ДОДАТОК А
СЛАЙДИ ПРЕЗЕНТАЦІЇ

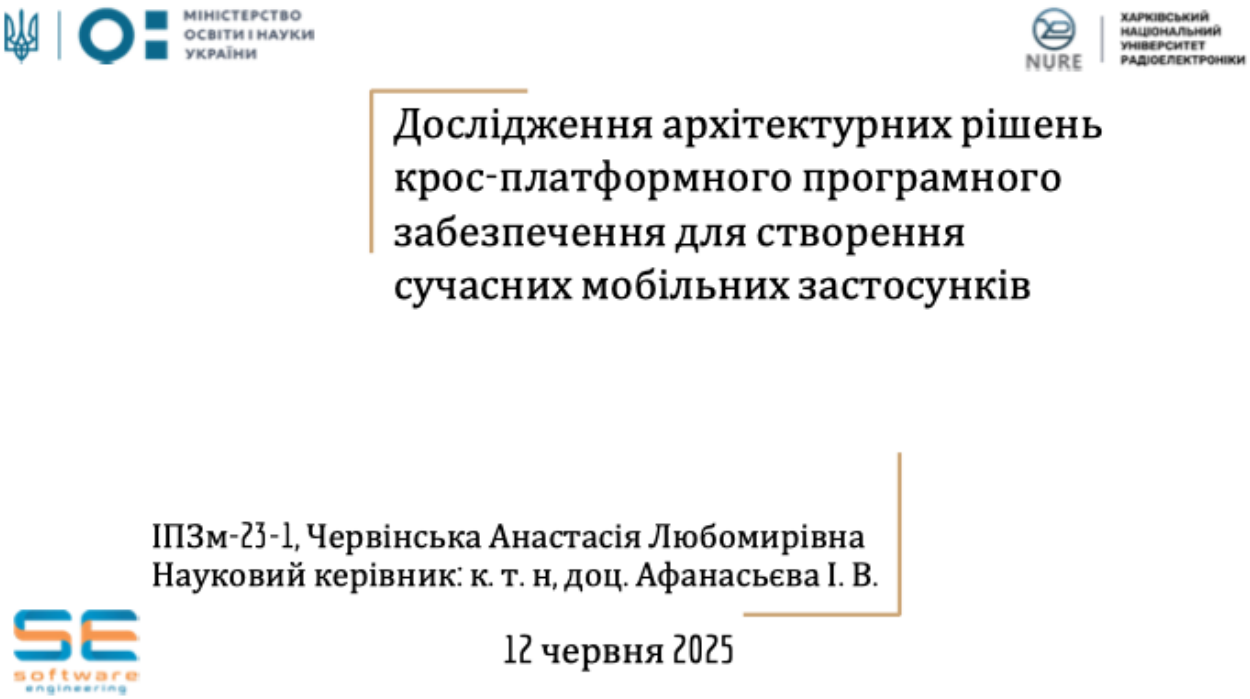


Рисунок А.1 – Слайд 1

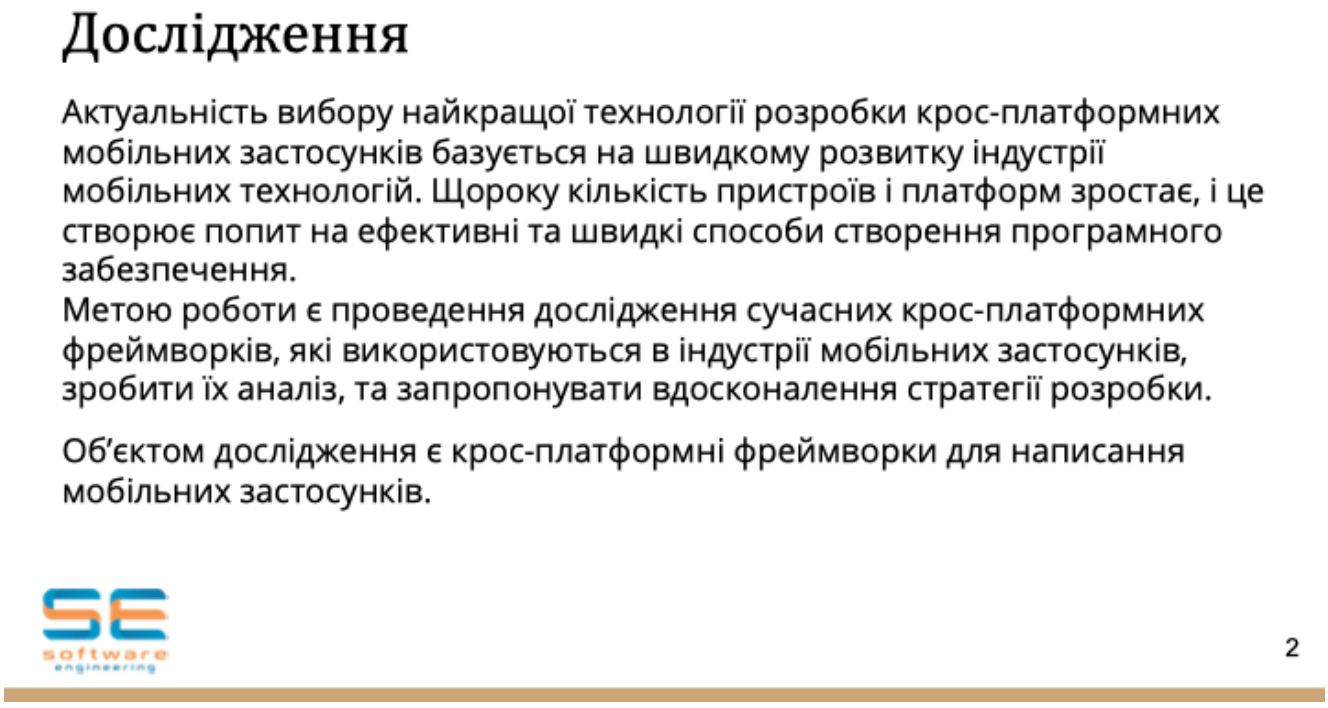


Рисунок А.2 – Слайд 2

Огляд літератури (аналогів)

- Дослідження ефективності фреймворку Flutter при розробці високо динамічних мобільних додатків
- Дослідження ефективності фреймворку React Native при розробці мобільних додатків
- Розробка програмного забезпечення для дослідження ефективності засобу Flutter при розробці мобільних додатків
- Дослідження можливостей .NET MAUI

3

Рисунок А.3 – Слайд 3

Прогалини у дослідженнях

- Відсутність порівняння недоліків
- Бракує емпіричних даних
- Обмежене охоплення UX/UI аспектів
- Ігнорується навчальна крива
- Не розглянута документація

4

Рисунок А.4 – Слайд 4

Постановка задачі

Розробити кросплатформні мобільні застосунки та визначити найбільш зручний та ефективний спосіб розробляти якісні мобільні застосунки.

Провести ретельний технічний огляд кожного фреймворку з Flutter, React Native та MAUI.

Провести аналіз мов програмування та архітектурних рішень, перевірити продуктивність кожного фреймвоку в різних умовах.

Надати комплексне розуміння у виборі технологічного рішення для майбутніх проєктів у сфері мобільної розробки.

Рисунок А.5 – Слайд 5

Методологія

Методи дослідження:

- Теоретичне дослідження
- Практичне дослідження

Інструментарій та технології:

- Фреймворки Flutter, React Native, MAUI.
- Мови програмування Dart, Java Script, C#.
- Середовище розробки Visual Studio Code, XCode, Android Studio.

Рисунок А.6 – Слайд 6

Зміст проведеного експерименту

- Три мобільні застосунки на кожен фресворк: React Native, Flutter, MAUI
- Отримання даних з API та вивід даних на екран
- Автоматичне вимірювання часу на відпрацювання запиту, запуск застосунку та вивід даних на UI

7

Рисунок А.7 – Слайд 7

Рекомендовані практики Flutter

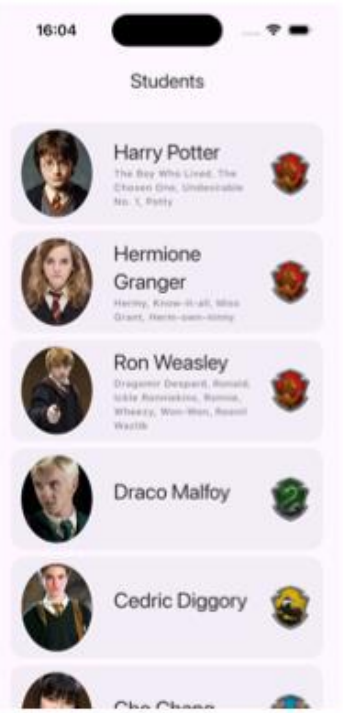
- Архітектурний патерн BLoC
- Інтеграція із RESTful API за допомогою бібліотеки Retrofit
- Використання Material Design і Cupertino
- Уникання непотрібного setState()
- Віддавання переваги одному віджету на файл
- Використання SizedBox замість контейнерів
- Утілізація потоків лише за необхідності
- Використання константних конструкторів

8

Рисунок А.8 – Слайд 8

Практичне дослідження Flutter

- Запит відпрацював за 309 мс
- Запуск застосунку зайняв 1207 мс
- Рендеринг UI відбувся а 14 мс
- Зручне використання
- Детальна документація
- Якісний інтерфейс



9

Рисунок А.9 – Слайд 9

Рекомендовані практики React Native

- Інтеграція із RESTful API за допомогою Axios
- Використання Redux для управління станами
- Тримати бізнес-логіку окремо від коду інтерфейсу
- Використання PureComponent і memo
- Уникання використання індексу, як ключової властивості
- Економне використання вбудованих стилів
- Використання вбудованого драйвера, коли це можливо

10

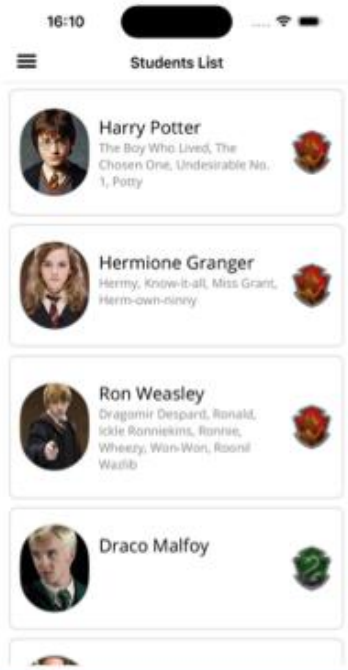
Рисунок А.10 – Слайд 10

Рисунок А.11 – Слайд 11

Рисунок А.12 – Слайд 12

Практичне дослідження MAUI

- Запит відпрацював за 262 мс
- Запуск застосунку зайняв 888 мс
- Рендеринг UI відбувся за 33 мс
- Зручне використання
- Стисла документація
- Якісний інтерфейс



13

Рисунок А.13 – Слайд 13

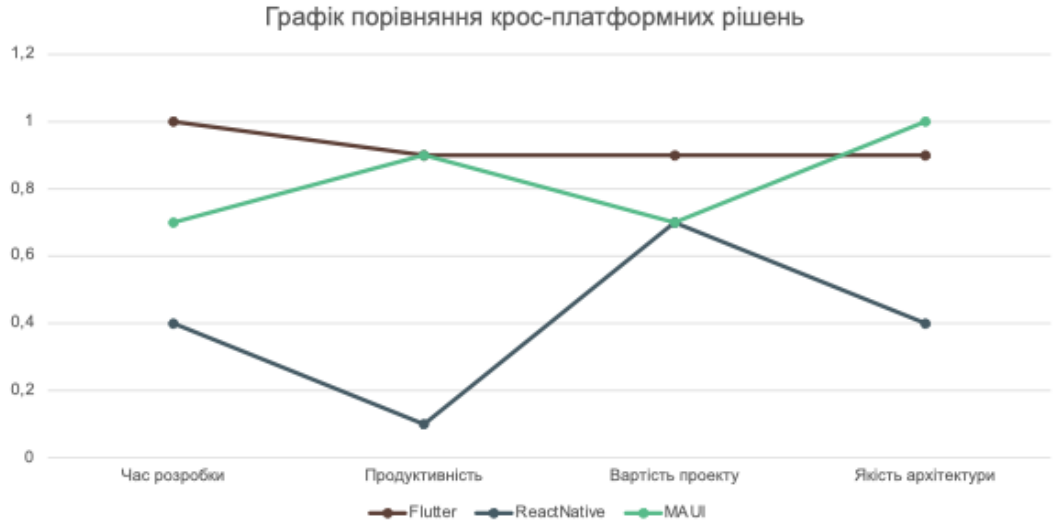
Порівняльна таблиця

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 мс)	309	333	262
Час запуску застосунку (в мс)	1207	25803	888
Час рендерінгу (в мс)	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 спільноти розробників	9	10	5

14

Рисунок А.14 – Слайд 14

Графік порівняння крос-платформних програмних рішень



15

Рисунок А.15 – Слайд 15

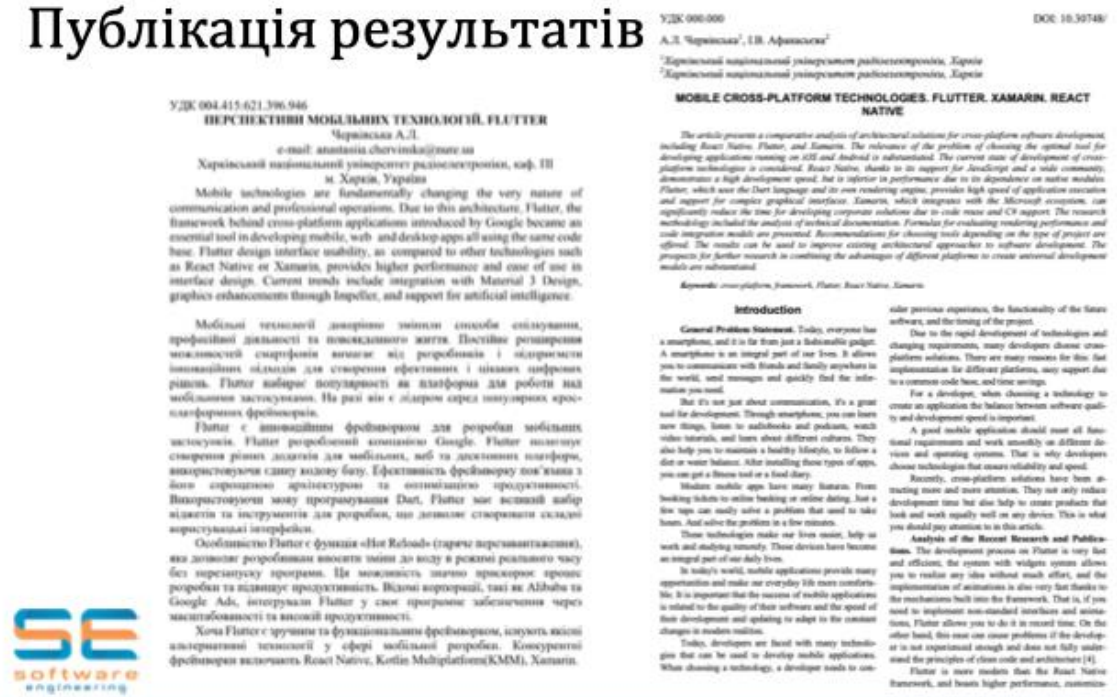
Аналіз отриманих результатів

- Flutter має найкращі показники за часом рендерінгу
- Flutter має кращу документацію
- React Native показав себе як не зручний та повільний фреймворк
- MAUI має хороші показники швидкості
- MAUI має можливість створити багато потоків

16

Рисунок А.16 – Слайд 16

Публікація результатів



УДК 004.415:621.396.946

ПЕРСПЕКТИВИ МОБІЛЬНИХ ТЕХНОЛОГІЙ. FLUTTER

Червінська А.Л.
e-mail: anastasiia.chervinska@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

Mobile technologies are fundamentally changing the very nature of communication and professional operations. Due to this architecture, Flutter, the framework behind cross-platform applications introduced by Google became an essential tool in developing mobile, web and desktop apps all using the same code base. Flutter design interface usability, as compared to other technologies such as React Native or Xamarin, provides higher performance and ease of use in interface design. Current trends include integration with Material 3 Design, graphics enhancements through Impeller, and support for artificial intelligence.

Мобільні технології докорінно змінили способи спілкування, професійної діяльності та повсякденного життя. Постійне розширення можливостей смартфонів вимагає від розробників і підприємств інноваційних підходів для створення ефективних і цінних цифрових рішень. Flutter набирає популярності як платформа для роботи над мобільними застосунками. На разі він є лідером серед популярних крос-платформних фреймворків.

Flutter є інноваційним фреймворком для розробки мобільних застосунків. Flutter розроблений компанією Google. Flutter полегшує створення різних додатків для мобільних, веб та десктопних платформ, використовуючи єдину кодову базу. Ефективність фреймворку пов'язана з його спрощеною архітектурою та оптимізацією продуктивності. Використовуючи мову програмування Dart, Flutter має великий набір віджетів та інструментів для розробок, що дозволяє створювати складні користувацькі інтерфейси.

Особливістю Flutter є функція «Hot Reload» (гаряче перезавантаження), яка дозволяє розробникам вносити зміни до коду в режимі реального часу без перезапуску програми. Це можливість значно прискорює процес розробки та підвищує продуктивність. Відомі корпорації, такі як Alibaba та Google Ads, інтегрували Flutter у свої програмне забезпечення через масштабованості та високій продуктивності.

Хоча Flutter є зручним та функціональним фреймворком, існують якісні альтернативні технології у сфері мобільної розробки. Конкуруючі фреймворки включають React Native, Kotlin Multiplatform(KMM), Xamarin.

УДК 000.000 DOI: 10.30748/

А.Л. Червінська[1], І.В. Афанасьєва[2]
[1]Харківський національний університет радіоелектроніки, Харків
[2]Харківський національний університет радіоелектроніки, Харків

MOBILE CROSS-PLATFORM TECHNOLOGIES. FLUTTER. XAMARIN. REACT NATIVE

The article presents a comparative analysis of architectural solutions for cross-platform software development, including React Native, Flutter, and Xamarin. The relevance of the problem of choosing the optimal tool for developing applications running on iOS and Android is substantiated. The current state of development of cross-platform technologies is considered. React Native, thanks to its support for JavaScript and a wide community, demonstrates a high development speed, but is inferior in performance due to its dependence on native modules. Flutter, which uses the Dart language and its own rendering engine, provides high speed of application execution and support for complex graphical interfaces. Xamarin, which integrates with the Microsoft ecosystem, can significantly reduce the time for developing corporate solutions due to code reuse and C# support. The research methodology included the analysis of technical documentation. Formulas for evaluating rendering performance and code integration models are presented. Recommendations for choosing tools depending on the type of project are offered. The results can be used to improve existing architectural approaches to software development. The prospects for further research in combining the advantages of different platforms to create universal development models are substantiated.

Keywords: *cross-platform, framework, Flutter, React Native, Xamarin.*

Introduction

General Problem Statement. Today, everyone has a smartphone, and it is far from just a fashionable gadget. A smartphone is an integral part of our lives. It allows you to communicate with friends and family anywhere in the world, send messages and quickly find the information you need.

But it's not just about communication, it's a great tool for development. Through smartphone, you can learn new things, listen to audiobooks and podcasts, watch video tutorials, and learn about different cultures. They also help you to maintain a healthy lifestyle, to follow a diet or water balance. After installing these types of apps, you can get a fitness tool or a food diary.

Modern mobile apps have many features. From booking tickets to online banking or online dating, just a few taps can easily solve a problem that used to take hours. And solve the problem in a few minutes.

These technologies make our lives easier, help us work and studying remotely. These devices have become an integral part of our daily lives.

In today's world, mobile applications provide many opportunities and make our everyday life more comfortable. It is important that the success of mobile applications is related to the quality of their software and the speed of their development and updating to adapt to the constant changes in modern realities.

Today, developers are faced with many technologies that can be used to develop mobile applications. When choosing a technology, a developer needs to consider previous experience, the functionality of the future software, and the timing of the project.

Due to the rapid development of technologies and changing requirements, many developers choose cross-platform solutions. There are many reasons for this: fast implementation for different platforms, easy support due to a common code base, and time savings.

For a developer, when choosing a technology to create an application the balance between software quality and development speed is important.

A good mobile application should meet all functional requirements and work smoothly on different devices and operating systems. That is why developers choose technologies that ensure reliability and speed.

Recently, cross-platform solutions have been attracting more and more attention. They not only reduce development time but also help to create products that look and work equally well on any device. This is what you should pay attention to in this article.

Analysis of the Recent Research and Publications. The development process on Flutter is very fast and efficient, the system with widgets system allows you to realize any idea without much effort, and the implementation of animations is also very fast thanks to the mechanisms built into the framework. That is, if you need to implement non-standard interfaces and animations, Flutter allows you to do it in record time. On the other hand, this ease can cause problems if the developer is not experienced enough and does not fully understand the principles of clean code and architecture [4].

Flutter is more modern than the React Native framework, and boasts higher performance, customiza-

Рисунок А.17 – Слайд 17

Підсумки

У рамках кваліфікаційної роботи:
- Проведено теоретичне дослідження
- Сформульовано критерії для порівняння 3-х фреймворків
- Для проведення практичного дослідження створено три застосунки з використанням різних фреймворків
- Зроблено аналіз отриманих результатів
- Подано статтю у журнал «Системи обробки інформації»
- Опубліковано тези у XXVIII МІЖНАРОДНИЙ МОЛОДІЖНИЙ ФОРУМ «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

Рисунок А.18 – Слайд 18

ДОДАТОК Б
АПРОБАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

Рисунок Б.1 – Перша сторінка збірника XXVIII Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» (за даними [13])

УДК 004.415:621.396.946

ПЕРСПЕКТИВИ МОБІЛЬНИХ ТЕХНОЛОГІЙ. FLUTTER

Червінська А.Л.
e-mail: anastasiia.chervinska@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

Mobile technologies are fundamentally changing the very nature of communication and professional operations. Due to this architecture, Flutter, the framework behind cross-platform applications introduced by Google became an essential tool in developing mobile, web and desktop apps all using the same code base. Flutter design interface usability, as compared to other technologies such as React Native or Xamarin, provides higher performance and ease of use in interface design. Current trends include integration with Material 3 Design, graphics enhancements through Impeller, and support for artificial intelligence.

Мобільні технології докорінно змінили способи спілкування, професійної діяльності та повсякденного життя. Постійне розширення можливостей смартфонів вимагає від розробників і підприємств інноваційних підходів для створення ефективних і цікавих цифрових рішень. Flutter набирає популярності як платформа для роботи над мобільними застосунками. На разі він є лідером серед популярних крос-платформних фреймворків.

Flutter є інноваційним фреймворком для розробки мобільних застосунків. Flutter розроблений компанією Google. Flutter полегшує створення різних додатків для мобільних, веб та десктопних платформ, використовуючи єдину кодову базу. Ефективність фреймворку пов'язана з його спрощеною архітектурою та оптимізацією продуктивності. Використовуючи мову програмування Dart, Flutter має великий набір віджетів та інструментів для розробки, що дозволяє створювати складні користувацькі інтерфейси.

Особливістю Flutter є функція «Hot Reload» (гаряче перезавантаження), яка дозволяє розробникам вносити зміни до коду в режимі реального часу без перезапуску програми. Ця можливість значно прискорює процес розробки та підвищує продуктивність. Відомі корпорації, такі як Alibaba та Google Ads, інтегрували Flutter у своє програмне забезпечення через масштабованості та високій продуктивності.

Хоча Flutter є зручним та функціональним фреймворком, існують якісні альтернативні технології у сфері мобільної розробки. Конкурентні фреймворки включають React Native, Kotlin Multiplatform(KMM), Xamarin.

Розроблений Facebook, React Native використовує JavaScript для створення мобільних застосунків. Підтримка крос-платформного повторного використання коду та велика екосистема бібліотек роблять його універсальним інструментом. Проте Flutter має вищу продуктивність під час

Рисунок Б.2 – Перша сторінка тез у XXVIII Міжнародному молодіжному форумі «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» (за даними [13])

виконання програми та більш цілісний дизайн інтерфейсу на основі віджетів, усуваючи залежність від специфічних для платформи компонентів.

Google визнан Kotlin найкращою мовою для розробки під Android. Можливості KMM дозволяють повторно використовувати код на Android та iOS. Зате, стандартна архітектура віджетів Flutter спрощує узгодженість інтерфейсу на різних платформах, що скорочує час розробки.

Фреймворк Xamarin від Microsoft використовує мову C# для розробки крос-платформних застосунків. Його безшовна інтеграція з екосистемою .NET робить його кращим вибором для рішень корпоративного рівня. Однак сучасні інструменти Flutter та функція гарячого перезавантаження пропонують більш динамічну та привабливу розробку, а також більш якісний та приємний сучасний інтерфейс.

Еволюція мобільних технологій характеризується швидкими трансформаційними тенденціями, серед таких нещодавні інновації Flutter:

- Flutter тепер підтримує Material 3 Design, що дає змогу створювати динамічні кольорові теми та персоналізовані інтерфейси;
- покращена продуктивність завдяки Impeller, сучасному движку рендерингу, що забезпечує більш плавну анімацію та покращені графічні можливості;
- завдяки плагінам та інтеграціям Flutter полегшує інтеграцію моделей машинного навчання, дозволяючи розробникам створювати більш розумні та адаптивні додатки.

Інновації мобільних технологій продовжують розвиватись, пропонуючи безпрецедентні можливості для розробників і кінцевих користувачів. Такі фреймворки, як Flutter, у поєднанні з технологічним прогресом у сфері зв'язку та штучного інтелекту, стануть рушійною силою наступного етапу розвитку мобільних технологій. Траєкторія розвитку мобільного зв'язку підкреслює майбутнє, що характеризується підвищеною взаємопов'язаністю, розширеною функціональністю та трансформаційним потенціалом.

Список використаних джерел:

1. 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.

2. Chajjed, Sandip. Probabilistic Prediction of Coalescence Flutter Using Measurements: Application to the Flutter Margin Method. Carleton University. 2016.

3. Sufyan bin Uzayr. Basics of Flutter. CRC Press. 2022.

Рисунок Б.3 – Друга сторінка тез у XXVIII Міжнародному молодіжному форумі «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ» (за даними [13])

УДК 000.000 DOI: 10.30748/

А.Л. Червінська[1], І.В. Афанасьєва[2]

[1]*Харківський національний університет радіоелектроніки, Харків*
[2]*Харківський національний університет радіоелектроніки, Харків*

MOBILE CROSS-PLATFORM TECHNOLOGIES. FLUTTER. XAMARIN. REACT NATIVE

The article presents a comparative analysis of architectural solutions for cross-platform software development, including React Native, Flutter, and Xamarin. The relevance of the problem of choosing the optimal tool for developing applications running on iOS and Android is substantiated. The current state of development of cross-platform technologies is considered. React Native, thanks to its support for JavaScript and a wide community, demonstrates a high development speed, but is inferior in performance due to its dependence on native modules. Flutter, which uses the Dart language and its own rendering engine, provides high speed of application execution and support for complex graphical interfaces. Xamarin, which integrates with the Microsoft ecosystem, can significantly reduce the time for developing corporate solutions due to code reuse and C# support. The research methodology included the analysis of technical documentation. Formulas for evaluating rendering performance and code integration models are presented. Recommendations for choosing tools depending on the type of project are offered. The results can be used to improve existing architectural approaches to software development. The prospects for further research in combining the advantages of different platforms to create universal development models are substantiated.

Keywords: *cross-platform, framework, Flutter, React Native, Xamarin.*

Introduction

General Problem Statement. Today, everyone has a smartphone, and it is far from just a fashionable gadget. A smartphone is an integral part of our lives. It allows you to communicate with friends and family anywhere in the world, send messages and quickly find the information you need.

But it's not just about communication, it's a great tool for development. Through smartphone, you can learn new things, listen to audiobooks and podcasts, watch video tutorials, and learn about different cultures. They also help you to maintain a healthy lifestyle, to follow a diet or water balance. After installing these types of apps, you can get a fitness tool or a food diary.

Modern mobile apps have many features. From booking tickets to online banking or online dating. Just a few taps can easily solve a problem that used to take hours. And solve the problem in a few minutes.

These technologies make our lives easier, help us work and studying remotely. These devices have become an integral part of our daily lives.

In today's world, mobile applications provide many opportunities and make our everyday life more comfortable. It is important that the success of mobile applications is related to the quality of their software and the speed of their development and updating to adapt to the constant changes in modern realities.

Today, developers are faced with many technologies that can be used to develop mobile applications. When choosing a technology, a developer needs to consider previous experience, the functionality of the future software, and the timing of the project.

Due to the rapid development of technologies and changing requirements, many developers choose cross-platform solutions. There are many reasons for this: fast implementation for different platforms, easy support due to a common code base, and time savings.

For a developer, when choosing a technology to create an application the balance between software quality and development speed is important.

A good mobile application should meet all functional requirements and work smoothly on different devices and operating systems. That is why developers choose technologies that ensure reliability and speed.

Recently, cross-platform solutions have been attracting more and more attention. They not only reduce development time but also help to create products that look and work equally well on any device. This is what you should pay attention to in this article.

Analysis of the Recent Research and Publications. The development process on Flutter is very fast and efficient, the system with widgets system allows you to realize any idea without much effort, and the implementation of animations is also very fast thanks to the mechanisms built into the framework. That is, if you need to implement non-standard interfaces and animations, Flutter allows you to do it in record time. On the other hand, this ease can cause problems if the developer is not experienced enough and does not fully understand the principles of clean code and architecture [4].

Flutter is more modern than the React Native framework, and boasts higher performance, customiza-

Рисунок Б.4 – Перша сторінка статті у журналі «Системи обробки інформації»

tion, and better documentation [5]. It was found that the native approach has the highest product quality, but also the highest development cost, as it requires two separate development teams.

A hybrid application has a lower development cost, but also a lower quality of the product. The best option was a cross-platform application that combines the strengths of both approaches, namely, relatively low development costs and high quality of the developed solution [6]. In its turn, React Native continues to outperform Flutter in the following parameters:

– the current number of developers who are able to develop a program using this technology is greater than that of Flutter; Using a more common programming language programming language (JavaScript instead of Dart);

– some parts of the logic can be reused in web applications that are written with the React.js library. There are still more vacancies on the market than Flutter.

In turn, Xamarin has been on the market since 2011 and during this time it has gained wide recognition among developers, as well as a stable base of tools and libraries. Xamarin provides access to all the APIs and features of the iOS and

Android operating systems using Xamarin.iOS and Xamarin.Android. This allows developers to work with the platform at the native level, which is not always possible to achieve with React Native or Flutter.

Thanks to compilation to native code, Xamarin provides high application performance. In comparison, React Native uses a JavaScript bridge to communicate with native modules, which can create delays. Flutter although it has high performance, depends on its own rendering engine, which can take up more resources.

Aim of the Research. This research is aimed at studying current methods and technologies used in the mobile application software industry and exploring the advantages and disadvantages of frameworks.

Main part

1. Analysis of the subject area

Nowadays, when everyone uses mobile apps in their daily lives, it is important to know which technologies applications in their daily lives, it is important to know which technologies best meet the requirements of the market and user requirements. Technical solutions have a direct impact on functionality, performance, and user experience. They are the basis for the success of a mobile application's success.

Increasingly, developers are choosing cross-platform solutions instead of native development. There are several reasons for this:

– cross-platform frameworks allow developers to use the same code for different platforms, which saves time and resources compared to writing separate native apps for Android and iOS;

– development for both platforms at the same time allows you to release a product faster product to the market faster, reducing the time from idea to implementation;

– development costs are much lower, which allows you to invest more money in marketing and attract more users.

In today's world of mobile app development, the question arises of the most optimal and effective cross-platform solution.

The choice is complicated by the fact that there are many different technologies available on the market. Let us consider the most popular ones: Flutter, React Native, and Xamarin.

React Native is a cross-platform open-source framework for the development of mobile applications for IOS and Android platforms, which was created by Facebook in 2015. It uses Java Script as a programming language. React Native has the following advantages:

– React Native's live reload feature allows you to see and work with changes in real time; the developer can make corrections in the code while the application is loading, and it will be reflected in the application with automatic reloading;

– React Native has extensive rendering capabilities and uses a component-based approach that makes it easy to create both simple and complex interface designs;

– the ability to reuse React Native code, which helps to save development costs;

– React Native offers many different plugins, including native and based on Java Script;

– React Native has a community of more than 100,000 [1] active members, which can provide expert support;

– Facebook engineers are constantly updating and improving the framework.

This framework was used to create such well-known projects as Instagram, Bloomberg, and Pinterest.

Flutter is a set of tools and an open-source framework for mobile applications for Android and IOS, web applications and desktop applications for Windows, macOS and Linux using the Dart programming language developed by Google.

Flutter is popular because of:

– the “hot reload” function allows you to see updates in real time without overloading the program;

– Flutter apps work on par with native mobile apps;

– has a growing community that can help with problems;

– has clear documentation;

Рисунок Б.5 – Друга сторінка статті у журналі «Системи обробки інформації»

– there are many YouTube videos available for those who want to start learning Flutter or improve their skills;

– Flutter provides tools that simplify the testing process.

There are several large companies that actively use this framework: Google Ads, eBay Motors, SpaceX Go.

Xamarin is an open-source platform designed to create modern applications for IOS, Android, and Windows with .NET. It uses C# programming language.

Advantages of using Xamarin:

– the code can be used to develop applications on several platforms; the program coding is based on C# and .NET libraries;

– Xamarin offers many APIs and plugins and supports cross-platform application development, which allows developers to enjoy native-level interaction with the device hardware;

– Xamarin doesn't require developers to build code from scratch.

Several large applications have been written on this platform: Alaska Airlines, BBC Good Food, and Novarum DX.

With so many different technologies available, there is a need to research of these technologies and their architectural solutions.

The relevance of choosing the best technology for developing cross-platform mobile applications is based on the rapid development of the mobile technology industry. The number of devices and platforms is growing every year, and this creates a demand for efficient and fast ways to create software.

Study will help to determine the advantages, disadvantages, and limitations of each technology.

In addition, the research will focus on architectural solutions and their impact on the performance, adaptability, and stability of mobile applications.

2. Theoretical study

Flutter allows you to build applications using architectural patterns such as BLoC (Business Logic Component), MVVM (Model-View-ViewModel), and Redux.

The most recommended for cross-platform development is BLoC because of the following advantages:

– ensuring a clear separation of business logic and interface;

– the possibility of code reuse;

– ease of testing components;

– each state that BloC goes through can be recorded and analyzed;

– BloC is easily combined with other Flutter libraries;

– scalability for large projects.

BLoC implements the «streams» pattern, which allows you to separate events and states, which simplifies the management of complex scenarios. The key features are support for reactive programming, which provides automatic updating of the of the interface in response to state changes, as shown in Fig. 1.

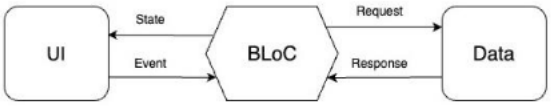


Fig. 1. Scheme of BLoC operation
Source: developed by the authors.

UI is short for User Interface.

Events define what should happen in the application. For example, a user presses a button, and this generates an IncrementCounter event. Events notify the BLoC that a certain action needs to be performed.

State reflects the current status of the application, for example: data loading, successful, error. Each state change updates the user interface.

The BLoC is a central component that receives events, processes them, and outputs new states. The main function of the BLoC is the mapEventToState method, which determines how an event is converted to a state.

Threads are used to pass events and states between components. Events are received by the BLoC through the add(Event) method. The state is passed back to the UI via a stream.

This approach improves productivity, reduces the risk of errors, and makes it easier to project support.

Integration with RESTful APIs is best done using the Retrofit library.

It has the following advantages:

– convenient support for HTTP requests (GET, POST, PUT, DELETE) through annotations;

– automatic deserialization of data in the model;

– high flexibility and extensibility due to interspersers and converters.

Retrofit is built on the Dio library.

Annotations are used to describe API methods, such as @GET, @POST, @PUT.

The library uses the build_runner package to generate client code.

Also, it supports serialization/deserialization of JSON data through libraries such as json_serializable.

Retrofit has an advantage over other approaches due to clear typing, automation of query processing, and the ability to work with large data sets.

Flutter allows you to use Keys to optimize the redrawing of components to optimize redrawing. For example, using a GlobalKey provides access to the state of a of a widget without the need for a complete redraw.

Flutter provides a single code base for Android and iOS, which allows you to follow Material Design and Cupertino guidelines. They provide a platform-specific user interface. Material Design is designed to provide a unified modern design on Android devices, with a focus on color

Рисунок Б.6 – Третя сторінка статті у журналі «Системи обробки інформації»

palettes, shadows, and animations. It includes ready-made components such as FloatingActionButton, SnackBar, and BottomNavigationBar. Cupertino, in its turn, is designed for iOS devices, providing users with an experience that complies with Apple's Human Interface Guidelines. Components such as CupertinoButton, CupertinoNavigationBar, and others, allow you to create interfaces that look and feel like native iOS apps.

This is how you can measure the time it takes to render an interface for further analysis:

```
Widget build(BuildContext context) {
  final stopwatch = Stopwatch()..start();
  final app = MaterialApp(home: app);
  stopwatch.stop();
  log('Build time: ${stopwatch.elapsedMilliseconds} ms');
  return app;
}
```

Fig. 2. An example of code to measure the time of the render interface
Source: developed by the authors.

To determine the time of data rendering, you can use WidgetsBinding.instance.addPostFrameCallback to measure the time from start to the end of the rendering page.

We also analyzed and created a list of Flutter best practices:

– using Material Design and Cupertino;
– avoiding unnecessary setState();
– preferring one widget per file;
– using SizedBoxes instead of containers;
– implementing threads only when necessary;
– use of constant constructors.

React Native supports many architectural patterns, such as Flux, Redux, and MVVM. The most common is Redux, which provides centralized state management and simplifies the interaction between components. Redux allows you to store state in a single container (store), which makes it ideal for complex programs (see Figure 3).

Store is a global object that stores all the state of an application. In Redux, there is only one store for the entire application.

Actions describe what happened in the application. These are ordinary objects with type (action name) and, if necessary, payload (additional data) fields.

Reducers are functions that take the current state and action, process it, and return a new state.

Dispatch is a method that sends actions to the repository.

Selectors are functions that allow you to get the data you need from a state.

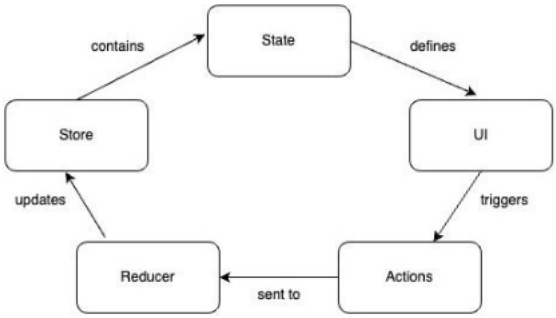


Fig. 3. The Redux workflow
Source: developed by the authors.

UI is short for User Interface.

The main advantages of Redux:

– centralized data storage;
– predictability of the state due to clearly defined rules (reducers);
– easy testing of application logic.

To integrate with the API, React Native usually uses Axios, which supports all standard HTTP requests (GET, POST, PUT, DELETE). In addition, Axios has the following advantages:

– easy to configure interspersers to handle requests and responses;
– built-in support for timeouts and error handling;
– the ability to work with tokens for authorization.

Axios integrates with Redux for state management and allows you to optimize network interaction by caching requests.

Axios in React Native works as an intermediary between the application and the server to exchange data via the HTTP protocol. When you want to send a request, Axios first generates a configuration that includes all the necessary parameters, such as the address, request method, headers, and data. This configuration is passed through special interceptors that can modify or add to it, such as adding an authentication token.

Axios then uses low-level tools, such as XMLHttpRequest in a browser or http in Node.js, to actually execute the request. The sent request goes to the server, which processes it and returns a response. This response also passes through interceptors, where you can check the status or extract useful data, and then returns as a promissory note. If an error occurs, it can be processed depending on the type: it can be a timeout, a network problem, or a server response with an error status.

React Native uses Virtual DOM to minimize component rendering of components. Dynamic interface updates are provided by the diffing mechanism that finds differences between the current and previous state and updates only the necessary parts.

React Native supports the use of platform-oriented components to create intuitive, user-friendly interfaces. For example:

Рисунок Б.7 – Четверта сторінка статті у журналі «Системи обробки інформації»

– for Android, use Material Design components such as TouchableNativeFeedback or BottomNavigation;

– for iOS, use Cupertino components such as ActionSheetIOS or DatePickerIOS.

React Native also provides performance monitoring tools such as Flipper. Using memoization (React.memo) and optimizing lists with FlatList or SectionList can significantly improve performance.

This is how you can measure the time it takes to render an interface for further analysis:

```
const start = performance.now();

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

Fig. 4. An example of code to measure the time of the render interface
Source: developed by the authors.

We also analyzed and created a list of React Native best practices:

– using the React Native CLI;

– keep the business logic separate from the UI code;

– using PureComponent and memo;

– avoid using index as a key property;

– use platform-oriented components to create intuitive interfaces;

– economical use of built–in styles;

– use the built-in driver whenever possible.

Xamarin allows you to use several architectural approaches, such as MVVM (Model-View-ViewModel), MVC (Model-View-Controller), and MVP (Model-View-Presenter). MVVM is the most common for Xamarin due to its integration with the Xamarin.Forms library and support for Data Binding (see Figure 5).

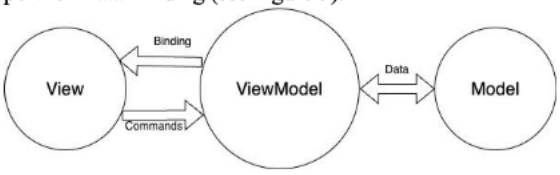


Fig. 5. The Redux workflow
Source: developed by the authors.

Model is responsible for working with data. It can include classes for working with APIs, databases, or business logic. A Model is independent of a View or ViewModel.

A View displays data to the user. In Xamarin, these are pages and interface elements (XAML + Code-Behind). It has a minimum of logic that relates exclusively to working with the UI.

ViewModel is an intermediate layer between Model and View. Contains logic that provides data representation for the View. It implements properties and commands that link user actions to the state of the application.

The main advantages of MVVM:

– clear separation of logic and interface;

– support for code reuse between platforms;

– simplified testing of business logic.

To integrate with the Xamarin API, the documentation recommends use the Refit library. It provides the following advantages:

– a simple and convenient interface for creating REST requests using annotations;

– automatic processing of JSON responses and conversion to typed objects;

– support for asynchronous requests.

Refit greatly simplifies the work with APIs thanks to a declarative approach to customizing requests. Instead of manually writing requests and processing responses, the developer describes the API as an interface with methods that correspond to each route or request. This interface contains special annotations that indicate what type of request is being used, what parameters are passed in the URL or request body.

At runtime, Refit automatically creates a client that implements this interface. When an interface method is called, the library generates a corresponding HTTP request, sends it to the server, and processes the response. Refit also automatically converts the response data into a convenient format, for example, into models used in the application.

In Xamarin.Forms, support for dynamic interface updates is implemented through the Data Binding mechanism. This allows you to automatically synchronize data between the model and the interface.

Xamarin.Forms allows you to create a single interface for both platforms, using such elements as Grid, StackLayout, and CollectionView. For platform-specific design, Xamarin supports the use of Custom Renderers. For Android, it is possible to implement Material Design through Visual Material.

For iOS, support for Apple Human Interface Guidelines through native components.

You can measure the application startup time:

```
Stopwatch buildStopwatch = new Stopwatch();
buildStopwatch.Start();
// код ініціалізації апки
buildStopwatch.Stop();
Console.WriteLine($"Час: {buildStopwatch.ElapsedMilliseconds}
```

Fig. 6. An example of code to measure the application startup time
Source: developed by the authors.

To measure the time it takes to draw data, you can also measure the time between data acquisition and its display on the screen can also be measured with Stopwatch.

General performance optimization can be done with the Xamarin Profiler. It allows you to identify performance bottlenecks, such as memory usage memory consumption or long pauses in execution (jank).

We also analyzed and created a list of Xamarin best practices:

– using Custom Renderers for platform-specific design;

– detecting XAML errors at the compilation level;

– avoiding dependencies on frameworks;

Рисунок Б.8 – П’ята сторінка статті у журналі «Системи обробки інформації»

– the advantage of using commands rather than event handlers;

– setting the BindingContext in the CodeBehind View;

– using Observable Collections instead of Lists;

– placing asynchronous method calls in OnAppearing in the CodeBehind View;

Conclusions

As a result of the work, popular cross-platform frameworks and identified their advantages and disadvantages. Flutter, React Native, and Xamarin are the three most popular frameworks for developing cross frameworks for developing cross-platform applications, each with its own strengths.

All three frameworks provide the ability to create a single code for different platforms, which significantly reduces the cost of developing and maintaining applications. Flutter stands out due to its own rendering engine and the ability to precisely control over the UI. React Native provides fast development by using JavaScript and many ready-made libraries.

Among the advantages of Flutter is that it relies on plugin libraries to access native features of the platforms. React Native makes it easy to integrate native code through Native Modules. Xamarin provides the tightest integration with platforms using native APIs.

If we compare the user interface, Flutter offers a single approach to UI through the Material and Cupertino libraries. They provide the same look and feel across all platforms. React Native uses native components, which guarantees a platform-specific design. Xamarin supports both native components and cross-platform elements through Xamarin.Forms.

Each of the technologies is a good tool for developing cross-platform mobile applications. All frameworks can provide high quality of the product.

The study opens opportunities for further research in optimizing cross-platform solutions and studying their impact on the development of large and complex mobile projects. Also, additional research may also consider new frameworks that will appear on the market.

List of references

1. Ібадов Д., Афанасьєва І. Crossplatform C++ library for multilayer perceptron learning. ScienceRise. 2016. № 2 (22). С. 19-25. https://doi.org/10.15587/2313-8416.2016.69588.

2. Корабельська Ю. К. Програмне забезпечення для береження даних на снові об'єктно-реляційного співвідношення у Java середовищі, Харків, 2021. - 112 с.

3. Осташко Є. В. Дослідження методів створення сервісно-орієнтованих та крос-платформених додатків за допомогою Flutter з серверною частиною Firebase, Харків, 2023. - 86 с. URL: https://openarchive.nure.ua/items/626371df-030-4a74-a857-0b6feda408f1 (дата звернення: 10.10.2024).

4. 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-21-1Куденко.pdf?sequence=1&isAllowed=y (дата звернення: 10.10.2024).

5. 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).

6. Sun, Enqiang. Cross-platform heterogeneous runtime environment. Boston, Massachusetts. 2016. С. 5-28. URL: https://repository.library.northeastern.edu/files/neu:cj82n975s (дата звернення: 16.04.2016). https://doi.org/10.17760/d2021316

7. Eke, Norbert. Cross-Platform Software Developer Expertise Learning. Carleton University. 2020. С.45-84. URL: https://repository.library.carleton.ca/concern/etds/x633f1838. https://doi.org/10.22215/etd/2020-14096

8. Marc Gregoire. Professional C++. John Wiley & Sons, Inc. 2018. С.76-211. https://doi.org/10.1002/9781394193202

9. Louis Kreitmann, Giselle D'Souza. A computational framework to improve cross-platform implementation of transcriptomics signatures. ScienceDirect. 2024. URL: https://www.sciencedirect.com/science/article/pii/S2352396424002391 (дата звернення: 10.01.2024). https://www.sciencedirect.com/science/article/pii/S2352396424002391

10. Qingqing Ji, Xiaoli Wang. Research on cross-platform dissemination of rational and irrational information based on a two-layer network. ScienceDirect. 2023. С.5-18. URL: https://www.sciencedirect.com/science/article/pii/S2405844024046012 (дата звернення: 06.12.2023). https://doi.org/10.1016/j.heliyon.2024.e28570

11. Márcio Luís Moreira De Souza MSc 1, Gabriel Ayres Lopes 2, Alexandre Castelo Branco MD 3, Jessica K Fairley MD, MSc 4, Lucia Alves De Oliveira Fraga PhD 1. Leprosy Screening Based on Artificial Intelligence: Development of a Cross-Platform App. ScienceDirect. 2020. URL: https://www.sciencedirect.com/org/science/article/pii/S2291522221001820 (дата звернення: 01.12.2020). https://doi.org/10.2196/23718.

12. Christoph Rieger, Tim A. Majchrzak. Towards the definitive evaluation framework for cross-platform app development approaches. ScienceDirect. 2018. С.6-24.. URL:https://www.sciencedirect.com/science/article/pii/S0164121219300743 (дата звернення: 20.11.2018). https://doi.org/10.1016/j.jss.2019.04.001.

13. Matteo Ciman, Ombretta Gaggi. An empirical analysis of energy consumption of cross-platform frameworks for mobile development. ScienceDirect. 2016. URL: https://www.sciencedirect.com/science/article/abs/pii/S1574119216303170 (дата звернення:12.07.2016), https://doi.org/10.1016/j.pmcj.2016.10.004.

Рисунок Б.9 – Шоста сторінка статті у журналі «Системи обробки інформації»

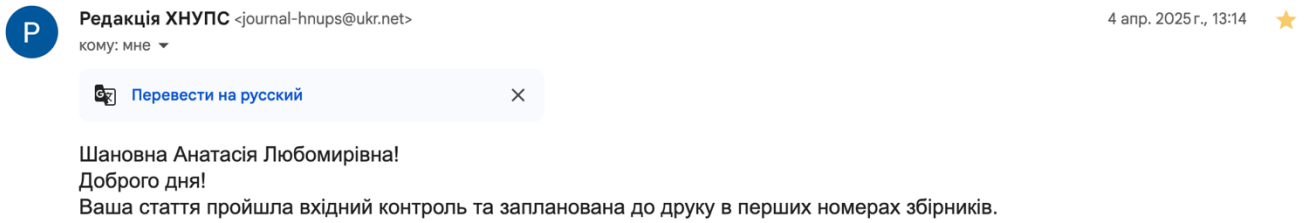


Рисунок Б.10 – Підтвердження про публікацію статті у журналі «Системи обробки інформації»

ДОДАТОК В

ТЕКСТ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ FLUTTER ЗАСТОСУНКУ

```
@RoutePage()
class HomePage extends StatelessWidget implements AutoRouteWrapper {
  const HomePage({super.key});

  @override
  Widget wrappedRoute(BuildContext context) {
    return BlocProvider(
      create: (context) => locator.get<HomeCubit>()..init(),
      child: const HomePage(),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(AppStrings.students),
      ),
      body: BlocConsumer<HomeCubit, HomeState>(
        listenWhen: (previous, current) => previous.status !=
current.status,
        listener: (context, state) {
          if (state.status == HomeStatus.error) {
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(
                content: Text(state.errorMessage),
                behavior: SnackBarBehavior.floating,
                backgroundColor:
Theme.of(context).colorScheme.onErrorContainer,
              ),
            );
          }
        },
        builder: (context, state) {
          if (state.status != HomeStatus.initial) {
            return const Center(
              child: CircularProgressIndicator(),
            );
          }

          return HomeBodyView(students: state.students);
        },
      ),
    );
  }
}

class HomeBodyView extends StatelessWidget {
  final List<Student> students;
```

```
  const HomeBodyView({
    super.key,
    required this.students,
  });

  @override
  Widget build(BuildContext context) {
    WidgetsBinding.instance.addPostFrameCallback((_) {
      log('Frame rendered');
    });

    return SafeArea(
      top: false,
      child: ListView(
        padding: const EdgeInsets.all(16),
        children: students.map((student) {
          return Container(
            margin: const EdgeInsets.all(4),
            padding: const EdgeInsets.symmetric(
              vertical: 8,
              horizontal: 12,
            ),
            decoration: BoxDecoration(
              color: Theme.of(context).colorScheme.surfaceContainer,
              borderRadius: BorderRadius.circular(16),
            ),
            child: Row(
              children: [
                ClipOval(
                  child: CachedNetworkImage(
                    height: 100,
                    width: 80,
                    fit: BoxFit.cover,
                    imageUrl: student.image,
                    errorWidget: (context, url, error) => Container(
                      height: 100,
                      width: 80,
                      color:

Theme.of(context).colorScheme.surfaceContainerHighest,
                    ),
                  ),
                ),
                SizedBox(width: 24),
                Expanded(
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text(
                        student.name,
                        style:
Theme.of(context).textTheme.headlineSmall,
```

```
                        ),
                        Text(
                          student.alternateNames.join(', '),
                          style:
Theme.of(context).textTheme.labelSmall?.copyWith(
                                color: Theme.of(context).dividerColor,
                              ),
                        ),
                      ],
                    ),
                  ),
                  SizedBox(width: 12),
                  Image.asset(
                    student.house.getImage(),
                    width: 50,
                    height: 50,
                  ),
                ],
              ),
            );
          }).toList(),
        ),
      );
  }
}
```

ДОДАТОК Г

ТЕКСТ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ REACT NATIVE ЗАСТОСУНКУ

```
export default function StudentsList ()  {
    const loading = useSelector(state => state.data.loading);
    const students = useSelector(state => state.data.students);
    const  dispatch = useDispatch();
    useEffect(() => {
        if (students.length > 0) {
            const start = performance.now();
            requestAnimationFrame(() => {
              const end = performance.now();
              console.log(`Render Time: ${end - start} ms`);
            });
        }
        dispatch(fetchStudents());
    }, [dispatch]);
    if (loading) {
        return (
            <View style={styles.centeredContainer}>
                <ActivityIndicator />
            </View>
        );
    }
    return (
        <SafeAreaProvider>
            <SafeAreaView style={styles.container}>
            <View>
                <FlatList
                    data={students}
                    keyExtractor={item => item.id}
                    renderItem={({    item    })    =>    <StudentItem
student={item} />}
                    contentContainerStyle={styles.listContainer}
                />
            </View>
            </SafeAreaView>
        </SafeAreaProvider>
    );
};
const StudentItem = ({ student }) => {
    const getHouseImage = (house) => {
        switch (house.toLowerCase()) {
            case 'gryffindor':
                return require('../assets/houses/gryffindor.webp');
            case 'slytherin':
                return require('../assets/houses/slytherin.webp');
            case 'hufflepuff':
                return require('../assets/houses/hufflepuff.webp');
            case 'ravenclaw':
                return require('../assets/houses/ravenclaw.webp');
            default:
```

```
                return null;
            }
        };
        return (
            <View style={styles.card}>
                {student.image == '' ?  null : <Image source={{ uri:
student.image }} style={styles.image} />}
                <View style={styles.detailsContainer}>
                    <Text style={styles.name}>{student.name}</Text>
                    <Text
style={styles.alternateNames}>{(student.alternate_names            ||
[]).join(', ')}</Text>
                </View>
                <Image               source={getHouseImage(student.house)}
style={styles.houseImage} />
            </View>
        );
};
const styles = StyleSheet.create({
    card: {
        flexDirection: 'row',
        backgroundColor: '#f1f1f1',
        borderRadius: 16,
        marginBottom: 16,
        padding: 16,
        alignItems: 'center',
    },
    image: {
        width: 80,
        height: 100,
        borderRadius: 200,
    },
    detailsContainer: {
        flex: 1,
        marginLeft: 16,
    },
    name: {
        fontSize: 24,
    },
    alternateNames: {
        fontSize: 14,
        color: '#555',
    },
    houseImage: {
        width: 50,
        height: 50,
    },
});
```

ДОДАТОК Д

ТЕКСТ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ MAUI ЗАСТОСУНКУ

```
<ContentPage NavigationPage.HasNavigationBar="False"
             Title="Students List"
             xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiDiplomApp.MainPage"
             xmlns:local="clr-namespace:MauiDiplomApp"
             xmlns:convert="clr-namespace:MauiDiplomApp.Converters"
             xmlns:model="clr-namespace:MauiDiplomApp.Model"
             >
    <ContentPage.Resources>
        <convert:HouseImageConverter x:Key="houseImage"/>
        <convert:StringJoinConverter x:Key="stringJoin"/>
        <convert:StringNotEmptyConverter x:Key="stringNotEmpty"/>
    </ContentPage.Resources>
    <Grid RowDefinitions="Auto, *">

        <CollectionView ItemsSource="{Binding Students}" Grid.Row="1"
Margin="8">
            <CollectionView.ItemsLayout>
                <LinearItemsLayout            Orientation="Vertical"
ItemSpacing="10"/>
            </CollectionView.ItemsLayout>
            <CollectionView.ItemTemplate >
                <DataTemplate x:DataType="model:Student">
                    <Frame Padding="10" Margin="8">
                        <Grid   ColumnDefinitions="Auto,   *,   Auto"
VerticalOptions="Center">

                            <Frame                    CornerRadius="40"
WidthRequest="80"           HeightRequest="100"           Padding="0"
IsClippedToBounds="True">
                                <Image    Source="{Binding    Image}"
WidthRequest="80" HeightRequest="100" Aspect="AspectFill"
                                       IsVisible="{Binding        Image,
Converter={StaticResource houseImage}}"/>
                            </Frame>
                            <VerticalStackLayout      Grid.Column="1"
Padding="10">
                                <Label     Text="{Binding     Name}"
FontSize="20" FontAttributes="Bold"/>
                                <Label                  Text="{Binding
Alternate_names,       Converter={StaticResource        stringJoin}}"
FontSize="14" TextColor="Gray"/>
                            </VerticalStackLayout>

                            <Image      Source="{Binding      House,
Converter={StaticResource houseImage}}"
                                WidthRequest="50"  HeightRequest="50"
Grid.Column="2"/>
```

```
                        </Grid>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </Grid>
</ContentPage>
public partial class MainPage : ContentPage
{
    private readonly StudentViewModel _viewModel;
    private readonly Stopwatch _stopwatch = Stopwatch.StartNew();

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

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

ДОДАТОК Е

ЗВІТ РЕЗУЛЬТАТІВ ПЕРЕВІРКИ НА УНІКАЛЬНІСТЬ ТЕКСТУ

StrikePlagiarism.com

Дата звіту 5/30/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗм-23-1_Червінська_А_Л_скорочений
Автор Науковий керівник / Експерт
Червінська Анастасія ЛюбомирівнаЄвген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

2.86% 2.86% КП 1

0.27% 0.27% КЦ

25 Довжина фрази для коефіцієнта подібності 2
6678 Кількість слів
52401 Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	¶	0
Парафрази (SmartMarks)	a	6

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://openarchive.nure.ua/bitstreams/d145af70-aec9-43dc-9b0c-4855131fb694/download	63 0.94 %
2	https://ir.nmu.org.ua/bitstream/handle/123456789/162381/122%D0%9C-21-1%20%D0%9A%D1%83%D0%BD%D0%B4%D0%B5%D0%BD%D0%BA%D0%BE.pdf?sequence=1	56 0.84 %
3	https://openarchive.nure.ua/bitstreams/d145af70-aec9-43dc-9b0c-4855131fb694/download	18 0.27 %
4	https://openarchive.nure.ua/bitstreams/d145af70-aec9-43dc-9b0c-4855131fb694/download	16 0.24 %

Рисунок Е.1 – Звіт результатів перевірки на унікальність тексту

ДОДАТОК Є
ЕКСПЕРТНИЙ ВИСНОВОК РЕЗУЛЬТАТІВ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА ВІДПОВІДНІСТЬ ОФОРМЛЕННЯ ВИМОГАМ ДСТУ 3008: 2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-1 (група)

Анастасія ЧЕРВІНСЬКА
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

04.06.2025

Рисунок Є.1 – Звіт результатів перевірки на відповідність оформлення вимогам