіку окремо від коду інтерфейсу;
- використання PureComponent і memo;
- уникання використання індексу як ключової властивості;
- використання платформо-орієнтованих компонентів для створення інтуїтивних зрозумілих інтерфейсів;
- економне використання вбудованих стилів;
- використання вбудованого драйвера, коли це можливо.

4.5 Дослідження найкращих практик у MAUI

MAUI дозволяє використовувати декілька архітектурних підходів, таких як MVVM (Model-View-ViewModel), MVC (Model-View-Controller) і MVP (Model-View-Presenter) [12]. Найбільш поширеним для C# фреймворків є MVVM завдяки його простоті та можливості розподіляти обов’язки між класами.

Model (Модель) відповідає за роботу з даними. Може включати класи для роботи з API, базами даних або бізнес-логікою. Модель не залежить від View або ViewModel.

View (Представлення) відображає дані користувачеві. У MAUI це сторінки та елементи інтерфейсу (XAML + Code-Behind). Має мінімум логіки, яка стосується виключно роботи з UI.

ViewModel проміжний шар між Model і View. Містить логіку, яка забезпечує представлення даних для View. Реалізує властивості та команди (Commands), що зв’язують дії користувача зі станом програми (див. рис. 4.3).

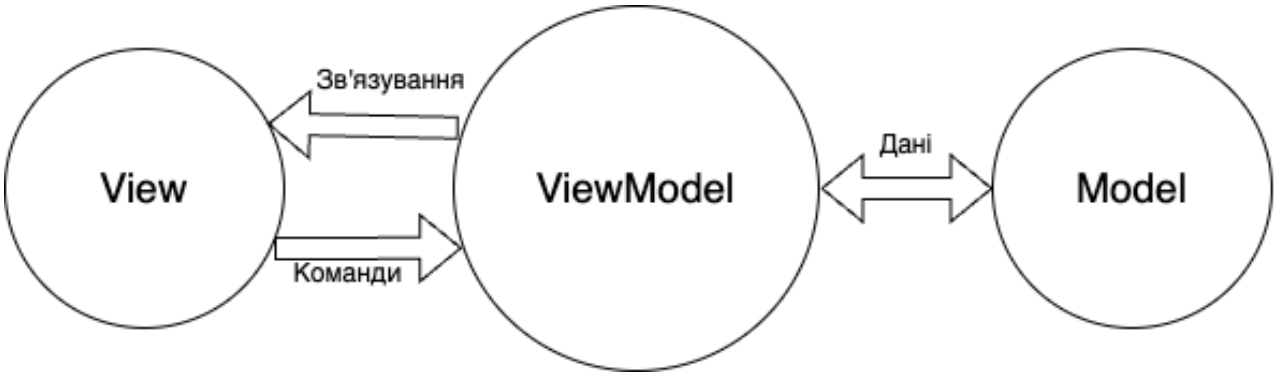

Рисунок 4.3 - Схема роботи MAUI (рисунок виконаний самостійно)

Основні переваги MVVM:

- чітке розділення логіки та інтерфейсу;
- підтримка повторного використання коду між платформами;
- спрощене тестування бізнес-логіки.

Для інтеграції з API у MAUI у рекомендовано використовувати бібліотеку HttpClient. Вона надає такі переваги:

- HttpClient працює на всіх підтримуваних платформах MAUI (iOS, Android, Windows, macOS);
- HttpClient дозволяє легко налаштовувати обробники запитів (наприклад, для логування чи аутентифікації);
- підтримка асинхронних запитів.

HttpClient значно спрощує роботу з API завдяки оптимальному підходу до налаштування запитів. Він забезпечує стабільність, гнучкість і найкращу продуктивність для роботи з мережевими запитами. HttpClient відкриває HTTP-з'єднання до сервера через вказаний URL. Він використовує HttpMessageHandler для низькорівневої обробки запитів. Формування HTTP-запиту відбувається через встановлення HTTP-методу (GET, POST тощо), далі за необхідністю додаємо заголовки (Headers), параметри запиту (Query Parameters) та тіло запиту (Content). Після надсилання запиту HttpClient отримує HTTP-статус код (200, 404, 500 тощо).

Далі читає тіло відповіді у вигляді string, byte[], Stream або десеріалізує у JSON/XML.

У MAUI підтримка динамічних оновлень інтерфейсу реалізована через механізм Data Binding. Це дозволяє автоматично синхронізувати дані між моделлю та інтерфейсом.

MAUI дозволяє створювати єдиний інтерфейс для обох платформ, використовуючи такі елементи, як Grid, StackLayout, і CollectionView. Для платформи-специфічного дизайну MAUI підтримує використання Мультитаргетинг (Platforms/ + #if PLATFORM. Для Android є можливість реалізації Material Design через використання MaterialComponents. Для iOS підтримка Apple Human Interface Guidelines за допомогою імпорту UIKit.

Час білду проекту зазвичай залежить від середовища, а не від самого коду, тому для вимірювання можна використовувати вбудовані інструменти (наприклад, у Visual Studio).

Але можна виміряти час запуску застосунку.

```
private static readonly Stopwatch _stopwatch = Stopwatch.StartNew();
public App()
{
  InitializeComponent();
}
protected override void OnStart()
{
  base.OnStart();
  _stopwatch.Stop();
  Debug.WriteLine($" Час запуску: {_stopwatch.ElapsedMilliseconds} мс");
}
```

Для вимірювання часу запитів у MAUI можна використовувати Stopwatch:

```
var stopwatch = Stopwatch.StartNew();

var response = await httpClient.GetAsync'https://request/get');
stopwatch.Stop();

Console.WriteLine($"Request duration: {stopwatch.ElapsedMilliseconds} ms");
```

Для вимірювання часу відмальовки даних можна виміряти час між отриманням даних та їх відображенням на екрані також за допомогою Stopwatch:

```
public partial class MainPage : ContentPage
{
  private readonly Stopwatch _stopwatch = Stopwatch.StartNew();

  public MainPage()
  {
    InitializeComponent();
    BindingContext = _viewModel;
  }

  protected override void OnAppearing()
  {
    base.OnAppearing();
    _stopwatch.Stop();
    Debug.WriteLine($"□          Час          завантаження          MainPage:
{_stopwatch.ElapsedMilliseconds} мс");
  }
}
```

Також, був проведений аналіз та створений список найкращих практик MAUI:

- використання x:Bind замість Binding;
- виявлення помилки XAML на рівні компіляції;
- очищення ресурсів вручну (Dispose());
- уникання залежностей від фреймворків;
- перевага у використані команди, а не обробники подій;
- використання CollectionView замість Lists.

5 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ

5.1 Практичне дослідження Flutter

Перед початком розробки мобільного застосунку на Flutter необхідно встановити на ноутбук або комп’ютер усі потрібні компоненти. Почнемо встановлення з Flutter SDK, посилання можна знайти в офіційній документації [9]. Наступним кроком встановлюємо XCode (доступний тільки для операційних систем macOS) для розробки iOS застосунків. Наступним кроком є встановлення CocoaPods також для розробки на платформі iOS. Далі необхідно встановити усі компоненти для розробки на платформі aOS, для цього встановлюємо Android Studio. Після її встановлення потрібно відкрити SDK Manager та обрати усі компоненти, які потрібні для Flutter. Повний список також доступний в офіційній документації. Багато розробників забувають про наступний крок – це згода на ліцензію Android. Це є не менш важливим етапом, бо без цього не вийде запустити застосунок на aOS девайс. Щоб перевірити встановлення усіх компонентів достастньо виконати команду flutter doctor в терміналі. Ця команда покаже де і які компоненти встановлені не правильно, що є дуже зручно. Також для зручності була встановлена Visual Studio Code та розширення Flutter. Можна відмітити, що налаштовувати все було достатньо просто, бо є дуже зрозуміла та детальна інструкція у документації.

Було розроблено клієнт-серверний застосунок з урахуванням найкращих практик, які були визначені у теоретичному розділі дослідження. Застосунок отримує дані студентів з відкритого API, відмальовує дані на UI та рахує скільки часу було витрачено на відпрацювання запиту з серверу, запуску самого застосунку та вивід даних на телефон. Під час розробки виникали мало помилок, які швидко вирішувались завдяки документації та великій спільноті розробників. На розробку було витрачено тиждень робочого часу.

Результати:

- запит відпрацював за 309 мілі секунд;
- запуск застосунку зайняв 1207 мілі секунд;
- відмалювання UI відбулося за 14 мілі секунд.

Користувацький інтерфейс зображено на рис. 5.1.

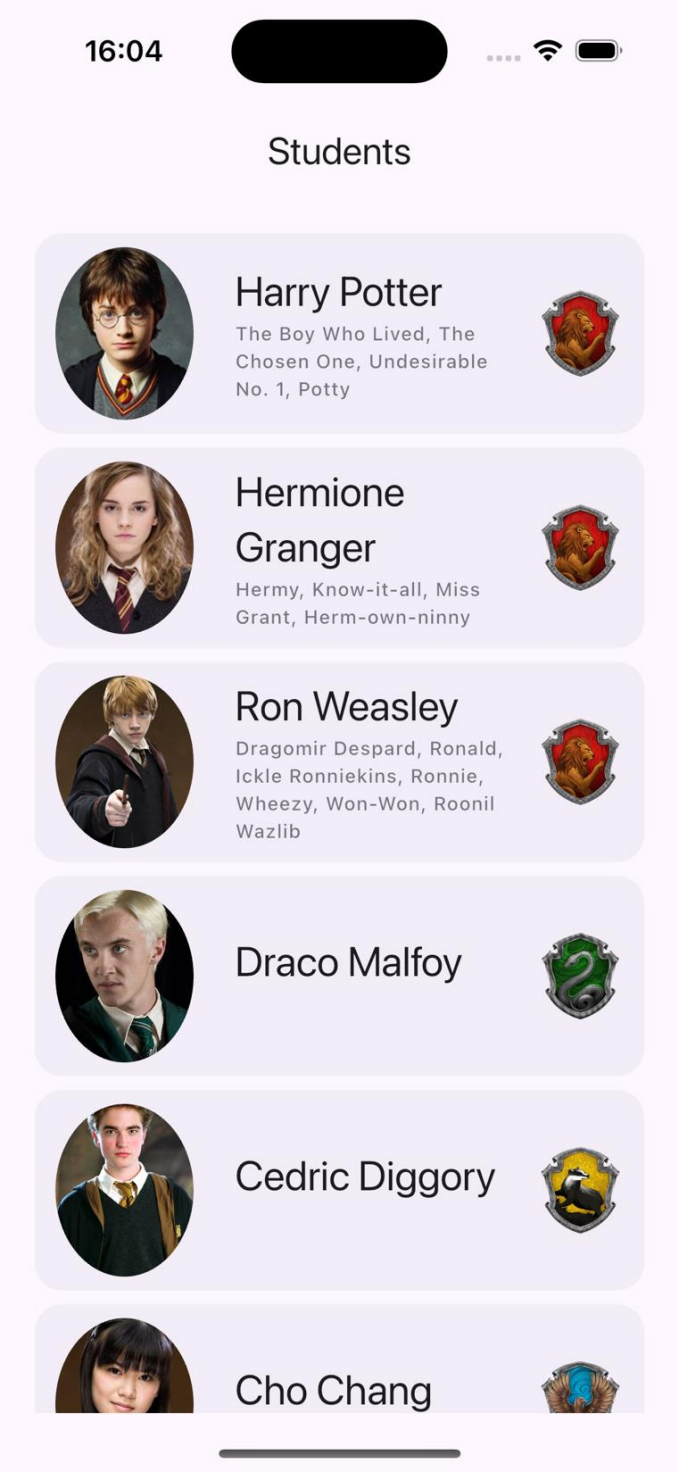

Рисунок 5.1 - Користувацький інтерфейс Flutter застосунку (рисунок виконаний самостійно)

Варто відзначити, що для створення користувацького інтерфейсу було використано тільки вбудовані можливості стилів фреймворку. Тобто не було

прописано жодних кольорів або шрифтів. Також UI вийшов мінімалістичним та зручним.

Flutter дуже зручний для використання. Щоб змінити або обрати девайс необхідно просто вибрати потрібний девайс у випадаючому списку внизу екрану. Для запуску застосунку достатньо натиснути на зелену кнопку запуску зверху зліва.

Під час розробки було знайдені деякі недоліки:

- великий розмір Flutter застосунків;
- Flutter не підтримує деякі старі версії Android та iOS, що може бути проблемою для деяких юзерів;
- при великих обчислень Flutter може показувати меншу продуктивність ніж нативні апки;
- мова dart має один потік, тому під час відтворення музики або ауді застосунок може «підвисати»;
- достатньо складно інтегрувати нативний код.

На кінець, варто відмітити документацію. Вона має чітку структуру та детальний опис всього, що може знадобитись. Також документація містить приклади коду, який можна запустити на сторінці та перевірити у реальному часі. Команда Google кожен тиждень публікує огляди на нові та корисні віджети, а також як їх використати. Тож документації можна поставити найвищу оцінку.

5.2 Практичне дослідження React Native

Перед тим, як розпочати розробку мобільного застосунку на React Native варто інсталювати на ноутбук або комп’ютер усі потрібні компоненти. Рекомендується встановити brew для швидкого встановлення усіх компонентів. Почнемо встановлення з npm, npm – це менеджер пакетів, які входять у склад Node.js. Далі встановлюємо XCode (доступний тільки для операційних систем macOS) для розробки iOS застосунків. Наступним кроком є встановлення CocoaPods також для розробки на платформі iOS. Далі необхідно встановити усі компоненти для розробки на платформі aOS, для цього встановлюємо Android

Studio. Після її встановлення потрібно відкрити SDK Manager та обрати усі компоненти, які потрібні. Нажаль на офіційній документації [3] немає повного списку, але є багато відео та інформації про це від спільноти розробників у мережі інтернет. Далі як во Flutter далі необхідно дати згоду на ліцензію Android. Наступним кроком необхідно встановити Flipper для відлатки застосунків, бо у React Native помилки у консолі не з'являються. З встановкою Flipper виникли проблеми, бо офіційна версія вважається підозрілою і не відкривається на macOS. Перевірити встановлення усіх компонентів ніяк не можна. Також для зручності була встановлена Visual Studio Code та розширення ReactNative. Відмічу, що під час налаштовування була невелика кількість проблем, але на усі запитання є відповіді у мережі інтернет.

Реалізовано клієнт-серверний застосунок відповідно до найкращих практик, визначених у теоретичному розділі дослідження. Застосунок так само отримує дані студентів з відкритого API, показує дані на телефон та вираховує скільки часу було витрачено на запуску самого застосунку, відпрацювання запиту з серверу та вивід даних на UI. Під час розробки виникало достатньо помилок, які без Flipper неможливо знайти. Завдяки великій спільноті розробників вдалося вирішити усі проблеми. На розробку було витрачено два тижні робочого часу.

Результати:

- запуск застосунку зайняв 25803 мілі секунд;
- запит відпрацював за 333 мілі секунд;
- відмалювання UI відбулося за 32 мілі секунд.

Одразу видно, що Flutter застосунок працює швидше.

Привертає увагу, що ReactNative не дуже зручний для використання. Щоб запустити проект, спочатку потрібно запустити npm host у консолі, потім написати, на якій платформі відкрити застосунок. Якщо на цьому моменті щось піде не так, то застосунок просто не відкриється без помилок. Якщо доступні два девайси для запуску застосунку, то воно запуститься на будь-якому без можливості вибору.

Необхідно підкреслити, що в ReactNative немає такої вбудованої теми, як у Flutter, тому усі стилі довелося прописати. Але UI вийшов непоганим та зручним.

Користувацький інтерфейс зображено на рис. 5.2.

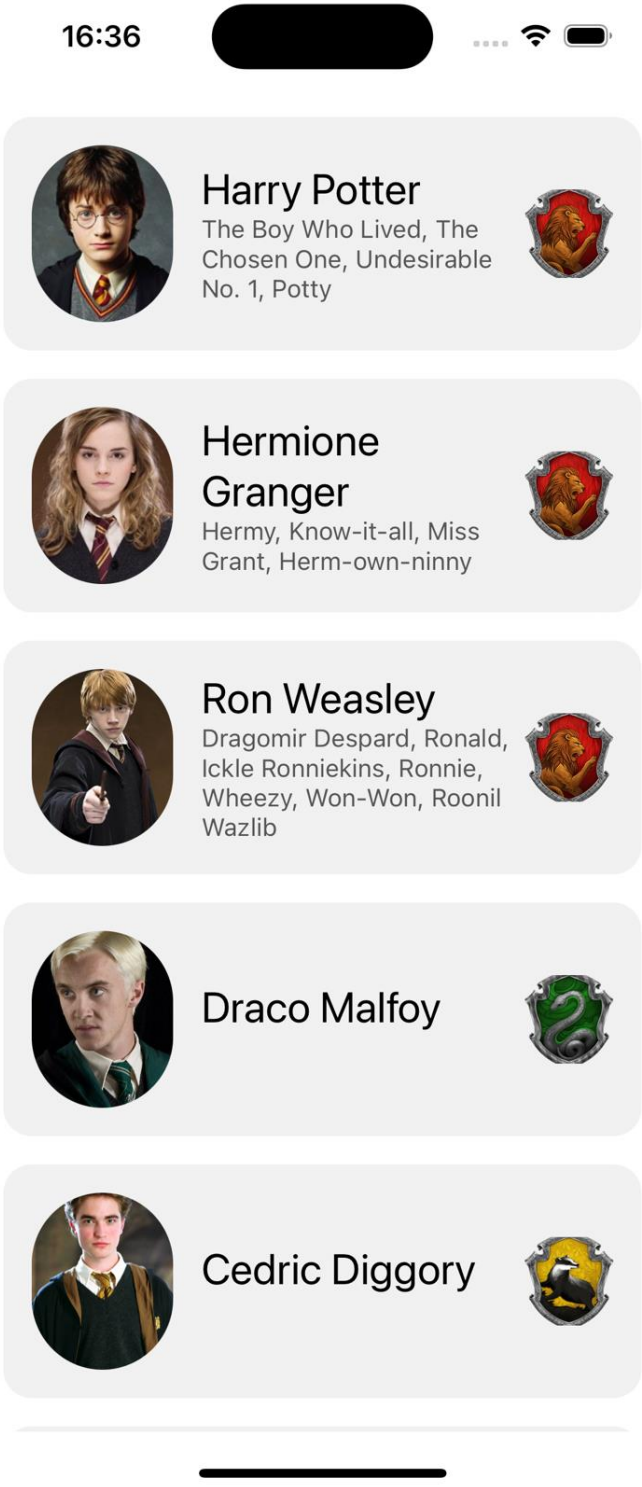

Рисунок 5.2 - Користувацький інтерфейс React Native застосунку (рисунок виконаний самостійно)

Перевагою у використані є те, що було написано набагато менше коду для такого самого функціоналу, ніж у Flutter.

Крім недоліків, описаних вище, було знайдені ще такі недоліки:

- React Native показує меншу продуктивність ніж нативні апки, особливо це помітно під час важких анімацій та високонавантажених обчисленнях;
- великий розмір React Native застосунків;
- перехід між версіями React Native часто проблемний через підтримку бібліотек та зворотну сумісність;
- React Native працює в одному потоці Java Script, що призводить до багатьох обмежень;
- деякі нативні можливості обмежені або взагалі відсутні;
- не всі npm-бібліотеки стабільно працюють у React Native.

Врешті-решт, проаналізуємо документацію. Вона має чітку структуру та детальний опис та картинки всього, що може знадобитись. Нажаль, у документації мало прикладів коду. Велика кількість відео на різних платформах від спільноти React Native розробників, які дійсно допомагають.

5.3 Практичне дослідження MAUI

Встановлення усіх компонентів для розробки на MAUI почалося з встановлення Visual Studio Code та розширення .NET MAUI. Також потрібно встановити розширення .NET Install Tool, воно допомагає легко та швидко завантажити усі потрібні компоненти для розробки на платформі .NET. Ці розширення показують список з того, що потрібно встановити та посилання, натискаючи на які починається завантаження. Таким чином було встановлено XCode (доступний тільки для операційних систем macOS) для розробки iOS застосунків, CocoaPods також для розробки на платформі iOS, Android Studio та .NET SDK. Також список показує, які компоненти були встановлені, а які – ні. Відмічу, що налаштовування було легким без великих проблем, але документація доволі стисла та мало інформації про усі налаштування у мережі інтернет.

Наступним кроком, було розроблено клієнт-серверний застосунок відповідно до найкращих практик, які були визначені у попередніх розділах дослідження. Застосунок працює так само, як і два попередні. Також він рахує скільки часу

витрачено на запуску застосунку, відпрацювання запиту з серверу та відмальовки даних на телефон. Під час розробки виникали помилки, які повільно вирішувались через недостатню документації. На розробку було витрачено півтора тижні робочого часу.

Користувацький інтерфейс зображено на рис. 5.3.

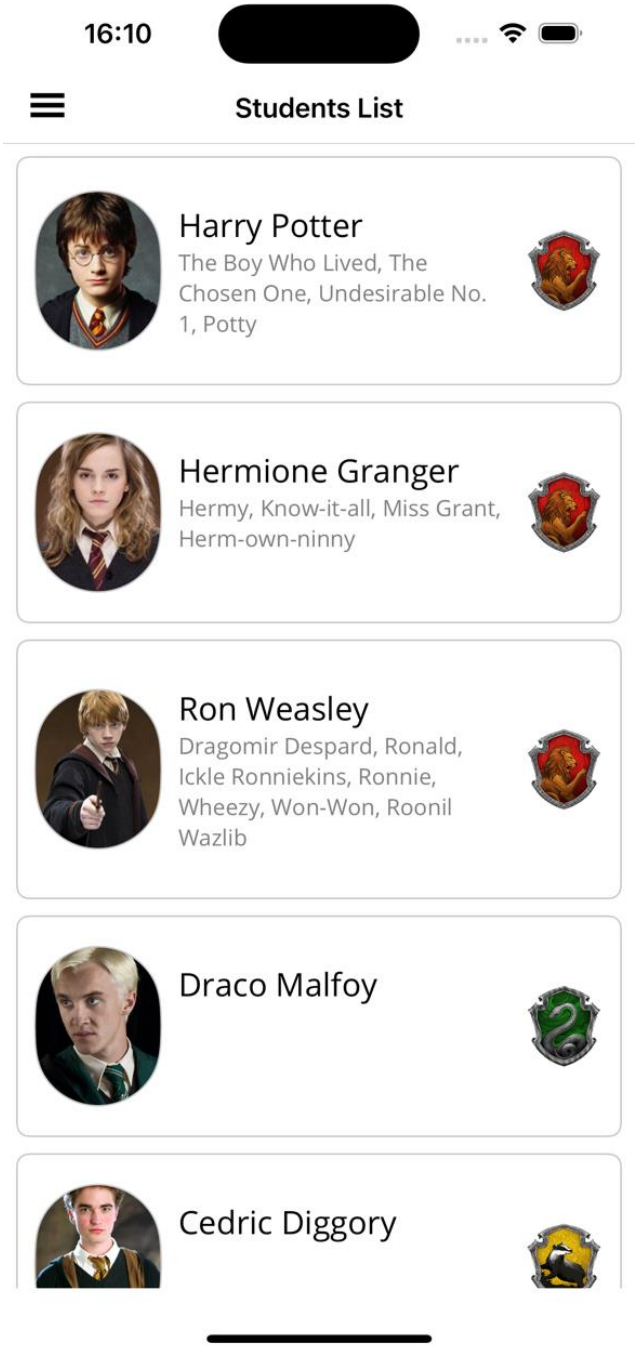

Рисунок 5.3 - Користувацький інтерфейс MAUI застосунку (рисунок виконаний самостійно)

Результати:

- запуск застосунку зайняв 888 мілі секунд;
- запит відпрацював за 262 мілі секунд;
- відмалювання UI відбулося за 33 мілі секунд.

Необхідно виділити, що MAUI також немає такої вбудованої теми, як у Flutter, тому усі стилі довелося прописати. Але UI вийшов гарним.

MAUI не поступається Flutter по зручності використання. Щоб змінити або обрати девайс необхідно просто натиснути на кнопку з назвою девайса внизу екрану. Для запуску застосунку достатньо натиснути на зелену кнопку запуску зверху зліва. Елиним недоліком у використані є те, що кнопка оновити застосунок не оновлює наявний білд, а зупиняє застосунок і запускає знову

Під час розробки було знайдені такі недоліки:

- MAUI нова технологія, тому може бути нестабільним;
- документація не ідеальна, спільнота набагато менша, ніж у Flutter та ReactNative;
- MAUI показує меншу продуктивність ніж нативні апки;
- великий розмір MAUI застосунків;
- не всі .NET-бібліотеки працюють коректно в MAUI, особливо ті, що залежать від специфічних платформ.

На відміну від React Native та Flutter, .NET MAUI надає можливість створювати декілька потоків, що є дуже важливим моментом.

На кінець, розглянемо документацію. Вона має чітку структуру та опис деяких елементів. Нажаль, у документації мало прикладів коду.

5.4 Загальне порівняння

Створимо графік порівняння крос-платформних програмних рішень для створення сучасних мобільних застосунків за критеріями, описаних в математичній моделі.

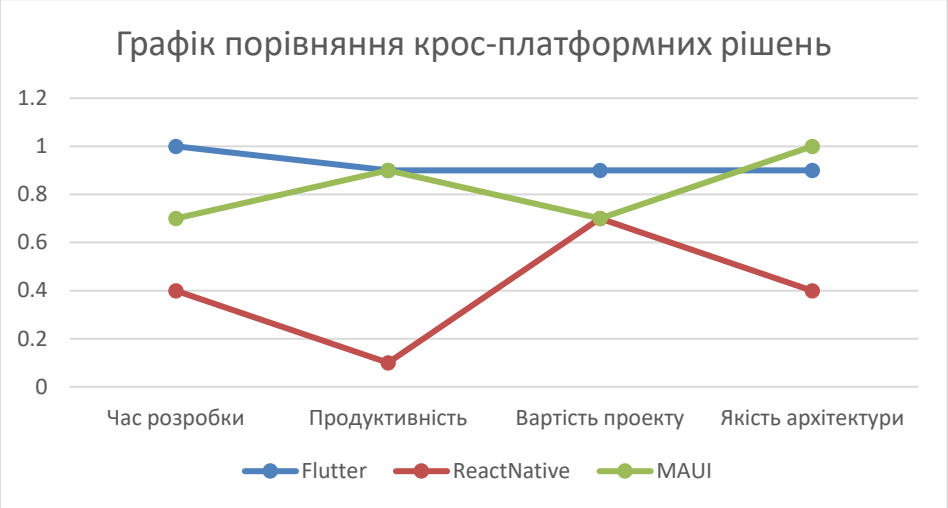

Рис. 5.1 – Графік порівняння крос-платформних програмних рішень (рисунок виконаний самостійно)

Створимо порівняльну таблицю трьох фреймворків на основі інформації попереднього розділу (див. табл. 5.1).

Таблиця 5.1 – Порівняльна таблиця

Критерій	Flutter	React Native	MAUI
Простота встановлення компонентів (від 1 до 10)	9/10	5/10	8/10
Частота виникнення помилок (від 1 до 10)	2	4	4
Час розробки (в годинах)	42	80	52
Час відпрацювання запиту (в мілі секундах)	309	333	262
Час запуску застосунку (в мілі секундах)	1207	25803	888
Час рендерінгу (зв мілі секундах)	14	32	33
Якість інтерфейсу (від 1 до 10)	10	7	9
Зручність використання (від 1 до 10)	10	6	10
Можливість багато поточності	Немає	Немає	Є
Якість документації (від 1 до 10)	10	8	5
Наявність інформації в мережі інтернет від спільноти розробників	9	10	5

З даних, які наведені, у таблиці та графіку видно, що React Native має найгірші показники. Цей фреймворк показав себе як не зручний та повільний порівняно з іншими фреймворками.

Flutter має дещо кращі показники у часі запуску застосунку та рендерінгу ніж MAUI, а також має набагато кращу документацію та більшу спільноту розробників. MAUI у свою чергу має можливість створення багато потоків, що є великою перевагою серед фреймворків. Можемо сказати, що якщо MAUI буде мати якіснішу документацію, то він може стати найкращим варіантом серед крос-платформних програмних рішень для створення сучасних мобільних застосунків.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи магістра було проаналізовано популярні крос-платформні фреймворки, ідентифікувано їхні переваги та недоліки. Flutter, React Native та MAUI є трьома найпопулярнішими фреймворками для розробки крос-платформних застосунків, кожен з яких має свої сильні сторони.

Усі три фреймворки забезпечують можливість створення єдиного коду для різних платформ, що значно знижує витрати на розробку та підтримку застосунків. Flutter виділяється завдяки власному механізму рендерингу і можливості точного контролю над UI. React Native забезпечує швидку розробку через використання JavaScript і великої кількості готових бібліотек.

Серед переваг Flutter є те, що він покладається на бібліотеки-плагіни для доступу до рідних функцій платформ. React Native дозволяє легко інтегрувати рідний код через Native Modules. MAUI забезпечує найбільш тісну інтеграцію з платформами завдяки використанню рідних API.

Якщо порівнювати користувацький інтерфейс, то Flutter пропонує єдиний підхід до UI через бібліотеки Material та Cupertino. Вони забезпечуючи однаковий вигляд на всіх платформах. React Native використовує рідні компоненти, що гарантує платформно-специфічний дизайн. MAUI підтримує як рідні компоненти, так і крос-платформні елементи.

Кожна з технологій має свої плюси та мінуси для розробки кросплатформних мобільних застосунків. React Native дещо поступається у зручності та швидкості іншим фреймворкам. MAUI та Flutter здатні вирішити поточні проблеми в повній мірі.

Дослідження відкриває можливості для подальших наукових дослідженнях у напрямку оптимізації кросплатформних рішень, а також вивчення їхнього впливу на розробку великих та складних мобільних проектів. Також, додаткові дослідження можуть розглядати нові фреймворки, що з'являться на ринку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Oleksandr Bezsmertnyi, Nataliia Golian, Vira Golian, Iryna Afanasieva. Behavior driven development approach in the modern quality control process. IEEE. 2020. С. 217-220.

2. Ібадов Д., Афанасьєва І. Crossplatform C++ library for multilayer perceptron learning. ScienceRise. 2016. № 2 (22). С. 19–25.

3. React Native. Overview: вебсайт. URL: https://reactnative.dev/community/overview (дата звернення: 15.11.2024).

4. Осташко Є. В. Дослідження методів створення сервісно-орієнтованих та крос-платформених додатків за допомогою Flutter з серверною частиною на Firebase, Харків, 2023. – 86 с. URL: https://openarchive.nure.ua/items/626371df-0e30-4a74-a857-0b6feda408f1 (дата звернення: 10.10.2024).

5. Попова Є. С. Дослідження ефективності фреймворку Flutter при розробці високо динамічних мобільних додатків, Харків, 2022. – 115 с. URL: https://ir.nmu.org.ua/bitstream/handle/123456789/162383/122M-21-1_диплом_Попова Є.С.pdf?sequence=1&isAllowed=y (дата звернення: 10.10.2024).

6. Куденко П. Р. Дослідження ефективності фреймворку React Native при розробці мобільних додатків, Дніпро, 2022. – 119 с. URL: https://ir.nmu.org.ua/bitstream/handle/123456789/162381/122M-21-1Кунденко.pdf?sequence=1&isAllowed=y (дата звернення: 10.10.2024).

7. Ситник Р. С. Розробка програмного забезпечення для дослідження ефективності засобу Flutter при розробці мобільних додатків, Дніпро, 2020. – 105 с. URL:https://ir.nmu.org.ua/bitstream/handle/123456789/157541/sytnyk_121m_19_1_diploma_last.pdf?sequence=1&isAllowed=y (дата звернення: 10.10.2024).

8. Глива В. А. Дослідження можливостей .NET MAUI в обробці даних картографічними API для навігаційних додатків, Тернопіль, 2024. – 79 с. URL: https://elartu.tntu.edu.ua/handle/lib/46893 (дата звернення: 10.03.2025).

9. Flutter documentation: вебсайт. URL: https://docs.flutter.dev/ (дата звернення: 15.11.2024).

10. Nataliia Golian, Vira Golian, Iryna Afanasieva. Black and white-box unit

testing for web applications. IEEE. 2022. С. 79-83.

11. React Native. Learn once, write anywhere: вебсайт. URL: https://reactnative.dev/ (дата звернення: 17.11.2024).

12. MAUI documentation: вебсайт. URL: https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-9.0 (дата звернення: 10.03.2025).

13. XXVIII МІЖНАРОДНИЙ МОЛОДІЖНИЙ ФОРУМ «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» Том 5 «Проблеми комп'ютерної інженерії та захисту інформації». URL: https://openarchive.nure.ua/server/api/core/bitstreams/27ddf287-dd41-4740-914a-436064fc44a2/content (дата звернення: 04.06.2025)

14. Відеоролик програмного продукту. URL: https://github.com/NChervinska/2025_М_ПІ_ІПЗ-23-1_Червінська_А_Л.mp4 (дата звернення: 29.05.2025).

15. Методичні вказівки до комплексного курсового проєктування для здобувачів спеціальності 121 – Інженерія програмного забезпечення, освітньо-наукова програма «Інженерія програмного забезпечення», другий (магістерський) рівень вищої освіти / Упоряд.: З.В. Дудар, В.І. Каук, І.А. Ревенчук, І.П. Сокорчук - Харків: ХНУРЕ, 2024. -30с.

16. Повний вихідний код програмного забезпечення. URL: https://github.com/NChervinska/2025_М_ПІ_ІПЗ-23-1_Червінська_А_Л(код).zip (дата звернення: 29.05.2025).

17. Презентація. URL: https://github.com/NChervinska/2025_М_ПІ_ІПЗ-23-1_Червінська_А_Л.pptx (дата звернення: 29.05.2025).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

1. Oleksandr Bezsmertnyi, Nataliia Golian, Vira Golian, Iryna Afanasieva. Behavior driven development approach in the modern quality control process. IEEE. 2020. С. 217-220.

2. Ібадов Д., Афанасьєва І. Crossplatform C++ library for multilayer perceptron learning. ScienceRise. 2016. № 2 (22). С. 19–25.

10. Nataliia Golian, Vira Golian, Iryna Afanasieva. Black and white-box unit testing for web applications. IEEE. 2022.. С. 79-83.