ДОДАТОК А

Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії

14. Falatiuk H., Shirokopetleva M., Dudar Z. Investigation of Architecture and Technology Stack for e-Archive System. 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv,Ukraine,8-11October2019.2019.,URL: https://doi.org/10.1109/picst47496.2019.9061407 (дата звернення: 30.05.2024).

15. DATA EXCHANGE MODEL IN THE INTERNET OF THINGS CONCEPT / I. Afanasieva et al. Telecommunications and Radio Engineering. 2019. Vol. 78,no. 10.P. 869–878.
URL: https://doi.org/10.1615/telecomradeng.v78.i10.30 (date of access: 30.05.2024).

16. Smelyakov S. V., Stoyan Y. G. Modelling of the space of paths in problems of constructing optimal trajectories. *USSR Computational Mathematics and Mathematical Physics*. 1983. Vol. 23, no. 1. P. 50–55.
URL: https://doi.org/10.1016/s0041-5553(83)80009-3 (дата звернення: 05.04.2024).

ДОДАТОК Б

СЛАЙДИ ПРЕЗЕНТАЦІЇ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Дослідження підходів до інтеграції картографічних систем у React Native застосунки

Коновалов Б. В., ІПЗм-22-5

Науковий керівник: проф. **Руткас А.Г.**

software engineering

Дослідження

Актуальність та стан розвитку галузі: Наше дослідження в галузі інтеграції картографічних систем у React Native застосунки є важливим, адже із зростом потреб на певний функціонал чи певні особливості картографічних SDK – зростає необхідність і в якісній інтеграції цих систем у мобільні React Native застосунки. Адже потреби користувачів та замовників можуть суттєво відрізнятись, а натомість оптимального методу інтеграції чи детальних досліджень – обмаль.

Чітке визначення напряму дослідження: Наше дослідження спрямоване на аналіз та порівняння веб-підходу та нативного підходу до інтеграції карт в мобільні застосунки. Мета полягає у визначенні, який метод найбільш ефективний у плані продуктивності, використання ресурсів та забезпечення оптимального досвіду користувачів.

Об’єкт дослідження: метод інтеграції картографічних систем у React Native застосунки.

software engineering

Постановка задачі

Постановка задачі: У ході нашої роботи буде проведено дослідження інтеграції картографічних систем у мобільні застосунки на платформі React Native, зосереджуючись на порівнянні веб-підходу та нативного підходу. Ми вивчимо, які є особливості у кожного підходу та обмеження і як це впливає на продуктивність у застосунку на користувацьких пристроях.

Методологія

Опис використаних методів дослідження: Заміри проведені в продакшн, виміряно CPU, Пам'ять, FPS, RAM, UI/UX за допомогою бенчмарк систем.

Інструментарій та технології, використані в роботі:

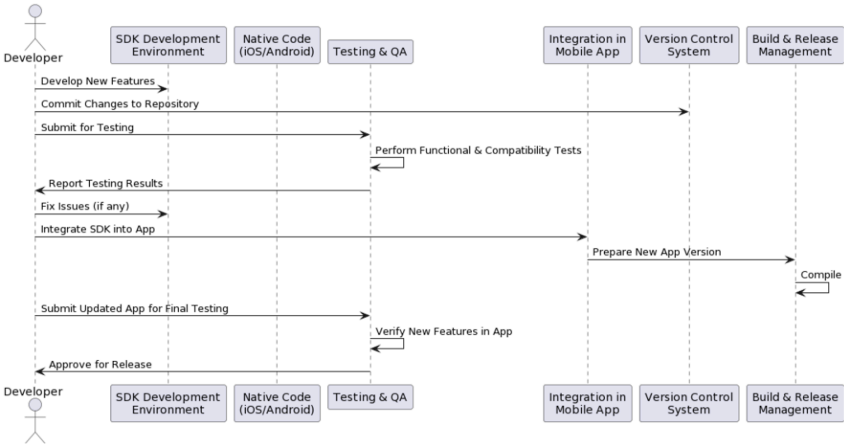
- Платформи: React Native
- Засоби тестування: Bundle Size, MacBook Screen Recorder, Android Studio Profiler, XCode profiler.
- Мова програмування: TypeScript, Swift, Java
- Бібліотеки: react-native-maps

Дослідження проблематики

- Було досліджено аналоги
- Досліджено приклади інтеграції існуючих SDK
- Було виявлено недостачу інформації про інтеграцію картографічних систем
- Було виявлено необхідність у тестуванні двох підходів інтеграції

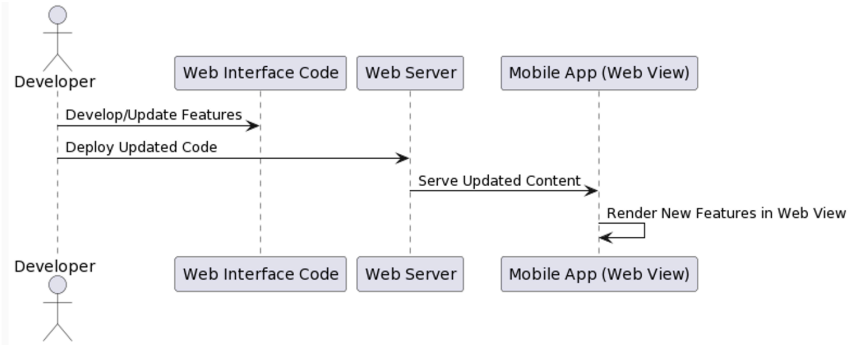
Дослідження підходів через SDK

- Було розглянуто та детально досліджено методи інтеграції через SDK.



Дослідження підходів Web підхід

- Було розглянуто та детально досліджено методи інтеграції через Web середовище.



Проведення дослідження

У нашому дослідженні ми застосовуємо комплексний підхід для оцінки продуктивності інтеграції картографічних систем у мобільні застосунки React Native, використовуючи два різні методи: веб-в'ю та нативний SDK. Наш вибір зосереджується на ключових метриках: розмір пакета, використання CPU та пам'яті, кадри на секунду (FPS), час відгуку та час запуску додатку.

Нами буде визначено наступні метрики:

- Bundle size;
- FPS;
- CPU;
- RAM.

РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ МЕТРИК

Для розрахунку зниження конверсійної ставки (CR) через збільшення розміру додатка використовується наступна формула:

$$\triangle CK = \left(\frac{\triangle S}{6}\right) \times 1\%$$

$\triangle CK$ – зміна конверсійної ставки;

$\triangle S$ – збільшення розміру додатка (в МБ).

Для визначення впливу FPS на задоволеність користувачів можна використовувати наступну формулу:

$$\triangle S = \left(\frac{\triangle FPS}{10}\right) \times 20\%$$

$\triangle S$ – зміна рівня задоволеності користувачів;

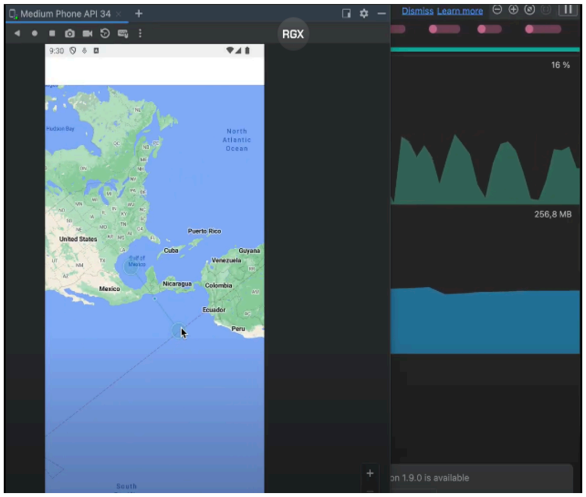
$\triangle FPS$ – зміна кількості кадрів в секунду.

Результати тестування Android

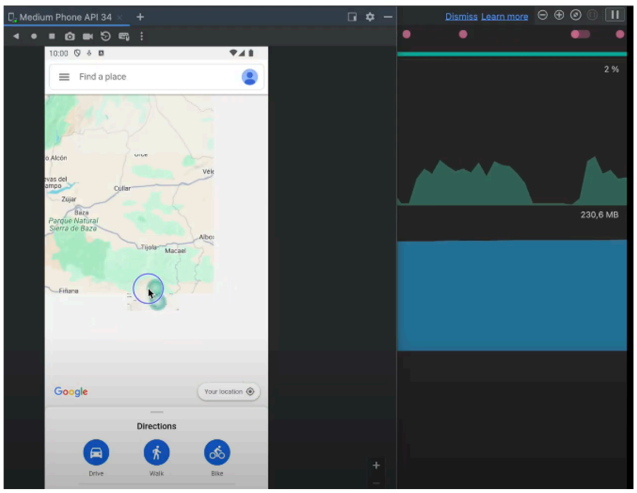
Bundle size y web

Bundle size y native

Результати тестування Android

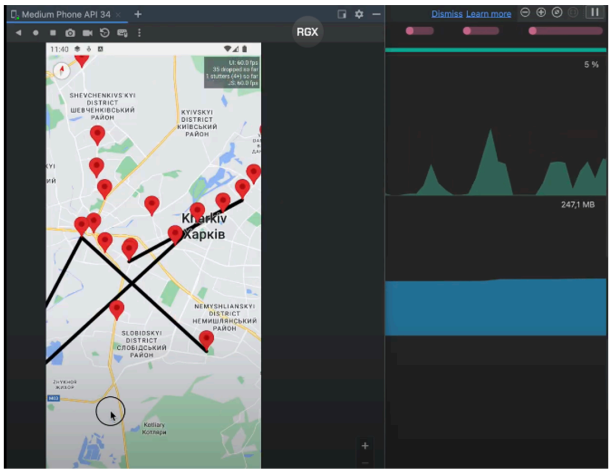


CPU y web

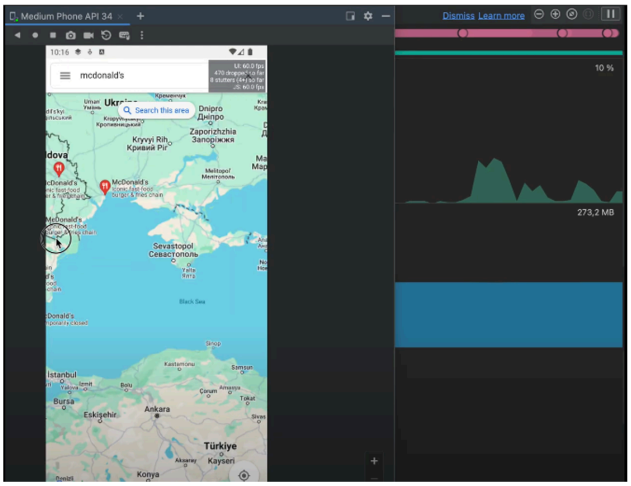


CPU y native

Результати тестування Android



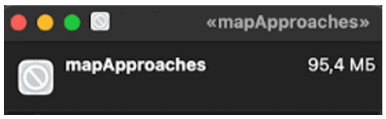
FPS y web



FPS y native

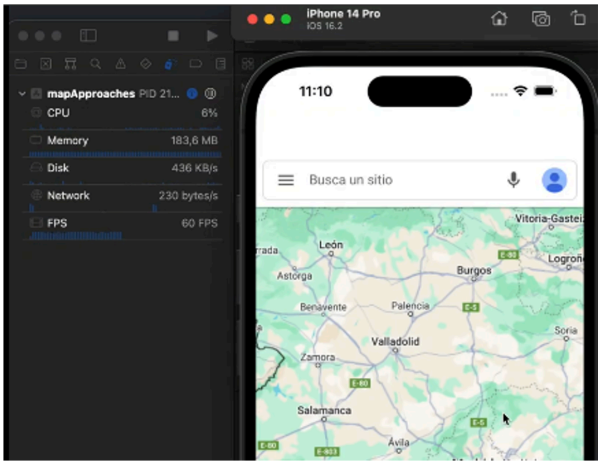
Результати тестування IOS

Bundle size y web

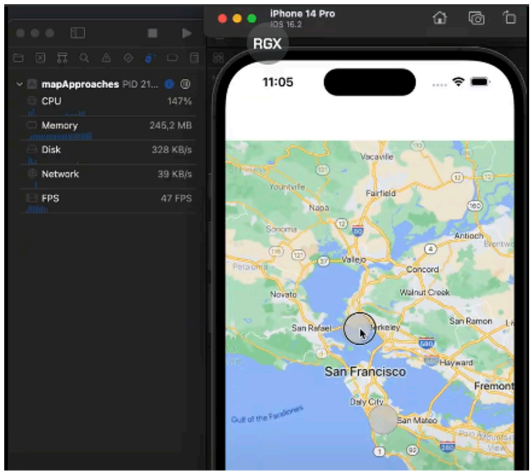


Bundle size y native

Результати тестування IOS

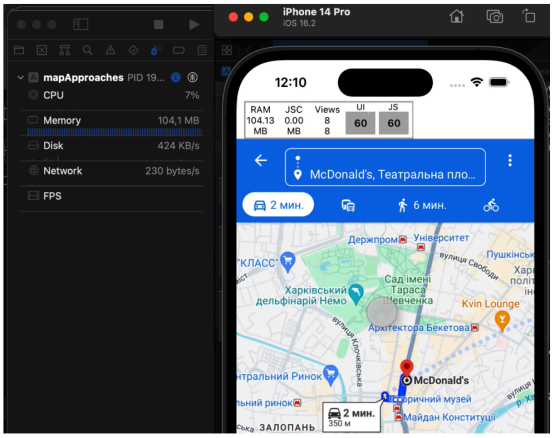


CPU y web

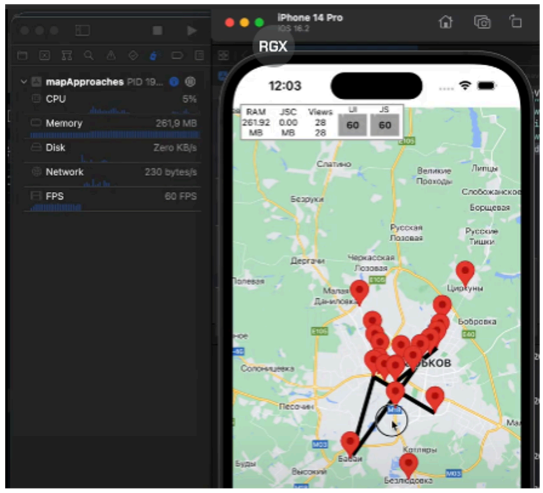


CPU y native

Результати тестування IOS



FPS y web



FPS y native

АНАЛІЗ РЕЗУЛЬТАТІВ

Платформа	Метрика	Веб підхід	Нативний підхід
Android	Bandle size	21.1 МБ	21.8 МБ
	FPS	59 FPS	55 FPS
	CPU	18 % завантаженості	40 % завантаженості
	Холодний старт	4.5 с.	4.2 с.
IOS	Bandle size	72 МБ	98 МБ
	FPS	60 FPS	45 FPS
	CPU	10 % завантаженості	35 % завантаженості
	Холодний старт	3.8 с.	1.9 с.

РЕЗУЛЬТАТИ ПЕРЕВАГИ НАТИВНОГО ПІДХОДУ

Платформа	Метрики	Результат втрати користувачів
Android	конверсія	0.12%
	FPS	8%
IOS	конверсія	4.33%
	FPS	30%

Висновки

- По результатах, можна зробити висновок що веб-в'ю підхід є більш підходящим для швидкої розробки та простих застосунків, особливо коли команда обмежена у ресурсах. Він дозволяє легко оновлювати контент, має гнучкість у дизайні та спрощує процес розробки. Проте, він може бути менш продуктивним і мати обмеження у використанні нативних функцій пристрою.
- Натомість SDK підхід вимагає більше часу та технічної експертизи для розробки, але пропонує кращу продуктивність, більші можливості кастомізації та повну інтеграцію з навивними можливостями пристрою. Цей підхід краще підходить для складних додатків та команд з досвідом у роботі з нативними платформами.

ДОДАТОК В
АПРОБАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

April 5, 2024 • Valencia, Kingdom of Spain • Collection of scientific papers «SCIENTIA»

SECTION 7.
COMPUTER AND SOFTWARE ENGINEERING

Bohdan Konovalov
Student at the Faculty of Computer Science
Kharkiv National University of Radio Electronics, Ukraine

Research supervisor: Anatolii Rutkas
Doctor of Physical and Mathematical Sciences, Professor,
Chair Professor of Software Engineering Department
Kharkiv National University of Radio Electronics, Ukraine

REVIEW OF APPROACHES TO INTEGRATING CARTOGRAPHIC USER INTERFACES INTO REACT NATIVE MOBILE APPLICATIONS

React Native is a cross-platform framework for mobile application development. In 2013, when the Facebook team began its creation, the main goal was to provide users with the ability to interact with their service not through a mobile browser, but through a separate application on their smartphone. This was intended to address the problem of underdeveloped mobile browsers, on which the social network could not achieve the same UI/UX performance as on the desktop web version [1]. When the opportunity arose to generate native interface elements using the JavaScript language, the team realized that they could leverage their flagship React library in the mobile development world. Such a combination allowed for the use of the same code for both iOS and Android platforms, which would speed up development, make the design and content of the application identical [2]. Additionally, website developers could now implement features in the mobile world as well, thanks to the identical programming language and similar paradigms. Thus, React Native was born.

As we can see, nothing was mentioned about interaction with cartographic systems. The focus was on simple user interface elements such as buttons, lists, blocks, text, etc. Has anything changed now?

According to the latest public activity of Meta (formerly Facebook) team representatives, we observe quite frequent emphasis on the concept of "Lean Core" in React Native. This means that the framework should implement only the most necessary things that will be used by developers in any project in 90% of cases, while everything with less common usage will be outsourced to separate libraries maintained by the community [3].

Therefore, from the above, we can conclude that cartography in React Native has not been and will not be built-in. So, how does such a popular framework for 11 years enable developers to integrate maps into their applications? As mentioned above - through separate JavaScript libraries.

Currently, in the world of React Native, there are the following more or less stable cartographic libraries: react-native-maps, @rnmapbox/maps. The choice is quite limited, considering the fact that they all provide a different set of features, have different architectures, and limitations. Also, not all popular map provider systems are represented in our list. And here we come to the main problem: What if the library of the required provider is missing or not suitable for various reasons?

To find the answer to this question and consider approaches to its resolution, it is necessary to understand what constitutes a cartographic library for React Native. It consists of two layers: the visualization layer and the control layer. Approaches are built depending on visualization, methods - depending on how we plan to exercise control. To see the approaches, we first need to clarify what types of cartographic systems can exist (or, in simpler terms, be displayed) in the mobile environment.

In the mobile world, there can be 2 types of cartographic systems:

1) built for native (iOS/Android) environments;
2) built for browser (Web) environments;

From this, we can draw a logical conclusion. If a cartographic system is not designed for native or browser environments, meaning it does not work in a browser and on iOS/Android mobile phones, then it cannot be integrated into React Native.

Now let's consider our approaches. Let's start with integrating the cartographic system through the native environment.

As we remember from the first paragraph, React Native combines two worlds: the JavaScript world and the native world (iOS/Android). Currently, React Native has the capability to build software for Windows, watchOS, tvOS, and even visionOS, but for the purposes of this work, we are only considering the operating systems that run on mobile phones - iOS and Android. Therefore, if React Native communicates with the native world to reproduce simple things, can it also allow developers to interact with something more complex, such as cartographic systems? Indeed, React Native enables developers to communicate with the native world independently. Within the scope of this work, we do not explore the possibilities of the new React Native architecture because at the time of writing, it is in an experimental mode. Currently, the implementation of such interaction is possible through two entities: Native Module [4] and Native Component [5]. One of the main differences between these two lies in whether we need to visually display something to the user or just retrieve data from the native world. In the case of a map, we need to display the map, so we are interested in the second entity.

Implementation of a Native Component for the cartographic system entails having SDKs (Software Development Kits) for iOS and Android. It's the SDK in the native environment that we will control through our entity. And it's the SDK that will display our map on the smartphone screen. This approach has the following advantages:

- canonicality, meaning this is how integrating maps is recommended by the documentation and the React Native community [5];
- more mobile phone functions are provided by Mobile SDKs;
- better performance since Mobile SDKs can utilize all phone resources and are not limited to browser APIs (Application Programming Interfaces) alone;
- React Native provides all necessary methods for effective and consistent control over Mobile SDKs (props, synchronous and asynchronous commands, UI element reconciliation mechanism with optimizations, etc.);
- flexibility in implementing application functions, building cartographic integration architectures;
- cartographic system design adapted to mobile standards.

Among the disadvantages, we can mention:

- developers need to have programming experience for iOS and Android.

Let's move on to the second approach.

As we know, browsers are widely used on smartphones. Today, every browser has a built-in mechanism - canvas, which allows working with various 2D graphics. It is thanks to this tool that cartographic system providers create their libraries for browsers. To use the browser in a mobile application, you need to use the WebView component. By the way, it is also provided as an RN library recently according to "Lean Core". For programmatic interaction with the map (control)

within this approach, we have three methods: 1) the "injectedJavaScript" property or "injectJavaScript" method, 2) "postMessage", 3) "onMessage". This approach will have the following advantages:

- no need for skills to program for iOS and Android (for the average RN developer, programming in JavaScript in the browser is more understandable);
- the browser regulates the use of smartphone resources, and developers will have fewer optimization problems;
- faster initial integration time, launching a native project requires more time and resources.

The following disadvantages are present:

- limited number of functions available through the browser API;
- worse performance;
- limited set of control tools, which does not allow building projects with heavy loads on the cartographic system fully;
- design not adapted to mobile standards.

In conclusion, it is important to note that the majority of existing cartographic libraries in React Native are implemented using the first (native) approach. The second approach can be used as a temporary alternative at the beginning of a project or at the MVP (Minimum Viable Product) stage. Also, the browser-based approach may be acceptable if there is no mobile implementation for the cartographic system or if there is a lack of programming skills for iOS/Android within the team. In all other cases, the native approach should be the foundation, as it is capable of handling cartographic integrations of maximum complexity, providing developers with all the necessary control tools, which ensures long-term solutions.

References:

1. Occhino T. React Native: Bringing modern web techniques to mobile. *Engineering at Meta*. URL:
2. https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/ (date of access: 14.03.2024).
3. Varshneya K. The History of React Native: Facebook's Open Source App Development Framework. *Techahead*. URL:
4. https://www.techaheadcorp.com/knowledge-center/history-of-react-native/ (date of access: 14.03.2024).
5. Sciandra L. The New React Native Architecture Explained: Part Four. *Nearform Commerce*. URL: https://commerce.nearform.com/blog/2019/lean-core-part-4/ (date of access: 14.03.2024).
6. Native Modules Intro. *React Native Official Documentation*. URL: https://reactnative.dev/docs/native-modules-intro (date of access: 14.03.2024).
7. iOS Native UI Components. *React Native Official Documentation*. URL: https://reactnative.dev/docs/native-components-ios (date of access: 14.03.2024).

ДОДАТОК Г

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-22-5 (група)

Коновалов Богдан Володимирович

(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	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	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	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

15.06.2024

ДОДАТОК Д

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

UNICHECK
by Turnitin

Ім'я користувача:
Нечволод Вадим Юрійович каф. ПІ

ID перевірки:
1016351303

Дата перевірки:
12.06.2024 12:29:54 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2024 13:12:01 EEST

ID користувача:
94949

Назва документа: 2024_М_ПІ_ІПЗм-22-5_Коновалов_Б_В_скорочений

Кількість сторінок: 51 **Кількість слів:** 7898 **Кількість символів:** 59260 **Розмір файлу:** 2.61 MB **ID файлу:** 1016154962

0.87%
Схожість

Найбільша схожість: 0.3% з джерелом з Бібліотеки (ID файлу: 1016134939)

0.46% Джерела з Інтернету	4	Сторінка 53
0.84% Джерела з Бібліотеки	48	Сторінка 53

0% Цитат

Не знайдено жодних цитат

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел