УДК 004.7 DOI: https://doi.org/10.30837/IYF.IIS.2024.459

# ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ТА ПРИСКОРЕННЯ РЕНДЕРА ЕЛЕМЕНТІВ FLATLIST У МОБІЛЬНОМУ ДОДАТКУ

Петроченков П. М.
Науковий керівник – ст. викл. Широкопетлєва М. С.
Харківський національний університет радіоелектроніки, ПІ
м. Харків, Україна
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:

– 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 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. Pronina D., Kyrychenko I. 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.