УДК 004.89 DOI: https://doi.org/10.30837/IYF.IIS.2024.421

ДОСЛІДЖЕННЯ ТА РОЗРОБКА ЕФЕКТИВНИХ СТРАТЕГІЙ ОПТИМІЗАЦІЇ ВЕБ-ДОДАТКІВ НА ОСНОВІ ТЕХНОЛОГІЇ REACT

Загнойко І. Ю.
Науковий керівник – професор Галуза О. А.
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна
e-mail: ihor.zahnoiko@nure.ua

This work addresses optimizing React web app performance by reducing unnecessary re-renders, optimizing data caching/loading, minimizing API requests, and decreasing bundle sizes. Key techniques like React.memo, useMemo, client-side caching, code splitting, and lazy loading are utilized. The proposed methods aim to decrease render times, improve responsiveness, reduce API requests, and accelerate load times for an enhanced user experience. Implementation leverages tools like Webpack, React Profiler, and Lighthouse. This comprehensive approach delivers more efficient, high-performing React apps.

Актуальність та постановка проблеми. Оптимізація продуктивності веб-додатків, створених із використанням фреймворку React, є актуальним завданням, оскільки швидкість роботи безпосередньо впливає на залученість і задоволеність користувачів. Основними проблемами, які потребують вирішення, є перерендеринги компонентів, неефективне кешування та завантаження даних, що призводить до зайвих запитів до серверного API, а також великі розміри JS та CSS бандлів, які уповільнюють завантаження та обробку сторінки. У цій роботі ставиться за мету дослідити та запропонувати ефективні стратегії оптимізації продуктивності React додатків. Розглядаються основні проблеми та шляхи їх вирішення на різних рівнях – від архітектури до окремих компонентів.

Методи та засоби вирішення проблеми. Оптимізація на рівні компонентів, а саме React.memo застосовується для запобігання непотрібних перерендерингів шляхом меморизації результатів обчислень пропсів. useMemo та useCallback використовуються для меморизації результатів важких обчислень та функцій, які передаються як пропси дочірнім компонентам. Перевірка на зміну пропсів та стану в shouldComponentUpdate дозволяє запобігти непотрібних перерендерингів класових компонентів.

Оптимізація запитів до серверного API, декілька запитів об'єднуються в один для зменшення кількості HTTP-запитів. GraphQL використовується для ефективного отримання лише необхідних даних. Відповіді API кешуються на клієнтській стороні для уникнення повторних запитів. При швидкому скролі запити до API відкладаються для уникнення затримок.

Оптимізація розміру JS та CSS бандлів: застосовується розбиття коду на чанки (code splitting) за допомогою динамічного імпорту для

завантаження лише необхідних частин додатку. Динамічне підвантаження компонентів відбувається за допомогою React.lazy для завантаження компонентів лише при необхідності. Невикористовуваний код виключається (tree shaking) за допомогою сучасних інструментів збірки. Мініфікація та стиснення коду зменшують розмір файлів.

Очікувані результати. Застосування запропонованих методів та засобів оптимізації продуктивності React додатків дозволить значно зменшити кількість непотрібних перерендерингів компонентів та обчислень, скоротивши час рендерингу та підвищивши відгук інтерфейсу. Завдяки оптимізації завантаження та кешування даних буде зменшена кількість зайвих запитів до серверного API, що призведе до прискорення відображення даних. Розмір JS та CSS бандлів зменшиться, що позитивно вплине на час першого рендерингу та прискорить завантаження та обробку сторінок.

Як результат, покращення швидкодії та відгуку додатку сприятиме підвищенню задоволеності користувачів. Загалом, застосування цих методів та засобів оптимізації дозволить створювати більш ефективні та продуктивні React додатки, орієнтовані на забезпечення кращого користувацького досвіду.

Висновок. Оптимізація продуктивності React додатків вимагає комплексного підходу на всіх рівнях – від архітектури до окремих компонентів. Застосування запропонованих методів та засобів, таких як React.memo, useMemo, useCallback, кешування даних, оптимізація запитів до API, розбиття коду на чанки, динамічне підвантаження компонентів, дозволить суттєво підвищити швидкодію та продуктивність як нових, так і вже існуючих проектів на React. Це, в свою чергу, позитивно вплине на залученість та задоволеність користувачів, забезпечивши плавний та відгукний користувацький досвід, що є надзвичайно важливим для успіху будь-якого веб-додатку.

Список використаних джерел:

1. Optimize Your React App: A Practical Guide to Better React Performance. URL: https://www.copycat.dev/blog/react-performance/ (дата звернення: 02.03.2024).

2. Gackenheimer C. Introduction to React. Berkeley: Apress, 2015. 188 p.

3. Research of Ways to Increase the Efficiency of Functioning Between Firewalls in the Protection of Information Web-Portals in Telecommunications Networks / Z. Dudar та ін. Current Trends in Communication and Information Technologies. Cham, 2021. С. 272–292. URL: https://doi.org/10.1007/978-3-030-76343-5_14 (дата звернення: 10.03.2024).