ДОДАТОК А

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

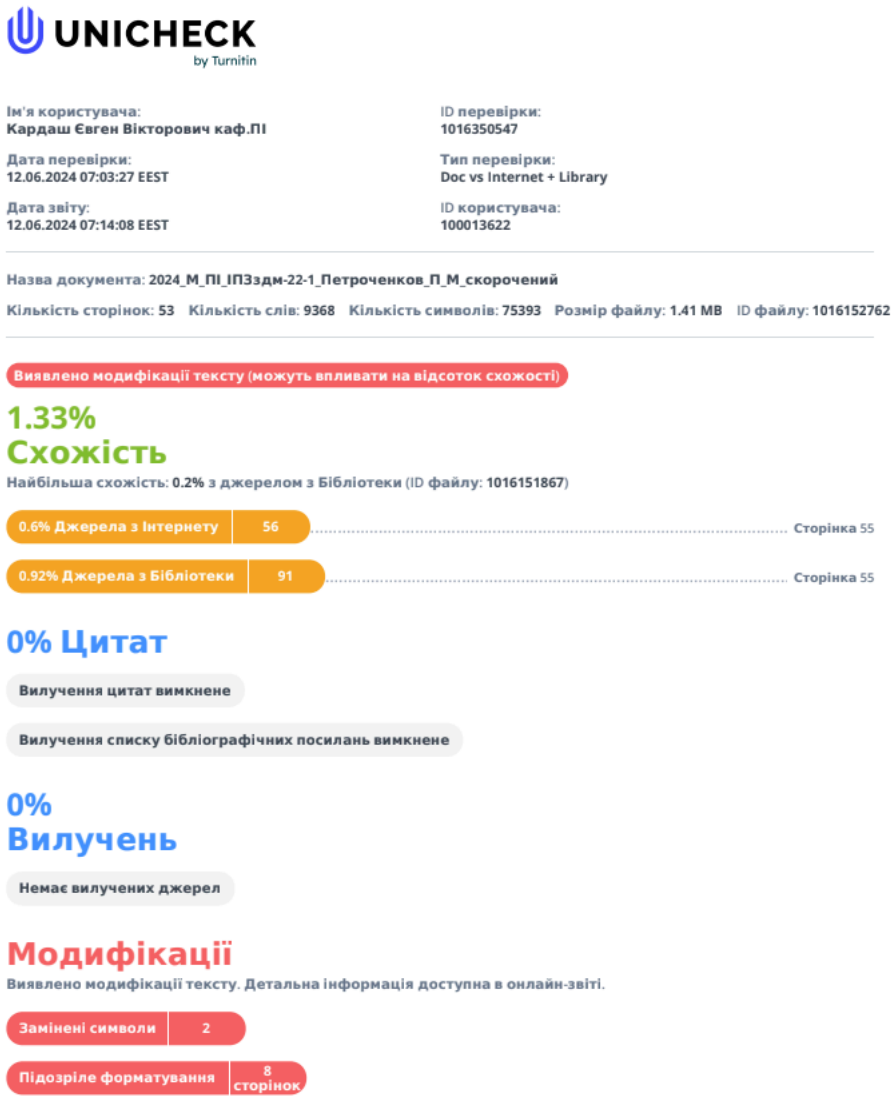
13. Daria Pronina, Iryna Kyrychenko. Comparison of Redux and React HooksMethods in Terms of Performance. 6th International Conference onComputational Linguistics and Intelligent Systems (COLINS-2022), 2022, Gliwice, Poland. – CEUR Workshop Proceedings 3171, Volume I: Main, PP. 791 – 800.

29. Smelyakov, K., Datsenko, A., Skrypka, V., Akhundov, A., The efficiency of images reduction algorithms with small-sized and linear details Smelyakov, K., Datsenko, A., Skrypka, V., Akhundov, A. 2019 IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 - Proceedings, 2019, стр. 745–750, 9061250 DOI 10.1109/PICST47496.2019.9061250

31. Tereshchenko, I., Tereshchenko, A., Bilous, N., Shtangey, S., Warsza, Z.L. Risk Analysis Method by the Extreme Data of Dependent Exogenous Variables Tereshchenko, I., Tereshchenko, A., Bilous, N., Shtangey, S., Warsza, Z.L. Journal of Automation, Mobile Robotics and Intelligent Systems, 2021, 2021(3), pp. 44–53

ДОДАТОК Б

Звіт з результатами перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016350547

Дата перевірки:
12.06.2024 07:03:27 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2024 07:14:08 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗздм-22-1_Петроченков_П_М_скорочений

Кількість сторінок: 53 Кількість слів: 9368 Кількість символів: 75393 Розмір файлу: 1.41 MB ID файлу: 1016152762

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.33%
Схожість

Найбільша схожість: 0.2% з джерелом з Бібліотеки (ID файлу: 1016151867)

0.6% Джерела з Інтернету 56 Сторінка 55

0.92% Джерела з Бібліотеки 91 Сторінка 55

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 2

Підозріле форматування 8 сторінок

ДОДАТОК В

Апробація роботи

УДК 004.42; 004.62

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ТА ПРИСКОРЕННЯ РЕНДЕРА ЕЛЕМЕНТІВ FLATLIST У МОБІЛЬНОМУ ДОДАТКУ

Петроченков П.М.
Науковий керівник: Широкопетлєва М.С.
Харківський національний університет радіоелектроніки, Харків, Україна
+38 066 5158834, pavlo.petrochenkov.cpe@nure.ua

The study thoroughly examines enhancing the productivity of mobile applications, with a focus on optimizing lists in React Native. It proposes analyzing the existing implementation of FlatList and developing a more efficient version through enhancements such as optimizing API usage and list item handling. Performance measurements are suggested to evaluate each FlatList version, with conclusions and recommendations provided in a table format. The characteristics of FlatList impacting performance are outlined, including virtualization, item caching, and lazy loading. Despite its advantages, FlatList has drawbacks such as manual key management and inefficiencies with dynamic data rendering. Approaches for improving FlatList performance are proposed, including memory management enhancement, advanced virtualization techniques, and optimized item rendering. These strategies aim to achieve better virtualization and performance optimization compared to the standard FlatList component, providing more efficient resource utilization and improved interaction with large data sets.

Мобільні застосунки розробляються з використанням різних ПЗ, в тому числі фреймворку React Native. Питання підвищення продуктивності – це питання оптимізації роботи окремих елементів, зокрема списків, навігації, анімації, елементів прокрутки, так і застосунків в цілому які використовують JavaScript як головний рушій [1].

В роботі пропонується дослідити можливості підвищення продуктивності роботи зі списками в програмних застосунках, розроблених з використанням фреймворку React Native, тобто метою роботи є проведення аналізу вже існуючої імплементації списку FlatList та визначення шляхів підвищення продуктивності FlatList за рахунок вдосконалення роботи з елементами списку та API FlatList.

FlatList є компонентом для відображення списку який надається фреймворком React Native [2]. Зазвичай він використовується для відображення великої кількості даних, бо FlatList в React Native має декілька способів підвищення продуктивності рендерингу даних таких як віртуалізація списку, кешування та інше.

Визначимо основні характеристики FlatList у React Native, які впливають на продуктивність:

Рисунок В.1 – Перша сторінка тез доповіді на 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»

- Віртуалізація: FlatList виконує рендеринг лише видимих елементів, що дозволяє оптимізувати використання пам'яті та підвищити продуктивність, особливо при роботі з великими списками даних.
- Кешування елементів: FlatList автоматично кешує елементи, які були попередньо оброблені для зменшення часу рендерингу при прокручуванні списку.
- Підтримка лінивого завантаження (lazy loading): дозволяє оптимізувати використання ресурсів та підвищити швидкість завантаження списків з великим обсягом даних [3].

Хоча FlatList є потужним інструментом для відображення списків даних у мобільних додатках, він також має деякі недоліки:

- Потреба ручного управління ключами елементів: для забезпечення правильної роботи ререндерингу елементів FlatList необхідно надавати унікальні ключі для кожного елемента. Це може бути важким у випадку, коли дані мають змінюватися або сортуватися, оскільки ключі також повинні відповідно змінюватися.
- Неефективне відображення динамічних списків зі змінним обсягом даних: якщо обсяг даних у FlatList постійно змінюється, наприклад, під час динамічного завантаження, це може призвести до некоректного відображення та неефективного використання пам'яті.
- Залежність продуктивності виконання запитів від списків та відображення даних від версії операційної системи та технічних можливостей пристроїв: спостерігається зменшення продуктивності при зниженні обсягу оперативної пам’яті та частоти процесора.

React Native має відкритий код, що надає змогу проаналізувати та змінити не тільки API та зовнішні елементи списки але і змінювати на системному рівні поведінку FlatList.

Пропонуються такі підходи для досягнення підвищення продуктивності роботи із FlatList:

- Ефективне управління пам'яттю: вдосконалення алгоритмів управління пам'яттю, що дозволяють ефективніше використовувати ресурси пам'яті пристрою. Вдосконалення має містити зменшення фрагментації пам'яті та уникнення витоку пам'яті. Для досягнення даної задачі пропонується змінювати процес роботи як на рівні зовнішніх елементів за допомогою меморізації так і на рівні реалізації FlatList за допомогою зміни поведінки FlatList, пов’язаної із роботою та обробкою пам’яті.
- Покращена віртуалізація елементів: пропонується застосовувати підходи з асинхронним завантаженням даних та з використанням більш розвинутих інструментів пов’язаних з віртуалізацією великих об’ємів даних, що дозволяє ефективно відображати великі списки даних з меншим впливом на продуктивність. Для цього пропонується

Рисунок В.2 – Перша сторінка тез доповіді на 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»

використовувати сучасні State Management інструменти такі як Context API та Redux подібні системи [4].

- Оптимізований рендеринг елементів: пропонується використовувати оптимізований рендеринг елементів, що містить у собі надання розмірів блоків з даними для ефективнішого відображення при швидкій зміні та роботи елементів прокрутки. Для цього пропонується модифікувати як API FlatList та і саму реалізацію, а саме додати додаткові поля які будуть відповідати за розміри елементів списку, кількість елементів які відображаються в момент показу та змінити начальні налаштування FlatList з додаванням цих полів до реалізації рендерингу. Ця інформація повинна підвищити продуктивність рендерингу та полегшити прорахування розмірів та розміщення як елементів списку та і самого списку.

Ці функції та стратегії дозволяють досягти покращеної віртуалізації та оптимізації продуктивності у порівнянні зі стандартним компонентом FlatList, забезпечуючи більш ефективне використання ресурсів та покращену взаємодію з великими списками даних.

В даній роботі проведено дослідження можливостей підвищення продуктивності роботи зі списками в програмних застосунках, розроблених з використанням фреймворку React Native. Було проаналізовано FlatList та виявлено слабкі сторони даного списку, а також запропоновано методи для покращення роботи FlatList, такі як зміна та модифікація API FlatList, робота з елементами списку за допомогою Context API та модифікацію поведінки FlatList на рівні системи. Надалі пропонується провести заміри продуктивності різних реалізацій компонента FlatList для визначення впливу кожного із запропонованих в роботі методів та їх комбінацій на час завантаження елементів в залежності від навантажень системи та обсягу даних для завантаження.

Список використаних джерел:

1. JavaScript: The Definitive Guide, David Flanagan, 2020, p.706.
2. The React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications, Akshat Pol, 2019, p.237.
3. Introduction to Algorithms. Fourth Edition, Thomas H. Cormen, 2022, p.1312.
4. D.Pronina, I.Kyrychenko. Comparison of Redux and React Hooks Methods in Terms of Performance. 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2022), 2022, Gliwice, Poland. – CEUR Workshop Proceedings 3171, Volume I: Main, PP. 791 – 800.

Рисунок В.3 – Перша сторінка тез доповіді на 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»

ДОДАТОК Г

Слайди презентації

Рисунок Г.1 – Слайд 1

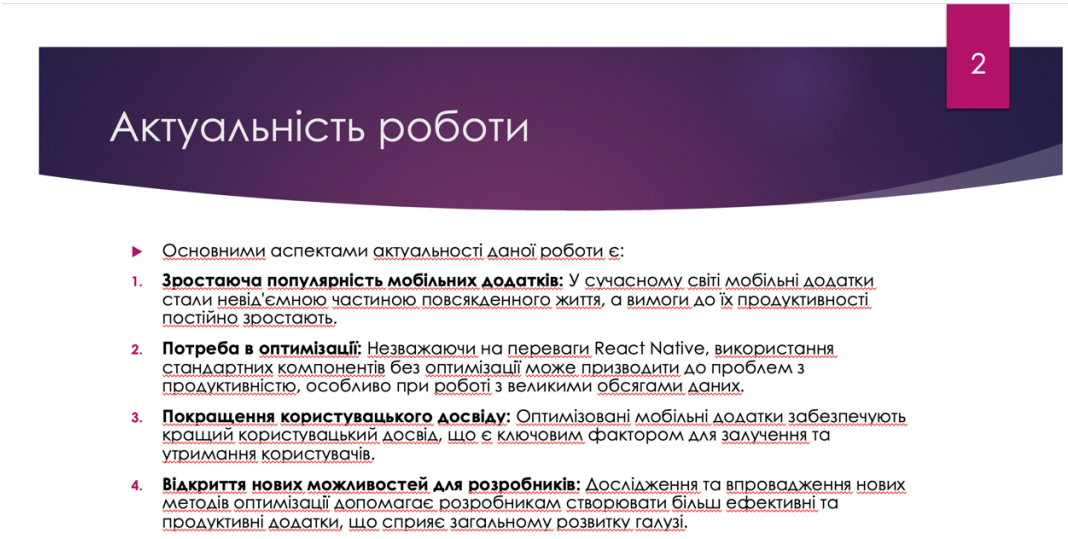


Рисунок Г.2 – Слайд 2

3

Опис об’єкта і предмета досліджень

- Об'єктом даного дослідження є мобільні додатки, розроблені з використанням фреймворку React Native. React Native є популярною технологією для створення кросплатформних додатків, яка дозволяє використовувати однокодові бази для різних платформ (iOS, Android). Головний елемент, що підлягає дослідженню, - це компонент FlatList, який використовується для відображення великих списків даних з можливістю прокручування.
- Предметом дослідження є методи та стратегії оптимізації рендерингу списків у мобільних додатках на базі React Native, зокрема:
  - стандартний FlatList,
  - різні комбінації оптимізацій,
  - альтернативні реалізації.

Рисунок Г.3 – Слайд 3

4

Постановка задачі

- Мета дослідження:
  - визначити властивості FlatList які мають вплив на перфоманс та швидкодію списку.
- Задачі дослідження:
  - визначити критерії для порівняння продуктивності роботи компонента FlatList,
  - розробка рекомендацій з оптимізації продуктивності роботи компонента FlatList,
  - експериментальна оцінка ефективності запропонованих рекомендацій.

Рисунок Г.4 – Слайд 4

5

Властивості які впливають на роботу FlatList

- Amount of Items
- Image
- Complex component
- Dynamic data
- Dumb element
- Key extractor
- Predefined sizes
- Recycling way of rendering
- Lazy loading
- Memoization
- Caching images

Рисунок Г.5 – Слайд 5

6

Метрики

- CumulativeBlankArea — це загальна порожня область, яку бачив користувач під час прокручування списку. Він вимірюється в пікселях і є сумою всіх проміжків між елементами, які ще не були відрендерені. Менше значення означає менше пустого місця та кращу продуктивність;
- Time loading – це час завантаження компонента. Менше значення означає скоріше завантаження.
- MaxBlankArea — максимальна порожня область, яку бачив користувач під час прокручування списку. Він також вимірюється в пікселях і є найбільшим розривом між елементами, які ще не були відрендерені. Менше значення означає менш помітний порожній простір і кращий досвід користувача.

Рисунок Г.6 – Слайд 6

7

Формули

- В роботі було використано дві формули.

Формула для вирахування середнього покращення кожної властивості наведеною нижче .

$$P_c = \frac{v_d - v_c}{v_d} * 100\%$$

де V_d – це Default значення при увімкненої властивості (наприклад isDumpElement);

V_c – це значення Current властивості;

P_c – це процент покращення Current властивості відносно Default.

Формула для прорахування середнього відсотку зміни

$$R = \frac{P_1 * \dots * P_k}{k}$$

де P_k – це процентне значення зміни перфомансу;

K– це кількість властивостей;

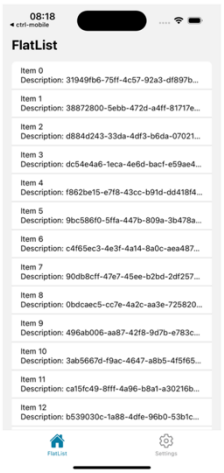
R – середній відсоток зміни характеристик для властивості.

Рисунок Г.7 – Слайд 7

8

Приклади налаштувань елементу списку для проведення експериментальних досліджень

Головна сторінка проекту



Сторінка налаштувань

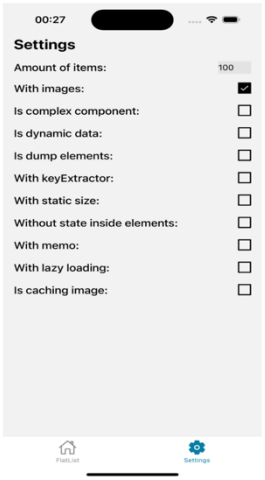


Рисунок Г.8 – Слайд 8

9

Конфігурації елементів списку

- Перша конфігурація – це список в якому усі елементи мають лише текст, та не мають ніяких додаткових ускладнень роботи
- Друга конфігурація – це список, який має зображення та увімкненою властивість isComplexData, яке в свою чергу додає навантаження на UI thread. В даному випадку це досягаються завдяки властивостям shadows. У реальному проекті – це може бути як складні стилі так і анімації на компонентах списку.
- Третя конфігурація – це список, такий самий як друга конфігурація, але з увімкненою властивістю isDynamicData. Що буде імітувати оновлення даних кожну секунду.

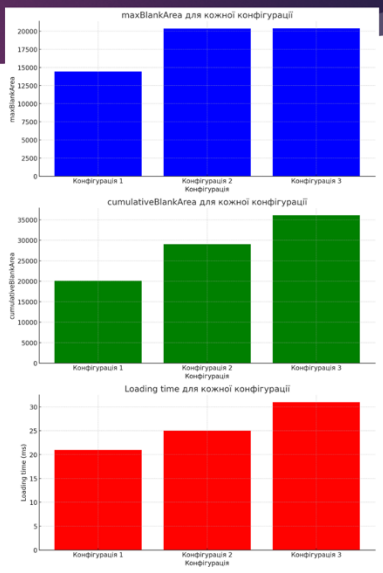


Рисунок Г.9 – Слайд 9

10

Графіки порівнянь властивостей

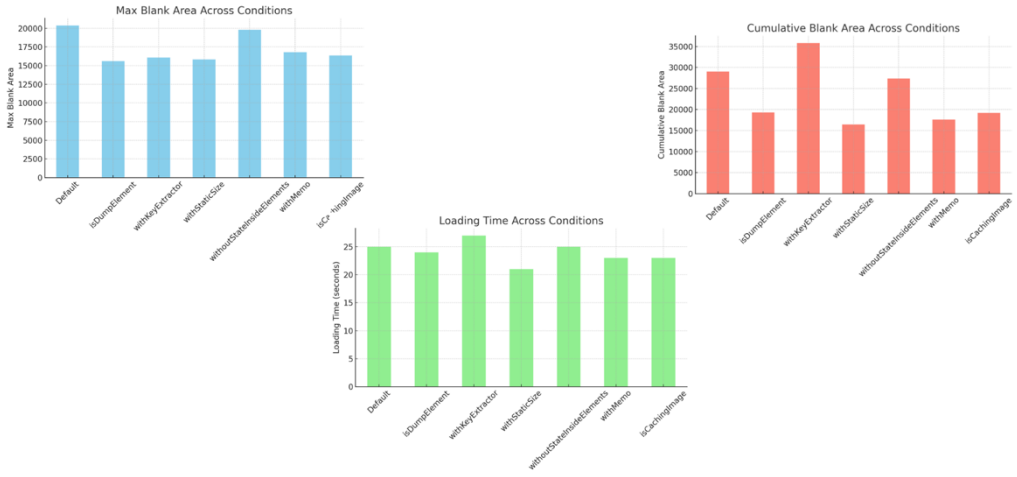


Рисунок Г.10 – Слайд 10

11

Графіки порівнянь комбінацій

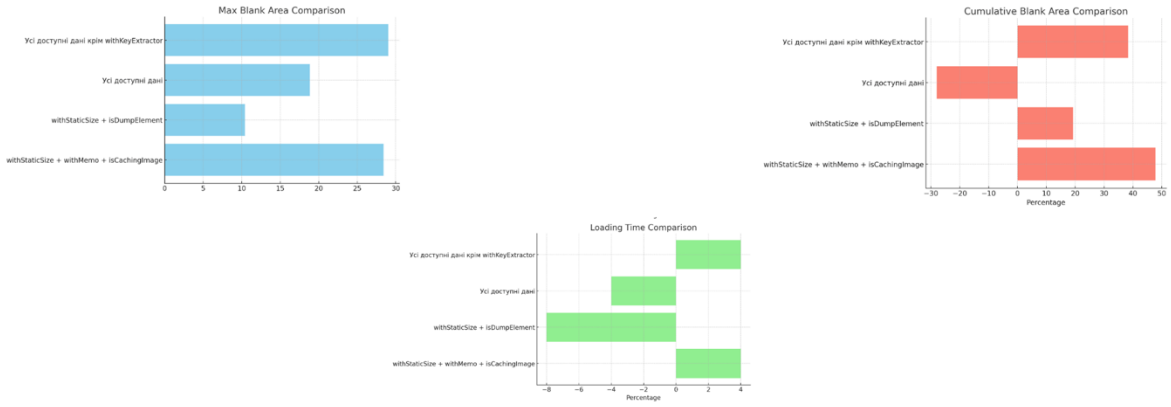


Рисунок Г.11 – Слайд 11

12

Рекомендації

- Властивості withStaticSize - показало найкращі результати (покращення на 27.28%), що вказує на ефективність фіксації розмірів елементів для оптимізації рендерингу.
- Властивість мемоізації (withMemo) та кешування зображень (isCachingImage) також показали значні покращення (понад 20%), що робить їх корисними для підвищення продуктивності UI.
- Слід звернути увагу на властивість withKeyExtractor, яка показала зниження продуктивності (-3.38%), що свідчить про її негативний вплив на рендеринг списків.
- Також було проаналізовано декілька комбінацій властивостей, та було виявлено, що комбінація “withStaticSize + withMemo + isCachingImage” властивостей має найкращий показник покращень середнього розміру списку, що складає 26.7% для середнього розміру списку.

Рисунок Г.12 – Слайд 12

13

Практична цінність

- Практична цінність дослідження:

- розробка і тестування різних конфігурацій FlatList дозволяє визначити найефективніші підходи до рендерингу списків у мобільних додатках на базі React Native. Це сприяє підвищенню продуктивності додатків, зменшенню часу рендерингу та оптимізації використання пам'яті.

- Використання результатів дослідження:

- результати дослідження можуть бути безпосередньо застосовані для оптимізації існуючих мобільних додатків на базі React Native. Використання різних конфігурацій FlatList, оптимізованих стратегій рендерингу та кешування зображень дозволить значно покращити швидкість роботи та загальну продуктивність додатків,

- отримані результати та рекомендації можуть бути впроваджені на етапі проектування та розробки нових мобільних додатків. Це забезпечить ефективне використання ресурсів та високу продуктивність додатків з самого початку.

Рисунок Г.13 – Слайд 13

14

Висновки по роботі

- В роботі було проаналізовано різні способи оптимізації списку та їх комбінації.
- Робота надає інформацію, щодо альтернативних готових рішень.
- В роботі надані рекомендації щодо використання списку та оптимізації використання FlatList в мобільних додатках, які можуть допомогти розробникам та бізнес-організаціям у прийнятті обґрунтованих рішень щодо вибору технологій для оптимізації та покращення продуктивності списку, що забезпечуючи оптимальну балансування між продуктивністю та доступністю.
- Результати дослідження було апробовано на 28-му Міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті».

Рисунок Г.14 – Слайд 14

ДЯКУЮ ЗА УВАГУ

Рисунок Г.15 – Слайд 15

ДОДАТОК Д

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимог ДСТУ 3008:2015

1

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗздм-22-1 (група)

Петроченков П. М.
(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	

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

15.06.2024