УДК 004.77

**В. П. Бусигіна**
**І. В. Богач**

# МЕТОДИ ТА ЗАСОБИ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ КЛІЄНТСЬКОЇ ЧАСТИНИ ВИСОКОНАВАНТАЖЕНИХ ВЕБДОДАТКІВ

Вінницький національний технічний університет

***Анотація***

*В цій роботі розглядаються сучасні підходи до підвищення продуктивності клієнтської частини вебдодатків, що є критично важливим для забезпечення якісного користувацького досвіду (UX) та ефективного SEO-ранжування. Основна увага приділяється аналізу архітектурних патернів, таких як клієнтський (CSR) та серверний рендеринг (SSR), а також методам зменшення обсягу переданих даних. Досліджуються технології лінивого завантаження, розділення коду та оптимізації медіаресурсів відповідно до метрик Core Web Vitals.*

**Ключові слова**: продуктивність вебдодатків, Core Web Vitals, SSR, CSR, розділення коду, ліниве завантаження, вебсистема.

***Abstract***

*This work examines modern approaches to improving the client-side performance of web applications, which is critically important for ensuring a high-quality user experience (UX) and effective SEO ranking. The primary focus is on analyzing architectural patterns such as client-side (CSR) and server-side rendering (SSR), as well as methods for reducing the volume of transmitted data. The research explores technologies for lazy loading, code splitting, and media resource optimization in accordance with Core Web Vitals metrics.*

**Keywords:** web application performance, Core Web Vitals, SSR, CSR, code splitting, lazy loading, web system.

## Вступ

Сучасні тенденції розвитку інтернету вимагають від вебдодатків не лише багатого функціоналу, але й високої швидкості роботи. Зростання частки мобільного трафіку та використання пристроїв з обмеженими апаратними можливостями роблять питання оптимізації продуктивності клієнтської частини (фронтенду) одним із пріоритетних завдань при розробці програмного забезпечення. Повільне завантаження сторінок призводить до зниження конверсії, погіршення показників утримання користувачів та падіння позицій у пошукових системах. Тому впровадження ефективних методів оптимізації є необхідною умовою для успішного функціонування будь-якої вебсистеми.

## Результати дослідження

Для об’єктивної оцінки продуктивності вебдодатків сьогодні стандартом є використання набору метрик Core Web Vitals, запропонованих консорціумом W3C та компанією Google. Вони включають ключові показники: швидкість завантаження основного контенту (LCP), час реакції на першу взаємодію користувача (INP) та візуальну стабільність інтерфейсу (CLS) [1]. Досягнення високих результатів за цими критеріями вимагає комплексного підходу, починаючи від етапу проєктування архітектури до налаштування процесів збірки фінального проєкту.

Одним із найпоширеніших джерел проблем із продуктивністю є використання виключно клієнтського рендерингу (CSR), що є характерним для класичних односторінкових додатків (SPA). При CSR браузер змушений завантажувати і виконувати великий обсяг JavaScript-коду перед тим, як користувач зможе побачити контент сторінки [2]. Ефективним вирішенням цієї проблеми є впровадження технології серверного рендерингу (SSR) або статичної генерації сайтів (SSG) за допомогою таких сучасних фреймворків, як Next.js. Застосування SSR дозволяє

генерувати готову HTML-розмітку на сервері, що суттєво зменшує показник LCP та пришвидшує індексацію сторінок пошуковими ботами [3]. У таких архітектурах клієнтський пристрій отримує вже сформований візуальний інтерфейс, після чого відбувається процес гідратації (hydration), який підключає обробники подій і робить сторінку інтерактивною.

Навіть за умови використання прогресивних архітектур, критично важливим залишається контроль над розміром JavaScript-бандлів. Для цього застосовується метод розділення коду (code splitting), який дозволяє розбити монолітний файл скриптів на дрібніші фрагменти. Ці фрагменти завантажуються модульно, лише коли користувач переходить на відповідну сторінку [4]. У комбінації з лінивим завантаженням (lazy loading) для зображень та некритичних компонентів інтерфейсу, цей підхід дозволяє радикально зменшити обсяг даних, які передаються мережею під час першого відвідування сайту [5].

Додатковим фактором успіху є правильна оптимізація статичних активів. Перехід на сучасні алгоритми стиснення даних, такі як Brotli, та використання оптимізованих форматів графіки (WebP, AVIF) забезпечує зниження навантаження на канал зв'язку без втрати візуальної якості [6]. Це особливо важливо для мобільних мереж з високою затримкою передачі пакетів.

Універсальність та потужність сучасних модульних збирачів (наприклад, Webpack або Vite) в оптимізації вебдодатків базується на їхній унікальній архітектурі, в центрі якої лежить абстрактне синтаксичне дерево (AST). На відміну від застарілих мініфікаторів, які лише видаляли пробіли та скорочували назви змінних, сучасний збирач спочатку повністю розбирає вихідний JavaScript-код і будує його логічну модель у пам'яті, що відображає всі залежності проєкту [5]. Це дозволяє програмі глибоко «розуміти» структуру додатку: які модули дійсно використовуються, де знаходяться динамічні імпорти, а де - недосяжний код. Завдяки такому підходу процес оптимізації бандлу стає інтелектуальним, оскільки розробник може втручатися в процес компіляції на етапі збірки, наприклад, автоматично виділяючи спільні бібліотеки в окремі файли (vendor chunks) або налаштовуючи специфічні стратегії кешування. Особливу роль відіграє вбудована система статичного аналізу та підтримка механізму «скидання гілок» (Tree Shaking), який автоматизує видалення невикористаного коду [4]. Під час розробки програмісти часто підключають масивні сторонні бібліотеки, що є досить типовим для сучасного ПЗ, проте збирач здатний проаналізувати ці записи і залишити лише ті функції, які фактично викликаються. Під час генерації фінальних клієнтських файлів програма автоматично переписує шляхи імпортів та створює оптимальні зв'язки між модулями, перетворюючи важкий монолітний скрипт на набір легких асинхронних фрагментів. Це докорінно змінює досвід користувача, адже перевантажений скриптами інтерфейс перетворюється на високопродуктивну систему, де сторінка завантажується миттєво, а показник часу до повної інтерактивності (TTI) зводиться до мінімуму [5].

Додатковим фактором успіху є правильна оптимізація статичних активів. Перехід на сучасні алгоритми стиснення даних, такі як Brotli, та використання оптимізованих форматів графіки (WebP, AVIF) забезпечує зниження навантаження на канал зв'язку без втрати візуальної якості [6]. Це особливо важливо для мобільних мереж з високою затримкою передачі пакетів.

## Висновок

В результаті дослідження методів та засобів оптимізації продуктивності вебдодатків було встановлено, що забезпечення високої швидкості роботи вимагає системного підходу на всіх етапах розроблення. Перехід від чистого клієнтського рендерингу до гібридних архітектур на базі SSR чи SSG є необхідним кроком для покращення метрик Core Web Vitals. Паралельне використання методів розділення коду, лінивого завантаження та сучасних алгоритмів стиснення ресурсів гарантує створення швидких, масштабованих та конкурентоспроможних вебсистем, здатних миттєво реагувати на дії користувача на будь-якому пристрої.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Web Vitals. [Електронний ресурс] – Режим доступу: https://web.dev/vitals/ (дата звернення: 20.03.2026).

2. Rendering on the Web. [Електронний ресурс] – Режим доступу: https://web.dev/rendering-on-the-web/ (дата звернення: 20.03.2026).

3. Next.js Architecture and SSR. [Електронний ресурс] – Режим доступу: https://nextjs.org/learn/foundations/about-nextjs (дата звернення: 20.03.2026).
4. Code Splitting in Modern Web Apps. [Електронний ресурс] – Режим доступу: https://react.dev/reference/react/lazy (дата звернення: 20.03.2026).
5. Lazy Loading. Mozilla Developer Network. [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Performance/Lazy_loading (дата звернення: 20.03.2026).
6. Image Optimization with WebP and AVIF. [Електронний ресурс] – Режим доступу:https://web.dev/fast/use-imagemin-to-compress-images/ (дата звернення: 20.03.2026).

***Бусигіна Вероніка Павлівна*** – студентка групи 1ПКТ-24б, кафедра комп'ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: veronikabysygina@gmail.com

***Богач Ілона Віталіївна*** – к.т.н., доцент, професор кафедри автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: ilona.bogach@gmail.com

***Busyhina Veronika Pavlivna*** – student of 1PCT-24b group, Department of Computer Science, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: veronikabysygina@gmail.com

***Bogach Ilona Vitaliivna*** – PhD, Associate Professor of Automation and Intelligent Information Technologies, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: ilona.bogach@gmail.com