Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Оцінювання сучасних методів
оптимізації швидкості завантаження веб-сторінок
(тема)

Виконав:
здобувач _2_ року навчання, групи _ПМм-23-2 _
Нагорець С.І.
(прізвище, ініціали)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник доц. Бринза Н.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Сидоров М.В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _113 Прикладна математика
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“ _25_ ” _листопада_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _Нагорцю Сергію Івановичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Оцінювання сучасних методів оптимізації швидкості завантаження веб-сторінок

затверджена наказом по університету від 22 листопада 2024 р. № 1223 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 6 січня 2025 р.

3. Вихідні дані до роботи _математична модель оцінювання швидкості завантаження веб-сторінок

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод оптимізації
5. Результати експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	25 листопада – 1 грудня 2024 р.	виконано
2	Вибір та обґрунтування методу	2 – 8 грудня 2024 р.	виконано
3	Розробка алгоритму і програми	9 – 22 грудня 2023 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	23 – 29 грудня 2024 р.	виконано
5	Робота над текстом пояснювальної записки	30 грудня 2024 р. – 9 січня 2025 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 січня 2025 р.	виконано

Дата видачі завдання _25_ листопада 2024 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Бринза Н.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 94 с., 4 табл., 15 рис., 1 дод., 17 джерел.

ВЕБ-САЙТ, КЕШУВАННЯ, КОМПРЕСІЯ, ОПТИМІЗАЦІЯ, РЕНДЕРИНГ, ШВИДКІСТЬ ЗАВАНТАЖЕННЯ, GOOGLE LIGHTHOUSE.

Об’єкт дослідження – сучасні методи оптимізації швидкості завантаження веб-сторінок.

Мета роботи – аналіз, оцінювання та порівняння сучасних методів оптимізації швидкості завантаження веб-сторінок.

Методи дослідження – системний аналіз, експериментальне тестування, порівняльний аналіз результатів оптимізації.

У роботі досліджено сучасні підходи до оптимізації швидкості завантаження веб-сторінок, зокрема такі методи, як зменшення кількості запитів, оптимізація зображень, застосування технік «lazy loading» та асинхронного завантаження, мінімізація розміру ресурсів, кешування тощо.

Проведено експериментальне оцінювання вибраних методів на тестовому веб-сайті із використанням популярного інструменту для аналізу продуктивності – Google Lighthouse.

На основі результатів розроблено рекомендації щодо застосування цих методів для покращення швидкості завантаження веб-сторінок.

Значимість роботи полягає в можливості покращення зручності користування веб-ресурсами та підвищенні SEO-показників.

Рекомендації щодо використання результатів роботи включають інтеграцію методів оптимізації в існуючі проєкти, використання їх для навчальних і дослідницьких цілей. Подальші дослідження можуть бути спрямовані на автоматизацію процесів оптимізації з використанням алгоритмів машинного навчання.

ABSTRACT

Introductory note: 94 pages, 4 tables, 15 figures, 1 appendix, 17 sources.

CACHING, COMPRESSION, GOOGLE LIGHTHOUSE, LOADING SPEED, OPTIMIZATION, RENDERING, WEBSITE.

Object of research – modern methods for optimizing web page loading speed.

Purpose of work – to analyze, evaluate, and compare modern methods for optimizing web page loading speed.

Methods of research – systematic analysis, experimental testing, and comparative analysis of optimization results.

The study explores modern approaches to optimizing web page loading speed, including methods such as reducing the number of requests, image optimization, implementing «lazy loading» and asynchronous loading techniques, minimizing resources, caching, and more.

Experimental evaluation of selected methods was conducted on a test website using a popular performance analysis tool – Google Lighthouse.

Based on the results, recommendations were developed for applying these methods to improve web page loading speed.

Significance of the work lies in the potential improvement of user experience with web resources and the enhancement of SEO metrics.

Recommendations for the use of the results include integrating optimization methods into existing projects and using them for educational and research purposes. Further research may focus on automating optimization processes using machine learning algorithms.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

CDN – Content Delivery Network (мережа доставки контенту);

CSS – Cascading Style Sheets (каскадні таблиці стилів);

DOM – Document Object Model (об'єктна модель документа);

HTTP/2 – HyperText Transfer Protocol Version 2 (протокол передачі гіпертексту версії 2);

SEO – Search Engine Optimization (оптимізація для пошукових систем).

UI – User Interface (користувацький інтерфейс);

UX – User Experience (користувацький досвід);

WebP – формат зображень із стисненням без втрат;

AVIF – AV1 Image File Format (формат зображень на базі кодека AV1).

ВСТУП

Актуальність теми. У сучасному світі веб-технології відіграють ключову роль у розвитку суспільства, економіки та інформаційного обміну. Швидкість завантаження веб-сторінок є критично важливим фактором, який впливає як на користувацький досвід, так і на комерційну ефективність веб-ресурсів. Дослідження показують, що затримка завантаження навіть у декілька секунд може значно знизити рівень залучення користувачів, призвести до втрати доходів і негативно вплинути на рейтинги у пошукових системах. Наприклад, за даними компанії Google, збільшення часу завантаження сторінки з 1 до 3 секунд підвищує ймовірність відмови користувачів на 32%.

Важливість оптимізації швидкості завантаження підтверджується також тенденціями глобальної цифровізації, коли кількість користувачів мобільного інтернету постійно зростає. В умовах обмеженої пропускної здатності мереж, особливо в регіонах із повільним інтернет-з'єднанням, швидкість завантаження веб-сторінок є ключовим показником ефективності веб-сайтів.

Пошук методів оптимізації швидкості завантаження веб-сторінок став важливим напрямом досліджень як у наукових колах, так і в практичній діяльності. Провідні наукові установи й організації, такі як W3C, а також корпорації, зокрема Google, Microsoft і Meta, активно працюють над вдосконаленням стандартів, технологій та інструментів для забезпечення швидкого завантаження веб-контенту.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є аналіз, оцінювання та порівняння сучасних методів оптимізації швидкості завантаження веб-сторінок. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі «оптимізації швидкості завантаження веб-сторінок»;

– дослідити методи та інструменти, що використовуються для оптимізації, включаючи стискання, кешування, оптимізацію зображень та інших ресурсів;

– оцінити ефективність існуючих методів на основі експериментального аналізу з використанням популярних інструментів тестування, таких як PageSpeed Insights, GTmetrix чи Google Lighthouse;

– розробити практичні рекомендації, на основі проведеного аналізу, щодо впровадження оптимізацій у реальних проектах.

Об'єктом дослідження є сучасні методи оптимізації швидкості завантаження веб-сторінок, зокрема кешування, компресії, мінімізації ресурсів та інші.

Предметом дослідження є сучасні підходи до зменшення часу завантаження сторінок, способи мінімізації затримок передачі даних між сервером і клієнтом, а також інструменти для оцінки ефективності цих методів.

Методи дослідження. У роботі використовуються методи системного аналізу, експериментального тестування, порівняльного аналізу результатів оптимізації для обґрунтування вибору оптимальних підходів.

Публікації. Результати, отримані у роботі, було представлено на III Міжнародній молодіжній науково-практичний конференції «Learning & Teaching: in the World after the War» (м. Харків, 8 листопада, 2024 р.) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз задачі оптимізації швидкості завантаження веб-сторінок

Швидкість завантаження веб-сторінок – це один із критичних факторів, який визначає загальну ефективність та привабливість веб-ресурсу. Це ключовий показник ефективності роботи веб-сайту, що безпосередньо впливає на користувацький досвід, рівень задоволеності користувачів та SEO (Search Engine Optimization) [2].

Поняття "швидкість завантаження" означає час, який потрібен для того, щоб веб-сторінка повністю завантажилась і стала доступною для взаємодії користувача з її елементами. Від цієї метрики залежить не тільки враження користувачів, але й успішність бізнесу, оскільки тривалі затримки можуть призвести до втрати відвідувачів та зниження рівня конверсій.

На швидкість завантаження впливають багато факторів. Багатокомпонентні сторінки з великою кількістю інтерактивних елементів або мультимедійного контенту можуть завантажуватися повільніше, що підкреслює важливість балансування між дизайном, функціональністю та швидкістю.

Важливість оптимізації швидкості завантаження виходить за межі лише технічних характеристик. Повільне завантаження може негативно вплинути на пошукову оптимізацію, оскільки пошукові системи, такі як Google, використовують швидкість сторінки як один із факторів ранжування.

Користувачі також очікують швидкого завантаження. Згідно з дослідженнями, більшість із них залишають сайт, якщо сторінка завантажується більше ніж 3 секунди. Таким чином, швидкість завантаження впливає не лише на видимість сайту в пошукових системах, але й на рівень конверсій та загальну ефективність веб-ресурсу в досягненні бізнесових цілей.

Детальне розуміння швидкості завантаження та її впливу на досвід користувачів допомагає приймати більш обґрунтовані рішення щодо оптимізації.

Проблема оптимізації швидкості завантаження веб-сторінок є надзвичайно важливою. Розглянемо ключові причини цього.

Причина 1. Користувацький досвід. Користувачі очікують швидкої роботи веб-сайтів. Затримка у завантаженні сторінки може призвести до незадоволення та втрати інтересу. Дослідження показують, що навіть затримка у декілька секунд може значно зменшити конверсії та збільшити кількість відмов (bounce rate).

Причина 2. SEO та ранжування в пошукових системах. Пошукові системи використовують швидкість завантаження як один з факторів ранжування веб-сторінок. Швидкі сайти мають більше шансів потрапити на вищі позиції в результатах пошуку, що підвищує їх видимість та трафік.

Причина 3. Конверсії та прибутки. Для комерційних сайтів швидкість прямо впливає на конверсії. Швидкий веб-сайт заохочує більше взаємодій та покупок. Наприклад, великі компанії, як Amazon чи Walmart, відзначають, що кожна секунда затримки у завантаженні може коштувати їм мільйони доларів.

Причина 4. Зниження використання ресурсів. Швидкі сторінки зазвичай легші та менш вимогливі до ресурсів, що зменшує навантаження на сервери та інтернет-канали, а також скорочує споживання енергії. Це важливо як для екологічної стійкості, так і для зменшення витрат на інфраструктуру.

Причина 5. Доступність для глобальної аудиторії. У багатьох регіонах світу доступ до високошвидкісного інтернету може бути обмеженим або дорогим. Оптимізовані веб-сторінки, які швидко завантажуються навіть на повільних з'єднаннях, роблять контент доступнішим для ширшої аудиторії.

Отже, оптимізація швидкості завантаження веб-сторінок є важливим аспектом сучасної веб-розробки, що впливає на успішність бізнесу, зручність користувачів та технічну ефективність.

Тенденції зростання мобільного трафіку та необхідність оптимізації для мобільних пристроїв також стали одними з ключових аспектів сучасної веб-розробки через кілька важливих факторів [3]. Розглянемо такі фактори.

Фактор 1. Зросла кількість користувачів мобільних пристроїв:

– у всьому світі мобільні пристрої, такі як смартфони та планшети, стали основними засобами доступу до Інтернету і станом на 2024 рік мобільний трафік становить понад 60% загального інтернет-трафіку, і ця тенденція продовжує зростати;

– у країнах, що розвиваються, мобільні пристрої є основним інструментом для доступу до Інтернету, оскільки стаціонарний інтернет може бути дорогим або недоступним.

Фактор 2. Змінилася поведінка користувачів:

– користувачі все частіше віддають перевагу мобільним пристроям для виконання повсякденних завдань, таких як пошук інформації, онлайн-шопінг, перегляд соціальних мереж, споживання контенту та відео, і згідно з дослідженнями, користувачі очікують, що мобільні сайти завантажуватимуться швидше, ніж на комп'ютерах, і якщо сайт працює повільно, вони швидко залишають його та переходять до конкурентів;

– дослідження показують, що користувачі частіше завершують покупку або взаємодіють із контентом на веб-сайтах, які швидко завантажуються та зручні у використанні на мобільних пристроях, і навпаки, сайти, які не оптимізовані для мобільних пристроїв, втрачають потенційних клієнтів, оскільки поганий користувацький досвід на мобільних пристроях може призвести до підвищення відмов та зниження продажів.

Фактор 3. Відбувся стрімкий розвиток мобільних технологій:

– мобільні технології, такі як 4G і 5G, значно покращили швидкість інтернет-з'єднань, однак навіть з такими покращеннями, оптимізація веб-сторінок для швидкого завантаження на мобільних пристроях залишається важливою, оскільки мобільні мережі часто можуть бути нестабільними або повільними у віддалених регіонах чи місцях з великим навантаженням на мережу [4];

– Google та інші пошукові системи перейшли на індексацію за принципом "mobile-first", тобто вони використовують мобільну версію сайту як основну для визначення його рейтингу у пошукових результатах, а веб-сторінки, що не оптимізовані для мобільних пристроїв, можуть значно втратити свої позиції у

видачі;

– швидкість завантаження мобільних сторінок також є важливим фактором у SEO, наприклад, Google знижує рейтинг сайтів, які повільно завантажуються на мобільних пристроях, що впливає на органічний трафік.

З огляду на зростання мобільного трафіку та зміну поведінки користувачів, оптимізація веб-сторінок для мобільних пристроїв стала необхідною умовою для успішного функціонування сучасного веб-ресурсу.

Основні вимоги у підходах до розробки сучасних веб-сайтів змістились у бік:

– швидкодії: оскільки мобільні користувачі зазвичай знаходяться в русі та використовують різні типи підключень (наприклад, 3G, 4G, Wi-Fi), швидкість завантаження є критично важливою; чим швидше завантажується сайт, тим краще;

– адаптивності: дизайн сторінок має бути адаптивним, тобто пристосовуватися до різних розмірів екранів, щоб забезпечити зручний перегляд і взаємодію з контентом;

– сучасного дизайну: веб-розробка все більше орієнтується на підхід "mobile-first", де сайти спочатку проєктуються для мобільних пристроїв і лише потім адаптуються для комп'ютерів, що дозволяє створювати максимально оптимізовані та швидкі сторінки для мобільних користувачів [5].

Оптимізація мобільних сайтів забезпечує швидке завантаження, покращує SEO, підвищує конверсії та забезпечує зручний користувацький досвід, що є вирішальним фактором у конкурентному інтернет-середовищі [6].

Існує низка технічних та інфраструктурних факторів, які можуть впливати на швидкість завантаження веб-сторінки:

– оптимізація зображень та ресурсів: великі та неоптимізовані зображення можуть бути однією з головних причин повільного завантаження;

– наявність кешування: використання кешування на стороні клієнта та сервера дозволяє зменшити кількість запитів до сервера та прискорити завантаження повторно відвідуваних сторінок;

– затримки з боку серверу: віддаленість користувача від сервера або використання неконкурентоспроможної інфраструктури може підвищувати час отримання перших даних від сервера;

– стиснення даних: використання або не використання методів стиснення HTTP-запитів, може суттєво впливати на розмір переданих даних між клієнтом та сервером, а від так на час завантаження сторінки;

– мережеві затримки та пропускна здатність: якість та швидкість підключення користувача до інтернету також значною мірою впливають на швидкість завантаження сторінки;

– оптимізація самої сторінки: неоптимізовані скрипти можуть значно впливати на час до інтерактивності сторінки, наприклад, мініфікація (процес зменшення розміру), об'єднання та асинхронне завантаження скриптів допомагає зменшити навантаження на браузер і прискорити завантаження;

– кількість HTTP-запитів: велика кількість запитів до сервера (наприклад, на завантаження зображень, скриптів, стилів) може затримувати завантаження сторінки, а оптимізація шляхом об'єднання ресурсів та використання спрайтів може значно покращити продуктивність.

Оцінка швидкості завантаження веб-сторінки за допомогою метрик є одним із ключових кроків на шляху до оптимізації швидкості завантаження вебсторінок. Це процес, що дозволяє виміряти, наскільки швидко користувач може отримати доступ до вмісту сторінки, а також оцінити різні етапи завантаження.

Метрики швидкості завантаження допомагають отримати кількісні дані про час, необхідний для завантаження сторінки, що дозволяє ідентифікувати вузькі місця та оптимізувати їх [7]. Розглянемо популярні метрики.

Метрика 1. Час до першого байту (Time to First Byte, TTFB) – це час, який проходить від моменту запиту браузером до отримання першого байту даних із сервера. Високий TTFB може вказувати на проблеми із сервером або мережевою інфраструктурою. Цей показник важливий, оскільки він вказує на те, наскільки швидко сервер обробляє запит і надсилає відповідь.

Метрика 2. Час до першого відображення (First Contentful Paint, FCP) – по-

казник часу, коли браузер починає відображати перший візуальний елемент сторінки. Це може бути текст, зображення або інший контент, що дає користувачеві відчуття, що сторінка почала завантажуватись. Ця метрика важлива, оскільки визначає, як швидко користувач бачить перший результат завантаження.

Метрика 3. Найбільша частина контенту (Largest Contentful Paint, LCP) визначає час, коли найбільший елемент контенту на сторінці (найчастіше це зображення або великий текстовий блок) завантажується та стає видимим для користувача. LCP є важливим показником, який дає уявлення про завершення завантаження основного контенту сторінки. LCP часто використовується для оцінки основної продуктивності завантаження сторінки, оскільки він показує, коли користувач може побачити основний контент.

Метрика 4. Сукупний зсув макету (Cumulative Layout Shift, CLS) – це показник стабільності макету сторінки під час завантаження. Він вимірює, наскільки елементи сторінки несподівано зміщуються під час завантаження, що може призводити до поганого користувацького досвіду. Наприклад високі значення можуть викликати незручності для користувачів, коли вони бачать раптові зсуви елементів під час взаємодії з веб-сторінкою.

Метрика 5. Латентність введення (Time to Interactive, TTI) – це показник, що відображає, коли сторінка стає повністю інтерактивною, тобто реагує на дії користувача без значних затримок. Показник вимірюється після завантаження всіх ресурсів та виконання скриптів, коли користувач може взаємодіяти з елементами сторінки без затримок.

Метрика 6. Затримка першої взаємодії (First Input Delay, FID) – це час, який проходить від моменту, коли користувач взаємодіє з елементом на сторінці (наприклад, кнопкою або посиланням), до моменту, коли браузер починає обробляти цю взаємодію. FID показує, наскільки швидко сторінка стає інтерактивною і є важливим для оцінки того, наскільки швидко сайт реагує на дії користувача.

Метрика 7. Загальний час блокування (Total Blocking Time, TBT) – час який проходить з моменту початку завантаження сторінки до того, як вона стане повністю інтерактивною. Ця метрика визначає, коли користувач може поча-

ти безперешкодно взаємодіяти з усіма елементами сторінки.

Метрика 8. Повний час завантаження (Fully Loaded Time, FLT) – це час, коли сторінка повністю завантажена, включаючи всі зображення, CSS, JavaScript, сторонні ресурси тощо. Це одна з найбільш загальних метрик, яка показує час, коли всі ресурси сторінки були завантажені та виконані. Визначає час коли користувач може повністю взаємодіяти з усіма елементами сторінки.

Всі ці метрики разом формують загальну картину продуктивності сторінки. Наприклад, низький TTFB та швидкий FCP можуть вказувати на те, що сервер добре працює і контент відображається швидко, але якщо TTI або TBT занадто великі, це означає, що сторінка може виглядати завантаженою, але фактично бути неінтерактивною через завантаження скриптів [8].

Ці метрики можна вимірювати за допомогою інструментів, таких як Google Lighthouse, PageSpeed Insights або інших веб-аналітичних сервісів.

Використання метрик для оцінки швидкості завантаження також важливе для обґрунтування рішень з оптимізації веб-сторінки. Наприклад, аналізуючи дані метрики, можна прийняти рішення про зменшення розміру зображень, оптимізацію JavaScript або зменшення кількості запитів до сервера.

Таким чином, метрики не тільки інформують про поточний стан веб-ресурсу, а й вказують на конкретні кроки для покращення його продуктивності, забезпечуючи кращий досвід для користувачів та підвищуючи конкурентоспроможність сайту в умовах сучасного цифрового середовища.

1.2 Аналіз сценаріїв вирішення задачі оптимізації швидкості завантаження веб-сторінок

За останні роки дослідження в галузі оптимізації продуктивності веб-сторінок значно активізувалися, оскільки швидкість завантаження веб-сайтів безпосередньо впливає на якість користувацького досвіду та ефективність взаємодії з ресурсом. Швидкість завантаження стає критичним фактором, який не

лише формує загальне враження від сайту, але й сприяє підвищенню показників конверсії, таких як кількість завершених покупок, реєстрацій та інші цільові дії [9].

У сучасній науково-технічній літературі дослідники розглядають різні підходи та сценарії для досягнення оптимального часу завантаження веб-сторінок. Окрему увагу приділено використанню нових технологій і підходів, що дозволяють підвищити швидкість відображення контенту. Серед найбільш актуальних напрямків оптимізації можна виділити кілька важливих категорій, які активно розвиваються та досліджуються у сучасних умовах швидкого розвитку веб-технологій:

– оптимізація часу першого відображення контенту (First Contentful Paint, FCP): дослідження показують, що значну увагу потрібно приділяти попередньому завантаженню критичних ресурсів, таких як шрифти та стилі;

– застосування “відкладеного завантаження” для зображень та інших важких елементів: останні дослідження показали, що відкладене завантаження зображень (lazy loading) і відео значно покращує час повного завантаження сторінок (Total Blocking Time, TBT), впровадження нативної підтримки цього підходу у браузерах стало можливим завдяки атрибуту loading="lazy" для зображень та iframe;

– мережеві оптимізації: дослідження акцентують увагу на впливі використання HTTP/2 та HTTP/3 на зниження затримок та оптимізацію паралельного завантаження ресурсів; завдяки мультиплексуванню (multiplexing), багато ресурсів можуть бути завантажені одночасно через одне з’єднання, що значно скорочує час очікування;

– підхід “Critical Rendering Path": багато досліджень зосереджено на критичному шляху рендерингу, зокрема, на зменшенні блокуючого CSS та JavaScript, і показали, що асинхронне завантаження скриптів та використання CSS лише для елементів, що відображаються на екрані під час першого завантаження, дає позитивний результат [10];

– кешування та CDN: застосування мережі доставки контенту (CDN) та

продумана система кешування значно покращують загальну продуктивність, а дослідження підтверджують, що оптимізоване кешування дозволяє повторно використовувати файли і зменшити кількість запитів до сервера.

Оптимізація швидкості завантаження веб-сторінок є складним і багатогранним процесом, що потребує комплексного підходу з урахуванням сучасних технологій і методів.

Сьогоднішні дослідження підтверджують, що застосування комплексних методів оптимізації, які включають комбінацію серверних та клієнтських покращень, дозволяє значно зменшити час завантаження веб-сторінок. Проведений аналіз сценаріїв показує, що жоден з методів не може вважатися універсальним і придатним для всіх типів веб-сайтів. Кожен підхід має свої переваги та обмеження, що вимагає ретельного вибору інструментів та технологій для досягнення максимальної ефективності.

Сучасні стандарти веб-технологій також значною мірою впливають на швидкість завантаження веб-сторінок, оскільки вони надають нові можливості для оптимізації коду, покращення взаємодії з сервером і зниження навантаження на клієнтську сторону.

Завдяки розвитку стандартів, таких як HTML5, CSS3 та JavaScript ES6+, веб-сторінки стали більш адаптивними та легшими для завантаження. Ці технології дозволяють використовувати полегшені способи відображення контенту, що зменшує час рендерингу та підвищує загальну швидкість.

Наприклад, нові можливості CSS3 дозволяють відмовитися від багатьох зображень та графічних елементів, замінюючи їх анімаціями, градієнтами та ефектами, створеними засобами CSS, що, у свою чергу, зменшує кількість ресурсів, які потрібно завантажувати. Крім того, CSS3 дозволяє створювати більш динамічні та гнучкі макети завдяки впровадженню Flexbox та CSS Grid. Ці технології значно скорочують кількість DOM-елементів, що потрібно створювати, та дозволяють реалізовувати складні макети без використання JavaScript, що позитивно впливає на швидкість завантаження.

HTML5 містить оптимізовані теги для роботи з мультимедійним контен-

том, як-от <video> і <audio>, що дозволяє відмовитися від громіздких плагінів і значно полегшує роботу браузера.

JavaScript також значно еволюціонував завдяки новим версіям ES6+ і таким інструментам, як WebAssembly, що дають змогу виконувати складні обчислення швидше та ефективніше. Асинхронне завантаження ресурсів за допомогою “async” і “defer” атрибутів для скриптів дозволяє браузеру завантажувати скрипти без блокування рендерингу сторінки, що зменшує час відображення контенту для користувачів. Також важливим є впровадження асинхронних функцій та Promises, що дозволило краще керувати затримками при завантаженні даних, не блокуючи основний потік [11].

Важливими є підходи, орієнтовані на скорочення критичних ресурсів та оптимізацію передачі даних через мережу. Так протокол HTTP/2 значно покращив продуктивність завдяки функції мультиплексування, що дозволяє надсилати кілька запитів одночасно через одне з’єднання. Це значно зменшує затримки та дозволяє браузеру отримувати більше ресурсів паралельно. HTTP/2 також підтримує стиснення заголовків (header compression), що зменшує розмір даних які передаються, це особливо корисно при роботі з мобільними мережами.

Питання оптимізації швидкості завантаження веб-сторінок є багатогранним і включає в себе аналіз багатьох аспектів, що впливають на час, який витрачається на завантаження контенту. На цей процес впливають різноманітні фактори, які умовно можна розділити на кілька груп:

– компоненти серверної частини (backend);

– компоненти клієнтської частини (frontend);

– компоненти мережевої інфраструктури.

Важливо враховувати, як кожен із цих компонентів взаємодіє між собою та впливає на загальну продуктивність веб-сторінки. Для проведення аналізу цих компонентів розглянемо, як кожна частина впливає на загальну продуктивність.

З боку серверної частини (backend), на швидкості завантаження веб-сторінок можуть впливати різні чинники. Розглянемо основні з них.

Чинник 1. Продуктивність самого сервера:

– процесор та оперативна пам'ять: якщо сервер не має достатньої потужності для обробки запитів, це призводить до затримок у відповіді; низька потужність або перевантаження сервера може спричинити високі значення TTFB метрики;

– навантаження на сервер: чим більше запитів обробляє сервер, тим більше часу потрібно на їх обробку; сервер може сповільнюватись під великим навантаженням, що впливає на загальний час відповіді;

– тип серверного програмного забезпечення: використання сучасних технологій, таких як Nginx чи Apache з відповідними конфігураціями, впливає на те, як ефективно сервер відповідає на запити.

Чинник 2. Оптимізація бази даних:

– запити до бази даних: неоптимізовані або надмірно складні запити до бази даних можуть значно сповільнити сервер, а використання індексів, кешування запитів і правильна структура бази даних дозволяють значно прискорити відповіді сервера;

– кешування: використання серверних технологій кешування (наприклад, Redis або Memcached) дозволяє зменшити навантаження на базу даних і підвищити швидкість відповіді на повторні запити.

Чинник 3. Компресія ресурсів – Gzip або Brotli стиснення: сервери можуть використовувати ці методи для стиснення HTML, CSS, JavaScript перед їх передачею клієнту, що зменшує обсяг переданих даних і пришвидшує їх завантаження.

Розглянемо фактори, які впливають на швидкість завантаження вебсторінок з боку клієнтської частини (frontend).

Фактор 1. Оптимізація ресурсоємних елементів:

– обсяг і оптимізація зображень: великі та неоптимізовані зображення збільшують час завантаження, а використання форматів зображень нового покоління (наприклад, WebP) та компресія можуть знизити розмір файлів;

– мінімізація та злиття файлів: об'єднання кількох CSS або JavaScript

файлів у один може зменшити кількість запитів до сервера, що прискорить завантаження. Мінімізація коду також знижує його розмір;

– асинхронне завантаження скриптів: використання атрибутів “async” та “defer” для скриптів дозволяє браузеру завантажувати контент сторінки до повного завантаження всіх JavaScript-файлів, що пришвидшує час відображення контенту;

– Lazy-loading зображень та інших ресурсів: ця техніка дозволяє відкладати завантаження зображень і медіаресурсів до моменту, коли вони фактично потрібні, що зменшує початкове навантаження і прискорює завантаження видимого контенту.

Фактор 2. Оптимізація вмісту:

– великий DOM (Document Object Model): структура HTML-коду впливає на те, як швидко браузер може його обробити, наприклад, великі та складні DOM-дерева уповільнюють продуктивність браузера;

– зменшення кількості бібліотек: використання великих фреймворків або зайвих бібліотек збільшує кількість завантажуваних ресурсів, що впливає на загальний час завантаження;

– критичний CSS: виділення та завантаження лише критичних стилів, необхідних для першого відображення сторінки (Critical CSS), допомагає пришвидшити відображення сторінки користувачам;

– анімації та інтерактивні елементи: складні анімації або інтерактивні елементи можуть негативно впливати на продуктивність, особливо на мобільних пристроях;

– кількість та тип шрифтів: велика кількість різних шрифтів, а також їх завантаження з зовнішніх ресурсів можуть значно уповільнити час завантаження сторінки.

Фактор 3. Кешування та попереднє завантаження:

– кешування в браузері: використання заголовків кешування дозволяє зберігати статичні ресурси в кеші браузера користувача, що знижує кількість повторних запитів при відвідуванні сторінки знову;

– використання механізмів попереднього завантаження: механізми попереднього завантаження ресурсів, такі як “prefetch”, “preload”, “preconnect” та інші, дозволяють зменшити затримку завантаження та покращити користувацький досвід, завантажуючи важливі ресурси до їх реального використання.

Розглянемо фактори, які впливають на швидкість завантаження вебсторінок з боку мережевої інфраструктури (network infrastructure).

Фактор 1. Пропускна здатність мережі:

– ширина каналу: пропускна здатність мережі між сервером і користувачем визначає, наскільки швидко дані можуть бути передані; чим вища пропускна здатність, тим швидше передаються файли;

– затримка мережі: відстань між користувачем і сервером також відіграє роль у затримці; затримка мережі (latency) може вплинути на час завантаження, особливо якщо сервер знаходиться далеко від користувача.

Фактор 2. Використовувані мережеві протоколи:

– HTTP/2 та HTTP/3: ці протоколи пришвидшують завантаження сторінки порівняно з HTTP/1.1 за рахунок таких функцій, як мультиплексування (можливість передавати кілька запитів через одне з'єднання), серверне натискання (Server Push) та стиснення заголовків;

– DNS-резолюція: час на перетворення доменного імені на IP-адресу (DNS-резолюція) може впливати на загальний час завантаження; швидкий та ефективний DNS-сервер допомагає зменшити затримку в цьому процесі.

Фактор 3. Техніка доставки контенту:

– мережа доставки контенту (CDN): використання CDN дозволяє доставляти контент через географічно розподілені сервери, що зменшує відстань між користувачем і сервером та підвищує швидкість завантаження сторінок;

– кешування на рівні мережі: кешування контенту на проміжних серверах (наприклад, CDN) дозволяє зменшити час доступу до ресурсу, зменшивши затримки та навантаження на основний сервер.

Для досягнення найкращої швидкості завантаження сторінки необхідно працювати над оптимізацією на всіх рівнях. На рівні сервера слід впроваджува-

ти технології кешування, оптимізувати бази даних, використовувати CDN. На клієнтському рівні оптимізувати HTML, CSS, JavaScript, використовувати техніки зменшення кількості запитів і обсягів даних. На рівні мережевої інфраструктури використовувати сучасні протоколи, покращувати мережеві з'єднання, налаштовувати оптимальне кешування [12].

Кожен з цих компонентів впливає на загальну продуктивність сторінки, і їх правильне налаштування та оптимізація дозволяє забезпечити максимально швидке завантаження веб-ресурсів.

Традиційні підходи до оптимізації швидкості завантаження веб-сторінок переважно зосереджувалися на зменшенні розміру файлів і кількості запитів. Це включало мінімізацію HTML, CSS, JavaScript, а також використання технік, як-от об'єднання файлів (concatenation) для скорочення кількості запитів до сервера. Стискання зображень до мінімальних розмірів без значної втрати якості також стало стандартною практикою, оскільки зображення, зазвичай, займають велику частку трафіку на сторінці. Використання кешування в браузері дозволяло уникати повторного завантаження незмінних ресурсів, що також сприяло пришвидшенню завантаження сторінок для повторних користувачів.

Однак з розвитком веб-технологій стали доступними нові, більш ефективні методи оптимізації. HTTP/2, наприклад, радикально змінює підхід до мережевих запитів. На відміну від HTTP/1.x, який обмежує кількість одночасних запитів до одного хосту, HTTP/2 дозволяє мультиплексування – одночасну передачу кількох ресурсів через одне з'єднання. Це означає, що браузер може запитувати і завантажувати кілька файлів одночасно, що значно зменшує час очікування. Також HTTP/2 використовує функцію пріоритизації запитів, що дозволяє швидше доставляти важливі ресурси, такі як CSS та JavaScript, на відміну від менш важливих, як-от зображення чи фонові скрипти.

Серед нових форматів файлів, які отримали популярність, можна відзначити WebP – формат зображень, розроблений Google. Він забезпечує значно краще стиснення порівняно з традиційними форматами, такими як JPEG або PNG, що дозволяє передавати зображення з вищою якістю при меншому розмі-

рі файлу. Це особливо важливо в контексті мобільних пристроїв, де швидкість завантаження суттєво впливає на користувацький досвід.

Крім того, новий алгоритм стиснення Brotli став значно популярнішим для стиснення текстових файлів, таких як HTML, CSS та JavaScript. Brotli забезпечує ефективніше стиснення порівняно з попередніми стандартами, такими як Gzip. Хоча Gzip залишається популярним і підтримується більшістю браузерів, Brotli забезпечує менший розмір файлів, що напряму впливає на швидкість завантаження веб-сторінки. Його використання також добре поєднується з іншими сучасними методами оптимізації, як-от HTTP/2.

Отже, якщо традиційні методи покладалися на мануальні процеси мінімізації та об’єднання файлів, то нові підходи, такі як HTTP/2, WebP та Brotli, дозволяють автоматично оптимізувати завантаження сторінок за рахунок глибшої інтеграції з веб-браузерами та серверами.

Ці нові технології не тільки підвищують швидкість завантаження, але й покращують загальний користувацький досвід, знижуючи потребу у відкладених або компромісних рішеннях, які були поширені раніше.

1.3 Змістовна та формальна постановка задачі

Основна мета дослідження полягає у визначенні та оцінюванні ефективності сучасних методів оптимізації швидкості завантаження веб-сторінок, а також їх впливу на загальну продуктивність веб-сайтів. У сучасному цифровому світі швидкість завантаження веб-сторінок є ключовим фактором, що впливає на взаємодію користувача з веб-ресурсом, його поведінкові реакції, а також на успішність бізнесу. Згідно з численними дослідженнями, затримка завантаження на кілька секунд може призвести до значного зниження кількості відвідувачів, зростання кількості відмов і, відповідно, втрат прибутку для компаній, що ведуть свою діяльність в інтернеті.

З огляду на це, дослідження має на меті:

– аналіз основних критеріїв швидкості завантаження: визначення ключових метрик, які використовуються для оцінки швидкості завантаження веб-сторінок, таких як First Contentful Paint (FCP), Largest Contentful Paint (LCP), Largest Contentful Paint (LCP), Fully Loaded Time (FLT) та інших, що відображають продуктивність веб-сайтів;

– огляд сучасних методів оптимізації: дослідження різних технічних рішень і методів, що використовуються для підвищення швидкості завантаження веб-сторінок; це включає використання кешування, стиснення зображень, мінімізації HTML, CSS та JavaScript, застосування lazy loading та інші технології, що спрямовані на зменшення часу очікування користувача;

– виявлення найефективніших методів: на основі проведеного аналізу і тестування, буде визначено, які з сучасних методів є найбільш ефективними у різних умовах (наприклад, для великих або малих сайтів, у мобільних та десктопних середовищах);

– практичні рекомендації щодо впровадження оптимізацій: розробка рекомендацій щодо того, які методи оптимізації доцільно застосовувати залежно від конкретних завдань і потреб, а також ресурсів веб-проєкту, що дозволить підвищити продуктивність веб-сайтів, знизити час очікування для користувачів і покращити загальну ефективність веб-ресурсів.

Ця мета дослідження є важливою з огляду на постійно зростаючі вимоги користувачів до швидкості та зручності роботи з веб-сайтами, а також необхідність відповідати вимогам пошукових систем, які все більше враховують швидкість завантаження як фактор ранжування. Оптимізація швидкості завантаження є одним із ключових інструментів для забезпечення конкурентоспроможності в онлайн-просторі, тому оцінка існуючих методів та їх впровадження має стати пріоритетом для розробників і власників веб-сайтів.

Загальна задача дослідження полягає в оцінюванні сучасних методів оптимізації швидкості завантаження веб-сторінок з метою їх впровадження в практику розробки та покращення користувацького досвіду.

Для досягнення цієї мети необхідно вирішити кілька конкретних завдань,

кожне з яких вимагає детального аналізу та розгляду на різних етапах дослідження.

Завдання 1. Аналіз поточного стану методів оптимізації швидкості завантаження веб-сторінок. Першочерговим завданням є здійснення огляду сучасних методів, що використовуються для прискорення завантаження веб-сторінок. Це включає вивчення різних технік, таких як мінімізація HTTP-запитів, оптимізація зображень, використання кешування на стороні клієнта та сервера, відкладене завантаження ресурсів, стиснення даних та асинхронне завантаження JavaScript. Зібрати інформацію про ці методи з наукової літератури, технічної документації та реальних прикладів впровадження на сучасних веб-платформах.

Завдання 2. Оцінка ефективності інструментів для вимірювання продуктивності веб-сторінок. Одним із важливих завдань є аналіз наявних інструментів для оцінювання швидкості завантаження веб-сторінок. Необхідно ознайомитись з такими сервісами, як Google PageSpeed Insights, GTmetrix, WebPageTest та Lighthouse, щоб зрозуміти, які показники вони використовують для оцінки швидкості та як їх результати можуть бути використані для подальшої оптимізації. Оцінка інструментів включає вивчення їх точності, простоти використання, а також їх здатності надавати рекомендації для оптимізації.

Завдання 3. Ідентифікація основних факторів, що впливають на швидкість завантаження веб-сторінок. Для успішної оптимізації важливо ідентифікувати фактори, які найбільше впливають на швидкість завантаження веб-сторінок. Цими факторами можуть бути швидкість відповіді сервера, розмір і кількість ресурсів, що завантажуються, кількість HTTP-запитів, методи стиснення даних та інші. Необхідно дослідити, як саме кожен із цих факторів впливає на продуктивність і як їх оптимізація може покращити загальний досвід користувачів.

Завдання 4. Розробка тестової платформи для експериментів із різними методами оптимізації. Для практичного дослідження необхідно створити контрольоване середовище для вимірювання впливу кожного з методів на продуктивність. Використання такого середовища дозволить порівняти ефективність

різних технік на практиці.

Завдання 5. Проведення серії експериментів з метою порівняння методів оптимізації. Наступним завданням є проведення експериментів для порівняння ефективності різних методів оптимізації. Це включає вимірювання часу завантаження сторінок, кількості переданих даних, споживання ресурсів браузера та сервера до і після впровадження кожного з методів. Завдяки експериментам можна буде оцінити, які техніки забезпечують найбільший приріст продуктивності

Завдання 6. Оцінка результатів дослідження та формулювання висновків. Головним завданням є оцінка отриманих результатів і формулювання висновків щодо ефективності сучасних методів оптимізації швидкості завантаження веб-сторінок. Це включає аналіз переваг і недоліків кожного з досліджених методів, а також розробку рекомендацій для розробників веб-сторінок щодо їх впровадження на практиці.

Завдання 7. Розробка рекомендацій щодо вибору методів оптимізації: завданням також є розробка практичних рекомендацій щодо вибору методів оптимізації в залежності від специфіки веб-сторінок. Це може бути поділ на статичні та динамічні сторінки, одно-сторінкові застосунки (SPA), контентні сторінки або сайти електронної комерції. Для кожного типу веб-сторінок можуть бути визначені найбільш ефективні методи оптимізації, враховуючи їх особливості та вимоги.

Таким чином, вирішення зазначених завдань дозволить досягти мети дослідження та сформувати практичні рекомендації щодо оптимізації швидкості завантаження веб-сторінок на основі сучасних методів та інструментів.

Для оцінки ефективності сучасних методів оптимізації швидкості завантаження веб-сторінок важливо розробити чітку методику вимірювання результатів. Оскільки швидкість завантаження залежить від багатьох факторів, необхідно використовувати кілька різних метрик і підходів, щоб отримати повну картину впливу оптимізацій.

Опишемо основні методи, які будуть застосовані для вимірювання ре-

зультатів дослідження.

Перш за все, це використання веб-аналітичних інструментів для вимірювання продуктивності.

Для вимірювання швидкості завантаження та оцінки впливу оптимізацій на продуктивність буде обрано і застосовано один з популярних інструментів веб-аналітики. Серед доступних безкоштовних інструментів:

– Google Lighthouse – інструмент від Google, який вимірює продуктивність, доступність та SEO-оптимізацію. Він надає оцінку як для десктопних, так і для мобільних версій сторінки, а також надає рекомендації щодо покращення коду та загального UX. Працює як у браузері (Chrome DevTools), так і через API;

– PageSpeed Insights – онлайн-інструмент, заснований на Google Lighthouse, для аналізу продуктивності, який надає оцінки на основі даних лабораторного тестування і реальних користувачів. Зосереджується на оптимізації швидкості завантаження та UX та має простий інтерфейс із практичними рекомендаціями;

– GTmetrix – інструмент для комплексного вимірювання часу завантаження сторінки, кількості запитів, а також для оцінки загального рівня продуктивності з різних країн і пристроїв.

Кожен із цих інструментів буде використаний для вимірювання швидкості завантаження до і після впровадження різних методів оптимізації, що дозволить порівняти їх ефективність на основі об’єктивних показників.

Для більш глибокого аналізу ефективності оптимізацій будуть застосовані наступні ключові метрики продуктивності:

– Time to First Byte (TTFB) – час, за який браузер отримує перший байт даних з сервера. Ця метрика показує затримки, які можуть виникати через завантаженість сервера, мережеві затримки та обробку запитів на сервері;

– First Contentful Paint (FCP) – час, за який на екрані користувача вперше відображається будь-який контент сторінки, що дає користувачу перше уявлення про завантаження сторінки;

– Largest Contentful Paint (LCP) – час, за який завантажується найбільший елемент сторінки, важлива метрика для визначення загальної швидкості та користувацького досвіду;

– Cumulative Layout Shift (CLS) – метрика, що вимірює стабільність макету сторінки під час її завантаження, показуючи, наскільки елементи на сторінці змінюють своє положення;

– Total Blocking Time (TBT) – час між FCP та TTI, коли сторінка частково або повністю блокує користувача від взаємодії. Вимірює затримки, спричинені виконанням довгих JavaScript завдань;

– Fully Loaded Time (FLT) – час, за який усі ресурси сторінки повністю завантажуються, що показує загальний час, необхідний для повного відображення контенту сторінки.

Вимірювання цих метрик дозволить оцінити, наскільки зменшується час завантаження та поліпшується користувацький досвід після впровадження методів оптимізації.

Для формального опису задачі оптимізації швидкості завантаження веб-сторінок можна сформулювати наступну математичну модель, що охоплює основні аспекти оптимізації швидкості завантаження веб-сторінок.

Цільова функція: мінімізація загального часу завантаження веб-сторінки T, який розбивається на дві основні частини – мережеві затримки та час рендерингу:

$$\min T = \min\left(T_{\text{network}} + T_{\text{rendering}}\right),$$

де T_{network} – час, необхідний для передачі ресурсів по мережі (в секундах);

$T_{\text{rendering}}$ – час, необхідний для обробки та рендерингу основного контенту (в секундах), та залежить від розміру самої сторінки та кількості елементів.

Час мережі обраховується як:

$$T_{\text{network}} \approx \frac{S_{\text{total}}}{B} + L,$$

де S_{total} – загальний розмір сторінки та її усіх ресурсів (в байтах);

B – середня пропускна здатність мережі (в байтах/с);

L – середня затримка мережі (в секундах).

Час рендерингу визначається як функція розміру сторінки S та кількості елементів N на сторінці:

$$T_{\text{rendering}} = f(S, N),$$

де $f(S, N)$ – емпірична функція, що враховує складність обробки DOM-елементів (S) і ресурсів сторінки (N).

При оптимізації швидкості завантаження веб-сторінок враховуються наступні обмеження.

Обмеження 1. Обмеження на загальний розмір ресурсів: для зменшення часу завантаження:

$$S_{total} \leq S_{\max},$$

де $S_{\max}$ – максимально допустимий розмір ресурсів (у кілобайтах);

S_{total} – загальний розмір ресурсів веб-сторінки (у кілобайтах).

Обмеження 2. Обмеження на кількість HTTP-запитів: для зменшення часу завантаження:

$$R_{total} \leq R_{\max},$$

де $R_{\max}$ – максимально допустима кількість HTTP-запитів;

R_{total} – загальна кількість HTTP-запитів.

Обмеження 3. Цільовий рівень рендерингу: для покращення сприйняття швидкості завантаження сторінки важливо, щоб перший видимий контент з'являвся якомога швидше. Це можна виразити умовою для метрики LCP:

$$LCP \leq LCP_{\text{target}},$$

де LCP_{target} – цільове значення для швидкого відображення основного контенту.

Таким чином, задача оптимізації формулюється як мінімізація загального часу завантаження сторінки T за умов дотримання зазначених обмежень.

Ця модель акцентує увагу на двох ключових аспектах – зменшенні загального розміру ресурсів, кількості запитів та досягненні швидкого першого рендерингу основного контенту.

Щоб оцінити ефективність кожного методу оптимізації, буде проведено порівняння результатів продуктивності до та після впровадження конкретних технік. Для цього буде використано контрольний веб-сайт, на якому будуть застосовуватися оптимізації, та де різні техніки оптимізації будуть реалізовані. Порівняння таких показників, як загальний час завантаження, кількість HTTP-запитів, розмір переданих даних та інші метрики, дозволить оцінити реальний вплив кожної оптимізації.

Методи вимірювання результатів у цьому дослідженні базуватимуться на використанні кількісних та якісних метрик, інструментів для тестування продуктивності, отриманих даних та порівнянні результатів до і після впровадження оптимізацій. Це дозволить отримати об'єктивні й надійні дані для оцінки ефективності сучасних методів оптимізації швидкості завантаження веб-сторінок та сформувати рекомендації щодо їх застосування.

1.4 Постановка задач дослідження

Основною метою даного дослідження є аналіз сучасних методів оптимізації швидкості завантаження веб-сторінок та визначення їхньої ефективності у реальних умовах.

Підвищення продуктивності веб-сторінок є ключовим завданням для сучасних веб-розробників, оскільки швидкість завантаження безпосередньо впливає на користувацький досвід, показники конверсії, рейтинг у пошукових системах та загальну успішність веб-сайту.

З огляду на це, метою дослідження є проведення комплексного аналізу існуючих підходів до оптимізації швидкості завантаження веб-сторінок, зокрема інструментів і технологій, які використовуються для прискорення обробки запитів, скорочення обсягу переданих даних і підвищення ефективності рендерингу контенту на стороні клієнта.

Для досягнення цієї мети необхідно дослідити та оцінити різні методи оптимізації, які застосовуються на різних етапах процесу завантаження веб-сторінок. Це включає аналіз оптимізації, зокрема, з використання кешування, зменшення розміру ресурсів, оптимізацію зображень, мінімізацію CSS та JavaScript файлів, а також впровадження методів асинхронного завантаження ресурсів. Окрім того, необхідно дослідити популярні інструменти для оцінювання продуктивності веб-сторінок, такі як Google PageSpeed Insights, Lighthouse, GTmetrix та інші, які надають комплексні показники, що дозволяють виявити можливі вузькі місця у процесі завантаження сторінки.

Однією з основних цілей дослідження є порівняння різних підходів до оптимізації веб-сторінок та оцінка їхньої ефективності за допомогою сучасних інструментів аналізу веб-продуктивності. Особливу увагу буде приділено швидкості рендерингу контенту, часу до повного завантаження сторінки та часу коли користувач може побачити основний контент (Largest Contentful Paint).

Оцінка цих параметрів дозволить виявити найбільш ефективні методи для зменшення часу завантаження сторінки без погіршення функціональності або

якості відображення контенту.

Таким чином, головною метою цього дослідження є не лише теоретичне обґрунтування та аналіз сучасних методів оптимізації швидкості завантаження веб-сторінок, але й практична оцінка ефективності цих методів, з акцентом на реальні сценарії їх застосування та вплив на кінцевий результат.

Нижче наведені основні завдання, що потребують вирішення:

– провести аналіз існуючих методів оптимізації швидкості завантаження веб-сторінок;

– визначити ключові критерії ефективності методів оптимізації для швидкого завантаження контенту;

– оцінити переваги та недоліки сучасних інструментів для тестування швидкості завантаження веб-сторінок (наприклад, Google PageSpeed Insights, GTmetrix тощо);

– здійснити порівняльне дослідження різних методів оптимізації, застосувавши їх до реального веб-сайту;

– розробити рекомендації щодо впровадження найбільш ефективних методів оптимізації для покращення швидкості завантаження веб-сторінок;

– оцінити вплив запропонованих методів на користувацький досвід та продуктивність веб-ресурсу.

Ці завдання спрямовані на забезпечення комплексного підходу до дослідження методів оптимізації швидкості завантаження веб-сторінок та обґрунтування вибору найбільш доцільних інструментів та технік для підвищення продуктивності веб-ресурсів.

Успіх у процесі оптимізації швидкості завантаження веб-сторінок буде визначатися через досягнення конкретних результатів, які мають покращити швидкість завантаження веб-сторінок. Виділимо кілька ключових критеріїв успіху.

Критерій 1. Скорочення часу завантаження сторінки – одним із головних показників успіху є значне зменшення загального часу завантаження. Для веб-сайту, що проходить оптимізацію, очікується, що час завантаження скоротиться мінімум на 20-30% залежно від типу сторінки та впроваджених технік оптимі-

зації. Це скорочення вимірюється за допомогою таких показників, як: зменшення часу, за який сторінка починає відображати контент; зменшення часу, за який найбільший елемент на сторінці повністю завантажується; зменшення часу до того моменту, коли сторінка стає повністю інтерактивною для користувача. Якщо час завантаження зменшується відповідно до цих показників, це буде вважатися успіхом оптимізації.

Критерій 2. Підвищення оцінок продуктивності – під час оптимізації веб-сторінок буде використовуватися кілька інструментів для оцінки їхньої продуктивності, таких як Google PageSpeed Insights, GTmetrix, Lighthouse тощо. Успіхом оптимізації буде вважатися підвищення оцінок продуктивності за такими інструментами мінімум на 10-15 балів (за шкалою до 100). Це підвищення оцінок свідчитиме про покращення швидкості завантаження та зменшення впливу ресурсів, які уповільнюють сторінку.

Критерій 3. Зменшення кількості HTTP-запитів та обсягу переданих даних – оптимізація має на меті зменшити кількість HTTP-запитів, що здійснюються під час завантаження веб-сторінки, а також скоротити обсяг даних, які необхідно передати. Успішним результатом буде вважатися зменшення кількості запитів щонайменше на 15-20%, а також зменшення розміру переданих даних (наприклад, через стиснення зображень та ресурсів) на 25-30%.

Критерій 4. Покращення стабільності макета (Cumulative Layout Shift - CLS) – показник Cumulative Layout Shift (CLS) відображає, наскільки стабільним є макет сторінки під час її завантаження. Для досягнення успіху у процесі оптимізації потрібно досягти значення $CLS \leq 0.1$, що вважається оптимальним рівнем для відсутності значних зрушень елементів сторінки під час завантаження.

Отже успіхом процесу оптимізації буде вважатися досягнення значного покращення швидкості завантаження веб-сторінок, підвищення оцінок продуктивності за інструментами аналізу, а також зменшення кількості запитів і обсягу даних. Оптимізація буде успішною, якщо вона забезпечить високу продуктивність на різних пристроях і в різних мережевих умовах, що приведе до зменшення часу завантаження і підвищення задоволеності користувачів.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ШВИДКОСТІ ЗАВАНТАЖЕННЯ ВЕБ-СТОРІНОК

2.1 Сучасні методи вирішення задачі оптимізації швидкості завантаження веб-сторінок

У сучасному світі зростаючого інтернет-трафіку та конкуренції між вебсайтами, зменшення часу завантаження стає одним із пріоритетів для розробників і власників сайтів.

Оптимізація швидкості завантаження веб-сторінок передбачає комплекс заходів, спрямованих на покращення взаємодії з користувачем шляхом зменшення часу завантаження та підвищення плавності роботи інтерфейсу.

Це питання стало особливо актуальним у зв’язку з тим, що сучасні веб-сторінки містять величезну кількість різноманітних ресурсів, таких як зображення, відео, інтерактивні елементи, анімації, які можуть значно збільшувати обсяг даних для завантаження. Незважаючи на розвиток високошвидкісного інтернет-зв’язку, важливо забезпечити ефективну доставку контенту користувачам із різними рівнями доступу до Інтернету та з використанням різних пристроїв, від потужних настільних комп'ютерів до мобільних телефонів.

Визначимо основні чинники, що впливають на швидкість завантаження веб-сторінок, на яких буде зосереджено дослідження (див. табл. 2.1).

Для покращення швидкості завантаження веб-сторінок важливо брати до уваги всі перелічені фактори.

Вони впливають на різні етапи завантаження та рендерингу сторінки, тому оптимізація повинна відбуватися на всіх рівнях – від правильного вибору технологій та форматів до налаштування сервера й ефективного використання протоколів передачі даних.

Таблиця 2.1 – Основні чинники, що впливають на швидкість завантаження веб-сторінок

Чинник	Опис	Вплив	Оптимізація
Розмір та кількість HTTP-запитів	Кожен елемент вебсторінки, такий як зображення, CSS, JavaScript, шрифти, робить окремий HTTP-запит на сервер. Велика кількість запитів та великий розмір ресурсів суттєво впливають на швидкість завантаження	Чим більше HTTP-запитів потрібно зробити для завантаження сторінки, тим більше часу займе процес, особливо при повільному інтернет-з'єднанні	Зменшення кількості HTTP-запитів за допомогою об’єднання файлів (concatenation), використання спрайтів для зображень або кешування ресурсів
Вага зображень та мультимедіа	Зображення та відео зазвичай складають найбільшу частину ваги вебсторінок. Неправильно оптимізовані або надто великі файли можуть значно збільшити час завантаження	Важкі зображення можуть суттєво уповільнити завантаження, особливо на мобільних пристроях	Використання сучасних форматів зображень, таких як WebP, адаптивні зображення для різних розмірів екранів, а також інструменти для стиснення зображень без втрати якості
Затримка мережі та географічне розташування	Фізична відстань між сервером та користувачем може впливати на швидкість завантаження через затримки в передачі даних (латентність).	Чим більша відстань між користувачем і сервером, тим більший час очікування на отримання даних	Використання мережі доставки контенту (CDN) дозволяє розміщувати копії вебсторінки на серверах по всьому світу, що скорочує затримки
Використання кешування	Кешування дозволяє браузеру зберігати статичні ресурси (зображення, файли CSS та JavaScript) локально, щоб не завантажувати їх кожен раз під час відвідування сторінки	Відсутність ефективного кешування призводить до того, що користувачі завантажують всі ресурси знову, навіть якщо вони не змінювалися	Використання заголовків кешування (Cache-Control, ETag) для того, щоб вказати браузеру зберігати ресурси на певний час
Оптимізація CSS	Подібно до JavaScript, файли CSS можуть блокувати рендеринг сторінки, якщо вони великі або завантажуються синхронно	Велика кількість CSS-файлів або блокуючий CSS може затримати час відображення сторінки	Використання мінімізації CSS-файлів, асинхронне завантаження стилів, зменшення кількості зовнішніх шрифтів, які також впливають на швидкість

Кінець таблиці 2.1

Чинник	Опис	Вплив	Оптимізація
Робота з JavaScript	JavaScript-файли часто блокують відображення контенту сторінки до повного завантаження. Особливо це стосується великих файлів або синхронних завантажень	Тривале завантаження або виконання JavaScript може затримати рендеринг сторінки, що погіршує користувацький досвід	Асинхронне завантаження скриптів (атрибути async та defer), зменшення розміру JavaScript-файлів (мінімізація та об'єднання), використання lazy loading для неважливих скриптів
Кількість зовнішніх плагінів і бібліотек	Використання зовнішніх бібліотек (наприклад jQuery) може додавати значний обсяг зайвого коду, який не завжди потрібен для конкретної вебсторінки	Завантаження та ініціалізація великих бібліотек уповільнюють завантаження сторінки, якщо вони не використовуються ефективно	Використання лише необхідних частин бібліотек або плагінів, відмова від завантаження зайвих скриптів
Шрифти веб-сторінок	Веб-шрифти можуть суттєво впливати на швидкість завантаження сторінки, оскільки їх потрібно завантажити перед тим, як текст буде відображений	Велика кількість шрифтів або відсутність налаштувань для попереднього відображення тексту може уповільнити процес завантаження	Використання системних шрифтів або нативних шрифтів, попереднє завантаження шрифтів (preload), мінімізація кількості шрифтів та гарнітур
Серверні ресурси та архітектура	Продуктивність сервера, використання ефективних баз даних та архітектура бекенд-інфраструктури також впливають на швидкість відповіді на запити	Якщо сервер перевантажений або не оптимізований, час відповіді на запити може значно збільшитися, що вплине на загальну швидкість	Оптимізація запитів до баз даних, балансування навантаження на сервери, використання серверів з високою продуктивністю, а також асинхронна обробка запитів.
Протокол передачі даних	Протоколи HTTP/1.1, HTTP/2 та HTTP/3 впливають на те, як дані передаються між сервером та клієнтом	HTTP/1.1 передбачає послідовне завантаження ресурсів, тоді як HTTP/2 і HTTP/3 дозволяють паралельне завантаження, знижуючи затримки	Перехід на HTTP/2 або HTTP/3 забезпечує швидше завантаження сторінок завдяки мультиплексуванню та оптимізації передачі даних

Основні сучасні підходи до вирішення задачі оптимізації швидкості завантаження веб-сторінок базуються на застосуванні таких методів, як мінімізація

коду (JavaScript, CSS, HTML), зменшення розміру зображень, використання сучасних форматів зображень (WebP, AVIF), використання кешування, асинхронне завантаження ресурсів, скорочення кількості HTTP-запитів, застосування Content Delivery Networks (CDN), попереднє завантаження контенту (prefetching) та інші [13].

Однією з ключових технологій, яка відіграє центральну роль в оптимізації швидкості, є lazy loading – техніка, яка дозволяє відкладати завантаження невидимих на екрані зображень та інших ресурсів до того моменту, коли вони стануть необхідними. Цей підхід значно зменшує початковий час завантаження сторінки, покращуючи як візуальне сприйняття, так і реальну швидкодію вебсайтів.

Інший важливий аспект – це мінімізація коду та зменшення розміру файлів, які передаються користувачу. Важливими інструментами для цього є мініфікація та агрегація файлів, які дозволяють скоротити кількість та розмір запитів до сервера.

Сучасні методи оптимізації швидкості завантаження веб-сторінок активно використовують і такі підходи, як адаптивний рендеринг та використання серверного рендерингу (Server-Side Rendering, SSR), які забезпечують швидше завантаження та відображення сторінок на різних типах пристроїв. Наприклад, AMP (Accelerated Mobile Pages) є прикладом технології, що дозволяє значно пришвидшити завантаження сторінок для мобільних пристроїв за рахунок обмеження використання важких елементів та оптимізації контенту.

Серед усіх методів оптимізації особливо виділяються кешування та компресія, оскільки вони дозволяють суттєво зменшити обсяг передаваних даних, підвищуючи ефективність завантаження веб-контенту. Ці техніки вже давно зарекомендували себе як одні з найбільш важливих і ефективних підходів для оптимізації.

Кешування – це процес тимчасового збереження даних для їх повторного використання без необхідності знову отримувати їх з серверу [14].

Основні види кешування включають:

– кешування на стороні клієнта (браузера). Браузери можуть зберігати певні елементи веб-сторінок, такі як CSS, JavaScript, зображення тощо. Це дозволяє значно знизити кількість запитів до сервера при повторних відвідуваннях сторінки, оскільки браузер використовує збережені дані;

– кешування на стороні сервера. Сервер може зберігати згенеровані сторінки чи їхні фрагменти для швидкого обслуговування наступних запитів. Це особливо корисно для динамічних сайтів, де сторінки часто генеруються у реальному часі. Замість повторного формування сторінки сервер може просто видати збережену копію;

– кешування на рівні мережі (CDN). Мережі доставки контенту (CDN) зберігають копії веб-ресурсів у різних точках по всьому світу. Це зменшує час затримки для користувачів, що знаходяться далеко від основного сервера, оскільки вони можуть отримати дані з найближчого до них центру обробки даних.

Компресія даних – це процес зменшення розміру файлів для їх швидшої передачі через мережу. Найбільш розповсюдженими методами компресії є Gzip і Brotli. Використання компресії дозволяє суттєво скоротити час завантаження сторінок, особливо для веб-сторінок з великою кількістю ресурсів.

Gzip – це найпоширеніший формат компресії, який підтримується практично усіма сучасними веб-серверами і браузерами. Gzip стискає текстові файли (CSS, JavaScript, HTML) і дозволяє зменшити їх розмір до 70-90%.

В той час як Brotli – це більш сучасний алгоритм компресії, розроблений Google, який пропонує ще вищий рівень стиснення, ніж Gzip, при цьому забезпечуючи швидше розпакування на стороні браузера. Хоча Brotli не такий широко підтримуваний, як Gzip, він поступово стає більш популярним, особливо для користувачів Chrome і Firefox.

Використання сучасних підходів до кешування дає ряд переваг:

– зменшується навантаження на сервер: завдяки кешуванню зменшується кількість запитів до сервера, що дозволяє знизити витрати на ресурси і покращити продуктивність сайту;

– підвищується загальна швидкість завантаження: користувачі отримують збережені дані швидше, особливо при повторному відвідуванні сайту;

– оптимізується використання пропускної здатності: кешування зменшує обсяг передаваних даних, що є важливим фактором для користувачів з обмеженим інтернет-з'єднанням або мобільним доступом.

В той час як застосування компресії даних забезпечує:

– зменшення розміру файлів. Дозволяє значно зменшити обсяг передаваних даних, що особливо важливо для великих сайтів з багатьма CSS і JavaScript файлами;

– швидке завантаження для користувачів з повільним інтернет-з'єднанням. Стискання даних дозволяє скоротити час завантаження сторінки для користувачів з обмеженою пропускною здатністю;

– підвищення SEO-позицій. Оскільки швидкість завантаження сторінки є одним з факторів ранжування, використання компресії може сприяти підвищенню позицій сайту у пошукових системах.

Найбільшого ефекту в оптимізації швидкості завантаження можна досягти, комбінуючи кешування і компресію. Наприклад, зберігаючи стиснені файли у кеші браузера або на сервері, ми можемо забезпечити як мінімальні розміри передаваних даних, так і мінімізувати кількість повторних запитів.

Загалом, досягнення оптимальних показників швидкості завантаження можливе за умови комплексного підходу, що включає різні методи і техніки.

Комбінація кешування, компресії, мінімізації коду, оптимізації зображень, використання CDN, асинхронного завантаження та відкладеного завантаження є ефективним набором рішень, що дозволяють значно покращити швидкість та ефективність завантаження веб-сторінок. Використання таких підходів дає можливість суттєво покращити досвід користувачів, підвищити показники SEO та, як результат, сприяти успіху веб-ресурсу в умовах сучасного цифрового світу.

Впровадження цих методів потребує постійного моніторингу та налаштування, але результати виправдовують витрачені зусилля, оскільки оптимізація швидкості завантаження веб-сторінок позитивно впливає як на технічні, так і на комерційні показники веб-проєкту.

2.2 Обґрунтування вибору методів оптимізації

Вибір методів оптимізації швидкості завантаження веб-сторінок є одним із ключових етапів процесу розробки та покращення продуктивності веб-сторінок. Оскільки сучасні веб-сайти стали значно складнішими, з великою кількістю динамічного контенту, інтерактивних елементів та мультимедійних ресурсів, виникає потреба у застосуванні гнучких та ефективних методів оптимізації, щоб забезпечити швидке завантаження сторінок навіть на пристроях з обмеженими ресурсами або в умовах низької швидкості інтернет-з'єднання.

При виборі конкретних методів для оптимізації, слід враховувати як технологічні обмеження, так і умови проєкту. Наприклад, важливо врахувати тип контенту веб-сторінок, характеристики цільової аудиторії (зокрема, їх пристрої та доступ до інтернету), а також очікувані навантаження на сервери.

Основними критеріями вибору методів оптимізації у даному дослідженні стали:

– ефективність зменшення часу завантаження сторінки;

– сумісність з існуючою інфраструктурою та технологіями проєкту;

– мінімізація негативного впливу на якість контенту;

– простота впровадження та підтримки методів.

У цьому дослідженні було обрано кілька основних методів оптимізації, які були ретельно проаналізовані з точки зору їх практичної ефективності та відповідності умовам проєкту. Розглянемо які основні методи включено.

Метод 1. Зменшення кількості запитів. Мінімізація HTTP-запитів є критично важливою для прискорення завантаження веб-сторінок, особливо в умовах обмеженої пропускної здатності мережі або високого навантаження на сервер. Така оптимізація націлена не лише на скорочення обсяг коду, а й на зниження затримок, пов'язаних із завантаженням сторонніх скриптів, що в результаті підвищує загальну швидкість і продуктивність сайту.

Метод 2. Оптимізація зображень. Оптимізація зображень є одним з найважливіших аспектів. Було обрано методи стиснення зображень без втрати яко-

сті (в тому числі з використання формату WebP). WebP є оптимальним форматом для зображень через його кращі показники стиснення у порівнянні з традиційними форматами (JPEG, PNG), що дозволяє знизити розмір файлів і, як наслідок, скоротити час завантаження сторінок.

Метод 3. Ліниве завантаження контенту. Це рішення є особливо ефективним для довгих сторінок з великою кількістю графіки або відео-контенту. Обраний метод був виправданий вимогами до коли швидкості з якою користувач може побачити основний контент.

Метод 4. Асинхронне завантаження скриптів. Цей підхід здатен забезпечити паралельне завантаження JavaScript-файлів без блокування основного потоку рендерингу сторінки, що забезпечує значне покращення швидкість завантаження.

Метод 5. Використання CDN (Content Delivery Network). Використання CDN для завантаження статичних ресурсів дозволяє зменшити час відповіді сервера та підвищити надійність. Файли зберігаються на географічно розподілених серверах, що забезпечує швидке завантаження ресурсів незалежно від місцезнаходження користувача, а також дозволяє їх ефективно кешувати.

Метод 6. Об’єднання і мініфікація ресурсів. Мінімізація та об’єднання файлів допомагає зменшити кількість HTTP-запитів і розмір передаваних файлів, що значно покращує швидкість завантаження сторінок.

Метод 7. Кешування. Використання кешування дозволяє значно прискорити процес завантаження сторінки шляхом повторного використання ресурсів. В даному дослідженні були обрані методи кешування на рівні браузера (через заголовки HTTP) і на рівні серверів (використання CDN для статичних файлів шрифтів). Це рішення дозволяє знизити кількість запитів до сервера і покращує загальну швидкість завантаження, особливо при повторних відвідуваннях [15].

Вибір цих методів обумовлений їх високою ефективністю у зниженні часу завантаження та покращенні загального користувацького досвіду.

По-перше, досліджуваний сайт орієнтований на широку аудиторію з різними типами пристроїв, тому необхідно було врахувати можливі обмеження у

швидкості інтернет-з’єднання та потужності пристроїв користувачів. Обрані методи (такі як оптимізація зображень, lazy loading та кешування) дозволяють покращити продуктивність сайту незалежно від цих факторів.

По-друге, важливою вимогою було забезпечити швидке завантаження навіть при великій кількості медіа-контенту, тому методи мінімізації файлів, використання CDN та кешування виявилися найбільш ефективними для цього завдання. Кожен метод був обраний з урахуванням можливості його швидкої інтеграції у поточну інфраструктуру проєкту, що дозволяє підтримувати високу продуктивність сайту без значних витрат часу зміну архітектури.

Таким чином, вибрані методи оптимізації швидкості завантаження вебсторінок мають гарантувати свою ефективність у широкому діапазоні сценаріїв та умов, забезпечуючи стабільну роботу веб-сайту.

2.3 Вибір інструментів для тестування

Однією з важливих складових успішної оптимізації швидкості завантаження веб-сторінок є правильний вибір інструментів для тестування продуктивності.

Для отримання точних та надійних даних щодо швидкості завантаження веб-ресурсу необхідно використовувати різноманітні інструменти, які надають детальний аналіз продуктивності та допомагають ідентифікувати ключові фактори, що впливають на час завантаження. У цьому розділі розглянемо найпоширеніші інструменти для аналізу швидкості завантаження, зокрема PageSpeed Insights, GTmetrix та Google Lighthouse. А також надамо обґрунтування вибору найвідповіднішого з них для даного дослідження.

Існує безліч інструментів для тестування продуктивності веб-сайтів, але не всі з них однаково підходять для конкретних типів досліджень або задач. Важливо розглянути основні можливості та особливості інструментів, що найчастіше використовуються для оцінювання швидкості завантаження веб-

сторінок.

PageSpeed Insights – це безкоштовний інструмент від Google, що дозволяє аналізувати продуктивність веб-сторінок на мобільних пристроях та комп'ютерах. Він надає детальний звіт про час завантаження різних елементів сторінки та рекомендації щодо покращення швидкості.

Основні функції:

– аналізує продуктивність веб-сторінок на мобільних та настільних пристроях;

– оцінка базується на метриках Core Web Vitals, включаюно з First Contentful Paint (FCP), Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS).

Переваги:

– інтеграція з Lighthouse дозволяє отримувати детальні рекомендації щодо оптимізації;

– видає оцінки на основі даних лабораторного тестування і реальних користувачів;

– зосереджується на оптимізації швидкості завантаження та UX;

– має простий інтерфейс із практичними рекомендаціями.

Недоліки:

– деякі метрики можуть бути складними для інтерпретації без розуміння технічних аспектів.

GTmetrix – ще один популярний інструмент для тестування швидкості завантаження веб-сторінок, який надає детальний аналіз продуктивності. GTmetrix використовує два основних механізми для тестування: Google Lighthouse та технологію від WebPageTest.

Основні функції:

– надійний інструмент для аналізу швидкості завантаження сторінок та загальної продуктивності;

– дає можливість вибирати місце розташування серверів для тестування (корисно для глобальних сайтів);

– оцінює веб-сторінки за такими параметрами, як PageSpeed Score і YSlow Score, надаючи розширені звіти;

– дозволяє побачити waterfall chart – візуалізацію завантаження ресурсів сторінки.

Переваги:

– гнучкі налаштування тестування (вибір локації, тип пристрою);

– можливість порівняти різні версії сторінок (після змін) для відстеження покращень.

Недоліки:

– безкоштовна версія має обмежені можливості налаштування;

– оцінки можуть відрізнятися від реальних умов, залежно від обраних налаштувань тестування.

Google Lighthouse – це один з найпопулярніших інструментів для тестування продуктивності веб-сторінок. Він інтегрований в браузер Chrome і надає можливість проводити аналіз без необхідності встановлення додаткових програм.

Основні функції:

– інструмент для аудиту веб-сторінок, інтегрований в Chrome DevTools;

– аналізує сторінки за кількома категоріями: продуктивність, доступність, SEO та прогресивні веб-додатки (PWA);

– оцінює сторінку за ключовими метриками, такими як FCP, LCP, CLS, TTI, і надає детальні рекомендації для оптимізації.

Переваги:

– можливість запуску на будь-якому сайті прямо з браузера;

– розгорнуті звіти з порадами щодо підвищення продуктивності.

Недоліки:

– для більш детального аналізу ресурсів або складних сценаріїв завантаження можуть знадобитися додаткові інструменти.

Pingdom Tools – це набір веб-інструментів для моніторингу та аналізу продуктивності веб-сайтів. Сервіс окрім іншого надає можливість тестувати

швидкість завантаження сторінок, відстежувати час безвідмовної роботи та аналізувати взаємодію користувачів з веб-ресурсами.

Основні функції:

– інструмент для моніторингу швидкості завантаження веб-сайтів;

– пропонує глобальну мережу тестування та надає детальні звіти щодо часу завантаження елементів, HTTP-запитів та розмірів сторінки;

– дозволяє відстежувати продуктивність у реальному часі та отримувати сповіщення у випадку проблем.

Переваги:

– інтуїтивно зрозумілий інтерфейс та візуалізація даних;

– підходить для моніторингу продуктивності в реальному часі.

Недоліки:

– деякі корисні функції доступні лише у платній версії.

WebPageTest – це потужний інструмент для глибокого аналізу продуктивності веб-сторінок. Він дозволяє проводити детальне тестування в різних умовах: з різними типами мережі, пристроями, браузерами та іншими параметрами.

Основні функції:

– потужний інструмент для детального аналізу продуктивності веб-сторінок;

– можливість вибору геолокації, браузера та типу підключення для точного аналізу;

– пропонує розширені можливості, такі як блокування вмісту для аналізу впливу конкретних елементів на продуктивність.

Переваги:

– глибокий аналіз з можливістю тонкого налаштування тестів;

– генерує водоспадні графіки, що дають візуальне уявлення про процес завантаження сторінки;

Недоліки:

– інтерфейс може бути складним для новачків.

Приведемо оглядову таблицю інструментів для тестування продуктивнос-

ті та швидкості завантаження веб-сторінок (табл. 2.2).

Таблиця 2.2 – Порівняльна таблиця інструментів для аналізу швидкості завантаження веб-сторінок

Інструмент	Опис	Додаткові особливості	Вартість
PageSpeed Insights	Інструмент від Google, який аналізує продуктивність на основі реальних даних користувачів	Включає рекомендації для оптимізації; інтеграція з Google Lighthouse для глибокого аналізу	Безкоштовно
GTmetrix	Надійний інструмент для комплексного аналізу продуктивності з великою кількістю метрик	Підтримка різних браузерів і локацій; історія тестів; можливість налаштування сповіщень	Безкоштовно (є платні послуги)
Google Lighthouse	Інструмент з відкритим кодом для аудиту якості веб-сторінок, розроблений Google	Оцінює сторінку за ключовими метриками; інтегрований у Chrome DevTools; аналізує продуктивність, доступність, SEO параметри; створює PDF-звіти; можлива інтеграція з CI/CD-процесами	Безкоштовно
WebPageTest	Інструмент для тестування швидкості завантаження з різних локацій та на різних пристроях	Можливість симуляції різних мереж; детальні таймлайн звіти; підтримка відеоаналітики	Безкоштовно (є преміум)
Pingdom Tools	Інструмент для базового аналізу швидкості завантаження та моніторингу	Підтримка моніторингу доступності та алертів; інтеграція з сервісами для довгострокового аналізу	Безкоштовно (є преміум)

Для оцінювання сучасних методів оптимізації швидкості завантаження веб-сторінок серед численних доступних інструментів зупинимо свій вибір на Google Lighthouse. Цей вибір обґрунтовано низкою ключових переваг, які роб-

лять цей інструмент ідеальним для проведення досліджень у сфері оптимізації веб-продуктивності. Нижче детально розглянемо причини, чому саме Google Lighthouse став основою нашого аналізу.

Популярність та довіра користувачів. Google Lighthouse є одним із найпоширеніших і найпопулярніших інструментів для аналізу продуктивності вебсайтів у світі. Він широко використовується як професіоналами в сфері веброзробки, так і аналітиками, що займаються оптимізацією швидкості завантаження сторінок. Популярність цього інструменту обумовлена його надійністю, інтуїтивністю та високим рівнем деталізації аналізу.

Крім того, Google Lighthouse має репутацію інструмента, який постійно оновлюється та вдосконалюється відповідно до сучасних вимог і стандартів. Це гарантує, що результати аналізу залишаються актуальними, а рекомендації відображають найкращі практики у сфері оптимізації.

Безкоштовний доступ. Однією з головних переваг Google Lighthouse є те, що він є абсолютно безкоштовним для використання. У порівнянні з іншими інструментами, які можуть вимагати платної підписки для отримання повного доступу до функціональності, Google Lighthouse пропонує широкий спектр можливостей без жодних фінансових витрат.

Безкоштовний доступ дозволяє не тільки зменшити витрати на проведення дослідження, а й забезпечити прозорість результатів, оскільки інструмент доступний кожному бажаючому для перевірки отриманих даних.

Інтеграція в Google Chrome. Ще однією важливою перевагою Google Lighthouse є його інтеграція в браузер Google Chrome. Завдяки цьому користувачі можуть запускати аналіз веб-сторінок без необхідності встановлювати додаткові програми або розширення. Інструмент доступний через розробницькі інструменти Chrome (Chrome DevTools), що робить його використання максимально простим і швидким.

Інтеграція з браузером також забезпечує зручність у проведенні тестів на різних етапах розробки веб-сайтів. Наприклад, розробники можуть миттєво тестувати зміни у продуктивності після впровадження оптимізацій, що значно

економить час.

Повний набір необхідних метрик. Google Lighthouse пропонує всі необхідні метрики для проведення глибокого аналізу швидкості завантаження веб-сторінок та їх продуктивності.

Комплексний підхід до аудиту. Окрім аналізу продуктивності, Google Lighthouse також оцінює веб-сторінки за іншими критеріями, такими як:

– доступність (Accessibility): перевіряє, наскільки сайт відповідає вимогам доступності для користувачів з особливими потребами;

– SEO: надає рекомендації для покращення видимості сайту у пошукових системах.

– прогресивні веб-додатки (PWA): аналізує відповідність стандартам сучасних веб-додатків.

Цей комплексний підхід робить Google Lighthouse універсальним інструментом, який допомагає покращити не тільки швидкість завантаження, але й загальну якість веб-сайту.

Рекомендації на основі найкращих практик. Google Lighthouse надає детальні рекомендації щодо оптимізації на основі даних аналізу. Ці рекомендації базуються на найкращих практиках, прийнятих у веб-індустрії, та враховують сучасні вимоги до продуктивності, доступності та зручності використання.

Наприклад, Lighthouse може запропонувати:

– оптимізацію зображень;

– зменшення розміру JavaScript;

– впровадження механізмів кешування;

– використання сучасних форматів файлів (WebP для зображень тощо).

Вибір Google Lighthouse як основного інструменту для оцінювання методів оптимізації швидкості завантаження веб-сторінок є обґрунтованим і доцільним. Його популярність, безкоштовний доступ, інтеграція в Google Chrome, а також повний набір необхідних метрик роблять цей інструмент найкращим вибором для дослідження.

Висновки за розділом 2

У розділі було детально проаналізовано сучасні підходи до оптимізації швидкості завантаження веб-сторінок, що є важливим етапом у забезпеченні високої продуктивності веб-ресурсів.

Розгляд методів, таких як оптимізація зображень, кешування, асинхронне завантаження скриптів, використання Content Delivery Networks (CDN), мінімізація коду та зменшення кількості HTTP-запитів, дозволяє сформувати цілісний підхід до вирішення проблем, пов’язаних зі швидкістю завантаження сторінок.

Також було розглянуто важливість компресії даних, яка дозволяє суттєво зменшити розмір файлів, скорочуючи час завантаження навіть за умов обмеженого доступу до Інтернету. А також важливість використання сучасних протоколів передачі даних, таких як HTTP/2 та HTTP/3, що підвищують ефективність передачі інформації між сервером і клієнтом.

Розглянуті стратегії дозволяють підвищити швидкість завантаження, покращити користувацький досвід, знизити затримки при доступі до ресурсів і, як наслідок, сприяти зростанню SEO-показників.

Впровадження описаних методів потребує ретельного планування, моніторингу та тестування. Вибір Google Lighthouse як основного інструменту для оцінювання ефективності запропонованих підходів є обґрунтованим, оскільки він забезпечує надійний аналіз метрик, зручність використання та можливість інтеграції в різні етапи розробки.

Застосування комплексного підходу до оптимізації швидкості завантаження веб-сторінок гарантує досягнення оптимальних результатів та відповідає сучасним вимогам до веб-продуктивності.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ОПТИМІЗАЦІЇ ШВИДКОСТІ ЗАВАНТАЖЕННЯ ВЕБ-СТОРІНОК

3.1 Опис умов та методів проведення експерименту

Для практичного дослідження ефективності сучасних методів оптимізації швидкості завантаження веб-сторінок необхідно створити контрольоване середовище, яке дозволить забезпечити точність і відтворюваність результатів експерименту.

Ключовим елементом дослідження стане тестовий веб-сайт, який буде використовуватися для моделювання реальних умов завантаження веб-сторінок та для оцінювання впливу різних методів оптимізації на швидкість завантаження веб-сторінок. Для цього потрібно виконати наступні етапи.

Етап 1. Розробка тестового веб-сайту. Створення веб-сторінки, що містить елементи різного типу (зображення, текст, мультимедійний контент, скрипти), які характерні для сучасних веб-ресурсів. Важливо забезпечити відповідність структури сторінки типовим прикладам із реального середовища, включно з використанням популярних технологій, таких як HTML5, CSS3 та JavaScript.

Етап 2. Налаштування інструментів тестування. Використання відповідних інструментів для вимірювання швидкості завантаження та інших метрик продуктивності. Для цього будемо використовувати Google Lighthouse.

Етап 3. Проведення серії експериментів. Кожен метод оптимізації буде поетапно застосований до тестового веб-сайту. Наприклад, на першому етапі виконується аналіз вихідного стану сайту без будь-якої оптимізації. Потім додається один із методів, наприклад, стиснення зображень, і повторно оцінюються метрики продуктивності. Усі отримані дані фіксуються для подальшого аналізу. Ключовою умовою є те, що в кожному експерименті змінюється лише один фактор, тоді як інші параметри залишаються незмінними, що дозволяє ізолювати вплив конкретного методу.

Етап 4. Аналіз результатів. Після завершення експериментів всі отримані дані обробляються та порівнюються. Це дозволяє визначити, який із методів має найвищу ефективність.

Таким чином, проведення експериментів на тестовому веб-сайті дозволить оцінити як якісний, так і кількісний вплив різних методів оптимізації на продуктивність завантаження веб-сторінок. Це забезпечить можливість зробити обґрунтовані висновки щодо їх практичного застосування в реальних проектах.

Дослідження будемо проводити на прикладі тестового веб-сайту доступного за адресою https://nahorets.me (рис. 3.1).

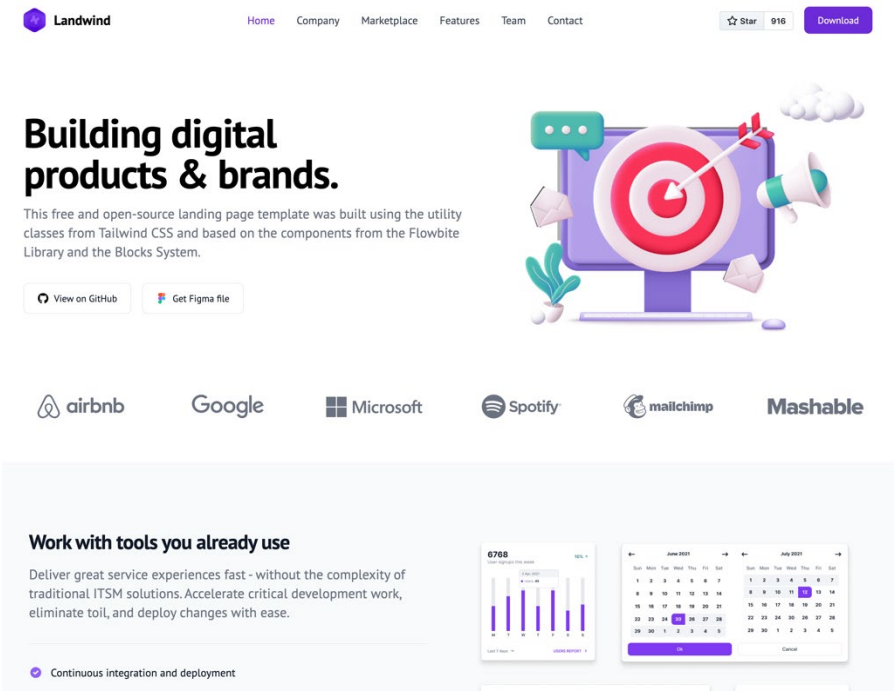


Рисунок 3.1 – Зовнішній вигляд експериментального веб-сайту

Сайт створено з використанням сучасних веб-технологій, які широко застосовуються в реальній розробці веб-додатків. Розглянемо їх.

HTML5 – є основою для створення структури сучасних веб-сторінок. Він забезпечує підтримку семантичних тегів, які сприяють кращій організації контенту та підвищенню доступності. HTML5 є стандартом для веб-розробки, і є базовим для всіх сучасних веб-сайтів.

CSS3 – відповідає за стиль і візуальне оформлення сторінки. Включення

цієї технології дозволяє використовувати сучасні підходи до стилізації, такі як адаптивний дизайн, що робить сайт зручним для користувачів як на десктопах, так і на мобільних пристроях.

PHP – обрано для серверної частини сайту завдяки його гнучкості та широкій підтримці веб-серверів. Ця технологія дозволяє створювати динамічні сторінки, забезпечувати роботу з базами даних, обробляти форми, а також забезпечувати реалізацію складної бізнес-логіки. PHP залишається одним із найпопулярніших серверних мов програмування через його легкість у навчанні, активну спільноту та широке використання у реальних проектах.

JavaScript – є невід'ємною частиною фронтенд-розробки, яка забезпечує динамічну взаємодію користувачів із веб-сторінками. Використання JS дозволяє реалізувати функціонал на клієнтській стороні, такий як валідація форм, інтерактивні анімації та асинхронний обмін даними з сервером за допомогою технологій AJAX та Fetch API. Ця технологія є критично важливою для створення сучасного, інтерактивного досвіду користувача.

TailwindCSS – було обрано як утилітарний CSS-фреймворк, що дозволяє швидко і гнучко створювати стиль сайту. Він надає набір готових класів для розміщення елементів, кольорових схем, відступів, шрифтів і багато іншого, що спрощує розробку, зменшує дублювання стилів і покращує читабельність коду.

Дизайн-шаблон Landwind, створений на базі TailwindCSS, взято за основу оскільки він включає в себе набір попередньо стилізованих компонентів, таких як форми, кнопки, таблиці, навігаційні панелі тощо. Використання цього шаблону дозволяє зосередитися на тестуванні методів оптимізації, а не витрачати багато часу на створення базової структури сайту. Цей шаблон відповідає вимогам сучасного дизайну та є чудовим прикладом, який демонструє, як можна використовувати TailwindCSS у реальних проектах.

Цей набір технологій було обрано з огляду на їх актуальність, популярність і здатність відображати реальні умови розробки сучасних веб-сайтів.

Веб-сайт розміщується на шаредному-хостингу сучасного хостинг-провайдеру https://www.spaceship.com, сервери якого фізично розташовані в

Сполучених Штатах Америки.

Для доступу до сайту і оцінювання методів оптимізації швидкості завантаження веб-сторінок було обрано браузер Google Chrome, як один із найпоширеніших інструментів серед користувачів. Для вимірювання ключових показників продуктивності використовувався інструмент Google Lighthouse (рис. 3.2), інтегрований у розробницькі інструменти DevTools браузера Chrome.

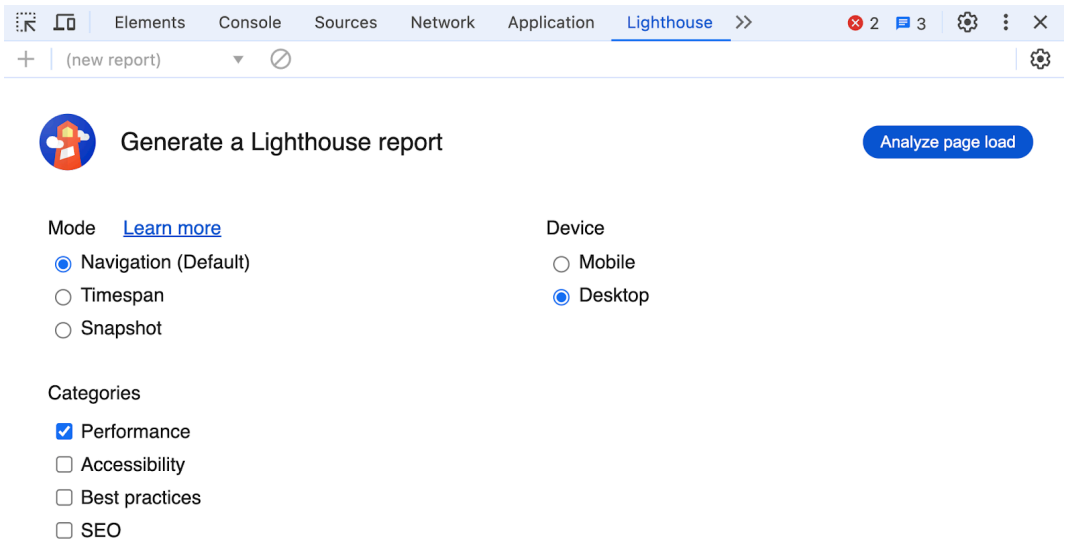


Рисунок 3.2 – Інструмент Google Lighthouse

Опишемо тестове середовище:

– операційна система: macOS Sonoma;

– версія браузера Chrome: 131;

– інструменти: Chrome DevTools та Google Lighthouse;

– тестове інтернет-з’єднання: стабільне з середньою швидкістю 100 Мбіт/с;

– тестова сторінка: було обрано головну сторінку веб-сайту https://nahorets.me.

Налаштування Google Lighthouse, що застосовувались:

– тип аудиту: Performance (швидкодія);

– емуляція пристрою: Desktop (настільний);

– емуляція мережі: не використовувалась.

Послідовність дій для проведення замірів:

– відкриття сторінки. Сторінка завантажувалася в новому інкогніто-вікні Chrome для виключення впливу кешування чи інших локальних факторів;

– запуск аудиту. У DevTools відкривався розділ Network та Lighthouse. Вибиралися попередньо налаштовані параметри аудиту. Після натискання кнопки “Analyze page load” відбувався збір даних про продуктивність;

– повторність замірів. Тестування проводилося 10 разів для отримання більш точних результатів. Результати для кожного тесту зберігалися у вигляді звітів у форматі HTML;

– обробка даних. Для кожного показника (наприклад, Largest Contentful Paint, First Contentful Paint, Total Blocking Time) розраховувалося середнє значення з 10 замірів. Значення округлювалися до першого знаку після коми.

Основні показники для аналізу:

– First Contentful Paint (FCP), секунд – час, за який перший елемент сторінки стає видимим;

– Largest Contentful Paint (LCP), секунд – час завантаження найбільшого видимого елементу;

– Cumulative Layout Shift (CLS), умовних одиниць – показник стабільності візуального відображення;

– Total Blocking Time (TBT), мілісекунд – час, протягом якого сторінка була недоступною для взаємодії;

– Speed Index, секунд – швидкість візуального завантаження;

– Lighthouse score, бали від 1 до 100 – це загальна оцінка продуктивності веб-сторінки, яку надає інструмент Google Lighthouse.

Додатково, з допомогою вкладки Network аналізувались (рис. 3.3):

– Total number of requests, ціле число – відображає загальну кількість HTTP-запитів, які були надіслані під час завантаження веб-сторінки і включає: запити на HTML, CSS, JavaScript, зображення; запити до API; запити на сторонні ресурси (наприклад, шрифти, скрипти, стилі) тощо;

– Total size of transferred resources, кілобайт – загальний об'єм даних, які були передані з сервера до клієнта (браузера) під час завантаження веб-сторінки. Це включає в себе всі ресурси, такі як HTML, CSS, JavaScript, зображення, шрифти та інші медіа-файли.

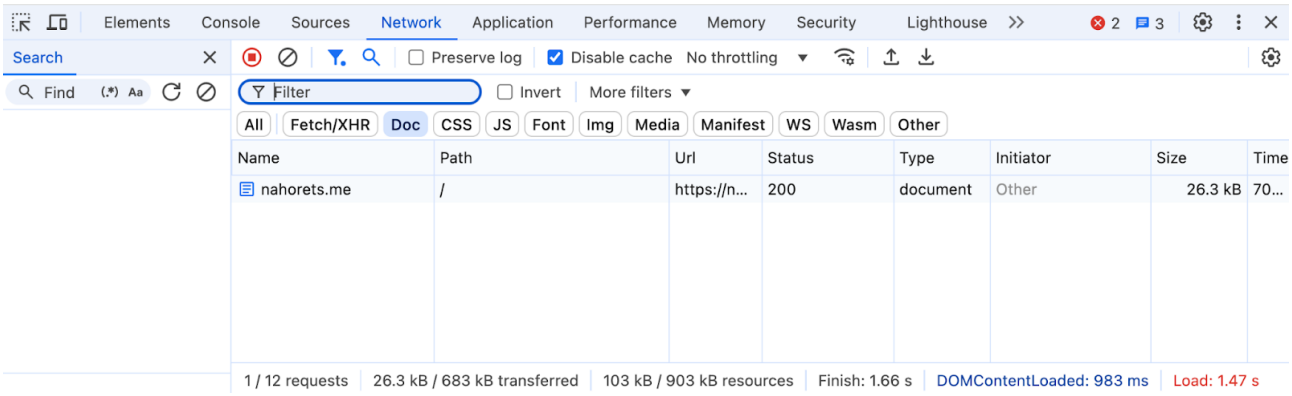


Рисунок 3.3 – Аналіз даних в Chrome DevTools

На основі середніх значень для кожної сторінки проводилося порівняння показників до та після застосування методів оптимізації. Результати представлялися у вигляді графіків і таблиць для візуалізації динаміки змін продуктивності.

Застосування описаної методики дозволяє об’єктивно оцінити ефективність сучасних методів оптимізації швидкості завантаження веб-сторінок, забезпечуючи відтворюваність і точність експерименту.

Далі проведемо серію експериментів з метою застосування різних методів клієнтської оптимізації головної сторінки сайту і порівняння результатів оптимізації. Це включає вимірювання основних метрик, часу завантаження сторінки, кількості переданих даних до і після впровадження кожного з методів.

3.2 Опис практичної реалізації методів оптимізації

У цьому розділі буде розглянуто практичну реалізацію методів оптиміза-

ції швидкості завантаження веб-сторінок. Основною метою є – мінімізація часу завантаження тестової веб-сторінки.

Для досягнення цієї мети було застосовано наступні методи оптимізації.

Метод 1. Зменшення кількості сторонніх скриптів. Використання сторонніх бібліотек часто призводить до перевантаження веб-сторінки через великий розмір і функціональність, яка може бути надлишковою.

Провівши аналіз вихідного коду тестового веб-сайту, було помічено що окрім інших, на ньому використовуються наступні сторонні скрипти: Moment.js та Lodash.

Moment.js – це популярна JavaScript-бібліотека для роботи з датами та часом у JavaScript. Вона спрощує маніпуляції з датами, такими як: форматування дат у зручному для користувача вигляді; парсинг рядків у об'єкти дат; маніпуляції з датами (додавання, віднімання днів, місяців тощо); локалізацію для відображення дат у відповідних форматах різних мов; забезпечує підтримку часових зон через додаткові модулі тощо.

З огляду на те, що головна сторінка веб-сайту не містить, за своєю природою, складних маніпуляцій із датами або необхідності локалізації чи обчислень часу, а основна інформація статична – вважаємо підключення бібліотеки Moment.js зайвим.

Видалення підключення бібліотеки зменшує вагу JavaScript-коду, що завантажується браузером. Це дасть можливість прискорити завантаження сайту, особливо для мобільних пристроїв із повільним інтернет-з'єднанням.

Менший розмір JavaScript означає швидше виконання коду браузером, що покращує показники Core Web Vitals, зокрема Largest Contentful Paint (LCP) та Total Blocking Time (TBT). Крім того видалення бібліотеки знижує кількість зовнішніх залежностей, що підвищує стабільність і безпеку проєкту.

Інша бібліотека Lodash – надає набір утиліт для роботи з масивами, об'єктами, рядками та іншими структурами даних. Основні функції, які вона спрощує: маніпуляції з масивами та об'єктами; глибоке копіювання та порівняння об'єктів; роботу з функціональним програмуванням; оптимізацію обчислень тощо.

Головна сторінка часто має статичний контент або прості запити до API. Після аналізу присутніх на сторінці JavaScript скриптів, було знайдено і замінено наступні виклики бібліотеки Lodash стандартними функціями JavaScript (табл. 3.1).

Таблиця 3.1 – Альтернативи функціям бібліотеки Lodash у JavaScript

Функції з бібліотеки Lodash	Альтернативний код на JavaScript
_.capitalize(string)	string.charAt(0).toUpperCase() + string.slice(1)
_.trim(string)	string.trim()

Всі виклики функцій в вихідному JavaScript коді були замінені їх аналогами, без змін у функціоналі веб-сайту.

Відмова від Lodash зменшує кількість залежностей, що покращує стабільність і безпеку застосунку. Менший загальний об’єм коду означає швидший час виконання скриптів і зменшення затримок при взаємодії з інтерфейсом.

Під-час аналізу додатково було помічено посилання на неіснуючий файл маніфесту “<link rel=’manifest’ href=’/site.webmanifest’>” виклик якого повертав 404 помилку (сторінку не знайдено).

Оскільки ми не потребуємо підтримки прогресивного веб-додатку (PWA), видалення посилання є виправданим і дозволить кількість HTTP-запитів які обробляються браузером. Це також дасть змогу поліпшити семантичну чіткість розмітки (видалення непотрібного елемента робить код сайту більш зрозумілим і чітким, усуваючи двозначності).

Отже нами було проведено рефакторинг коду для видалення бібліотеки Moment.js, а також видалення бібліотеки Lodash з відповідною підміною вихідного коду. Також було прибрано посилання на неіснуючий маніфест-файл.

Метод 2. Оптимізація зображень. Візуальний контент є важливим елементом сучасних веб-сторінок, проте неоптимізовані зображення значно впливають

на швидкість завантаження. Застосування формату WebP та зменшення розмірів зображень дозволить суттєво знизити їх вагу без втрати якості.

На сторінці використовуються зображення форматів PNG та SVG. Скориставшись сервісом https://tinypng.com для оптимізації PNG зображень ми отримали наступні результати (рис. 3.4).

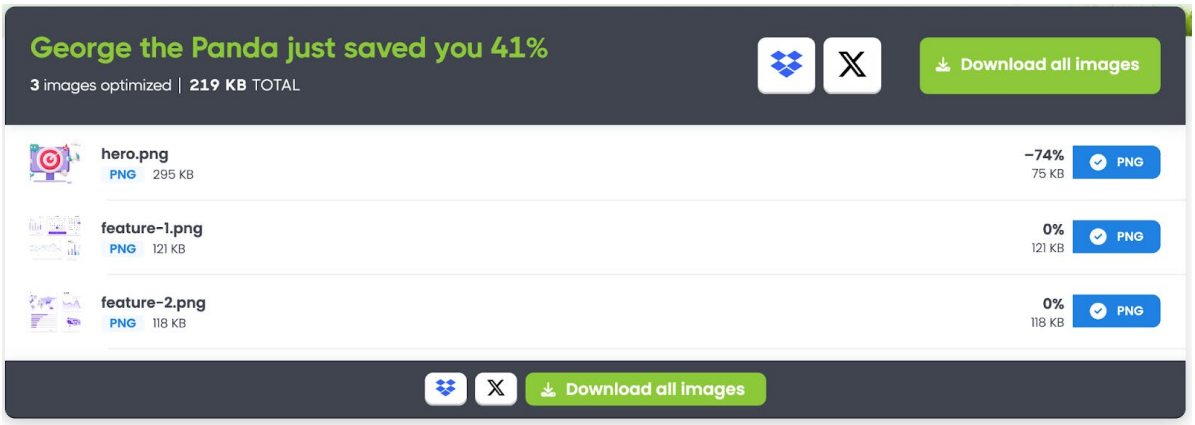


Рисунок 3.4 – Результати оптимізації PNG зображень

Як видно з рисунку, об’єм зображення hero.png вдалось скоротити на -74,5% (з 295 Кб до 75 Кб), без видимої втрати якості зображення.

Оптимізація інших PNG зображень не дала помітного результату. Це може бути спричинено тим що вихідний файл уже оптимізований або особливостями самих зображень.

Для більшої оптимізації скористаємось конвертацією форматів PNG у WebP, який має вищий рівень стиснення зі збереженням якості. Для цього скористаємось сервісом https://convertio.co. Результати конвертації представлені на рис. 3.5. З рисунку видно, що досягнутий результат оптимізації для зображень feature-1.png та feature-2.png складає приблизно 10% (з 118Kb до 106K для першого зображення і з 115 Kb до 104K для другого, відповідно).

Також зробимо відповідні заміни розширення файлів у коді з "./images/feature-1.png" на "./images/feature-1.webp" та "./images/feature-2.png" на "./images/feature-2.webp", відповідно.

Логотип у форматі SVG також оптимізуємо шляхом прибирання XLM ін-

струкцій, мета-даних та мініфікції, з допомогою сервісу https://svgomg.net.

Отриманий результат для файлу logo.svg – зниження розміру файлу з 5Kb до 3Kb (тобто зниження розміру на 40%).

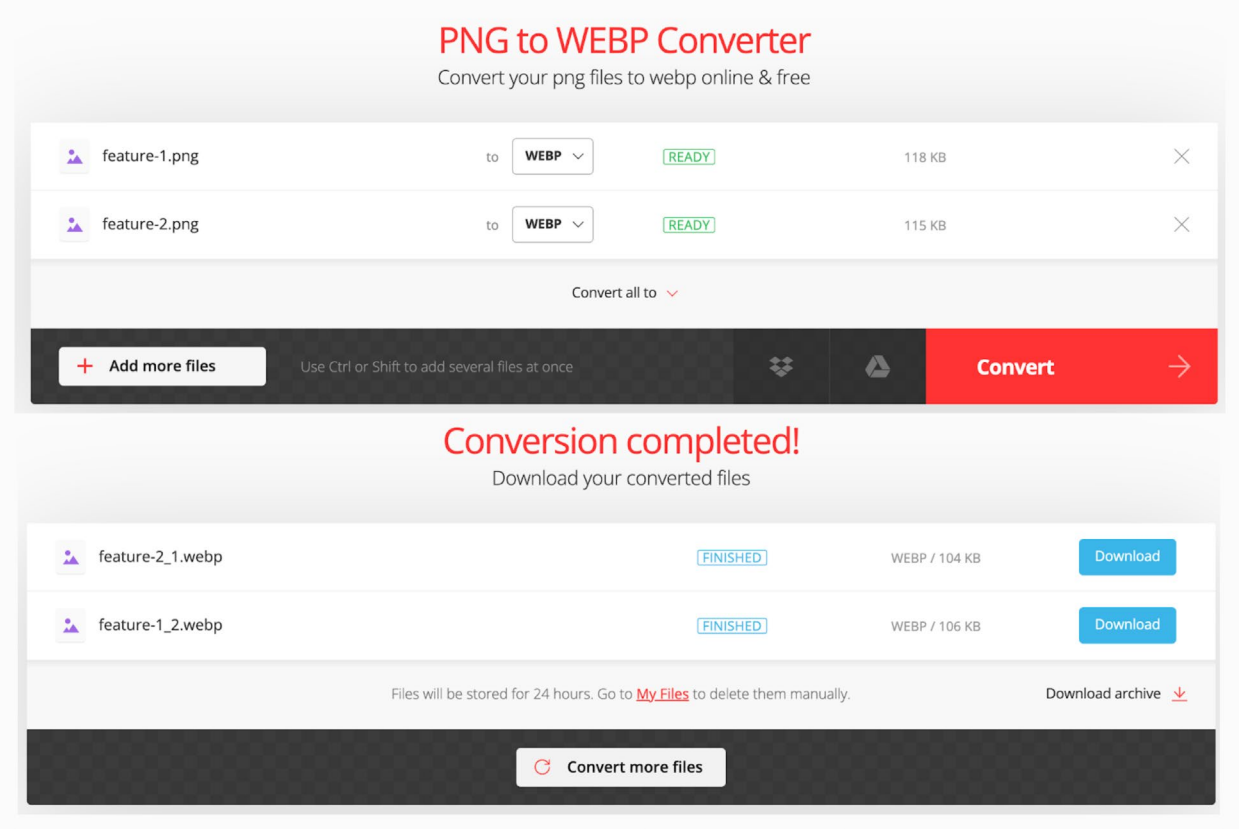


Рисунок 3.5 – Результати конвертації зображень у формат WEBP

Додатково були додані атрибути “width” і “height” до зображань, які дозволяють браузеру виділити відповідний простір на сторінці ще до завантаження зображення. Це допомагає уникнути зміщення контенту, коли зображення завантажується. Якщо ці атрибути не задані, браузер не знає розміру зображення до його завантаження. Це призводить до того, що після завантаження зображення контент зміщується, щоб врахувати його розмір. Зазначення розмірів усуває цю проблему, оскільки місце для зображення фіксується ще на етапі обробки HTML.

Така оптимізації додатково покращує продуктивність рендерингу та сприйняття сторінки користувачами, а також безпосередньо впливає на значення метрики Cumulative Layout Shift (CLS).

Метод 3. Використання техніки “lazy loading” для контенту. Наступним кроком оптимізації, для зменшення кількості одночасно завантажуваних ресурсів, буде впроваджено механізм відкладеного завантаження зображень та інших елементів за допомогою вбудованого атрибуту loading="lazy".

Техніка відкладеного завантаження контенту, або "lazy loading," має цілу низку переваг, які значно покращують ефективність роботи веб-сторінок та загальний користувацький досвід.

Відкладене завантаження дозволяє зменшити кількість ресурсів, які потрібно завантажити одразу під час першого завантаження веб-сторінки. Завдяки цьому знижується обсяг даних, які браузер завантажує в момент відкриття сторінки. Користувач швидше отримує доступ до основного контенту, що позитивно впливає на перше враження від сторінки.

Техніка "lazy loading" допомагає ефективніше використовувати пропускну здатність мережі, завантажуючи лише ті елементи, які користувач бачить на екрані (в межах "видимого вікна"). Це особливо корисно для користувачів з повільним або нестабільним інтернет-з'єднанням, оскільки ресурси, які не потрібні негайно, не споживають цінний трафік.

Оскільки сервер не обробляє запити на ресурси, які поки не потрібні користувачу, навантаження на сервер значно зменшується. Це може бути критично важливим для великих сайтів з великою кількістю відвідувачів, де економія серверних ресурсів напряму впливає на стабільність та швидкодію.

Крім того, у випадках, коли на сторінці є велика кількість зображень або інших важких елементів, їх завантаження може суттєво збільшити використання оперативної пам'яті браузером. Використання "lazy loading" дозволяє завантажувати лише ті елементи, які потрібні користувачеві в конкретний момент, тим самим зменшуючи обсяг використаної пам'яті.

Для реалізації оптимізації, додамо атрибут loading="lazy" для всіх тегів <img> та сторінці.

Метод 4. Асинхронне завантаження скриптів. Використання атрибутів async і defer для зовнішніх скриптів дозволить уникнути блокування основного

потоку завантаження сторінки.

Асинхронне завантаження скриптів із використанням атрибутів async і defer є ключовою практикою для оптимізації швидкості завантаження веб-сторінок. Ці атрибути дозволяють зовнішнім JavaScript-файлам завантажуватися окремо від основного потоку завантаження HTML-документа, що позитивно впливає на швидкість завантаження сайту та користувацький досвід.

За замовчуванням скрипти JavaScript блокують обробку DOM під час завантаження та виконання. Це означає, що браузер зупиняє побудову сторінки, поки скрипт не буде завантажено й виконано. Використання атрибутів async і defer дозволяє уникнути цієї проблеми, тому сторінка продовжує рендеринг навіть під час завантаження скриптів.

Зовнішні скрипти з атрибутом async завантажуються паралельно з іншими ресурсами, такими як зображення та CSS, що дозволяє зменшити загальний час завантаження сторінки. Для атрибута defer скрипти також завантажуються асинхронно, але їх виконання відкладається до завершення побудови DOM, що дозволяє браузеру швидше відображати сторінку для користувача.

Асинхронне завантаження також дозволяє уникнути зайвого блокування головного потоку, в якому браузер обробляє інші завдання, наприклад, взаємодію з користувачем, анімації чи прокручування сторінки. Це робить сторінку більш чутливою до дій користувача.

Для застосування асинхронного завантаження, додамо атрибути “async” та “defer” до тегів <script> які відповідальні за підвантаження бібліотек buttons.js та flowbite.js відповідно. Використання атрибуту “defer” у другому випадку виправдано специфікою його роботи, оскільки ця бібліотека вимагає взаємодії з вже завантаженим DOM-деревом.

Метод 5. Використання CDN. Використання CDN (Content Delivery Network) для завантаження зовнішніх ресурсів на веб-сторінці має низку суттєвих переваг, які позитивно впливають на продуктивність, доступність та безпеку веб-сайтів.

CDN дозволяє розподіляти статичний контент (наприклад, JavaScript,

CSS, зображення чи шрифти) через мережу серверів, розташованих по всьому світу. Коли користувач звертається до сторінки, ресурс завантажується з найближчого до нього сервера. Це мінімізує затримки, пов’язані з географічною відстанню, і суттєво скорочує час завантаження сторінки.

Наприклад, користувач з Європи отримає ресурс із європейського сервера, а не з центрального сервера в США. Це зменшує latency (затримки) та покращує загальний досвід користувача.

CDN забезпечує високу доступність ресурсів завдяки механізмам автоматичного резервування та балансування навантаження. У разі виходу з ладу одного з серверів запит автоматично перенаправляється на інший сервер мережі, що гарантує безперервний доступ до контенту. CDN також використовує розподілене кешування для зберігання популярних ресурсів на своїх серверах. Завдяки цьому браузер користувача отримує кешовану версію ресурсу з найближчого вузла мережі, що ще більше пришвидшує завантаження.

Окрім того, використання CDN дозволяє знизити навантаження на основний сервер веб-додатка, розподіляючи запити між численними вузлами. Це особливо важливо під час пікових навантажень (наприклад, у дні великих онлайн-продажів або під час популярних трансляцій).

Для статичних ресурсів, таких як шрифти, також провадимо завантаження через мережу доставки контенту (CDN). Це забезпечить більш швидке та надійне їх завантаження.

На тестовому веб-сайті використовується шрифт “PT Sans”, який підвантажуєтсья разом з рештою ресурсів. Для оптимізації додамо наступний код

“<link href="https://fonts.googleapis.com/css2?family=PT+Sans:ital,
wght @0,400;0,700;1,400;1,700&display=swap" rel="stylesheet">”

до коду веб-сторінки та видалимо посилання на існуючі локальні файли з директорії “./fonts” веб-сайту.

Таким чином ми оптимізуємо швидкість завантаження веб-сторінки шляхом зменшення часу завантаження шрифтів і підвищення надійності їхньої доставки користувачам.

Метод 6. Об’єднання і мініфікація ресурсів. Наступним кроком, для зменшення кількості HTTP-запитів, скористаємося технікою об’єднання CSS файлів, а також їх мініфікації.

Кожен окремий CSS файл вимагає окремого HTTP-запиту до сервера, що збільшує час завантаження сторінки, особливо для користувачів з повільним інтернет-з'єднанням. Коли CSS файли об’єднуються в один, браузер здійснює лише один запит для завантаження всіх необхідних стилів. Це суттєво скорочує загальний час завантаження сторінки, особливо якщо на сайті використовуються десятки CSS файлів.

Мініфікація CSS файлів – це процес видалення зайвих пробілів, коментарів, переносів рядків та інших непотрібних символів, які не впливають на функціональність коду. Це дозволяє значно зменшити розмір файлів. У поєднанні з об’єднанням це дає змогу скоротити трафік і прискорити завантаження сторінок навіть для мобільних пристроїв з обмеженими ресурсами.

Коли браузер отримує один оптимізований файл стилів, він швидше обробляє його, оскільки не потребує обробки декількох файлів окремо. Це призводить до швидшого рендерингу сторінки, забезпечуючи користувачам кращий досвід взаємодії.

Об’єднаний CSS файл зручно кешувати в браузері користувача. Це означає, що при повторному відвідуванні сайту браузер не завантажуватиме стилі знову, якщо вони не були змінені. Завдяки цьому значно зменшується час завантаження сторінок для постійних відвідувачів.

Менша кількість HTTP-запитів також зменшує навантаження на сервер, особливо під час пікових навантажень. Серверу стає простіше обробляти більшу кількість запитів одночасно, що підвищує стабільність роботи сайту.

Для об’єднання CSS файлів скористаємось будь-яким текстовим редактором, а для мініфікації скористаємось сервісом https://www.minifier.org.

Вигляд файлу до стилів до мініфікації наведено на рис. 3.6, а після – на рис. 3.7.

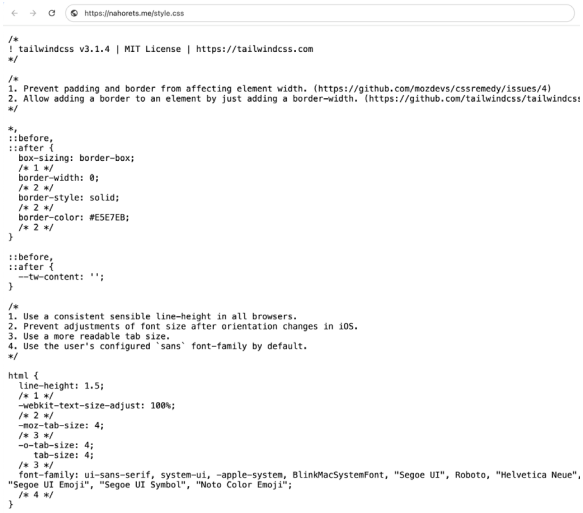


Рисунок 3.6 – Вигляд файлу до стилів до мініфікації

Рисунок 3.7 – Вигляд файлу до стилів після мініфікації

Метод 7. Кешування. Впровадження кешування на рівні сервера та браузера забезпечить повторне використання вже завантажених ресурсів, зменшуючи навантаження на сервер та час завантаження сторінок для постійних користувачів.

Кешування дозволяє зберігати вже завантажені ресурси, такі як зображення, файли JavaScript і CSS, на стороні клієнта або у проміжному кеші. Це означає, що при повторному зверненні до веб-сторінки сервер не отримує запит

на кожен ресурс, а відповідає лише за нові або змінені дані. Таким чином, значно скорочується кількість запитів до сервера, що допомагає знизити навантаження на його ресурси, особливо в пікові періоди відвідувань.

Повторне використання закешованих ресурсів дозволяє браузеру завантажувати сторінки швидше, оскільки більша частина інформації вже знаходиться локально на пристрої користувача. Це особливо важливо для великих веб-сайтів або веб-додатків, де завантаження статичних файлів може займати значний обсяг часу. Завдяки кешуванню користувач бачить готову сторінку практично миттєво.

Для мобільних користувачів або тих, хто працює у мережах з обмеженою пропускною здатністю, кешування має вирішальне значення. Повторне завантаження ресурсів споживає менше трафіку, що робить використання веб-сайту економічнішим для користувача.

Адміністратори веб-сайтів можуть гнучко керувати кешуванням, визначаючи, які ресурси і як довго мають зберігатися. Наприклад, статичні ресурси (зображення, стилі) можуть кешуватися на довгий час, тоді як динамічний контент оновлюється частіше. Це дозволяє підтримувати баланс між швидкістю роботи і актуальністю даних.

Для активації кешування на боці хостинг-провадера необхідно внести відповідні зміни до файлу “.htaccess” (який зазвичай розташовується в директорії “public_html”). Вихідний код який потрібно додати наведено в додатку А.

Пересвідчитись в успішності налаштувань можна за наявністю відповідних заголовків відповіді від серверу “Cache-Control”, “Expires” та інших.

Висновки за розділом 3

В рамках дослідження було реалізовано низку практичних заходів, спрямованих на оптимізацію швидкості завантаження веб-сторінок, із детальним аналізом кожного етапу.

Були детально описані умови та методи проведення експерименту, а також етапи практичної реалізації оптимізації. Серед цих етапів виділяються: зменшення кількості HTTP-запитів, зменшення обсягу завантажуваних даних через стиснення та оптимізацію ресурсів, використання CDN для доставки статичних елементів, а також мініфікація коду.

Ці оптимізації сприяли досягненню значного зменшення часу рендерингу й швидкого завантаження контенту, що особливо важливо для користувачів із низькою пропускною здатністю мережі або мобільних пристроїв.

Важливим компонентом оптимізації стало також впровадження механізмів кешування, які дозволяють повторно використовувати вже завантажені ресурси. Що дозволяє суттєво зменшити навантаження на сервер, оптимізувати використання пропускної здатності мережі та покращити швидкість роботи вебсайту при повторних відвідуваннях.

Кожен із застосованих підходів був спрямований на поліпшення окремих аспектів роботи веб-ресурсів, таких як зменшення обсягу переданих даних, зменшення кількості HTTP-запитів, застосування кешування, оптимізація об’єму ресурсів на стороні клієнта, а також скорочення часу рендерингу.

4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Аналіз результатів оптимізації

Одним із ключових етапів дослідження сучасних методів оптимізації швидкості завантаження веб-сторінок є експериментальне підтвердження їхньої ефективності. У цьому розділі представлено результати практичного застосування обраних методів оптимізації, а також проведено аналіз отриманих даних з метою оцінювання їхнього впливу на загальну швидкодію веб-ресурсів.

Основним завданням було перевірити, наскільки кожен із застосованих підходів сприяє зменшенню часу завантаження, підвищенню швидкості обробки контенту та покращенню користувацького досвіду.

У цьому розділі розглядаються такі аспекти:

– порівняння часу завантаження веб-сторінок до та після оптимізації;

– вплив окремих методів на різні метрики продуктивності, такі як First Contentful Paint (FCP), Largest Contentful Paint (LCP) та інші;

– аналіз ефективності обраних підходів у комплексі;

– рекомендації щодо подальшого використання методів оптимізації залежно від специфіки проєкту.

На основі отриманих даних буде зроблено висновки про доцільність застосування досліджуваних методів у сучасній практиці веб-розробки.

Для проведення експерименту з оцінювання сучасних методів оптимізації швидкості завантаження веб-сторінок був обраний тестовий веб-сайт із наступними початковими характеристиками (рис. 4.1):

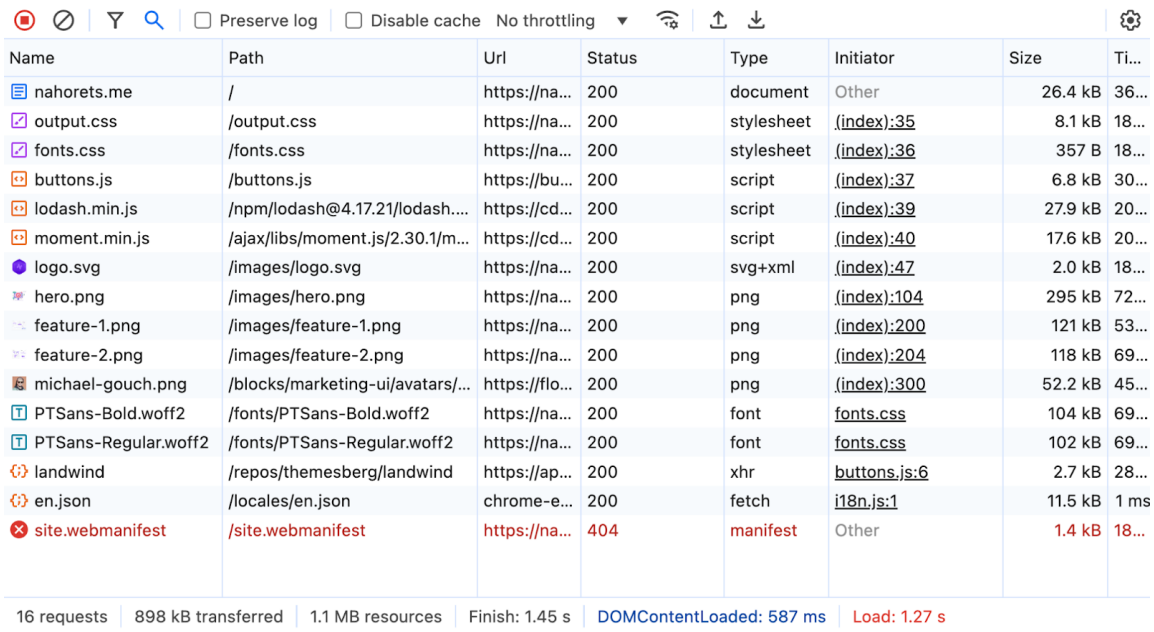
– кількість запитів до серверу: 16 HTTP-запитів;

– розмір отриманих даних: 898 Кб;

– загальний розмір ресурсів: 1.1 Мб;

– час повного завантаження сторінки: 1.45 секунди;

– час до події DOMContentLoaded: 587 мс;



Preserve log | Disable cache | No throttling

Name	Path	Url	Status	Type	Initiator	Size	Ti...
nahorets.me	/	https://na...	200	document	Other	26.4 kB	36...
output.css	/output.css	https://na...	200	stylesheet	(index):35	8.1 kB	18...
fonts.css	/fonts.css	https://na...	200	stylesheet	(index):36	357 B	18...
buttons.js	/buttons.js	https://bu...	200	script	(index):37	6.8 kB	30...
lodash.min.js	/npm/lodash@4.17.21/lodash....	https://cd...	200	script	(index):39	27.9 kB	20...
moment.min.js	/ajax/libs/moment.js/2.30.1/m...	https://cd...	200	script	(index):40	17.6 kB	20...
logo.svg	/images/logo.svg	https://na...	200	svg+xml	(index):47	2.0 kB	18...
hero.png	/images/hero.png	https://na...	200	png	(index):104	295 kB	72...
feature-1.png	/images/feature-1.png	https://na...	200	png	(index):200	121 kB	53...
feature-2.png	/images/feature-2.png	https://na...	200	png	(index):204	118 kB	69...
michael-gouch.png	/blocks/marketing-ui/avatars/...	https://flo...	200	png	(index):300	52.2 kB	45...
PTSans-Bold.woff2	/fonts/PTSans-Bold.woff2	https://na...	200	font	fonts.css	104 kB	69...
PTSans-Regular.woff2	/fonts/PTSans-Regular.woff2	https://na...	200	font	fonts.css	102 kB	69...
landwind	/repos/themesberg/landwind	https://ap...	200	xhr	buttons.js:6	2.7 kB	28...
en.json	/locales/en.json	chrome-e...	200	fetch	i18n.js:1	11.5 kB	1 ms
site.webmanifest	/site.webmanifest	https://na...	404	manifest	Other	1.4 kB	18...

16 requests | 898 kB transferred | 1.1 MB resources | Finish: 1.45 s | DOMContentLoaded: 587 ms | Load: 1.27 s

Рисунок 4.1 – Параметри веб-сайту до оптимізації

До проведення експерименту метрики продуктивності згідно з Google Lighthouse мали значення, наведені на рис. 4.2.

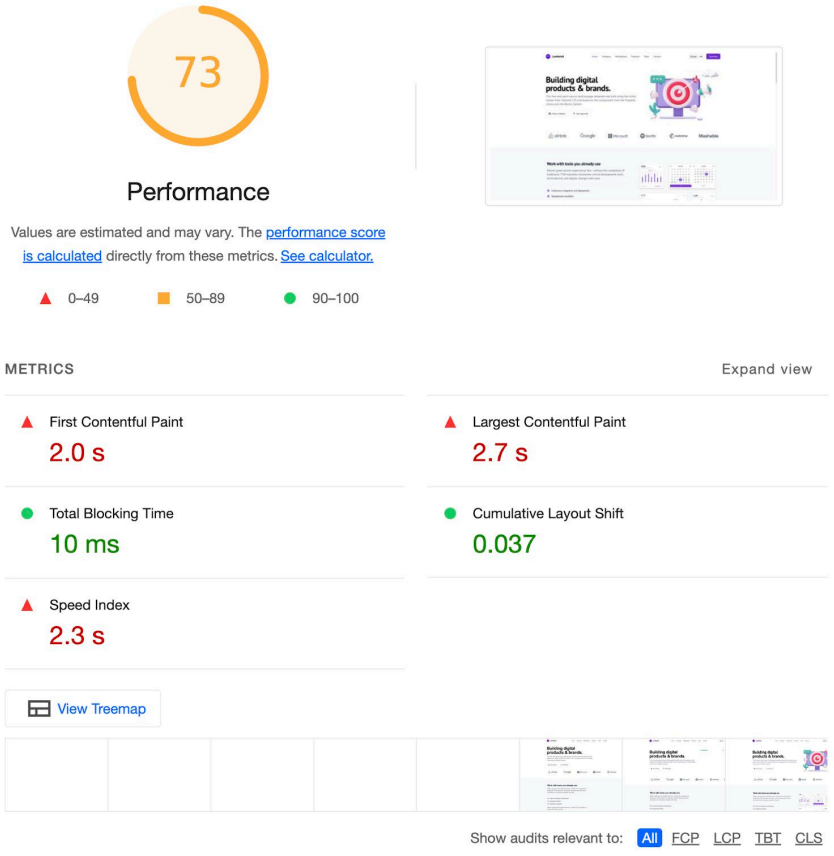


Рисунок 4.2 – Параметри веб-сайту до оптимізації в Google Lighthouse

Як видно з зображення, початкові показники метрик складали:

– загальний показник продуктивності: 73 (на рівні "середній");

– First Contentful Paint (FCP): 2.0 секунди (перевищує рекомендовані значення);

– Largest Contentful Paint (LCP): 2.7 секунди (перевищує рекомендовані значення);

– Cumulative Layout Shift (CLS): 0.037 (відповідає нормам);

– Total Blocking Time (TBT): 90 мс (в межах норми);

– Speed Index: 2.3 секунди (перевищує рекомендовані значення).

Також можна відзначити наступне:

– значення FCP та LCP свідчать про затримки у відображенні основного контенту;

– час повного завантаження сторінки (1.45 секунди) перевищує оптимальні значення, але залишається прийнятним.

Ці параметри було зафіксовано до застосування будь-яких оптимізацій. Подальші результати експерименту порівнюватимуться із зазначеними початковими значеннями для оцінки ефективності впроваджених методів оптимізації.

Далі проаналізуємо, як кожен етап оптимізації вплинув на показники швидкодії веб-сайту.

Етап 1. Зменшення кількості сторонніх скриптів. Після застосування оптимізації:

– розмір переданих даних: зменшено з 897 КБ до 851 КБ. Це зменшення пов'язане з видаленням важких бібліотек;

– кількість запитів: зменшилась з 16 до 13, оскільки відтепер менше ресурсів завантажується, окрім того прибрано надлишковий запит на маніфест-файл;

– показник FCP: зменшився до 1.4 с, оскільки зменшення кількості скриптів зменшують і час до першого відображення контенту;

– показник LCP: знизився до 2.0 с, оскільки відбувається менше затримок

від рендерингу через сторонні бібліотеки.

– Speed Index: покращився до 1.35 с;

– Загальний показник продуктивності: підвищився до 86.

Видалення сторонніх скриптів суттєво вплинуло на швидкість завантаження першого контенту (FCP) та загальну продуктивність сторінки, оскільки зменшилась кількість ресурсів, які необхідно завантажувати та виконувати під час рендерингу сторінки. Завдяки цьому користувачі отримали можливість швидше побачити основний контент, що позитивно впливає на досвід взаємодії з веб-сайтом.

Зменшення залежності від великих бібліотек, таких як Moment.js та Lodash, дозволило звільнити ресурси браузера для інших завдань, поліпшивши показники швидкодії. Крім того, усунення надлишкового JavaScript-коду знизило ймовірність потенційних конфліктів між скриптами, що також може сприяти стабільності роботи сторінки.

Варто зазначити, що ефективність видалення сторонніх скриптів залежить від типу та обсягу JavaScript-коду, який використовується на конкретному веб-сайті. Наприклад, якщо скрипти виконують критичні для функціонування сторінки задачі, то їх видалення може вимагати заміни більш легковаговими альтернативами або власними реалізаціями функціоналу. Водночас, якщо ці скрипти є надлишковими і не несуть значної функціональної цінності, їх видалення дає суттєвий приріст продуктивності.

Таким чином, оптимізація через видалення сторонніх скриптів не лише покращує технічні показники веб-сторінки, а й сприяє досягненню покращення загального враження від веб-сайту.

Етап 2. Оптимізація зображень. Розглянемо результати застосування оптимізованих зображень:

– розмір переданих даних: значно зменшився (до 607 КБ), оскільки оптимізація зображень суттєво скоротила об'єм даних, що передаються;

– кількість запитів: незмінна (13);

– показники FCP та LCP: знизилися до 1.3 с та 1.7 с відповідно;

– Speed Index: покращився до 1.27 с;

– загальний показник продуктивності: підвищився до 87.

Оптимізація зображень дала найбільший ефект на зменшення розміру переданих даних, що значно вплинуло на швидкість рендерингу та загальну продуктивність веб-сторінки. Перехід на сучасні формати, такі як WebP, а також зменшення розмірів зображень без втрати їхньої якості, дозволили значно знизити навантаження на мережу та браузер. У результаті користувачі отримали швидший доступ до контенту.

Крім того, зменшення розміру даних сприяло зниженню часу завантаження ключових елементів сторінки, таких як First Contentful Paint (FCP) і Largest Contentful Paint (LCP), що позитивно позначилося на взаємодії користувачів із веб-сайтом. Завдяки меншому обсягу завантажуваних ресурсів браузер міг швидше завершити процес рендерингу, забезпечуючи плавний і приємний досвід використання.

Важливо зазначити, що ефективність оптимізації зображень залежить від вихідних умов. Наприклад, якщо зображення вже були зменшені або використовували ефективний формат, ефект може бути менш вираженим. Проте для веб-сайтів, де раніше не застосовувались передові методи стиснення та оптимізації, покращення може бути дуже значним.

Окрім підвищення швидкості, оптимізація зображень також знижує витрати на передачу даних, що є важливим фактором для користувачів із обмеженим інтернет-планом. Це робить веб-сайт не лише технічно більш ефективним, а й більш доступним для широкої аудиторії. Таким чином, оптимізація зображень є критично важливим кроком у процесі вдосконалення веб-ресурсів, що впливає як на технічні аспекти роботи сторінки, так і на бізнес-результати, такі як покращення показника утримання користувачів та зниження показника відмов.

Етап 3. Застосування lazy-loading для контенту. Результати застосування технології “lazy loading” для контенту наступні:

– розмір переданих даних: зменшився до 555 КБ, оскільки не всі ресурси

завантажуються одразу;

– кількість запитів: знизилася до 12;

– показники FCP та LCP: ще більше зменшились (1.3 с та 1.7 с);

– Speed Index: покращився до 1.2 с;

– загальний показник продуктивності: зріс до 89.

Отже lazy loading ефективно зменшує кількість активних завантажень, знижуючи навантаження на сервер і мережу, що, у свою чергу, прискорює видимість контенту для користувачів. Цей підхід дозволяє завантажувати лише ті ресурси, які необхідні на даний момент, наприклад, зображення або відео, що знаходяться у видимій області екрана. Решта ресурсів завантажується поступово, у міру прокручування сторінки, що робить процес завантаження значно більш ефективним.

Використання lazy loading безпосередньо сприяє зниженню часу First Contentful Paint (FCP) і Largest Contentful Paint (LCP), оскільки браузер фокусується на рендерингу першочергового контенту, не витрачаючи ресурси на завантаження елементів, які поки не видно користувачеві. Це покращує досвід взаємодії з веб-сайтом, особливо для мобільних пристроїв і повільних інтернет-з’єднань, де швидкість є вирішальним фактором.

Lazy loading також знижує обсяг переданих даних. Однак варто враховувати, що lazy loading потрібно використовувати розумно. Неправильна реалізація або застосування до критичних елементів сторінки, таких як банери або основний контент, може призвести до негативного досвіду для користувачів. Тому ключовим аспектом є баланс між ефективністю завантаження та забезпеченням належної якості обслуговування.

Загалом, lazy loading виявився ефективним інструментом для оптимізації швидкодії, що дозволило суттєво покращити технічні показники та покращити користувацький досвід. Впровадження цієї технології є обов’язковим етапом для сучасних веб-ресурсів, які прагнуть відповідати вимогам часу і забезпечувати максимальну ефективність роботи.

Етап 4. Асинхронне завантаження скриптів. Застосування підходу з асин-

хронним завантаженням скриптів дозволило оптимізувати наступні показники:

– FCP та LCP: не змінились порівняно з lazy loading (1.3 с та 1.7 с);

– Speed Index: незначне покращення (1.2 с);

– загальний показник продуктивності: зріс до 90.

Асинхронне завантаження скриптів мінімізує блокування ресурсів, дозволяючи браузеру виконувати інші завдання, такі як рендеринг сторінки, замість очікування завантаження та виконання JavaScript. Завдяки використанню атрибутів async і defer, скрипти можуть завантажуватися незалежно від основного потоку завантаження сторінки, що покращує швидкодію, особливо для сайтів із великою кількістю скриптів.

Однак ефект від асинхронного завантаження стає менш помітним після попередніх кроків оптимізації, таких як видалення сторонніх скриптів або об'єднання та мініфікація ресурсів. Це пов'язано з тим, що більшість критичних задач вже було оптимізовано, і асинхронне завантаження виконує більше роль завершального етапу поліпшення продуктивності. Проте для складних сайтів із великою кількістю JavaScript-коду або інтерактивних елементів цей підхід залишається важливим.

Слід також враховувати, що некоректна реалізація асинхронного завантаження може призвести до несправності окремих функцій або неправильної послідовності виконання скриптів. Тому під час впровадження цієї техніки необхідно ретельно тестувати роботу сайту на всіх основних браузерах і пристроях.

Загалом, асинхронне завантаження є ефективним і сучасним методом оптимізації швидкодії веб-сайтів, особливо в умовах зростання складності вебдодатків.

Етап 5. Використання CDN. Застосування оптимізації шляхом перенесення завантаження шрифтів на CDN дало наступні результати:

– розмір переданих даних: залишився на рівні 555 КБ;

– кількість запитів: лишилась незмінна (12);

– показники FCP, LCP: не змінились;

– Speed Index: покращився до 1.1 с.

Використання CDN (Content Delivery Network) забезпечує стабільність і швидкість доступу до шрифтів та інших статичних ресурсів, зменшуючи затримки, які можуть виникати під час їх завантаження з основного сервера. Завдяки розподіленій мережі серверів, розташованих у різних регіонах світу, користувачі отримують доступ до контенту з найближчого вузла, що значно зменшує час відповіді і затримки, особливо для відвідувачів із віддалених географічних місць.

У даному випадку, хоча використання CDN і не дало значного покращення технічних показників, таких як FCP чи LCP, його впровадження сприяє підвищенню загальної надійності завантаження ресурсів. Це особливо актуально для великих веб-сайтів із глобальною аудиторією, де перебої в доступі до шрифтів або інших ресурсів можуть суттєво погіршити користувацький досвід.

Крім того, CDN дозволяє зменшити навантаження на основний сервер, що може бути критично важливим під час пікових навантажень або DDoS-атак. Використання CDN також часто забезпечує автоматичне стиснення даних, оптимізацію кешування та підтримку HTTP/2, що додатково сприяє покращенню продуктивності веб-сайту.

Варто зазначити, що ефективність використання CDN залежить від типу та обсягу ресурсів, що обслуговуються. У даному прикладі ефект був обмеженим, оскільки основна частина оптимізації була зосереджена на зображеннях, скриптах та інших елементах, тоді як внесок шрифтів у загальний обсяг даних був відносно невеликим.

Попри це, впровадження CDN є важливим компонентом сучасної архітектури веб-сайтів, оскільки воно забезпечує масштабованість і стабільність у довгостроковій перспективі. Хоча у короткостроковій перспективі ефект може бути менш помітним, його користь стає очевидною у разі зростання трафіку або розширення аудиторії сайту.

Етап 6. Об’єднання і мініфікація ресурсів. Застосування підходів з об’єднання і мініфікації статичних ресурсів дало наступні результати:

– розмір переданих даних: суттєво зменшився (до 346 КБ);

– кількість запитів: лишилася незмінною (12);

– показники FCP та LCP: значно зменшились (0.9 с та 1.3 с);

– загальний показник продуктивності: досяг 94.

Мініфікація і об’єднання ресурсів значно зменшили об'єм даних, які передаються між сервером і браузером, що позитивно вплинуло на швидкість завантаження сторінки. Зменшення обсягу CSS-файлів за допомогою видалення зайвих пробілів, коментарів та символів дозволило скоротити час завантаження ресурсів і підвищити ефективність їх виконання.

Ця оптимізація значно вплинула на показники First Contentful Paint (FCP) і Largest Contentful Paint (LCP), скоротивши час до появи контенту на екрані користувача. Також помітно покращився Speed Index, що свідчить про швидше завантаження видимого контенту.

Варто зазначити, що мініфікація і об’єднання ресурсів особливо ефективні для сайтів із великою кількістю зовнішніх скриптів та стилів. Однак для максимальної ефективності слід враховувати потенційні ризики, такі як проблеми з кешуванням, коли змінений файл вимагає оновлення у користувачів, або труднощі з налагодженням у разі виникнення помилок у мініфікованому коді. Для цього рекомендується використовувати сучасні інструменти, такі як Webpack або Gulp, які дозволяють автоматизувати процес оптимізації й уникнути типових помилок.

У результаті, мініфікація і об’єднання ресурсів є потужними інструментами для покращення продуктивності веб-сайтів. Вони не лише прискорюють завантаження сторінок, але й сприяють зниженню витрат на трафік та підвищенню зручності користувачів, що особливо важливо для сайтів із високим трафіком або глобальною аудиторією. Така оптимізація є невід'ємним етапом у створенні швидких, стабільних і конкурентоспроможних веб-ресурсів.

Етап 7. Кешування. Застосування кешування для статичних файлів дало наступні результати:

– розмір переданих даних: суттєво зменшено, до 13 КБ;

– кількість запитів: знизилася до 9;

– показники FCP та LCP: найкращі результати (0.7 с та 0.9 с);

– Speed Index: досяг 0.9 с;

– загальний показник продуктивності: підвищився до 98.

Кешування забезпечило найбільший приріст швидкодії шляхом мінімізації кількості запитів до серверу та повторного використання вже завантажених ресурсів.

Завдяки цьому значно скоротився обсяг переданих даних, що робить завантаження сторінки швидшим і ефективнішим, особливо для повторних відвідувачів сайту. У даному випадку кешування дозволило досягти мінімального розміру переданих даних – лише 13 КБ, а кількість запитів зменшилась до 9, що суттєво перевершує результати інших оптимізацій.

Однією з ключових переваг кешування є можливість знизити навантаження на сервер, що особливо актуально для веб-сайтів із високим трафіком або в пікові періоди. Локальне збереження ресурсів на стороні клієнта дозволяє браузеру швидше обробляти запити, оскільки ресурси вже доступні в кеші і не потребують повторного завантаження. Це також позитивно впливає на економію інтернет-трафіку для користувачів.

Крім того, кешування значно покращує технічні показники веб-сайту, такі як First Contentful Paint (FCP) і Largest Contentful Paint (LCP), зменшуючи час до появи видимого контенту. У цьому випадку показники FCP і LCP досягли найкращих значень – 0.7 с та 0.9 с відповідно, а загальний показник продуктивності виріс до 98, що демонструє винятковий рівень оптимізації.

Варто зазначити, що ефективність кешування залежить від правильної конфігурації. Наприклад, необхідно встановити відповідні заголовки кешування (Cache-Control, Expires, ETag), щоб визначити, які ресурси слід зберігати і як довго. Неправильне налаштування може призвести до проблем із оновленням контенту або використанням застарілих версій файлів. Крім того, для динамічного контенту може знадобитися часткове кешування або використання CDN для кешування ресурсів у різних регіонах.

Таким чином, кешування є одним із найефективніших методів оптимізації продуктивності веб-сайту, який забезпечує як технічні, так і бізнесові переваги. Воно сприяє підвищенню швидкості завантаження, зниженню навантаження на сервер, покращенню користувацького досвіду і підвищенню конверсій.

На основі результатів експериментів, представлених у таблиці 4.1, можна зробити висновок, що застосування сучасних методів оптимізації значно покращує показники продуктивності веб-сторінок.

Таблиця 4.1 – Результати оптимізації

Показник	Початкові показники	Зменшення кількості скриптів	Оптимізація зображень	Lazy loading	Асинхронне завантаження	Використання CDN	Об’єднання і мініфіка-ція	Кешування
Розмір даних, Кб	897	851	607	555	555	554	346	13
Кількість запитів	16	13	13	12	12	12	12	9
TTFB, мс	50	50	50	50	50	50	50	50
FCP, с	2	1.4	1.3	1.3	1.3	1.3	0.9	0.7
LCP, с	2.7	2	1.7	1.7	1.7	1.7	1.3	0.9
CLS	0.037	0.037	0	0	0	0	0	0
TBT, мс	90	90	90	90	90	90	90	90
Speed index, с	1.45	1.35	1.3	1.27	1.2	1.1	1	0.9
Загальний показник продуктивності	73	86	87	89	90	91	94	98

Як видно з наведених даних, найбільший позитивний ефект спостерігається при використанні кешування, що дозволило знизити обсяг переданих даних до 13 КБ та кількість запитів до сервера до 9, забезпечуючи найкращий загальний показник продуктивності – 98.

Інші методи, такі як об'єднання і мініфікація ресурсів, асинхронне завантаження скриптів та lazy loading контенту, також продемонстрували значне по-

кращення ключових метрик, таких як First Contentful Paint (до 0.7 с) та Largest Contentful Paint (до 0.9 с). Загальний показник продуктивності для цих методів коливається в межах 86 – 94, що підкреслює їхню ефективність у вирішенні задач оптимізації швидкості завантаження веб-сторінок.

Наведемо графіки які ілюструють зміни ключових показників продуктивності для кожного з методів оптимізації (рис. 4.3). На графіках буде видно, як кожен метод впливає на такі метрики, як швидкості завантаження (Speed index), обсяг переданих даних, кількість запитів до сервера, час до першого відображення контенту (FCP), найбільший контентний елемент (LCP) та загальний показник продуктивності. Це дозволить наочно порівняти ефективність різних підходів та обґрунтувати вибір оптимальних методів для впровадження.

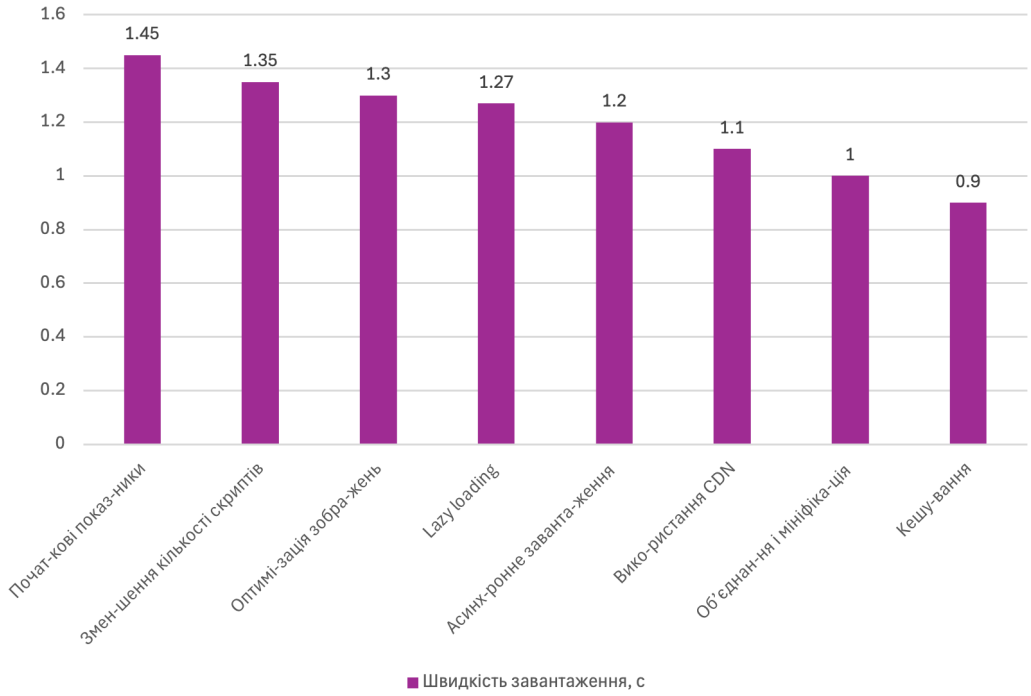


Рисунок 4.3 – Динаміка зміни швидкості завантаження внаслідок застосування методів оптимізації

Як бачимо, після впровадження комплексу оптимізаційних заходів, результати завантаження веб-сторінки суттєво покращилися. Графік на рисунку 4.3 наочно ілюструє послідовне зниження часу завантаження веб-сторінки з

впровадженням кожного наступного методу оптимізації. Сукупне скорочення часу завантаження становить – 0.55 с.

Графіки на рис. 4.4 та рис. 4.5 ілюструє ефективність кожного з застосованих методів оптимізації в зменшенні трафіку та покращенні продуктивності завантаження веб-сторінок.

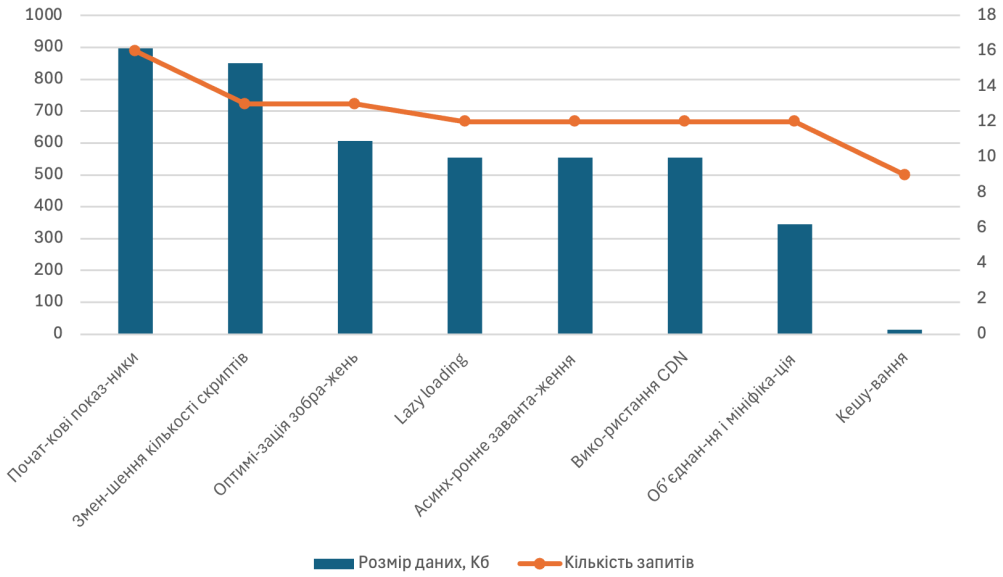


Рисунок 4.4 – Розмір даних що передаються та кількість запитів

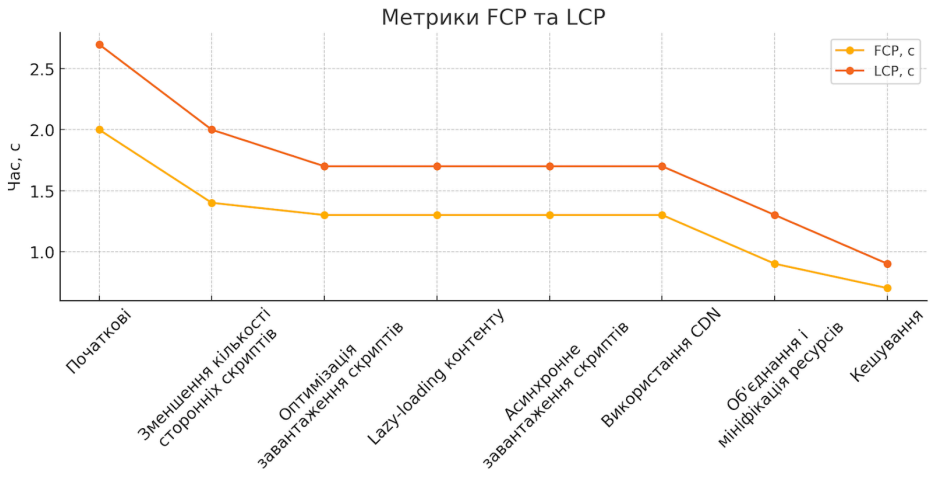


Рисунок 4.5 – Метрики FCP та LCP

Графік на рисунку 4.6 показує загальний показник продуктивності веб-сторінки, оцінений у балах (до 100), для різних методів оптимізації. Даний графік підтверджує важливість комплексного підходу до оптимізації, коли одноча-

сно використовуються кілька методів для забезпечення найкращих показників продуктивності.

Рисунок 4.6 – Загальний показник продуктивності

Фінальні метрики продуктивності після запроваджених оптимізацій становили (рис. 4.7):

– кількість запитів до серверу: 9;

– розмір передаваємих даних: 12.7 Кб;

– загальний розмір ресурсів: 382 Кб;

– час повного завантаження сторінки: 663 мс;

– час до події DOMContentLoaded: 332 мс.

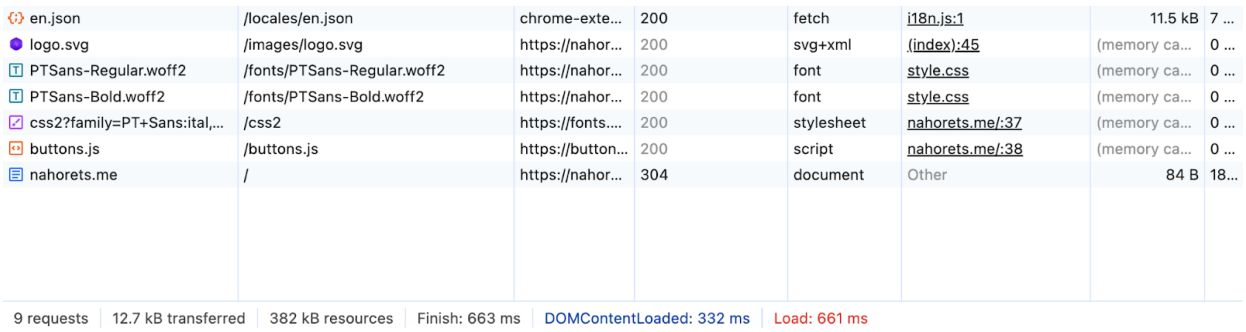


Рисунок 4.7 – Параметри веб-сайту після оптимізації

Після оптимізації параметри в Google Lighthouse покращились до значень

(рис. 4.8):

– загальний показник продуктивності: 98 (відмінний результат).

– First Contentful Paint (FCP): 0.7 секунди;

– Largest Contentful Paint (LCP): 0.9 секунди;

– Speed Index: 0.7 секунди;

– Cumulative Layout Shift (CLS): 0 (відсутні зсуви макету);

– Total Blocking Time (TBT): 100 мілісекунд.

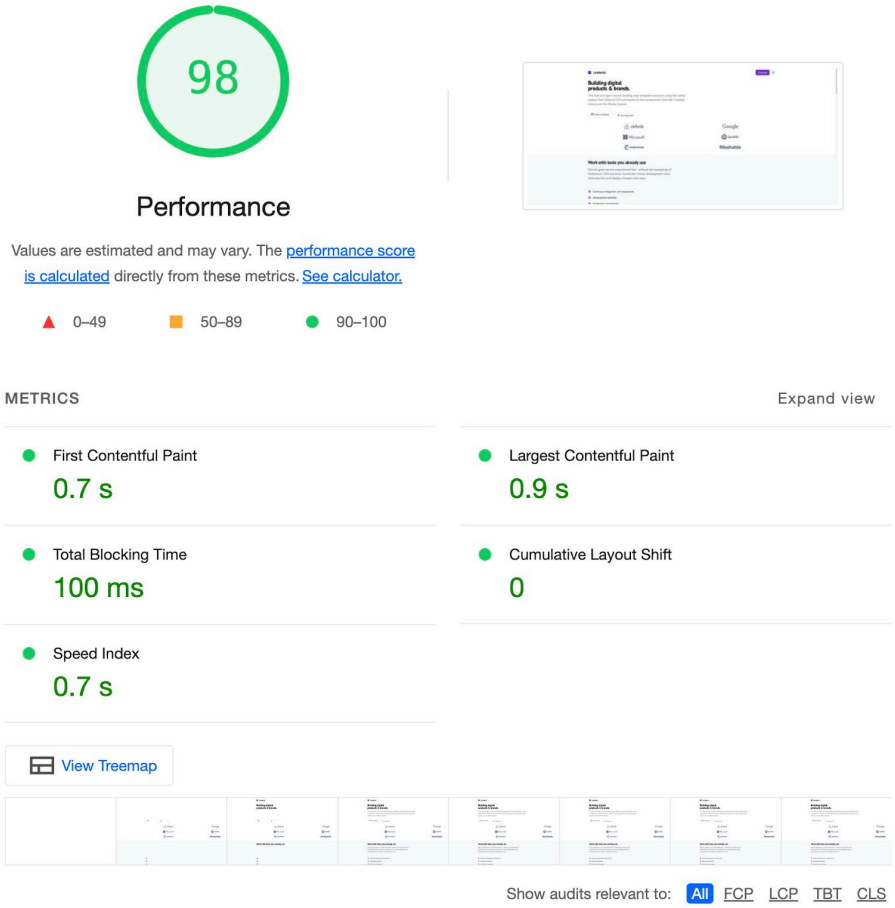


Рисунок 4.8 – Параметри веб-сайту після оптимізації в Google Lighthouse

Для глибшого аналізу результатів оптимізації швидкості завантаження веб-сторінок, досягнутих у ході експерименту, застосуємо математичну модель до отриманих результатів. Модель охоплює основні аспекти процесу завантаження, включаючи загальний час завантаження, обмеження розміру ресурсів, час мережі та метрику Largest Contentful Paint (LCP).

Цільова функція:

$$\min T = \min\left(T_{\text{network}} + T_{\text{rendering}}\right).$$

В експерименті отриманий загальний час завантаження сторінки становив $T = 0,663$ секунд.

Обчислимо час завантаження ресурсів за формулою:

$$T_{\text{network}} \approx \frac{S_{\text{total}}}{B} + L.$$

$S_{\text{total}} = 382$ Кб – загальний розмір ресурсів після оптимізації.

$B = 100$ Мбайт/с – середня пропускна здатність мережі.

$L = 50$ мс – середня затримка мережі.

$$T_{network} = \frac{382 \times 8}{100 \times 10^6} + 0,05 = 0,03556 + 0,05 = 0,08556 \text{ с.}$$

Час рендерингу $T_{rendering}$ суттєво залежав від кількості елементів N на сторінці. Успішна оптимізація DOM дозволила скоротити цей час.

Загальний розмір ресурсів відповідає обмеженню:

$$S_{total} \le S_{\max}.$$

У нашому випадку $S_{total} = 382$ Кб, що менше за стандартні $S_{\max} = 500$ Кб, забезпечуючи відповідність вимогам оптимізації.

Проведені оптимізації забезпечили також дотримання обмеження на кількість HTTP-запитів:

$$R_{total} \le R_{\max}.$$

У ході експерименту було скорочено кількість HTTP-запитів до 9, що значно нижче типового $R_{total} = 20$, рекомендованого для високопродуктивних сайтів.

Модель вимагає, щоб цільовий рівень рендерингу був нижче порогу:

$$LCP \le LCP_{\text{target}}.$$

В нашому випадку цільове значення $LCP_{\text{target}} = 2$ секунди. У результаті оптимізації вдалося досягти $LCP = 0{,}9$ секунди, що свідчить про високу ефективність методів оптимізації.

Комбінація обмежень на розмір ресурсів $S_{\max}$, кількість запитів $R_{\max}$ та цільові метрики LCP_{target} забезпечила досягнення оптимальних показників завантаження сторінки.

Отже, математична модель успішно підтвердила експериментальні дані, забезпечуючи кількісне обґрунтування ефективності застосованих методів оптимізації. Завдяки цьому вдалося досягти не лише відповідності сучасним стандартам швидкодії, але й закласти фундамент для подальшої автоматизації процесу оптимізації веб-ресурсів.

Застосовані методи оптимізації довели свою ефективність, що підтверджується суттєвим покращенням метрик продуктивності.

Проведений аналіз методів оптимізації швидкості завантаження вебсторінок демонструє, що для забезпечення максимальної продуктивності вебсайтів слід дотримуватись комплексного підходу до оптимізації та орієнтуючись на найефективніші методи залежно від специфіки проєкту.

Зменшення кількості сторонніх скриптів та оптимізація їхньої роботи допоможуть значно скоротити час завантаження сторінки. Видалення зайвих бібліотек або заміна їх легшими альтернативами рекомендується для проєктів, де

більшість контенту є статичним або не вимагає значної інтерактивності.

Оптимізація зображень є ключовим кроком для ресурсів із великою кількістю графічного контенту. Перехід на сучасні формати, такі як WebP, а також зменшення розмірів файлів без втрати якості знизять навантаження на сервер і покращать швидкість завантаження. Цей підхід особливо актуальний для інтернет-магазинів і фотогалерей, де швидке відображення візуального контенту має критичне значення.

Для довгих сторінок або ресурсів із великою кількістю динамічного контенту слід реалізувати технологію lazy loading, що дозволяє завантажувати лише видимий користувачем контент. Це знижує навантаження на сервер і прискорює рендеринг сторінки. Використання асинхронного завантаження скриптів додатково мінімізує блокування рендерингу та підвищує ефективність роботи браузера, особливо для динамічних веб-додатків.

Застосування CDN є обов’язковим для проєктів із глобальною аудиторією. Використання розподіленої мережі серверів зменшить час відповіді для користувачів із різних регіонів, а також підвищить стабільність доступу до ресурсів. Крім цього, об’єднання та мініфікація статичних ресурсів дозволять зменшити їхній розмір, що позитивно вплине на продуктивність і спростить управління файлами.

Необхідно також впроваджувати стратегії кешування, які дозволяють повторно використовувати вже завантажені ресурси, скорочуючи обсяг запитів до сервера. Це особливо ефективно для проєктів із високим рівнем відвідуваності, де кожна секунда завантаження сторінки має значення. Для максимального ефекту слід налаштувати відповідні заголовки кешування та забезпечити коректну роботу з динамічним контентом.

Таким чином, комплексна реалізація цих методів оптимізації не лише підвищить швидкість завантаження, але й забезпечить кращий досвід користувачів і більшу стабільність веб-ресурсів, враховуючи сучасні технологічні вимоги.

Висновки за розділом 4

У результаті проведеного експериментального дослідження сучасних методів оптимізації швидкості завантаження веб-сторінок вдалося досягти суттєвого підвищення продуктивності тестового веб-ресурсу. Під час аналізу ефективності різних підходів було виявлено, що кожен із методів робить свій унікальний внесок у покращення загальних показників. Наприклад, зменшення кількості сторонніх скриптів дозволило суттєво скоротити час до відображення першого контенту, а оптимізація зображень найбільше вплинула на зменшення обсягу переданих даних і пришвидшення рендерингу.

Інтеграція технології lazy loading показала себе як ефективний інструмент для зниження навантаження на сервер і підвищення швидкості завантаження видимого контенту, що є особливо актуальним для ресурсів із великою кількістю зображень чи мультимедійних файлів. Асинхронне завантаження скриптів і використання CDN допомогли покращити стабільність і доступність ресурсів, що критично важливо для сучасних динамічних веб-додатків із глобальною аудиторією.

Найбільший приріст у продуктивності було досягнуто завдяки реалізації кешування, яке забезпечило мінімальний обсяг запитів до сервера та максимальну швидкість доступу до контенту. Кешування стало ключовим елементом загальної архітектури оптимізації, дозволивши значно скоротити час завантаження сторінки для повторних відвідувань.

Комбіноване застосування мініфікації й об’єднання ресурсів забезпечило не лише підвищення швидкості завантаження, але й спростило керування файлами та зменшило ризик виникнення конфліктів між скриптами. Водночас проведений експеримент підтвердив важливість адаптивного підходу до вибору методів оптимізації, адже їхня ефективність значною мірою залежить від специфіки кожного окремого веб-ресурсу.

Таким чином, результати проведеного дослідження демонструють, що впровадження комплексного підходу до оптимізації, який включає декілька ме-

тодів, дозволяє не лише досягти сучасних стандартів швидкодії, але й забезпечити кращий досвід користувачів. Використання таких підходів сприяє покращенню бізнесових результатів, збільшенню утримання аудиторії та зменшенню витрат на підтримку веб-ресурсів. Це підтверджує доцільність і необхідність інтеграції сучасних методів оптимізації в процес розробки та підтримки веб-додатків.

ВИСНОВКИ

В рамках дослідження було проведено детальний аналіз методів оптимізації швидкості завантаження веб-сторінок, їхніх переваг та обмежень, а також інструментів, що дозволяють об’єктивно оцінювати результати оптимізації. Було вивчено сучасні підходи та стандарти веб-розробки, які сприяють підвищенню продуктивності та покращенню користувацького досвіду.

Особлива увага приділялася впровадженню таких методів, як кешування, стиснення та оптимізація завантаження ресурсів. В рамках дослідження було обґрунтовано вибір найбільш ефективних методів, що враховують специфічні обмеження проєкту, а також проведено порівняльний аналіз інструментів для тестування швидкості, таких як PageSpeed Insights, Google Lighthouse та GTmetrix. Розглянуті інструменти дозволяють визначати вузькі місця в продуктивності та забезпечують відстеження прогресу в процесі оптимізації.

Практична цінність результатів полягає в можливості застосування обраних методів та інструментів для підвищення швидкості завантаження веб-сторінок, що забезпечить більш високий рівень зручності користування та підвищення позицій в пошукових системах. Це особливо важливо для комерційних проєктів, де кожна секунда затримки може впливати на утримання користувачів і загальні доходи компанії.

В рамках дослідження було досліджено сучасні методи оптимізації швидкості завантаження веб-сторінок. Було розглянуто практичні підходи до оптимізації, які використовуються в сучасній веб-розробці. Зокрема, досліджено такі методи, як кешування, стиснення зображень, мінімізація CSS та JavaScript, використання мереж доставки контенту (CDN), відкладене завантаження (lazy loading) та інші техніки, які дозволяють підвищити швидкість завантаження. Комбінація цих підходів забезпечує значне скорочення часу завантаження та позитивно впливає на користувацький досвід.

Було проведено огляд популярних інструментів для аналізу швидкості завантаження. Було розглянуто інструменти для вимірювання швидкості заван-

таження, зокрема Google PageSpeed Insights, Lighthouse та GTmetrix. Їх використання допомагає отримати чіткі й репрезентативні дані для виявлення "вузьких місць" у процесі завантаження веб-сторінок. Інструменти, зокрема, дозволяють оцінити такі метрики, як First Contentful Paint (FCP), Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS) та інші, які є критичними для вимірювання швидкості й якості завантаження.

Було створено контрольоване середовище та проведено порівняння ефективності обраних методів оптимізації на базі тестового веб-сайту. Експериментальне дослідження відбувалось із поступовим застосуванням методів оптимізації та фіксацією ключових метрик продуктивності на кожному етапі. Завдяки цьому було отримано відтворювані результати, які дозволяють об’єктивно оцінити вплив кожного методу.

На основі отриманих результатів було сформульовано рекомендації для використання методів оптимізації залежно від специфіки веб-проєктів. Зокрема, підходи, спрямовані на зменшення обсягу даних і часу завантаження, рекомендовано використовувати для веб-ресурсів із високим навантаженням або глобальною аудиторією.

Виявлено критичні фактори впливу на швидкість завантаження вебсторінок. Дослідження показало, що найбільший ефект оптимізацій спостерігається при зменшенні кількості HTTP-запитів, використанні сучасних форматів ресурсів (наприклад, WebP для зображень) і правильному налаштуванні кешування.

Таким чином, результати дослідження підтверджують, що впровадження сучасних методів оптимізації дозволяє суттєво покращити швидкість завантаження веб-сторінок, забезпечуючи високу якість користувацького досвіду та підвищуючи конкурентоспроможність веб-ресурсів.

Проведене дослідження показало, що систематичне застосування сучасних технологій дозволяє суттєво покращити продуктивність веб-ресурсів, підвищити якість користувацького досвіду та забезпечити конкурентоспроможність у цифровому середовищі. У рамках експерименту було використано ком-

плексний підхід, який включав аналіз початкових метрик, реалізацію оптимізаційних заходів та оцінку отриманих результатів.

В рамках роботи було застосовано сім сучасних методів оптимізації, кожен із яких зробив суттєвий внесок у загальне покращення продуктивності веб-сайту.

Результати експерименту підтвердили, що застосування сучасних методів оптимізації є необхідним етапом для створення швидких, стабільних і конкурентоспроможних веб-ресурсів. Найбільший ефект досягається шляхом комплексного підходу, коли різні методи доповнюють один одного, дозволяючи врахувати специфіку проєкту та забезпечити максимальну ефективність.

Успішне впровадження оптимізаційних методів вимагає чіткого розуміння цільової аудиторії, технічних обмежень і характеру контенту веб-сайту.

Для досягнення найкращих результатів слід використовувати сучасні інструменти моніторингу продуктивності, такі як Google Lighthouse та PageSpeed Insights.

Вибір оптимізаційних методів має враховувати не лише технічні аспекти, але й бізнес-потреби, зокрема витрати на впровадження та підтримку оптимізацій.

Отримані результати можуть бути впроваджені у проєктах різного масштабу: від невеликих корпоративних сайтів до великих веб-додатків. Зокрема, кешування та CDN підходять для високонавантажених систем, lazy loading оптимальний для динамічних сторінок із великою кількістю мультимедійного контенту, а мініфікація та об’єднання ресурсів – для сайтів із великою кількістю скриптів та стилів.

Результати роботи сприяють покращенню користувацького досвіду, що безпосередньо впливає на утримання відвідувачів та зменшення кількості відмов. Економія ресурсів на серверній стороні та зниження обсягу переданих даних зменшують витрати як для власників веб-ресурсів, так і для кінцевих користувачів. Це робить оптимізовані сайти доступнішими, особливо для аудиторій з повільним інтернет-з’єднанням.

Подальші дослідження можуть бути зосереджені на автоматизації процесу оптимізації та адаптивних підходах до вибору методів залежно від характеристик проєкту. Наприклад, вивчення використання штучного інтелекту для динамічної оптимізації ресурсів або адаптації веб-контенту до умов кінцевого користувача. Також варто зосередитися на вивченні екологічного впливу оптимізації, зокрема зниження енергоспоживання дата-центрів.

Результати дослідження є вагомим внеском у розвиток методів оптимізації швидкості завантаження веб-сторінок і можуть бути використані як основа для подальших наукових та практичних розробок.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Nahorets S. Evaluation of modern methods for optimizing web page loading speed. Learning & Teaching: after War and during Peace [Electronic Edition]: *Conference Proceedings of III International Scientific & Practical Conference*, Kharkiv, Ukraine, 8 November, 2024. Kharkiv : KNPU, 2024. P. 232.

2. Tuning Your Website for Modern Page Speed Metrics // PaperStreet. 2020. URL: https://www.paperstreet.com/blog/tuning-your-website-for-modern-page-speed-metrics/ (дата звернення: 20.10.2024).

3. 12 Top Website Performance Tools for Site Owners // Semrush. 2024. URL: https://www.semrush.com/blog/website-performance-tools/ (дата звернення: 02.11.2024).

4. Швидкість завантаження сайту: від чого залежить? // Pingvin.Pro. 2024. URL: https://pingvin.pro/blogy/software-blogy/shvidkist-zavantazhennya-saitu-vid-chogo-zalezhit.html (дата звернення: 23.10.2024).

5. Website Speed Optimization: 14 Tips to Improve Performance // Sematext. 2021. URL: https://sematext.com/blog/improve-website-performance/ (дата звернення: 23.10.2024).

6. Веб-продуктивність: як пришвидшити завантаження сайту та покращити SEO // WhileWeb. 2023. URL: https://whileweb.com/uk/blog/veb-produktivnist-yak-prishvidshiti-zavantazhennya-sajtu-ta-pokrashiti-seo/ (дата звернення: 23.10.2024).

7. Швидкість сайту: чому вона така важлива і як її підвищити // Ranktracker. 2023. URL: https://www.ranktracker.com/uk/blog/website-speed-why-its-so-important-how-to-improve-it/ (дата звернення: 24.10.2024).

8. PageSpeed Insights Best Practices: Acing Google’s Assessment // Toptal. 2024. URL: https://www.toptal.com/site-speed-optimization/pagespeed-insights-best-practices (дата звернення: 24.10.2024).

9. Як покращити швидкість завантаження сайту: ефективні методи та рекомендації // SEO Evolution. 2024. URL: https://seo-evolution.com.ua/blog/seo-

prodvizhenie/yak-pokrashiti-shvidkist-sajtu-poradi-ta-rekomendaciyi (дата звернення: 24.10.2024).

10. Page Speed: How to evaluate and improve page speed // Moz. 2024. URL: https://moz.com/learn/seo/page-speed (дата звернення: 28.10.2024).

11. Оптимізація сайту – Що варто знати про швидкість завантаження сайту? // Golden-Web Digital. 2024. URL: https://golden-web.digital/blog/rubrika-seo/shvidkist-zavantazhennia-saitu-vse-pro-golovni-metriki-sposobi-vidstezhennia-danikh-i-chervnevii-apdeit-vid-google/ (дата звернення: 28.10.2024).

12. Швидкість завантаження сайту: як перевірити, від чого залежить та як покращити // WEDEX. 2024. URL: https://wedex.com.ua/blog/shvydkist-zavantazhennya-sajtu-yak-pereviryty-vid-chogo-zalezhyt-ta-yak-pokrashhyty/ (дата звернення: 28.10.2024).

13. Оптимізація продуктивності сайту: Методи та інструменти для підвищення швидкості // Stfalcon. 2023. URL: https://stfalcon.com/uk/blog/post/Web-Performance-Optimization-Techniques-and-Tools-to-Improve-Website-Speed (дата звернення: 28.10.2024).

14. 19 Website Speed Optimization Strategies for 2024 [New Data] // HubSpot Blog. 2023. URL: https://blog.hubspot.com/website/how-to-optimize-website-speed (дата звернення: 02.11.2024).

15. Розробка веб-дизайну, оптимізованого до розробки веб-сайту з високою продуктивністю // COI. – 2024. – URL: https://coi.ua/blog/DesignCo/load-optimization-fast-web-design-for-maximum-productivity/ (дата звернення: 02.11.2024).

16. Google PageSpeed Insights Guide – 21 Ways to Improve Your Score // SiteGround. 2024. URL: https://www.siteground.com/kb/google-pagespeed-insights-guide/ (дата звернення: 02.11.2024).

17. Ліваковський В.К., Хмелівський Ю.С. Оптимізація швидкості завантаження веб-сторінок: стратегії та інструменти. *Журнал інформаційних технологій*. 2023. № 1. С. 45–52.