Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Оптимізація продуктивності та SEO при масштабуванні проєктів на основі React і Next.js

Виконав: студент VI курсу, групи СНм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Старицький О.Т. (прізвище та ініціали)

Керівник (підпис) Дмитроца Л.П. (прізвище та ініціали)

Нормоконтроль (підпис) Никитюк В.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Осухівська Г.М. (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _передень захисту 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Старицькому О.Т.
(прізвище, ім'я, по батькові)

1. Тема роботи Оптимізація продуктивності та SEO при масштабуванні проєктів на основі React і Next.js

Керівник роботи Дмитроца Л.П., к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » листопада 2023 року № 4/7-1100

2. Термін подання студентом завершеної роботи 27 травня 2025р.

3. Вихідні дані до роботи наукова література з оптимізації та SEO додатків на основі JS-фреймворків, архітектура та методи побудови сучасних вебдодатків з використанням React та Next.js, розроблені програмні засоби для оптимізації додатків, документація та коди React та Next.js, перелік основних показників та метрик ефективності оптимізації, інструменти вимірювання метрик продуктивності та SEO, React Developer Tools, інструменти Google Analytics.

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз методів оптимізації та SEO додатків на основі JS-фреймворків
2. Архітектура та методи побудови сучасних вебдодатків з використанням React та Next.js
3 Дослідити та вибрати програмні засоби для оптимізації додатків
4 Проаналізувати та вибрати метрики вимірювання продуктивності
5. Розробити тестові додатки з використанням React та Next.js
6. Провести експериментальне дослідження ефективності програмних засобів для оптимізації

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Демонстраційний матеріал

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 24 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	04.12.2024	*Виконано*
2.	Підбір наукових джерел по темі оптимізація продуктивності та SEO при масштабуванні проєктів на основі React і Next.js	15.12.2025-31.12.2025	*Виконано*
3.	Опрацювання наукових публікацій та збір даних по темі роботи	15.01.2025-25.02.2025	*Виконано*
4.	Виконання дослідження згідно мети кваліфікаційної роботи	26.02.2025-07.04.2025	*Виконано*
5.	Оформлення розділу «АНАЛІЗ АРХІТЕКТУРИ ТА МЕТОДІВ ОПТИМІЗАЦІЇ ВЕБПРОЄКТІВ»	15.04.2025-18.04.2025	*Виконано*
6..	Оформлення розділу «МЕТОДИ ПОБУДОВИ ТА ІНСТРУМЕНТИ ОПТИМІЗАЦІЇ СУЧАСНИХ ВЕБДОДАТКІВ»	19.04.2025-25.04.2025	*Виконано*
7.	Оформлення розділу «ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ТА SEO»	26.04.2025-02.05.2025	*Виконано*
	...		
8.	Виконання завдання до підрозділу «Охорона праці»	03.05.2025-07.05.2025	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в		
	надзвичайних ситуаціях»	08.05.2025-10.05.2025	*Виконано*
10.	Оформлення кваліфікаційної роботи	11.05.2025-14.05.2025	*Виконано*
11.	Нормоконтроль	15.05.2025-16.05.2025	*Виконано*
12.	Перевірка на плагіат	17.05.2025	*Виконано*
13.	Попередній захист кваліфікаційної роботи	21.05.2025	*Виконано*
14.	Захист кваліфікаційної роботи	27.05.2025	*Виконано*

Студент _____ *Старицький О.Т.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Дмитроца Л.П.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Тема: Оптимізація продуктивності та SEO при масштабуванні проєктів на основі React і Next.js // Кваліфікаційна робота освітнього рівня «Магістр» // Старицький Олександр Тарасович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2025 // С. 95, рис. – 27, табл. – 1, лістингів коду – 8, додат. – 2, бібліогр. – 51.

Ключові слова: веброзробка, JavaScript фреймворки, продуктивність, React.js, Next.js, оптимізація, SEO.

У роботі проведено аналіз методів оптимізації продуктивності та SEO для додатків на основі JS-фреймворків. Досліджено архітектуру та методи побудови сучасних вебдодатків з використанням React та Next.js. На основі порівняльного аналізу програмного забезпечення для оптимізації додатків та побудов стратегії проаналізовано та обрано релевантні метрики вимірювання продуктивності. З метою порівняння ефективності методів оптимізації продуктивності та SEO розроблено та побудовано тестові програми з використанням React та Next.js. Проведено експериментальне дослідження ефективності програмних засобів для оптимізації продуктивності та SEO при масштабуванні додатків.

ANNOTATION

Subject: Optimizing performance and SEO when scaling projects based on React and Next.js // The educational level "Master" qualification work // Starytskyi Oleksandr // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2024 // P. 95, fig. - 27, tab. – 1, annexes - 2, ref. - 51.

Key words: web development, JavaScript Frameworks, performance, React.js, Next.js, optimization, SEO.

Thesis is devoted to the analysis of productivity and SEO optimization methods for applications based on JS frameworks. The architecture and methods of building modern web applications using React and Next.js are studied. Based on a comparative analysis of application optimization software and strategy building, relevant performance measurement metrics are analyzed and selected. In order to compare the effectiveness of productivity and SEO optimization methods, test applications are developed and built using React and Next.js. An experimental study of the effectiveness of software tools for optimizing productivity and SEO when scaling applications is made.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MVC (англ. Model-View-Controller) – архітектурний шаблон «моде́ль–вигляд–контролер».

SEO (Search Engine Optimization) – пошукова оптимізація сайту.

UI (англ. User Iinterface) – інтерфейс користувача.

SPA (англ. Single Page Application) – односторінковий додаток.

CSR (англ. Client-Side Rendering) – рендеринг на стороні клієнта.

SSR (англ. Server-Side Rendering) – рендеринг на стороні сервера.

SSG (англ. Static Site Generator) – генератор статичних сайтів.

DOM (англ. Document Object Model) – програмний інтерфейс для вебдокументів, об'єктна модель документа.

SERP (англ. Search engine results page) – вебсторінка, що генерується пошуковою системою у відповідь на пошуковий запит користувача.

JSX (JavaScript XML) – розширення синтаксису для JavaScript, яке дозволяє писати розмітку схожу на HTML, всередині JS-файлу.

CRA (Create React App) – інтерфейс командного рядка, що дозволяє створювати попередньо налаштовані React-проєкти коефіцієнт помилково-негативної ідентифікації.

CLI (англ. Command Line Interface,) – інтерфейс командного рядка.

API (англ. Application Programming interface) – інтерфейс програмування застосунків.

CTR (англ. Click-through rate) — рейтинг кліків або клікабельність, відношення числа кліків на оголошення до числа його показів.

CWV (Core Web Vitals) — це набір показників завантаження вебсторінок сайту.

LCP (Largest Contentful Paint) — момент відтворення найбільшого контенту.

INP (Interaction to Next Paint)[1], тобто відгук інтерфейсу на взаємодію.

CLS (Cumulative Layout Shift) — загальний зсув макета.

ЗМІСТ

ВСТУП.....8
1. АНАЛІЗ АРХІТЕКТУРИ ТА МЕТОДІВ ОПТИМІЗАЦІЇ ВЕБПРОЄКТІВ…...10
1.1 Архітектура сучасних вебдодатків.....10
1.1.1 Принципи побудови масштабованих вебзастосунків.....11
1.1.2 Архітектурні патерни.....12
1.2 Архітектурні моделі у веброзробці та їх вплив на продуктивність.....13
1.2.1 Патерн MVC у розробці вебзастосунків.....14
1.2.2 Односторінкові додатки (SPA): переваги та обмеження.....17
1.2.3 Робочий процес SPA.....18
1.3 Сучасні технології у сучасних вебпроєктах.....20
1.3.1 Особливості використання React для розробки продуктивних SPA.....21
1.3.2 Next.js як рішення для серверного рендерингу та статичної генерації.....22
1.4 Оптимізація продуктивності вебзастосунків.....24
1.4.1 Основні метрики продуктивності вебзастосунків.....25
1.4.2 Напрямки оптимізації продуктивності вебзастосунків.....26
1.5 Сучасні підходи до пошукової оптимізації у динамічних вебдодатках.....28
1.5.1 Індексація та ефективні рішення SEO-оптимізації SPA-додатків.....28
1.5.2 Технічні аспекти SEO.....30
1.6 Висновок до першого розділу.....32
2. МЕТОДИ ПОБУДОВИ ТА ІНСТУМЕНТИ ВЕБДОДАТКІВ.....34
2.1 JavaScript у високопродуктивній веброзробці.....34
2.1.1 Структурні особливості JavaScript……..34
2.1.2 Огляд сучасних бібліотек та фреймворків.....38
2.2 Бібліотека React та оптимізація.....40
2.2.1 Структура і компоненти.....40
2.2.2 Оптимізація у проєктах React……45
2.3 Фреймворк Next.js як інструмент оптимізації.....48
2.3.1 Організація та побудова проєктів у Next.js……..48
2.3.2 Практики SEO у Next.js……..52

2.4 Інструменти аналізу ефективності вебпроєктів.....54
2.4.1 Моніторинг та аналіз SEO-показників.....55
2.4.2 Технічні аспекти SEO для Next.js.....57
2.5 Метрики оцінки продуктивності та SEO.....59
2.5.1 Ключові показники продуктивності.....59
2.5.2 Метрики індексації та клікабельності.....61
2.6 Висновок до другого розділу.....63
3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ТА SEO 65
3.1 Постановка задачі та методологія експерименту 65
3.1.1 Постановка мети й завдань 66
3.1.2 Гіпотеза, змінні та індикатори ефективності 67
3.1.3 Вибір інструментів і технічного середовища 69
3.2 Базова конфігурація вебпроєкту для оптимізації 73
3.3 Впровадження оптимізації продуктивності 76
3.4 SEO-оптимізація вебдодатку 79
3.5 Аналіз результатів оптимізації 81
3.6 Висновок до третього розділу 88
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ.....91
4.1. Фактори, що впливають на функціональний стан користувачів комп’ютерів.....92
4.2 Питання щодо охорони праці.....95
4.3 Питання щодо безпеки в надзвичайних ситуаціях.....99
4.4 Висновок до четвертого розділу.....104
ВИСНОВКИ.....108
Перелік джерел.....111
ДОДАТКИ

ВСТУП

Актуальність теми. Фреймворки JavaScript були створені для того, щоб робота вебпрограмістів була більш простою та плідною, наприклад, на основі надання готових бібліотек, активної підтримки спільноти та надання розробникам вигоди від повторного використання компонентів. Технології розробки повинні відповідати вимогам сучасних додатків та забезпечувати високий рівень коду та продуктивності. Використання фреймворку Next.js, створеного на основі бібліотеки React.js дозволяє створювати сучасні масштабовані вебзастосунки з високою продуктивністю і SEO. Для оптимізації продуктивності та SEO додатків на основі React.js та Next.js постійно розробляються нові методи та підходи. Тому тема покращення сучасних методів оптимізації продуктивності є актуальним напрямом сучасних досліджень.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є удосконалення методів та алгоритмів оптимізації продуктивності та SEO в сучасних вебдодатках, створених на основі React.js та Next.js. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати дослідження в області створення сучасних вебдодатків з урахуванням їхньої архітектури та методів сучасної організації проєктів на основі фреймворків JavaScript, таких як React.js та Next.js.

– Дослідити існуючі на даний час методи оптимізації продуктивності та SEO в сучасних вебдодатках.

– Проаналізувати методи програмної імплементації оптимізації продуктивності та SEO.

– Виконати порівняння існуючих технологій веб проєктів, створених на основі фреймворків JavaScript, та їх методів оптимізації.

– Удосконалити його методи оптимізації для проєктів React.js та Next.js .

Об’єктом дослідження є процеси оптимізації продуктивності та SEO в сучасних вебдодатках на основі React.js та Next.js.

Предмет дослідження - методи оптимізації продуктивності у масштабованих вебдодатках з використанням інноваційних підходів до оптимізації продуктивності.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у тому, що на основі проведення порівняння оптимізації продуктивності для тестових проєктів, створених на основі React.js та Next.js, були удосконалені методи збільшення продуктивності, які є перспективними для великих проєктів.

Практичне значення одержаних результатів. Виконано макетування та прототипування сучасних вебдодатків, заснованих на ефективних алгоритмах оптимізації та SEO.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на XIII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 11-12 грудня 2024 року, XII Науково-технічній конференції «ІМСТ» – Тернопіль, 18–19 грудня 2024 року.

Публікації. Основні результати кваліфікаційної роботи опубліковано у працях конференції (Див. додатки А).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 53 найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає 109 сторінки, з них 87 сторінки основного тексту, який містить 27 рисунків.

1 АНАЛІЗ АРХІТЕКТУРИ ТА МЕТОДІВ ОПТИМІЗАЦІЇ ВЕБПРОЄКТІВ

У сучасному світі весь обмін інформацією та транзакції здійснюються за допомогою різних додатків та вебсервісів, необхідно, щоб кожен користувач мав знання про комп'ютер та його додатки, що надаються разом із ним. За ці роки швидке вдосконалення апаратних та програмних технологій зробило комп'ютерні пристрої, мобільні телефони, ноутбуки та інші цифрові технології зручнішими для користувача та доступними. Оскільки дані, інформація та підключення швидко зростають з кожним днем, вебсайти стикаються з проблемами довговічності, продуктивності, безпеки та обслуговування. Тому було створено багато технологій, нових архітектур вебдодатків та інструментів, які спеціально призначені для підтримки цих цілей.

У цьому розділі викладено основні принципи перегляду вебдодатків, розглянуто особливості та переваги підходу MVC (Model-View-Controller). Далі викладено переваги використання односторінкових додатків порівняно з багатосторінковими; розглядаються їх технологічні особливості. Після цього проведено огляд сучасних методів спрямованих на покращення продуктивності сучасних вебдодатків, аналізуючи погляди до оптимізації як з боку загального підходу, так і із застосуванням методів пошукової оптимізації (SEO, search engine optimization).

1.1 Архітектура сучасних вебдодатків

1989 року в ЦЕРНі (науково-дослідна лабораторія неподалік Женеви, Швейцарія) Тім Бернерс-Лі представив пропозицію про систему управління інформацією, яка б дозволила обмінюватися знаннями та ресурсами через комп'ютерну мережу [1]. Запропонована ним система поширилася в те, що справді можна назвати Всесвітньою павутиною, оскільки люди в усьому світі використовують її для найрізноманітніших цілей. Тім Бернерс-Лі спочатку просував Всесвітню павутину (The WorldWideWeb) як віртуальну бібліотеку,

систему контролю документів для обміну інформаційними ресурсами між дослідниками. Доступ до онлайн-документів можна було отримати за унікальною адресою документа, універсальним локатором ресурсів (URL). На ці документи можна було посилатися за допомогою гіпертекстових посилань. Однак здебільшого інформаційні ресурси, що поширювалися в Інтернеті, були статичними документами. Динамічні інформаційні сервіси – від пошукових систем до CGI-скриптів і пакетів, що з'єднували Інтернет з реляційними базами даних – все це змінили. З появою динамічного вебу планку підняли ще вище. Вже недостатньо було сказати, що розробляється «вебсайт» (на відміну від строкатої колекції «вебсторінок»). Стало необхідним розробляти вебзастосунки.

1.1.1 Принципи побудови масштабованих вебзастосунків

Не існує універсального визначення архітектури вебзастосунку, але найчастіше її описують як модель взаємодії між компонентами застосунку [2]. Іншими словами, існує скелет застосунку, який визначає, як і де будуть розміщені всі елементи, а також яка основна логіка взаємодії між клієнтом і сервером. Види вебзастосунків можна класифікувати відповідно до вимог і завдань, які вони вирішують. Для цього існує кілька причин, з яких важливо приділити особливу увагу проєктуванню архітектури вебдодатку:

1. Ефективна архітектура означає, що додаток буде менш схильним до помилок і проблем, добре справлятиметься з навантаженням і ефективно виконуватиме свої функції без будь-яких затримок або збоїв. Крім того, масштабована архітектура дозволяє розширювати функціональність додатка в майбутньому, додаючи нові актуальні функції та адаптуючись до змінних потреб ринку.

2. Завдяки масштабованій та ефективній архітектурі додатка, впровадження змін або додавання покращень не займає забагато часу та ресурсів. Крім того, високопродуктивний додаток має вищі рейтинги в

пошукових системах, а це означає, що він принесе більше трафіку, а отже, більше потенційних клієнтів.

3. Коли користувач взаємодіє з цифровим продуктом, йому потрібно лічені секунди, щоб сформувати враження, і навіть найменша проблема може його зіпсувати. Завдяки безперебійно функціонуючому застосунку, який однаково добре працює за різних умов, користувацький досвід залишатиметься на тому ж високому рівні.

1.1.2 Архітектурні патерни

Зазвичай можна виділити три основні сторони архітектури застосунків [3].

- Клієнтська частина: також відома як фронтенд, ця частина архітектури відповідає за взаємодію користувача з додатком.
- Сервер бази даних: ця частина надсилає дані клієнта на сервер.
- Серверна частина: також відома як бекенд-сховище, ця частина відповідає за зберігання даних, обробку запитів від сервера бази даних та надсилання відповідей назад.

Різні типи архітектури вебдодатків призначені для виконання конкретних завдань, тому їх компоненти можуть взаємодіяти по-різному. Виходячи з цього, важко класифікувати компоненти, що виникають. Тим не менш, прийнято вважати, що звичайний додаток має два основні типи своїх компонентів – компоненти інтерфейсу користувача (UI, user interface) та структурні. Вважають, що компоненти UI програми не надають суттєвого впливу на продуктивність програми та не прив'язані до архітектури. Тим не менш, ці компоненти відносяться до досвіду користувача і є частиною візуального інтерфейсу. Вони включають сповіщення, журнали, параметри конфігурації, панелі моніторингу тощо. З іншого боку, структурні компоненти пов'язані з функціональністю додатку, їх можна умовно розділити на дві групи. По-перше, це клієнтські компоненти, які є фронтенд-частина додатків; їх користувач

бачить у браузері та безпосередньо взаємодіє. Ці компоненти створюються за допомогою JavaScript, CSS та HTML та не потребують підключення пристрою для роботи. По-друге, компоненти сервера, які є внутрішніми частинами програми; вони розроблені з використанням таких технологій як Java, .NET, NodeJS, Python або PHP. Треба звернути увагу на те, що компоненти сервера складають базу даних і загалом відповідають за те, як дані зберігаються, обробляються та вилучаються.

Архітектурні компоненти самі по собі не мають значення доти, доки вони не організовані в проєкт на основі певного принципу. Протягом останніх кількох років вебсайти перейшли від простих HTML-сторінок з невеликою кількістю CSS до неймовірно складних додатків, над якими одночасно працюють тисячі розробників. Для роботи з цими складними вебдодатками розробники використовують різні шаблони проєктування для компонування своїх проєктів, щоб зробити код менш складним та легшим у роботі. Найпопулярнішим із цих шаблонів є MVC, також відомий як Model View Controller. Концепція MVC – це більш традиційний підхід до створення вебзастосунків, який значною мірою зосереджений на концепції ООП (об'єктно-орієнтованого програмування) [4]. Наприклад, Next.js, який побудовано на базі React, разом з іншими фреймворками, такими як Angular та Vue, змінив парадигму на більш функціональний підхід на основі використання концепції компонентів MVC для створення вебзастосунків. У компоненті представлення містить всю логіку (контролер) та дані (модель), необхідні для візуалізації в межах цього єдиного компонента.

1.2 Архітектурні моделі у веброзробці та їх вплив на продуктивність

Архітектуру MVC можна використовувати для розробки масштабованої, прозорої та портативної вебпрограми. Архітектура MVC навчає паралельній розробці, оскільки вона розділяє логіку програми на три рівні, дозволяючи кільком розробникам працювати над однією вебпрограмою одночасно.

Налаштування архітектури MVC зазвичай є дорогим. Невеликі MVC-проєкти та проєкти, що не пов'язані з MVC, часто займають однакову кількість часу для обробки, проте великі MVC-проєкти зазвичай обробляються швидше, ніж проєкти, що не пов'язані з MVC.

1.2.1 Патерн MVC у розробці вебзастосунків

MVC – це дизайн програмного забезпечення, який зазвичай використовується для створення користувацьких інтерфейсів, даних та логіки керування. Він зосереджений на розмежуванні бізнес-логіки та зовнішнього вигляду програмного забезпечення. Таке «розділення обов'язків» забезпечує краще використання робочої сили та обслуговування. Інші шаблони проєктування на основі MVC включають MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter) та MVW (Model-View-Web) (Model-View-Whatever).

Шаблон проєктування програмного забезпечення MVC можна розділити на три частини:

- Дані та бізнес-логіка обробляються самою моделлю.
- Макет та представлення обробляються View.
- Controller керує тим, як команди спрямовуються до моделі та елементів View.

Яку інформацію слід надавати в додатку, визначає модель. Модель зазвичай вказує на основний вигляд (щоб відображення можна було змінювати за потреби) та контролеру, коли змінюється статус цих даних (якщо потрібна окрема логіка для керування оновленим виглядом). Наприклад, модель визначатиме, яку інформацію слід включати до елементів списку, таку як термін дії, ціна тощо, а також які елементи списку наразі присутні.

Представлення визначає, як мають відображатися дані в додатку. Наприклад, як користувач бачить списки і отримує дані від моделі для представлення.

У відповідь на умови введення користувачем, контролер реалізує логіку, яка визначатиме вигляд моделі. Наприклад, поля введення та кнопки для додавання та видалення елементів можуть бути присутніми в списку покупок. Оскільки ці операції передбачають модифікацію моделі, дані передаються до контролера, який, у свою чергу, модифікує модель за потреби, перш ніж надсилати змінені дані до представлення (View). Щоб змінити спосіб відображення даних, треба просто відредагувати представлення. Наприклад, змінення порядок товарів на алфавітний або від найнижчої до найвищої ціни. Контролер міг би впоратися з цим у цій ситуації без необхідності оновлення моделі. На рисунку 1.1 подано архітектуру MVC та показано зв'язки між структурними елементами.

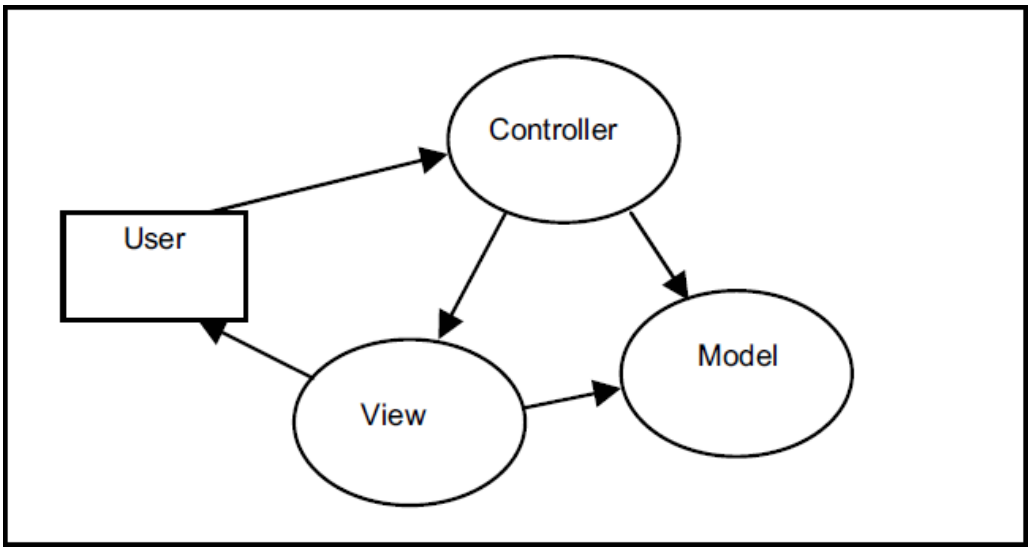

Рисунок 1.1 – MVC архітектура

Модель даних зберігається в таблиці змісту (чи традиційна серверна база даних, наприклад MySQL, або клієнтське рішення, наприклад IndexedDB). Керуючий код програми написано на html/js, а інтерфейс користувача на html/css,. Це схоже на MVC, але MVC вимагає, щоб ці компоненти дотримувалися суворішої структури.

Спочатку браузер надсилає контролеру команду запиту. Щоб доставити та отримати дані, контролер взаємодіє з моделлю. Контролер взаємодіє з

представленням (View) для візуалізації даних. Єдине завдання представлення (View) полягає в тому, щоб знати, як інформація доставляється, а не як вона представлена в кінцевому підсумку. Це буде динамічний HTML-файл, який візуалізує дані на основі вхідних даних контролера. Нарешті, View оброблятиме все своє остаточне представлення для контролера, яке контролер представить користувачеві. Важливо зазначити, що View та модель ніколи не взаємодіють. Вони можуть взаємодіяти один з одним лише через контролер. Це стосується як логіки програми, так і інтерфейсу користувача [5].

Далі розглянуто, що відбувається на простому прикладі пошуку фільмів (рис. 1.2).

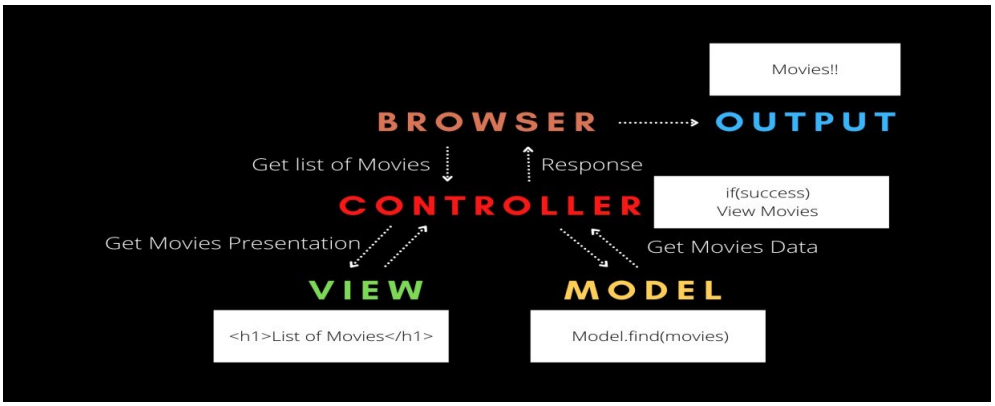

Рисунок 1.2 – Структура MVC для пошуку через веббраузер архітектура

Спочатку користувач вводить інформацію про список фільмів через веббраузер або мобільний додаток. Браузер повинен надіслати запит до контролера, щоб отримати список фільмів. Далі контролер запитує у моделі знайти список фільмів з основної бази даних. Потім модель шукає цю базу даних і надає контролеру список фільмів. Якщо контролер отримує список фільмів від моделі відповідно до запиту, він наказує представленню відобразити цей список. Представлення отримує запит і повертає контролеру відображений список фільмів у форматі HTML. Нарешті, контролер бере цей HTML-код і надає його користувачеві, що призводить до виведення списку фільмів [6].

1.2.2 Односторінкові додатки (SPA): переваги та обмеження

Наразі відомо, що найкращим підходом до розробки вебдодатків є односторінкові застосунки (SPA, single-page application) [4]. SPA переносять до браузера прийоми роботи, які звичні для персональних додатків.

Односторінковий додаток (SPA) – це тип вебдодатку, який динамічно завантажує та оновлює контент без оновлення всієї сторінки [7]. На відміну від традиційних вебсайтів, SPA використовують сучасні технології для покращення взаємодії з користувачем, мінімізуючи переривання та забезпечуючи більш плавний інтерфейс. Користувачі можуть безперешкодно взаємодіяти з додатком, подібно до використання програмного забезпечення для настільних комп'ютерів. Головною перевагою є усунення перезавантажень повної сторінки, що призводить до більш адаптивного та захопливого вебдосвіду. Це досягається шляхом забезпечення того, щоб браузер отримував усі необхідні коди HTML, JavaScript та CSS за один запит або оновлював необхідний контент на основі дій користувача. Коли натискається на щось у SPA, воно надсилає лише необхідну інформацію до вашого браузера, а браузер її відтворює, як показано на рисунку 1.3.

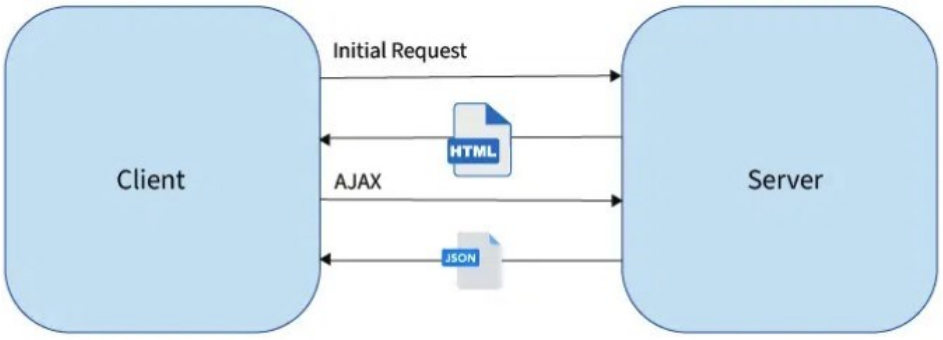

Рисунок 1.3 –Життєвий цикл сторінки SPA

Це відрізняється від традиційного завантаження сторінки, де сервер надсилає повну сторінку до браузера з кожним кліком (рис. 1.4).

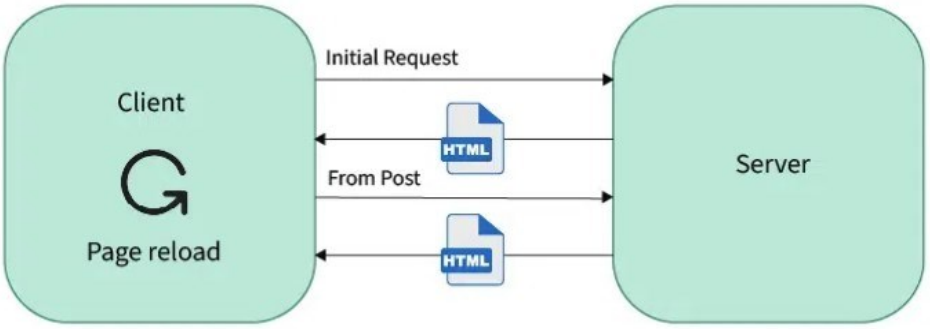

Рисунок 1.4 – Традиційний життєвий цикл сторінки

Переваги та доброзичлива для користувачів логіка SPA зумовила широкий спектр можливих застосувань. Так, для динамічної взаємодії SPA вибирається для додатків, що потребують частої взаємодії з користувачем без перезавантаження сторінки. Для оновлення в режимі реального часу SPA використовується, коли для безперебійного досвіду користувача критично важлива швидкість реагування в реальному часі. Використання SPA для інтерактивних інтерфейсів з такими функціями, як анімація та динамічний контент, дозволяє будувати багаті інтерфейси користувача. До того ж цей підхід забезпечує мінімальне навантаження у тому випадку, коли пріоритетом є зниження навантаження на сервер та використання смуги пропускання. Очевидно, що SPA є кращим для контенту, логічно представленого на одній сторінці, уникаючи необхідності в декількох сторінках. Але особливо важлива кросплатформова узгодженість SPA для підтримки однакового досвіду користувача на різних пристроях і платформах.

1.2.3 Робочий процес SPA

У SPA немає необхідності перегортати сторінки, щоб перейти до різних розділів. Отже, SPA функціонують в Інтернеті, плавно завантажуючи та оновлюючи контент, дозволяючи користувачам досліджувати цифровий світ без будь-яких затримок, які виникають на традиційних вебсайтах. Він включає три типи рендерингу (рис. 1.5):

1. Рендеринг на стороні клієнта (CSR, Client-Side Rendering). Спочатку браузер запитує HTML із сервера. Потім сервер швидко відповідає базовим HTML-файлом та пов'язаними стилями/скриптами. Після того, як виконується JavaScript, користувач бачить порожню сторінку або завантажувач. Додаток отримує дані, створює представлення (View в MVC) і впроваджує їх у DOM (Document Object Model). Слід зазначити, що CSR може бути повільнішим для простих вебсайтів, оскільки він використовує багато ресурсів пристрою. Проте він хороший для завантажених вебсайтів, прискорюючи роботу користувачів. З іншого боку, якщо потрібні різні варіанти соціального обміну, необхідно використовувати інші види рендерингу.

.

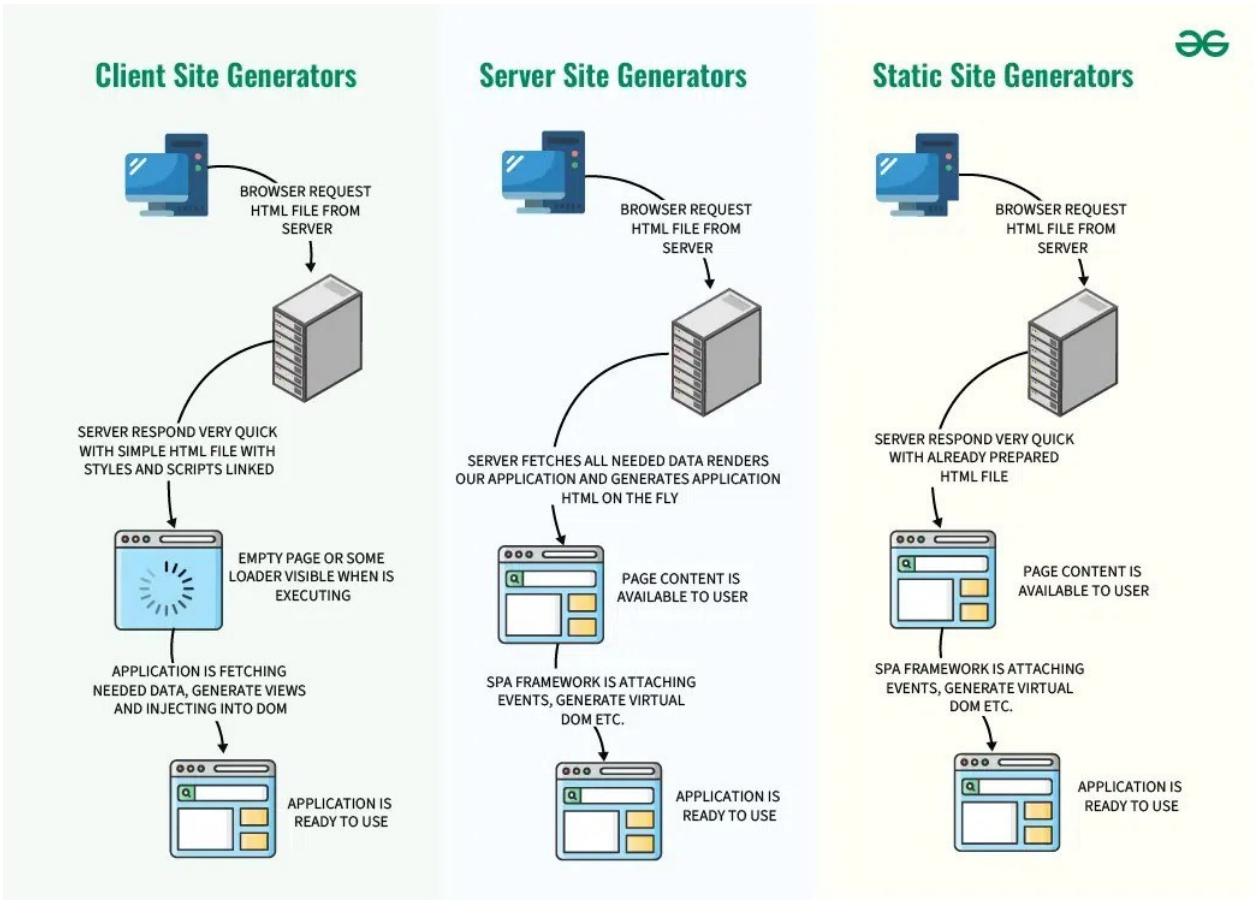

Рисунок 1.5 – Три види SPA рендерингу [7]

2. Рендеринг на стороні сервера (SSR, Server-Side Rendering). Браузер запитує файл HTML на сервері. Тепер сервер збирає необхідні дані, створює SPA і миттєво створює файл HTML. Користувач бачить готовий до

використання контент. Структура SPA додає події, створює віртуальний DOM і підтримує все до використання. Система вибирає SSR для досягнення швидкого досвіду роботи додатків, балансуючи швидкість односторонніх додатків, не завантажуючи браузер користувача, забезпечуючи оптимальну продуктивність додатків.

3. Генератор статичних сайтів (SSG, Static Site Generator). Браузери вимагають HTML, SSG швидко надають готові статичні сторінки. Сервер показує користувачам статичну сторінку для швидкого завантаження. SPA на сторінці отримують дані та вносять динамічні зміни на сторінку. SPA готовий до плавної взаємодії з користувачем після додавання даних. SSG відмінно підходять для швидких статичних сторінок, але можуть бути ідеальними для високодинамічних вебсайтів. Хоча SSG пропонують швидке та ефективне рішення, важливо відзначити, що вони можуть не ідеально підходити для вебсайтів з динамічним контентом. Їхня сила полягає в статичних сторінках, що відповідають їх назві. Робочий процес SPA слід, у багатьох випадках, слідує звичайним процедурам, прийнятим у цій галузі. А ось для створення SPA особливу ефективність показали бібліотеки та методи, які спочатку були націлені в цьому напрямку.

1.3 Сучасні технології у сучасних вебпроєктах

Створення SPA включає декілька ключових елементів, які притаманні роботі та над іншими проєктами. Це, перш за все, орієнтована команда, що складається з розробників frontend, backend та UX/UI-дизайнерів. При цьому необхідні глибокі знання у фреймворках JavaScript, таких як React, Angular або Vue.js, залежно від ваших уподобань. Усі етапи розробки базуються на основі JavaScript, CSS та HTML як основні технології розробки SPA. Поряд із використанням для розгортання фреймворків та бібліотек JavaScript (Ajax за потребою), необхідні бекенд-технології, такі як PHP, Node.js, а також бази даних, такі як MongoDB або MySQL.

1.3.1 Особливості використання React для розробки продуктивних SPA

Оскільки метою роботи є дослідження доступних підходів до оптимізації продуктивності в проєктах, створених на основі React та Next.js, то особливу увагу надалі приділено особливостям цієї бібліотеки та фреймворка, розгорнутого на його базисі, для побудови та функціонування SPA.

Бібліотека React не є обов'язковою вимогою для створення SPA [8]. React також можна використовувати для покращення невеликих частин існуючих вебсайтів за допомогою додаткової інтерактивності. Код, написаний на React, може мирно співіснувати з розміткою, що відтворюється на сервері за допомогою чогось на кшталт PHP, або з іншими клієнтськими бібліотеками. Фактично, саме так React використовується у Facebook. React - це бібліотека JS з відкритим вихідним кодом для створення інтерфейсу користувача, що використовується для SPA, і вона управляє представленнями (View в MVC) вебдодатків [9]. React допомагає змінювати дані без перезавантаження сторінки. Це популярна бібліотека на ринку завдяки своїй масштабованості та високій продуктивності. Односторінкові програми відрізняються від багатосторінкових програм, тому що SPA не переміщується на нові сторінки; натомість воно завантажує сторінки в один рядок на тій самій сторінці. Таким чином, якщо говорити про використання React для побудови програми, не можна забувати, що це просто чудова бібліотека JavaScript. Проте принципи побудови проєкту на основі MVC підходу відмінно вписуються у використанні методів React.

З іншого боку, React, ймовірно, є найбільш неконтрольованим варіантом, оскільки це лише бібліотека, тому необхідні функціональні можливості залишаються на розсуд розробника. React не використовує шаблони в стилі HTML, тому не має окремого файлу шаблону HTML. Весь код HTML-документа повністю визначено в JavaScript. React також вводить

використання віртуального DOM. Віртуальний DOM є проміжним місцем перед реальним DOM, де зміни даних вносяться спочатку через швидшу обробку.

Хоча React спрощує створення красивого динамічного контенту, він все ще залишається односторінковим застосунком. Він чудово підходить для динамічних застосунків, але не дуже добре для SEO. Метатегам важко описати кожен відповідний аспект застосунку. З другого боку, односторінкові застосунки не потребують додаткових бібліотек маршрутизації. Щоб встановити експертний рівень та авторитет в Інтернеті, застосунок повинен мати насичений контентом вебсайт із цінними статтями, семантичними ключовими словами та правильно позначеними тегами метаданими.

1.3.2 Next.js як рішення для серверного рендерингу та статичної генерації

Next.js повністю підтримує створення односторінкових застосунків (SPA) [10]. Це включає швидкі переходи між маршрутами з попередньою вибіркою, вибірку даних на стороні клієнта, використання API браузера, інтеграцію зі сторонніми клієнтськими бібліотеками, створення статичних маршрутів тощо. Якщо є існуючий SPA, можливо перейти на Next.js без значних змін у коді. Next.js дозволяє поступово додавати серверні функції за потреби. Next.js може автоматично розділяти код JavaScript-пакетів та генерувати кілька точок входу HTML у різні маршрути. Це дозволяє уникнути завантаження непотрібного JavaScript-коду на стороні клієнта, зменшуючи розмір пакета та забезпечуючи швидше завантаження сторінок. Компонент next/link автоматично попередньо вибирає маршрути, забезпечуючи швидкий перехід між сторінками SPA, але з перевагою збереження стану маршрутизації програми до URL-адреси для посилання та спільного використання. Next.js може починатися як статичний сайт або навіть як суворий SPA, де все відображається на стороні клієнта. Якщо проєкт зростає, Next.js дозволяє поступово додавати більше серверних функцій (наприклад, React Server Components, Server Actions тощо) за потреби.

Фреймворк Next.js вже підтримує простий механізм маршрутизації в папці сторінок, тому для маршрутизації не потрібні сторонні бібліотеки. Для перенаправлення між сторінками можна використовувати компонент посилання Next.js і передати відповідну URL-адресу для перенаправлення.

Згідно з офіційною документацією [11], на відміну від простих React.js-додатків, які підтримують отримання даних лише за допомогою CSR, фреймворк Next.js підтримує механізми отримання даних через CSR, SSR, SSG та інкрементальну регенерацію статичних сторінок (ISR, Incremental Static Regeneration). Таким чином, механізм отримання даних можна налаштувати відповідно до потреб застосунку.

Оскільки Next.js використовує SSR, то SSR конвертує код React у HTML під час першого запиту. Для порівняння, типовий SPA-сайт доставляє браузеру велике корисне навантаження на Javascript. Більшість HTML-коду, що відображається браузерами, генерується Javascript. Ця різниця в стратегії завантаження суттєво впливає на рейтинг у пошуку.

SEO-боти сканують Інтернет і перевіряють HTML кожної URL-адреси, з якою стикаються. Коли робота досягає сторінки SPA, яка не є SSR, вона використовує багато Javascript замість HTML. SEO-роботи не розуміють Javascript і не завжди чекають, поки він буде перетворений на HTML. Це означає, що роботи, які визначають, що відображається на першій сторінці результатів пошуку, не можуть прочитати цю сторінку. Якщо вони не можуть прочитати сторінку, Google не покаже її як «найкращий» варіант відповіді на пошуковий запит користувача. Саме тут допомагає Next.js.

Завдяки інкапсуляції SPA-сторінок у Next.js, усі запити до URL-адреси вебсайту завжди генерують зручні для роботів HTML-сторінки. Це пропонує найкраще з обох світів. Він може створювати SPA вебсайти, отримуючи при цьому переваги SEO традиційної архітектури вебсайтів.

Основним недоліком Next.js є високе навантаження на сервер. Оскільки фреймворк Next.js використовує метод SSR у застосунках, навантаження на сервер збільшується через часті запити до сервера. Потрібно дочекатися, поки

вебсайт стане інтерактивним. Попередній рендеринг за допомогою SSR (серверний рендеринг) може призвести до того, що користувач введе сторінку, а потім HTML-файл буде відображено та обслуговуване сервером. Однак, доки браузер успішно не запустить React, користувач може бачити лише вигляд сторінки та не може взаємодіяти зі сторінкою. Далі необхідно повне перезавантаження сторінки за допомогою SSR (серверний рендеринг): коли користувач змінює сторінку, вся сторінка перезавантажується, оскільки користувач повинен запросити HTML-файл, який був відображений на сервері.

Таким чином, React.js спрощує створення інтерактивних користувацьких інтерфейсів, тоді як Next.js пропонує такі функції, як рендеринг на стороні сервера для кращого SEO. Компоненти React та JSX покращують читабельність коду та можливість повторного використання. Однак результати показують, що React.js може вимагати додаткових бібліотек для маршрутизації та може бути складним для SEO. Next.js спрощує маршрутизацію, підтримує різні механізми отримання даних та покращує SEO завдяки рендерингу на стороні сервера. Однак це може призвести до високого навантаження на сервер та затримок в інтерактивності. У будь-якому випадку, перш ніж використовувати React або Next для певних проєктів, необхідно вийти на рівень налаштування, пов'язаного з процесами оптимізації.

1.4 Оптимізація продуктивності вебзастосунків

Хоча Всесвітня павутина починалася як глобальний обмін інформацією, вона стала найважливішою доступною платформою додатків, окрім своєї початкової мети. Одним із побічних ефектів цієї еволюції, можливо, були неоптимальні способи доставки контенту через Інтернет, що призводило до марнування ресурсів і бізнесу через втрату конверсій. Технічно кажучи, існує багато способів покращити продуктивність вебдодатків. У цьому розділі ми розглянемо наявні варіанти та останні тенденції, пов'язані з покращенням продуктивності вебдодатків.

1.4.1 Основні метрики продуктивності вебзастосунків

Враховуючи, що вебплатформа суттєво розвинулася за останні тридцять років, у веброзробників та авторів вебфреймворків є багато можливостей покращити якість своєї роботи, особливо щодо продуктивності. Далі викладено основні аспекти продуктивності вебзастосунків. Як аспекти цього підходу виділяють напрямки оптимізації, зокрема, затримку сервера, транспортний протокол, розвантаження та кешування.

Затримка сервера – це добре вивчена та зрозуміла проблема, враховуючи, що її неможливо уникнути в традиційних вебзастосунках. Мережа доставки вмісту та периферійні обчислення пропонують способи вирішення проблеми затримки сервера шляхом переміщення ресурсів та обчислень ближче до клієнта. За допомогою додаткового програмного забезпечення цю проблему можна вирішити, перемістивши обчислення до клієнта та мінімізуючи кількість циклів обміну даними з сервером.

Другий аспект пов’язано з транспортним протоколом. Відомо, що з розвитком Інтернету розвивалися й транспортні протоколи, що лежать в його основі. Починаючи з HTTP/2, транспортний протокол Інтернету базувався на потоковій передачі, і ситуація ще більше змінилася з HTTP/3, оскільки протокол перейшов на використання UDP поверх TCP/IP, що дозволило зняти попередні обмеження [12]. Як зазначалося в [12], навіть із потоковим протоколом продуктивність все ще обмежена явищем, яке називається блокуванням рендерингу, яке виникає, коли файли, що були передані потоком клієнту, повинні бути повторно зібрані, перш ніж рендеринг сторінки може продовжитися. Таким чином, підкреслюється необхідність уникати операцій блокування рендерингу на клієнті. Як потенційне рішення, автори [12] пропонують реорганізацію вмісту та структури сторінки для забезпечення зручності потокової передачі. Перевага цього підходу може бути помітна у

зменшенні FCP, оскільки браузер може швидше отримати доступ до контенту. Таким чином, найбільшою зміною став перехід до потокових підходів [12].

Третім аспектом є робота з розвантаження програми. Розвантаження відноситься до перенесення даних з одного цифрового пристрою на інший. Це рішення, при якому обчислення переносяться на ресурсні комп'ютери для збільшення можливостей мобільних пристроїв. Хоча середовища виконання JavaScript традиційно були однопоточними, впровадження нових примітивів, таких як Web Workers, дозволило розробникам подолати це обмеження [13]. Сучасні браузери добре підтримують Web Workers, а глобальна підтримка становить близько 98,4%. Як зазначається в [13], клієнт і сервер спільно виконують обчислювальні завдання, і через неоднозначність того, яку логіку виконувати, окрім засобів контролю, пов'язаних з безпекою, [13] цей тип обчислень назвали сірими обчисленнями. Сірі обчислення пов'язані з етичними проблемами, пов'язаними з тим, скільки ресурсів клієнта слід споживати, і є витрати та переваги, які слід враховувати, оскільки методи сірих обчислень забезпечують переваги, наприклад, у реагуванні вебсайту [13].

Четвертим аспектом є кешування. Як загальний метод, кешування можна застосовувати на різних рівнях стеку вебзастосунків для зменшення необхідної роботи.

1.4.2 Напрямки оптимізації продуктивності вебзастосунків

З огляду на те, що вебплатформи суттєво розвивалися протягом останніх тридцяти років, веброзробники та автори вебфреймворків мають багато можливостей покращити якість своєї роботи, застосовуючи певні методи оптимізації продуктивності [14]. У цьому розділі викладено напрямки оптимізації продуктивності вебзастосунків для сучасних веброзробників. Варто зазначити, що список, ймовірно, не є вичерпним, оскільки постійно з'являються нові методи.

Перший напрямок ґрунтується на підходах до рендерингу. Вебзастосунки можуть бути відображені як на стороні клієнта, так і на стороні сервера, і деякі з цих методів можуть доповнювати один одного [14]. Найбільш відомими прикладами методів рендерингу є CSR, SSR, DPR, ISR та SSG. Кожен фреймворк вебзастосунку за визначенням зобов'язаний реалізувати певний підхід до рендерингу. Чисто клієнтські фреймворки реалізують CSR, тоді як повностекові та серверні фреймворки можуть використовувати SSR, DPR, ISR та SSG.

Розробка, орієнтована на HTML, максимально враховує налаштування, пов'язані з безпосереднім використанням вебплатформи. Вона базується на попередніх ідеях витонченої деградації та прогресивного вдосконалення. Найкращими прикладами є елементи HTML, які можна використовувати безпосередньо, або методи, за допомогою яких CSS можна використовувати для забезпечення інтерактивності без JavaScript. Насправді такий шлях призводить до явного прогресу, але додаток втрачає свою функціональність, якщо вважати важливим фактором здатність до масштабування [15].

Стилізація з акцентом на корисність переосмислює CSS, переміщуючи примітиви стилізації до HTML-розмітки. Це може покращити продуктивність як побічний ефект завдяки меншим та ефективнішим корисним навантаженням CSS. Стилізація з акцентом на корисність – це техніка, яка може використовуватися фреймворком через зовнішні бібліотеки та інструменти. Зазвичай підхід до стилізації фреймворку залежить від розробника.

Програмне забезпечення, орієнтоване на локальне використання, переоцінює те, як слід створювати вебзастосунки, переносячи логіку та дані на клієнта та розглядаючи сервер як допоміжний [16]. Поки що не існує жодного відомого прикладу фреймворку, орієнтованого на локальне використання, оскільки цей підхід призначений для використання зверху.

Часткове завантаження було прийнято фреймворками, такими як Astro та багатьма іншими фреймворками [17], де можливо використовувати розділення

коду. Примітно, що в останньому випадку для отримання суттєвих переваг можуть знадобитися значні зусилля розробника.

Метод усунення непрацюючого коду зазвичай використовується у фреймворках, які використовують пакетувальник у процесі збірки. Гарним прикладом є інтеграція мініфікації до Next.js у продакшн-режимі [18].

Оскільки добре відомо, що менш корисні навантаження, що надсилаються клієнту, призводять до кращої продуктивності, тому до корисних навантажень можна застосовувати методи стиснення. Стиснення зазвичай доступне розробнику для повнофункціональних або серверних фреймворків. Наприклад, у Next.js стиснення доступне через конфігурацію [19].

Тим не менш, отримання найважливішої сукупності метрик, які оцінюють досвід перебування відвідувачів на сторінці (Web Vitals), пов'язані з пошуковою оптимізацією (SEO, Search Engine Optimization), а продуктивність є головною частиною рейтингу сторінки [20].

1.5 Сучасні підходи до пошукової оптимізації у динамічних вебдодатках

Традиційно SEO означало оптимізацію вебсайтів для того, щоб пошукові системи могли легко їх знаходити, але, як зазначають автори [21], значення змінилося на оптимізацію пошукового досвіду, зосередження уваги на користувачеві, а не на пошукових системах, оскільки зосередження на користувачеві має відчутні переваги для видимості вебсайту. Технічно SEO можна розділити на SEO на сторінці, SEO поза сторінкою та технічне SEO [21].

1.5.1 Індексація та ефективні рішення SEO-оптимізації SPA-додатків

У сучасному світі жорстка конкуренція присутня повсюдно в кожній галузі бізнесу, тому для всіх організацій та окремих осіб вкрай важливо мати високо рейтингований пошуковими системами вебсайт [22]. Процес створення

платформи таким чином, щоб вона була впізнаваною та зрозумілою для пошукових систем, називається пошуковою оптимізацією. SEO спрямовує неоплачуваний трафік (також відомий як органічний трафік) і не аналізує платний або прямий трафік. Правильна оптимізація є важливою, оскільки вона збільшує ймовірність того, що користувачі та, що найважливіше, потенційні клієнти знайдуть сторінку після введення фрази. Це також перевага для постійних відвідувачів, які можуть швидко повернутися на сторінку, безпосередньо покращуючи користувацький досвід [22]. Коли розглядається розробка вебсайтів, як динамічних, так і статичних, необхідно враховувати всі важливі фактори, які можуть збільшити трафік на вебсайт безпосередньо або через пошукові системи. Чим більше трафіку, тим вищий буде рейтинг вебсайту та вищий показник продажів. Розробники зазвичай використовують різні способи створення привабливих та вражаючих ефектів на вебсайтах, щоб зробити їх привабливими для користувачів, але власник не може нічого отримати від нього, якщо користувач не може бачити або знайти вебсайт через пошукові системи. Гарне SEO формує довіру та авторитет бренду в очах користувачів і дозволяє аналізувати потреби клієнтів. На основі SEO такі інструменти, як Google Analytics, аналізують сторінку результатів пошукової видачі (SERP) та надають цінну інформацію, таку як кількість людей, які відвідують певну сторінку, як довго вони залишаються, місцезнаходження, вплив ключових слів тощо.

Історично концепція SEO вперше була представлена в 1991 році, коли в Інтернеті було запущено першу сторінку [22]. У минулому термін SEO був відомий як маркетинг у пошукових системах (SEM) через його застосування в маркетингу та бізнес-послугах. Раніше, якщо компанія хотіла мати високий рейтинг, вона мала забезпечити, щоб кількість її ключових слів була вищою, ніж у конкурентів. Такі практики сьогодні відомі як спам і негативно впливають на рейтинг вебсайту. З точки зору бізнесу, SEO є економічно ефективним. Воно не тільки надає вищезгадані важливі дані, що характеризують трафік і відвідувачів, але й дешевше порівняно з вихідною тактикою. SEO

зосереджується на залученні користувачів, які вже шукають певні продукти чи послуги, без значних витрат на рекламу. Що стосується реклами з оплатою за клік (PPC), SEO забезпечує більш тривалі та автентичні довгострокові результати без додаткових витрат на неї. PPC може генерувати швидкий трафік, але це короткострокове рішення з обмеженими перевагами [23].

SEO-алгоритми в пошукових системах є ретельно охоронюваною комерційною таємницею та не розкриваються публічно в повній деталі. Однак, інформація та рекомендації, наприклад, надані Google, допомагають зрозуміти, як працюють ці алгоритми. Також є SEO-фахівці, які пропонують аналітичні дані на основі свого досвіду.

1.5.2 Технічні аспекти SEO

Загалом, пошукові системи використовують такі методи [24]:

• пошукові системи на основі сканерів;

• директорії, керовані людиною;

• гібридні пошукові системи.

Популярні пошукові системи, такі як Google, Bing та Yahoo, належать до першої категорії. Сканер (також званий «роботом» або «павуком») – це програма, призначена для автоматичного пошуку та сканування вебсторінок за URL-адресами [25]. Він здійснює пошук в Інтернеті, щоб знайти потенційні ключові слова, що відповідають пошуковому запиту, збираючи дані (зображення, позиції ключових слів, зворотні посилання та інші дані) з цих сторінок для подальшої обробки. Згодом, на рисунку 1.6 зображено поведінку пошукової системи. Вона створює індекс на основі отриманих копій вебсторінок та ранжує їх за допомогою різних алгоритмів ранжування.

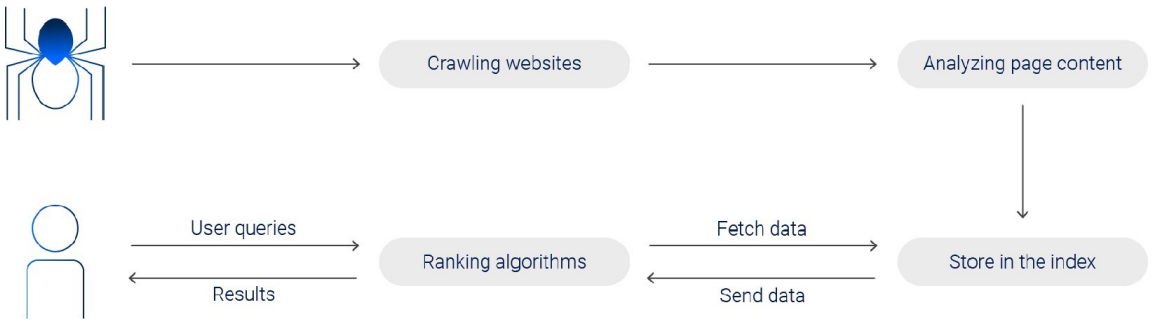

Рисунок 1.6 – Механізм роботи пошукової системи [24]

SEO поділяється на дві підгрупи: на сторінці та поза сторінкою. SEO внутрі сторінки включає методи оптимізації, що застосовуються безпосередньо на вебсайті під контролем розробника. Воно включає оптимізацію різних елементів, таких як тег заголовка, використання ключових слів, тег метаопису, файл robots.txt, оптимізована URL-адреса, карта сайту, організація контенту, заголовки, оптимізація зображень та зручність для мобільних пристроїв [25].

Поза сторінкою SEO передбачає дії та стратегії, які безпосередньо не контролюються розробником, включаючи поширення інформації на соціальних платформах, зіркові рейтинги, нарощування посилань та інші рекламні акції за межами цільового вебсайту.

Алгоритм PageRank, розроблений засновниками Google Ларрі Пейджем та Сергієм Бріном, оцінює важливість та популярність вебсторінок на основі структури посилань. Чим більше цінних та надійних сайтів посилається на сторінку, тим вищий її PageRank. PageRank також враховує значущість кожної сторінки, яка отримала голос, хоча оголошення деяких сторінок є важливими, і відповідно до цієї значущості сторінки, посилання на яку вони створені. Важливі сторінки підтримують більш високу оцінку PageRank і відображаються на перших позиціях результатів пошуку. Загальна формула для PageRank сторінки [redacted] така:

[redacted], (1.2)

де коефіцієнт демпінгу [redacted], загальна кількість сторінок [redacted], кількість вихідних посилань [redacted] на сайт [redacted], [redacted]множина сайтів, які пов'язано з [redacted]. Рівняння (1.2) визначає, власне теорію середнього поля. Іншими словами, вона не враховує флуктуації, які й визначають динаміку мережі.

Якщо говорити про дві групи SEO, обидва вони необхідні і повинні розглядатися пліч-о-пліч для досягнення мети SEO як для статичних, так і для динамічних вебсайтів. Дослідження зазвичай зосереджено на проблемах сканування/індексування динамічних вебсайтів на основних пошукових системах (Google, Yahoo і Bing від Microsoft). Існує безліч інструментів для SEO-досліджень, що зосереджені на різних аспектах для аналізу сканування/індексування та ранжування вебсторінок.. Однак більшість із них платні та пропонують лише кілька днів безкоштовної пробної версії [26]. Тому під час вибору платформи для дослідження враховувалися багатий ресурс вимірювань, що повертаються інструментом, та фінансовий аспект. Пошукові системи сканують вебсайти та отримують інформацію відповідно до своїх власних умов; пізніше ця інформація підсумовується для збереження у базах даних пошукової системи; і, нарешті, ця збережена інформація представляється як результат пошуку вебсайтів SERP у відповідь на пошуковий запит. Таким чином, створення вебсайту для сприяння відповідному скануванню/індексуванню може стати позитивним кроком на шляху до кращого рейтингу.

Зрештою, Seobility [27] і Google PageSpeed Insights [28] можуть бути використані. Seobility — це вебплатформа для аналізу та оптимізації вебсайтів з точки зору рейтингу пошукових систем. Основні функції та можливості, що пропонуються Seobility, включають багато функціоналу.

Google PageSpeed Insights (далі просто PageSpeed) – це безкоштовний інструмент, розроблений Google, який аналізує та оцінює продуктивність вебсайту. Він розділяє результати для мобільних та настільних версій програми.

Окрім розділів продуктивності та SEO, він також проводить вимірювання доступності та перевірених методів.

1.6 Висновок до першого розділу

В першому розділі описано основні принципи перегляду вебдодатків, розглянуто особливості та переваги підходу MVC (Model-View-Controller). Далі підкреслено переваги використання односторінкових додатків порівняно з багатосторінковими; розглядаються їх технологічні особливості. Після цього проведено огляд сучасних методів спрямованих на покращення продуктивності сучасних вебдодатків, аналізуючи погляди до оптимізації як з боку загального підходу, так і із застосуванням методів пошукової оптимізації (SEO, search engine optimization). Показано, що Next.js, який побудовано на базі React, разом з іншими фреймворками, такими як Angular та Vue, змінили парадигму на більш функціональний підхід на основі використання концепції компонентів MVC для створення вебзастосунків. Тому у цьому компоненті представлення містить всю логіку (контролер) та дані (модель), необхідні для візуалізації в межах цього єдиного компонента. Показано, що найкращим підходом до розробки вебдодатків є односторінкові застосунки (SPA, single-page application), що переносять до браузера прийоми роботи, які звичні для персональних додатків. Проаналізовано робочий процес SPA на основі видів рендерингу. Доведено, що React.js спрощує створення красивого динамічного контенту, він все ще залишається односторінковим застосунком. Він чудово підходить для динамічних застосунків, але не дуже добре для SEO. Якщо існуючий SPA, можливо перейти на Next.js без значних змін у коді. Було розглянуто наявні варіанти та останні тенденції, пов'язані з покращенням продуктивності вебдодатків. На основі цього викладено напрямки оптимізації продуктивності вебзастосунків для сучасних веброзробників. Проаналізовано технології SEO, розглянуті алгоритми індексації. Для отримання метрик для порівняльного аналізу проєктів на основі React та Next запропоновано використовувати

безкоштовні вебплатформи для аналізу та оптимізації вебсайтів з точки зору рейтингу пошукових систем.

2 МЕТОДИ ПОБУДОВИ ТА ІНСТРУМЕНТИ ОПТИМІЗАЦІЇ СУЧАСНИХ ВЕБДОДАТКІВ

У цьому розділі досліджено ефективність Next.js як фреймворку, що вирішує проблеми, що виникають через React.js, зокрема, в продуктивності, SEO та рівноправній вебдоступності. На основі розгляду структурних особливостей та можливостей імплементації проєктів на JavaScript викладено опис бібліотеки React. Проведено огляд сучасних бібліотек та фреймворків. Далі розглянуто різні методи та найкращі практики оптимізації продуктивності React-додатків, які допомагають розробникам створювати швидкий та адаптивний UI. У наступному розділі розглянуто фреймворк Next від організації проєктів до шляхів оптимізації. Проведено порівняльний аналіз інструментів аналізу ефективності вебпроєктів. Показано способи моніторингу та аналізу SEO-показників Next.js. Основні метрики оцінки продуктивності та SEO, які потрібно отримувати при порівняльному аналізі проєктів з використанням React і Next, викладено в заключному розділі.

2.1 JavaScript у високопродуктивній веброзробці

JavaScript — одна з найбільш широко використовуваних мов програмування в світі, необхідна для створення динамічних та інтерактивних вебсторінок. Застосування JavaScript відкриває безліч можливостей в індустрії веброзробки. [6]. JavaScript в основному використовується для створення інтерактивних вебзастосунків та динамічних інтерфейсів користувача. Деякі поширені випадки використання JavaScript включають:

1. Створення односторінкових вебзастосунків: Вебзастосунки, які динамічно оновлюють та змінюють вміст без необхідності оновлення сторінки, можна створювати за допомогою JavaScript.
2. Доступ та маніпулювання моделлю об'єктів документа (DOM): JavaScript можна використовувати для доступу та маніпулювання елементами

вебсторінки, що дозволяє створювати динамічні та інтерактивні вебсторінки.

3. Здійснення HTTP-запитів: Вебсторінка може отримувати або надсилати дані без оновлення за допомогою JavaScript для здійснення HTTP-запитів до сервера.
4. Створення розширень браузера: JavaScript можна використовувати для створення розширень браузера, які можуть додавати нові функції та можливості до веббраузера.
5. Створення мобільних додатків: Розробка мобільних додатків за допомогою JavaScript можлива завдяки таким фреймворкам, як React Native, що дозволяє розробникам створювати мобільні додатки за допомогою JavaScript.
6. Створення настільних додатків: JavaScript можна використовувати для створення настільних додатків за допомогою таких технологій, як Electron, що дозволяє розробникам створювати кросплатформні настільні додатки за допомогою вебтехнологій.
7. Машинне навчання: Використовуючи такі бібліотеки, як TensorFlow.js та Brain.js, JavaScript можна використовувати для створення моделей машинного навчання, які можна запускати безпосередньо у браузері.

2.1.1 Структурні особливості JavaScript

Мова програмування JavaScript в основному використовується для розробки динамічних інтерфейсів користувача та інтерактивних ве-додатків. Це мова сценаріїв, яка дозволяє вебдизайнерам додавати на вебсайти динамічні елементи, такі як перевірка форм, слайдери зображень та інтерактивні карти. Оскільки JavaScript є мовою, що керується подіями, вона може реагувати на взаємодію з користувачем, а також на інші події, такі як завантаження сторінок та кліки. Для цього використовуються слухачі подій – функції, що викликаються у відповідь на певні події. JavaScript також є однопотоковою мовою

програмування, що означає, що вона може обробляти лише одне завдання за раз. Але вона використовує модель паралельного виконання, яка називається «циклом подій», що забезпечує неблокуючі операції вводу-виводу. Це означає, що JavaScript може обробляти кілька завдань одночасно, таких як введення даних користувачем та мережеві запити, без зависання інтерфейсу користувача. Оскільки JavaScript — це мова програмування з низькою типізацією, змінні не завжди потрібно оголошувати з певним типом даних, і вони можуть зберігати різноманітні дані одночасно. Це може бути як перевагою, так і недоліком, оскільки підвищує гнучкість кодування, а також ускладнює налагодження.

Як і будь-яка інша мова програмування, JavaScript побудована на наступних визначеннях:

1. Змінні: Змінні використовуються для зберігання даних у JavaScript. Змінні оголошуються за допомогою ключового слова «var», «let» або «const» і можуть бути призначені значенням за допомогою оператора присвоєння =. Змінна може бути використана для зберігання будь-якого типу даних, таких як рядки, числа, логічні значення, масиви та об'єкти.

2. Типи даних: JavaScript має кілька типів даних, таких як рядки, числа, логічні значення та об'єкти. Розуміння цих типів даних та того, як з ними працювати, є важливим під час написання коду JavaScript.

3. Оператори: Оператори використовуються для виконання операцій над змінними та значеннями. JavaScript має кілька типів операторів, включаючи математичні, порівняльні, логічні та оператори присвоєння.

4. Умовні оператори: Умовні оператори використовуються для прийняття рішень у JavaScript. Найпоширенішим умовним оператором є оператор if-else, який дозволяє виконувати різний код залежно від того, чи є певна умова істинною чи хибною.

5. Цикли: Цикли використовуються для повторення блоку коду певну кількість разів. JavaScript має два типи циклів: for та while.

6. Функції: Функції – це блоки коду, які можна повторно використовувати та виконувати кілька разів. Функції визначаються за допомогою ключового слова «function» і можуть приймати параметри та повертати значення.

7. Масиви: Масиви використовуються для зберігання списків даних. Масиви JavaScript – це тип об'єктів, які можуть містити кілька значень будь-якого типу даних.

8. Об'єкти: Об'єкти використовуються для зберігання колекцій даних. Об'єкти JavaScript – це тип структури даних, яка може зберігати кілька властивостей, подібних до змінних, та методів, подібних до функцій.

9. Події: Події – це дії, які може виявити JavaScript, такі як натискання кнопки, завантаження сторінки або відправлення форми. Слухачі подій можна додавати до елементів для виконання функції, коли відбувається певна подія.

10. DOM: Об'єктна модель документа (DOM) – це представлення вебсторінки у вигляді деревоподібної структури, яке використовується JavaScript для доступу та маніпулювання елементами вебсторінки. Розуміння DOM є важливим для створення динамічних та інтерактивних вебсторінок.

Однією з важливих особливостей JavaScript є те, що це об'єктно-орієнтована мова. До того ж, JavaScript спочатку створено для веброзробки. Таким чином, він йде у зв'язці з HTML та CSS, які визначають дизайн програми (статика), а JavaScript пов'язаний з динамікою. Таким чином, можна маніпулювати об'єктами в більш широкому середовищі, навішуючи на них методи та властивості у відповідності до правил ООП. Зазначено, що у JavaScript існує чотири способи створити об'єкт: функція-контруктор (constructor); клас (class); зв'язування об'єктів (object linking to other object, OLOO); фабрична функція (Factory Function).

Таким чином, динаміка сайтів у мережі більшою мірою визначається мовою JavaScript. З іншого боку, правильна організація проєктів грає визначальну роль в оптимізації. Ці стратегії можуть вимагати більш складного підходу, але їхній потенційний вплив на видимість сайту занадто великий, щоб його ігнорувати.

2.1.2 Огляд сучасних бібліотек та фреймворків

JavaScript є популярним вибором для створення вебдодатків, оскільки його можна безпосередньо інтегрувати в HTML та CSS і запускати у веббраузері клієнта. Провідними варіантами для фронтенд-розробки є React [9], Angular [30] та Vue.js [31]. React, розроблений Facebook, відомий своєю компонентною архітектурою та віртуальним DOM. Angular [30] від Google пропонує комплексний фреймворк із двостороннім зв'язуванням даних, тоді як Vue.js відомий своєю простотою та гнучкістю. Для серверної розробки широко використовуються Node.js [29] та Express.js [32]. Node.js, побудований на рушії V8 Chrome, дозволяє використовувати JavaScript для серверних скриптів, що дозволяє повноцінну розробку. Express.js спрощує розробку серверних застосунків та API, а стек MERN (MongoDB, Express.js, React та Node.js) особливо популярний завдяки своїй гнучкості та ефективності. Це означає, що одну й ту саму мову можна використовувати як для фронтенду, так і для бекенду вебдодатку, що робить її популярним вибором для веброзробників.. Крім того, його часто поєднують з іншими інструментами веброзробки та вебфреймворками, такими як Angular або Next [11] на базі бібліотеки React.

Для мобільної розробки React Native [33] та Ionic дозволяють створювати кросплатформні додатки за допомогою вебтехнологій. React Native, розроблений Facebook, дозволяє використовувати мобільні додатки, забезпечуючи майже нативний досвід, тоді як Ionic, використовуючи HTML, CSS та JavaScript, може зіткнутися з проблемами продуктивності у складних додатках. Керування станом є ще одним критичним аспектом, оскільки такі бібліотеки, як Redux [34] та MobX, керують динамічними даними. Redux, відомий своїм передбачуваним контейнером станів, зазвичай використовується з React для централізації стану та логіки додатків. Крім того, допоміжні бібліотеки, такі як Lodash, покращують читабельність та зручність обслуговування коду, оптимізуючи розробку та спрощуючи вирішення щоденних завдань програмування.

Для кожної бібліотеки на основі JavaScript існують свої переваги та недоліки. Так, React [9] добре відомий своєю компонентною архітектурою, яка сприяє повторному використанню коду та легкості керування. Використання віртуального DOM у React забезпечує ефективне оновлення та рендеринг, що сприяє швидким та чуйним користувацьким інтерфейсам. Крім того, React має широку підтримку спільноти, багату екосистему бібліотек та інструментів, а також надійні інструменти для розробників, які покращують процес налагодження та розробки.

Angular [30] виділяється як комплексний фреймворк, який надає багатий набір інструментів та функцій «з коробки». Його двостороннє зв'язування даних спрощує синхронізацію між моделлю та представленням, а також він має сильну підтримку та регулярні оновлення від Google. Використання впровадження залежностей в Angular підвищує модульність та спрощує тестування.

Vue.js [31] відомий своєю простотою та легкістю інтеграції, що робить його доступним для початківців. Він пропонує гнучкість, яка добре масштабується як для малих, так і для великих застосунків, а його добре підтримувана та детальна документація допомагає у навчанні та усуненні неполадок. Vue.js має реактивну та ефективну систему зв'язування даних. Однак, він має меншу спільноту порівняно з React та Angular, що може вплинути на доступність сторонніх бібліотек та інструментів. Крім того, йому бракує підтримки корпоративного рівня, яку надають React та Angular, і він може зіткнутися з проблемами масштабованості у дуже великих застосунках.

За допомогою JavaScript можна створювати широкий спектр вебдодатків, від простих статичних вебсайтів до складних інтерактивних. Крім того, його можна використовувати для створення розширень для комп'ютерів, мобільних пристроїв та браузерів.

2.2 Бібліотека React та оптимізація

React був створений на початку 2010-х років для Facebook, тепер відомого як Meta, оскільки доступні на той час рішення були неадекватними для їхніх потреб. Пізніше бібліотека набула статусу програмного забезпечення з відкритим вихідним кодом у 2013 році і з того часу стала найпопулярнішою для сучасної фронтенд-розробки. React вплинув на сферу веброзробки. Наприклад, одна з центральних ідей React полягає в тому, що інтерфейс користувача (UI) є функцією стану програми, і це було широко прийнято пізніше іншими фреймворками. У цьому розділі викладено, як створювати проєкт у React, визначено компоненти та розглянуто його структуру. Розглянуто також можливості React із погляду оптимізації проєкту.

2.2.1 Структура і компоненти

React - всього лише велика бібліотека, проте він став дуже популярним завдяки використанню повторно використовуваних компонентів як основних будівельних блоків вебдодатків. Для шаблонизації React використовує JavaScript XML (JSX), який пізніше був прийнятий багатьма іншими фреймворками. JSX - це розширення синтаксису для JavaScript, яке дозволяє писати розмітку, схожу на HTML, всередині файлу JavaScript, який потім компілюється під час складання JavaScript

Рендеринг у контексті веброзробки – це процес, який виконують браузери. Він перетворює код JavaScript, HTML та CSS на візуальне та інтерактивне представлення. React використовує CSR, який має такі високорівневі кроки: 1) клієнт отримує мінімальний HTML від сервера, 2) клієнт отримує та виконує JavaScript від сервера та 3) клієнт рендерить вебсторінку. Цей метод забезпечує плавний UX, оскільки програма може оновлювати свій стан без оновлення браузера, але має недолік у вигляді повільного початкового завантаження. З SPA відстежувати інтерфейс користувача та підтримувати його стан складно і дуже

трудомістко. З React потрібно турбуватися лише про одне: кінцевий стан інтерфейсу користувача. Неважливо, в якому стані ваш інтерфейс користувача був спочатку, яку послідовність кроків зробили користувачі, щоб змінити інтерфейс користувача. Важливо лише те, де опинився інтерфейс користувача. React подбає про все інше. DOM – це інтерфейс, який використовується браузерами для рендерингу, і він визначає вебсторінку як деревоподібне представлення. Для рендерингу React використовує віртуальний DOM, який є представленням реального DOM у пам'яті. Базовий підхід до рендерингу детально описується наступним чином: коли щось змінюється, що запускає повторний рендеринг сторінки, React повторно рендерить відповідні частини віртуального DOM, починаючи з компонента, який змінився, і рекурсивно включаючи всі його дочірні елементи. Після цього React порівнює отриманий віртуальний DOM з попереднім віртуальним DOM, щоб визначити різницю. Після виявлення відмінностей React вносить ці зміни до фактичного DOM. Замість того, щоб розглядати візуальні елементи у застосунку як один монолітний блок, React розбиває візуальні елементи на все менші й менші компоненти (рис. 2.1).

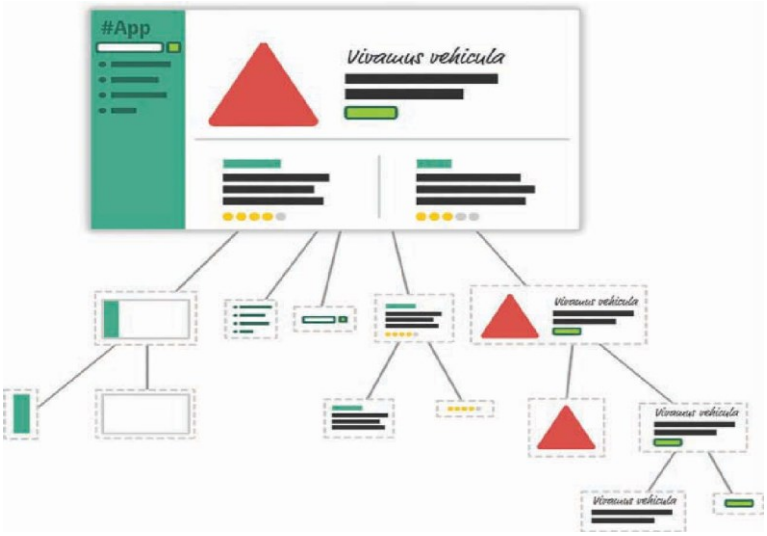

Рисунок 2.1 – Приклад того, як візуальні елементи вашого додатку можна розбити на менші частини

Як і з усім іншим у програмуванні, гарною ідеєю є робити речі модульними, компактними та самодостатніми. React поширює цю добре зарекомендовану ідею на користувацькі інтерфейси. Багато основних API React зосереджені на спрощенні створення менших візуальних компонентів, які пізніше можна комбінувати з іншими візуальними компонентами для створення більших та складніших візуальних компонентів.

Компоненти - це одна з частин, які роблять React чудовим інструментом. Вони є одним з основних способів визначення візуальних елементів та взаємодій, які складають те, що бачать люди, коли використовують програму. Рішення багатьох проблем можна знайти в компонентах React. Компоненти React — це багаторазові фрагменти JavaScript, які виводять (через JSX) HTML-елементи. Спроба створити застосунок на React без використання компонента — це щось на кшталт створення застосунку на основі JavaScript без використання функцій.

Нижче розглянуто підготовку готового проєкту в React. Вважається, що було проведено розбиття проєкту на компоненти, як на рисунку 2.1. Маніпуляції з ними закладено всередині React коду. У JavaScript це зробити неможливо, тому використовується синтаксис JSX. Що стосується процесів розробки та складання, то спільнота та екосистема JavaScript пропонують безліч варіантів. Один із найпростіших і найпоширеніших підходів — використання утиліти Create React App (CRA) (у якої є чудова документація [35]). Щоб використовувати CRA, на комп'ютері повинно бути встановлено Node.js. Щоб встановити Node.js, треба перейти на сайт [36] і завантажити програму встановлення, яка відповідає операційній системі. Треба дотримуватись інструкцій програми встановлення. Для перевірки її наявності можна через командний рядок запитати версію: `node -v`. Повинно вивести строку з версією, наприклад, `v20.11.1`. Коли все буде готове, можливо скористатися послугами, які надає утиліта диспетчера пакетів Node npm (скорочено від Node

Package Manager) – це пакетний менеджер, який дозволяє встановлювати та видаляти JavaScript пакети, керувати їх версіями та залежностями. Версія npm вводиться через командний рядок: `npm -v`. Отже, настав час встановити CRA:

```
npm install -g create-react-app
```

Для створення нового React-додатку використовують утиліту npx, яка постачається разом з Node.js. Вона дозволяє виконувати (звідси і x) сценарії пакетів Node. Треба запустити сценарій CRA один раз: він скачає та виконає останню версію, повністю налаштує програму.

```
npx create-react-app my-test
```

Таким чином створюється React-додаток під назвою my-test. Запуск сервера розробки створюється командою: `npm start` . Ці команди відкриють браузер і надішлють його на localhost:3000/, де можна побачити працюючий React-додаток.

Для подальшого кодування рекомендується використовувати якийсь редактор; рекомендується Visual Studio Code, але цілком підходить і Sublime Text. У редакторі (рис. 2.2) ліворуч дерево проєкту, праворуч – вміст папок. У цьому випадку виділяється App.js.

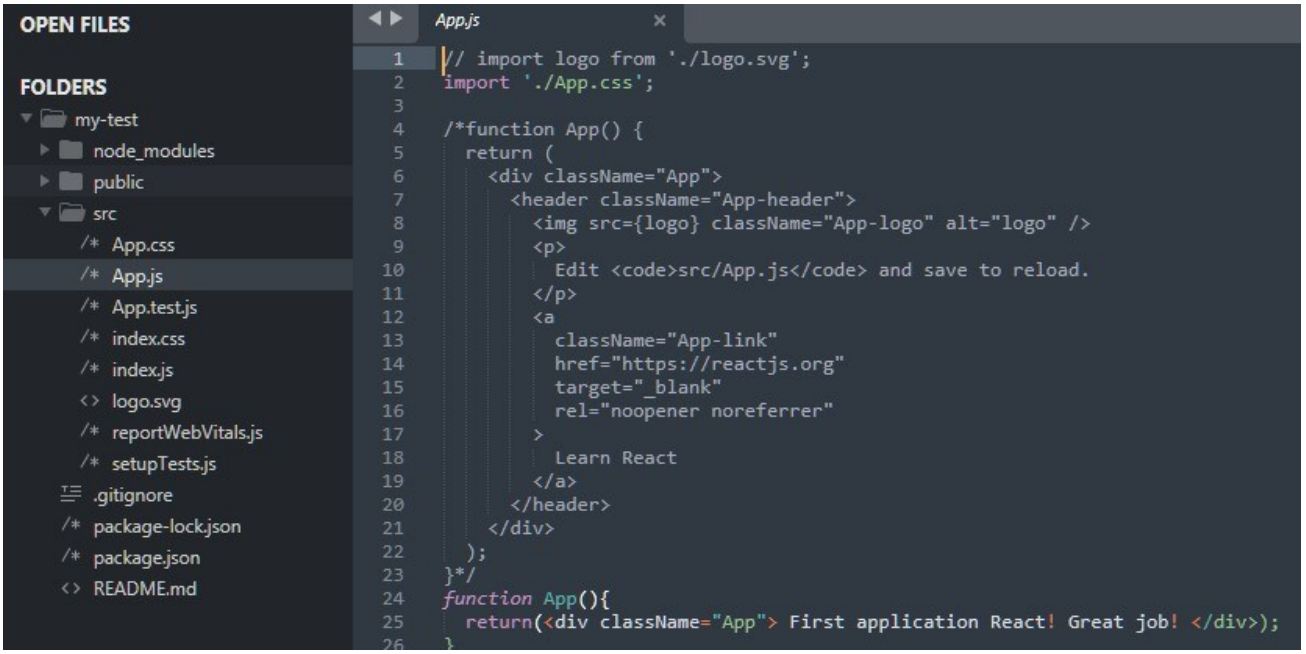

Рисунок 2.2 – Структура React-додатку

Закоментовані частини файлу App.js відповідають стандартному набору. Далі додається функція, яка у базовому блоковому елементі HTML виводить простий текст. Звичайно, для цього потрібно змінити App.css.

Далі наведено опис папок та їх призначення.

• node-modules – це папка, яка містить пакети (залежності), які потрібні нашій програмі. Їх завантажує та оновлює npm.

• public – це папка, в якій знаходиться сторінка index.html, що визначає HTML шаблон усієї нашої програми; окрім цього вона ще містить деякі інші файли для програми (favicon.ico, manifest.json, robots.txt і т.д.), а також у неї при необхідності можна помістити будь-які інші файли, які мають бути скопійовані в папку build незайманими, тобто без обробки їх за допомогою Webpack.

• src (скорочено від source) – каталог, який містить вихідний код програми. Розробка проєкту практично ведеться тут.

• .gitignore – потрібний для приховання файлів та папок від системи контролю версій GIT. Цей файл містить усі необхідні записи, якимсь певним чином його налаштовувати не потрібно.

• package.json – це набір метаданих про вашу програму: назва проєкту, версія, пакети від яких залежить проєкт і т.д. Слід зазначити, що package.json є ключовим файлом для будь-яких програм, заснованих на Node.js.

• package-lock.json – файл, який створює npm автоматично під час встановлення залежностей. Він містить повну інформацію про всі встановлені залежності, включаючи їх точні версії. Крім package-lock.json, може бути ще yarn.lock, якщо використовується Yarn для встановлення пакетів.

• README.md - це короткий довідковий посібник із використання інструменту create-react-app.

Після того як зібрано проєкт, проведено налагодження та демонстрацію в браузері, необхідно запустити процес складання та пакування додатку, готового до розгортання на сервері: `npm run build`. Це процес складання та пакування програми, готової до розгортання. Складання знаходиться в папці /build.

Далі розглянуто, як пов'язані проєкти React з оптимізацією продуктивності та SEO.

2.2.2 Оптимізація у проєктах React

React.js здобув величезну популярність завдяки своїй простоті та гнучкості у створенні динамічних користувацьких інтерфейсів. Однак, зі зростанням складності додатків, забезпечення оптимальної продуктивності стає критично важливим. У цьому розділі розглянуто різні методи та найкращі практики оптимізації продуктивності React-додатків, що допомагають розробникам створювати швидкий та адаптивний UI.

Оптимізація продуктивності в React включає покращення швидкості рендерингу, зменшення непотрібних повторних рендерів та оптимізацію використання ресурсів для покращення загального користувацького досвіду. Важливими є принципи процесу рендерингу в React, включаючи віртуальний DOM, узгодження та методи життєвого циклу компонентів.

Перед впровадженням оптимізації важливо виявити вузькі місця продуктивності за допомогою інструментів профілю, таких як React DevTools

[37], Chrome DevTools [38] (не обов'язково Lighthouse) або інших бібліотек, таких як React Profiler, за допомогою якої здійснюється профілювання продуктивності React-додатків. Для цього потрібно проаналізувати компоненти, які часто перевантажуються, виявити дорогі операції та неефективну вибірку даних, щоб визначити області для покращення.

Для запам'ятовування функціональних компонентів та запобігання непотрібним повторним рендерингам рекомендується використовувати React.memo(). Мемоїзація — це метод оптимізації, який використовується переважно для пришвидшення роботи комп'ютерних програм шляхом збереження результатів ресурсомістких викликів функцій та повернення кешованого результату, коли ті самі вхідні дані трапляються знову. Мемоїзована функція зазвичай швидша, тому що якщо функція викликається з тими ж значеннями, що й попередня, то замість виконання логіки функції вона отримує результат з кешу. Таким чином, мемоїзація кешує результат рендерингу компонента, гарантуючи, що він повторно рендериться лише тоді, коли змінюються його властивості.

Для того щоб проілюструвати способи запам'ятовування, розглядається код у лістингу 2.1. Тут усі дочірні елементи в UserDetails базуються на Props. Цей компонент без урахування стану буде повторно відображатися щоразу, коли Props змінюється. Якщо атрибут компонента UserDetails рідше змінюється, то він є гарним кандидатом для використання версії компонента memoize. Цей метод виконає поверхневе порівняння як Props, так і контексту компонента на основі рівності.

Лістинг 2.1 – Способи мемоїзація компонента `UserDetails`

```
import moize from 'moize';

const UserDetails = ({user, onEdit}) =>{
    const {title, full_name, profile_img} = user;

    return (
        <div className="user-detail-wrapper">
            <img src={profile_img} />
            <h4>{full_name}</h4>
```

```
            <p>{title}</p>
        </div>
    )
}
//Це убираємо
export default moize(UserDetails,{
    isReact: true
});
// Це використовуємо
export default React.memo(UserDetails)
```

Особливе значення для продуктивності рендерингу має оптимізація компонентів класу: для цього застосовується метод життєвого циклу shouldComponentUpdate. Цей метод дозволяє контролювати, коли компонент повинен рендеруватися на основі змін у властивостях або стані. Припустимо, перезаписується shouldComponentUpdate та перевіряється nextState на відповідність this.state, щоб переконатися, що повторно рендеряться компоненти лише тоді, коли відбуваються зміни у стані (дивиться лістинг 2.2).

Лістинг 2.2 – Метод життєвого циклу shouldComponentUpdate

```
shouldComponentUpdate(nextProps, nextState) {
    if (this.state.users !== nextState.users) {
      return true;
    }
    return false;
  }
```

Навіть якщо в масиві користувача відбудуться зміни, React не перерендерить інтерфейс користувача, оскільки це те саме посилання.

Оновлення стану в React є асинхронними, що означає, що його стан не обов'язково оновлюється миттєво. Тому рекомендується використовувати хуки useCallback та useMemo для мемоїзації функцій та обчислених значень відповідно, а також для запобігання непотрібним повторним рендерингам, спричиненим змінами властивостей. Ці хуки особливо корисні для оптимізації обробників подій та ресурсомістких обчислень.

З метою покращення продуктивності початкового завантаження в деяких випадках рекомендується реалізувати розділення коду та ліниве завантаження [39]. Ліниве завантаження (Lazy loading) - це стратегія, яка спрямована на визначення ресурсів, як не критичних, для того, щоб відкласти завантаження

цих ресурсів у той час, коли вони справді необхідні. Так можна скоротити довжину критичних етапів рендерингу, що призводить до зменшення часу завантаження програми. Тому треба розділити додаток на менші фрагменти та завантажити їх динамічно за потреби, зменшуючи початковий розмір пакета та покращуючи час взаємодії. React.lazy() та Suspense корисні для компонентів лінивого завантаження.

Оптимізація мережевих запитів реалізується за допомогою React Query, яка оптимізує вибірку та кешування даних. React Query - потужна бібліотека для керування станом сервера у програмах React. React Query надає хуки для вибірки, кешування та оновлення даних, скорочуючи надмірні мережеві запити та підвищуючи продуктивність вибірки даних.

Виконуючи ці кроки та впроваджуючи методи оптимізації продуктивності у додатку React, можна значно підвищити швидкість рендерингу, скоротити непотрібні повторні рендеринги та покращити спільну взаємодію з користувачем. Не слід забувати регулярно профілювати програми React, відстежувати показники продуктивності та здійснювати поточну оптимізацію для досягнення оптимальної продуктивності. Зосередившись на оптимізації продуктивності, можна створювати високопродуктивні програми React, які відповідають очікуванням користувачів за швидкістю та відгуком.

2.3 Фреймворк Next.js як інструмент оптимізації

Next.js [11] швидко став основним фреймворком для створення потужних, масштабованих вебдодатків. Здатність обробляти рендеринг на стороні сервера, генерацію статичних сайтів та маршрутизацію API робить його ідеальним вибором для великомасштабних проєктів [10]. Однак, зі зростанням складності застосунку, підтримка чіткої та ефективної структури проєкту є критично важливою для забезпечення довгострокового успіху.

2.3.1 Організація та побудова проєктів у Next.js

Оскільки Next.js є лише фреймворком бібліотеки React, то методика створення проєкту залишається схожою на те, що застосовується для материнської бібліотеки. Так, для створення проєкту (під назвою `my-next-app`) необхідно в командному рядку вибраної директорії ввести наступне:

```
npx create-next-app@latest my-next-app
```

Добре організована структура проєкту покращує зручність підтримки, читабельність та масштабованість. Вона допомагає команді зрозуміти, де знаходити файли, де додавати нові функції та як ефективно орієнтуватися в кодовій базі. Більше того, у міру масштабування програми чітка структура допомагає запобігти технічному боргу, полегшуючи залучення нових розробників та керування змінами.

На рисунку 2.3 зображено рекомендовану структуру для великомасштабного проєкту Next.js.

```
my-next-app/
├── public/                 # Static files (images, fonts, etc.)
├── src/
│   ├── components/         # Reusable UI components
│   ├── pages/              # Next.js pages and routes
│   │   ├── api/            # API routes
│   │   ├── _app.js         # Custom App component
│   │   └── index.js        # Home page
│   ├── styles/             # Global styles and CSS modules
│   ├── hooks/              # Custom React hooks
│   ├── contexts/           # React context providers
│   ├── utils/              # Utility functions and helpers
│   ├── services/           # API service calls and data fetching
│   ├── constants/          # Constant values used across the app
│   ├── assets/             # Images, icons, and other assets
│   └── layouts/            # Layout components for consistent design
├── .env.local              # Local environment variables
├── .gitignore              # Files to ignore in git
├── next.config.js          # Next.js configuration file
├── package.json            # Project dependencies and scripts
└── README.md               # Project documentation
```

Рисунок 2.3 – Структура проєкту Next.js

Тепер розглянемо розбивку кожного каталогу. По-перше, папка public/ містить статичні файли, які можна обслуговувати безпосередньо, такі як зображення та шрифти. До файлів, розміщених тут, можна отримати доступ через URL-адресу без будь-якої спеціальної обробки. Далі йде src/ - ядро програми, де знаходиться весь ваш код.

• components/: Містить компоненти інтерфейсу користувача, які можна використовувати повторно, такі як кнопки, модальні вікна та елементи форм. Кожен компонент може мати власну папку з відповідними стилями та тестами. node-modules – це папка, яка містить пакети (залежності), які потрібні нашій програмі. Їх завантажує та оновлює npm.

• pages/: Структура цієї папки визначає маршрутизацію програми. Кожен файл відповідає маршруту. Підпапка api/ призначена для безсерверних функцій. Файл користувача _app.js використовується для ініціалізації сторінок і може використовуватися для розміщення сторінок постачальників контексту або компонентів макета.

• styles/: Тут розміщуються глобальні стилі та модулі CSS. Ви можете організувати стилі компонентів або сторінок для ясності.

• hooks/: Користувацькі React-хуки для повторного використання. Зберігання ваших хуків тут сприяє чіткому розділенню логіки та інтерфейсу користувача.

• contexts/: Постачальники контексту React для керування станами. Це забезпечує централізовану та легку в управлінні глобальну логіку станів.

• utils/: Допоміжні функції, які можна використовувати у додатку. Це можуть бути форматери, валідатори або будь-яка логіка повторного використання.

• services/: Виклики служб API та логіка отримання даних. Таке розділення допомагає зберегти ваші компоненти чистими та зосередженими на рендерингу інтерфейсу користувача.

• constants/: Зберігаються константні значення, що використовуються у програмі, такі як значення конфігурації або статичні рядки.

• assets/: Місце для зображень, значків та інших медіаресурсів, що є частиною вашого додатку.

• layouts/: Компоненти макета, які можна використовувати для створення узгодженого дизайну на різних сторінках.

Поряд із файлами, які визначають функціонування додатків, у проєкті присутні допоміжні. Так, файл .env.local містить локальні змінні середовища, що запобігає потраплянню конфіденційної інформації до вашої кодової бази. Далі, next.config.js – це місце, де налаштовуються параметри Next.js, такі як перенаправлення, перезаписи та інші власні конфігурації. Файл package.json визначає залежності проэкту та містить скрипти для збірки, тестування та запуску програми. І нарешті, файл README.md має містити інформацію про ваш проєкт, зокрема інструкції з налаштування, рекомендації щодо використання та будь-які важливі примітки для розробників.

Слід зауважити, що така ускладнена конструкція додатку аж ніяк не сприяє оптимізації при завершенні збирання; проєкт вимагає додаткових налаштувань у цьому напрямі. Тим не менш, існує набір певних правил та практик для побудови успішного проєкту на базі Next.js.

1. Узгоджені правила іменування: Використовуються узгоджені правила іменування для файлів і папок, щоб покращити читабельність і зручність обслуговування.
2. Модульний код: Розбиваються складні компоненти на менші, керовані частини. Це сприяє повторному використанню та спрощує тестування.
3. Оптимізація продуктивності: Використовуються такі функції Next.js, як оптимізація зображень, генерація статичного сайту та поступова статична регенерація, для покращення продуктивності.
4. Тестування: Впроваджується тестування для компонентів і сторінок. Використовуються бібліотеки, такі як Jest і React Testing Library, щоб забезпечити надійність програми.

5. Документація: Підтримується документація в актуальному стані. Добре документований проєкт полегшує новим розробникам залучення та ефективний внесок.
6. Регулярно оновлюється версія Next.js та залежності, щоб використовувати нові функції та покращення.

Ефективне структурування великомасштабного проєкту Next.js є ключем до досягнення довгострокового успіху. Наслідуючи рекомендовану структуру проєкту та кращі практики, можна створювати підтримувану, масштабовану та ефективну програму, яка може зростати разом з потребами. Коли будується певний проєкт Next.js, треба пам'ятати, що ясність та організація прокладуть шлях до успішної розробки. Проте, коли від проєкту вимагається збалансована автоматизація та SEO, знадобляться додаткові методи та підходи.

2.3.2 Практики SEO у Next.js

Оптимізація програми Next.js для SEO має вирішальне значення для забезпечення того, щоб сайт був видимим і займав високі позиції у результатах пошуку. Next.js, будучи фреймворком React, що забезпечує рендеринг на стороні сервера, пропонує чудові можливості для оптимізації SEO [40]. У цьому розділі розглянуто кращі практики та стратегії для покращення SEO додатка Next.js.

SEO-оптимізація в Next.js передбачає налаштування додатку для кращої доступності та індексації пошуковими системами. Це включає покращення часу завантаження сторінок, забезпечення правильного відображення контенту на стороні сервера та легку навігацію сайтом як користувачами, так і пошуковими роботами. Кращі практики та стратегії SEO краще розділити на частини, кожна з яких пов'язана з можливостями фреймворку Next.js, як було зроблено для React у розділі 2.2.2.

Next.js дозволяє попередньо рендерити сторінки. Це означає, що HTML-код генерується заздалегідь, або під час збірки (статична генерація), або

під час запиту (серверний рендеринг), що робить контент одразу доступним для пошукових систем. З метою оптимізації SEO найкраще використовувати статичну генерацію для сторінок, де вимоги не змінюються часто. Це робиться за допомогою getStaticProps та getStaticPaths в Next.js. Далі рекомендується використовувати рендеринг на сторонньому сервері сторінок, яким потрібні дані першому етапі. SSR реалізується за допомогою getServerSideProps у компонентах сторінки. Відомо, що getServerSideProps - це метод, який дозволяє отримувати дані на серверній стороні перед тим, як рендерувати сторінку. Він запускається лише на серверній стороні та недоступний у браузері.

Теги meta є критично важливими для SEO [41]. Вони надають пошуковим системам метадані про вебсторінку. Компонент <Head> Next.js дозволяє динамічно встановлювати HTML-теги в заголовку сторінок. Для title тега необхідно переконатись, що кожна сторінка має унікальний та описовий заголовок. Що стосується Meta Description, в ньому має бути представлений чіткий і короткий опис вмісту сторінки. І вкрай важливо використання метатегів соціальних мереж (Open Graph, Twitter Cards тощо), щоб контролювати, як контент відображається при поширенні (дивиться лістинг 2.3).

Лістинг 2.3 – Оптимізація метатегів

```
Import Head from 'next/head';

const HomePage = () => (
  <>
    <Head>
      <title>Your Page Title</title>
      <meta name="description" content="A short description..."/>
      <meta property="og:title" content="Your Page Title" />
      <meta property="og:description" content="A detailed
description ..." />
      <meta property="og:image"
content="https://example.com/thumbnail.jpg" />
      <meta name="twitter:card" content="summary_large_image" />
    </Head>
    {/* Page content */}
  </>
);
```

Структуровані дані допомагають пошуковим системам краще розуміти вміст і контекст сторінок. Формат JSON-LD використовується, щоб додавати структуровані дані в компонент <Head>. Відомо, що JSON-LD - один із методів передачі пов'язаних даних з використанням текстового формату JSON. Формат має на меті спростити зусилля розробників з перетворення існуючих JSON-даних на JSON-LD.

Для автоматичної оптимізації зображень з точки зору швидкості та продуктивності рекомендується використовувати вбудований компонент Next.js Image.

```
import Image from 'next/image';
```

Компонент Image забезпечує ліниве завантаження зображень та використання таких форматів, як WebP, якщо вони підтримуються.

Швидкість завантаження сторінки є важливим фактором SEO. Найкраще використовувати функції Next.js, такі як автоматичний поділ коду, і постійно аналізувати розмір пакета, щоб видалити непотрібні. Такі інструменти, як PageSpeed Insights від Google, можуть допомогти визначити області для покращення.

Карти сайту допомагають пошуковим системам знаходити необхідні сторінки. Бібліотека next-sitemap використовується для автоматичного створення картки сайту.

Файл robots.txt повідомляє пошуковим роботам, які сторінки чи файли вони можуть або не можуть вимагати з сайту. Він є важливим для контролю трафіку веброботів.

Оптимізація застосунку Next.js для SEO вимагає багатогранного підходу, зосередженого на стратегіях рендерингу, метатегах, структурованих даних, оптимізації зображень, швидкості завантаження сторінок тощо.

2.4 Інструменти аналізу ефективності вебпроєктів

У сучасних клієнтських проєктах програмного забезпечення розробники часто відчувають брак часу для впровадження нових функцій. Хоча задоволення потреб клієнта важливо, також важлива і якість продукту. Таким чином, профілювання продуктивності можна використовувати для прийняття рішення про те, чи варто витрачати цінний час розробки на оптимізацію для поліпшення досвіду користувача. По-перше, рекомендується проаналізувати загальну продуктивність програми, щоб визначити, чи існує необхідність оптимізації. Крім того, проблеми оптимізації можна виявити за допомогою дослідницького тестування, з'ясовуючи, чи здаються якісь операції програми повільними. Крім того, профілювання продуктивності можна використовувати для аналізу переваг методів оптимізації. У цьому розділі розглянуто процес профілювання продуктивності за допомогою таких інструментів, як Chrome DevTools Lighthouse [38] та React Developer Tools [37], а також компонент React.js Profiler.

2.4.1 Моніторинг та аналіз SEO-показників

Lighthouse [42] — один із Chrome DevTools, набір інструментів розробника в Google Chrome, який можна використовувати для аудиту загальної продуктивності вебпрограми. Для отримання найбільш точних результатів слід використовувати оптимізоване виробниче складання програми для аудиту. Інші аспекти, такі як доступність, кращі практики сучасної веброзробки, SEO та прогресивні стандарти вебдодатків також можна аналізувати за допомогою Lighthouse. Аудит продуктивності, проведений Lighthouse, охоплює різні показники, пов’язані зі швидкістю завантаження сторінки, для розрахунку зваженого балу продуктивності в діапазоні від 0 до 100. Показники оцінюються та класифікуються за кольорами від червоного до зеленого, де червоний та помаранчевий бали означають потребу в покращенні. Загальний бал 90-100, що позначається зеленим кольором, ідеально підходить для гарного користувацького досвіду. Для визначення балів значення показників

порівнюються з даними продуктивності, зібраними з найбільш відвідуваних вебсайтів Інтернету.

Профілювальник React Developer Tools [37] надає інтерактивний метод для визначення причин виникнення певної проблеми продуктивності. Інструмент можна завантажити як розширення для браузера, його можна знайти на вкладці «Profiler» DevTools. Для профілювання програми записується сеанс браузера. Дані про продуктивність збираються при кожному рендерингу програми під час сеансу. Дані групуються за кожним комітом (рис. 2.4), що означає фазу, коли React.js застосовує зміни, такі як оновлення DOM. Висота комітів на відповідної панелі відповідає часу, необхідному для рендерингу. Більше того, коміти зі швидким часом рендерингу можна приховати з налаштувань, щоб відфільтрувати обсяг даних, що відображаються, і, таким чином, спростити процес аналізу [43].

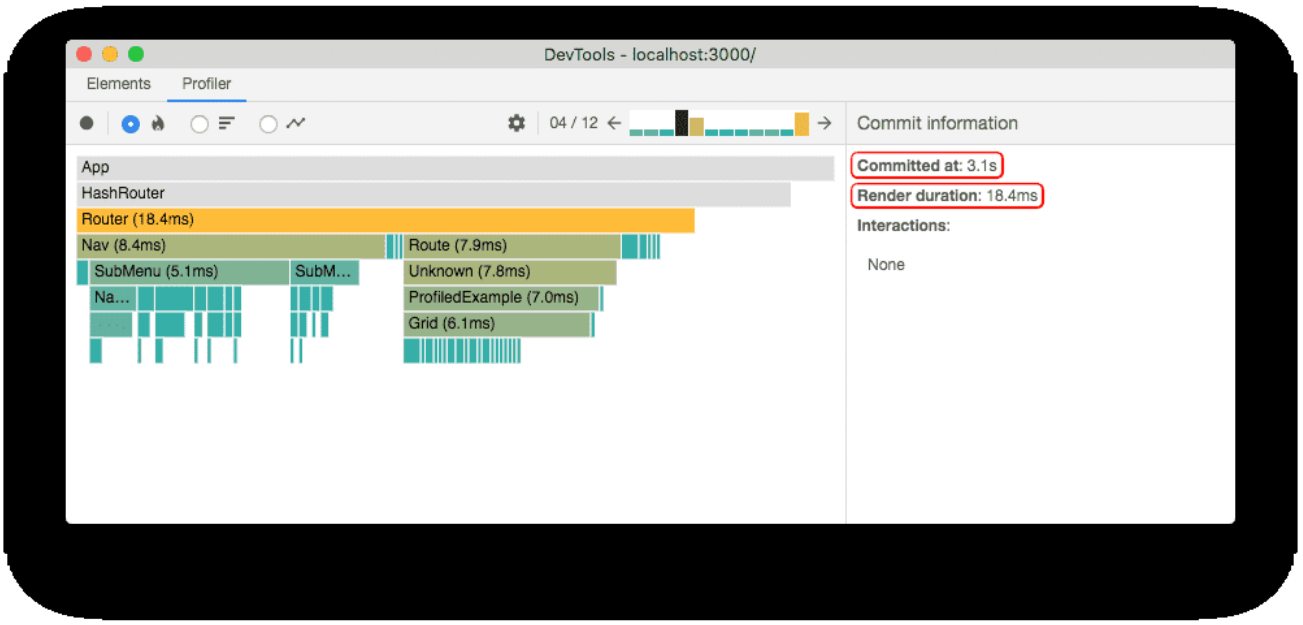

Рисунок 2.4 – Приклад React Developer Tools Profiler результатів [43]

Загалом, Profiler допомагає виявити вузькі місця продуктивності в застосунку. Діаграма може бути ефективним способом розпізнавання проблемних компонентів, оскільки вони займають більше загального візуального простору і тому їх легше помітити. Крім того, аналіз даних

компонентів у групах комітів може виявити непотрібні зміни стану або властивостей, які можуть впливати на продуктивність застосунку.

Функція зворотного виклику профайлера (дивиться лістинг 2.4) [44] виконується щоразу, коли компонент у піддереві виконує коміт. Першим параметром функції зворотного виклику є ідентифікатор профайлера, що корисно, якщо одночасно використовується кілька компонентів. Однак, з точки зору вимірювання продуктивності, найважливішими параметрами є actualDuration та baseDuration. ActualDuration вимірює час, витрачений на рендеринг профайлера та його нащадків. Якщо мемоїзація використовується, значення параметра має значно зменшитися на етапах оновлення, оскільки нащадки повинні повторно рендерити лише тоді, коли змінюються їхні властивості. З іншого боку, baseDuration оцінює найгірший час рендерингу без мемоїзації. Порівняння двох параметрів може бути корисним для визначення впливу використаних методів оптимізації [44].

Лістинг 2.4 – Profiling Navigation компонент у React.js

```
// React will call your onRender callback with information about
what was rendered.
function onRender(id, phase, actualDuration, baseDuration,
startTime, commitTime) {
  // Aggregate or log render timings...
}
//Wrap the <Profiler> component around a React tree to measure its
rendering performance.
<App>
  <Profiler id="Sidebar" onRender={onRender}>
    <Sidebar />
  </Profiler>
  <PageContent />
</App>
```

Отже, було розглянуто деякі інструменти вимірювання продуктивності, які використовуються в React. Звичайно, все це актуально і для фреймворку Next. Тим не менш, оскільки він працює на серверній стороні, краще використовувати інструменти мережі.

2.4.2 Технічні аспекти SEO для Next.js

Розуміння ефективності вебсайту з точки зору SEO є надзвичайно важливим у світі цифрового маркетингу. Один із найповніших способів моніторингу ефективності SEO-оптимізації Next.js полягає в отриманні аналітичних даних з надійних джерел, таких як Google Analytics [45] та Google Search Console [46].

Інтеграція Google Analytics надає значну користь для відстеження трафіку вебсайту та поведінки користувачів. Незалежно від того, чи розробка ведеться на багатонаціональної платформі електронної комерції або на невеликому блог-сайті, інтеграція Google Analytics може стати одним із найрозумніших кроків. Цей інструмент дозволяє відстежувати ключові показники, включаючи користувачів, сеанси, показники відмов, тривалість сеансу, перегляди сторінок тощо, що дає детальне уявлення про трафік вебсайту та поведінку користувачів. Вплітаючи Google Analytics в структуру вашого додатка Next.js за допомогою певних тегів, розміщених безпосередньо на сторінках або через менеджер тегів, стає легше розпізнавати, які частини сайту працюють добре з точки зору залучення користувачів або де необхідно внести покращення, щоб зрештою підвищити ваш рейтинг SEO Next.js. Крім того, на знак визнання його універсальності Google Analytics також пропонує розширені функції, такі як когортний аналіз та інші, які можуть надати більш глибокі та дієві ідеї для вдосконалення стратегій навколо залучення, утримання та монетизації користувачів.

Використання Google Search Console для аналізу індексації та продуктивності вебсайту призначене для тих, хто хоче отримати більш точну картину своєї наступної продуктивності JS SEO на рівні пошукової системи. Важливість цього безкоштовного інструменту не можна недооцінювати, коли прагнете плідної оптимізації вебсайту. Google Search Console дозволяє вимірювати ефективність роботи вебсканера Google на сайті (що критично важливо для покращення індексації), спостерігати за звітами в режимі

реального часу щодо кліків/показів/CTR для кожного ключового слова, а також надати прямі сповіщення про будь-які проблеми, що впливають на видимість на сторінках результатів пошуку (SERP). По суті, використання цього інструменту не лише сприятиме більшій прозорості між додатком Next.js та пошуковими роботами, але й надає відповідну інформацію, необхідну для вдосконалення SEO-стратегій, виправлення помилок, що впливають на продуктивність сайту, та отримання глибшого розуміння поведінки аудиторії в Інтернеті.

Нарешті, відстеження SERP залишається наріжним каменем оцінки працездатності стратегії SEO. Досягнення високої видимості органічно часто призводить до вищих показників клікабельності (CTR), збільшення вебтрафіку та потенційно вищих показників конверсії — тому відстеження коливань цих показників слід максимізувати, коли це можливо. У деяких випадках рейтинги SERP нерідко відчувають легку волатильність через різні фактори, такі як зміни стану. Тим не менш, різкі падіння можуть бути тривожними дзвіночками, що вимагають негайного розслідування та виправлення.

На закінчення цього розділу зазначимо, що ці інструменти є настільки ефективними, наскільки ефективно інтерпретуються їх дані. Усвідомлене поєднання Google Analytics, Google Search Console та регулярного моніторингу SERP та CTR значно підвищить шанси на успіх у досягненні чудових результатів SEO Next.js.

2.5 Метрики оцінки продуктивності та SEO

Показники SEO — це ключові показники, які кажуть, наскільки добре сайт працює в пошукових системах і де можливо досягти покращення для досягнення цілей. В цьому розділі розглянуто основні показники, які використовуються для оцінки продуктивності вебдодатків. В контексті продуктивності з Lighthouse обговорення деяких показників було проведено в розділі 2.4.1. Тому тут значну увагу приділено загальному описі, оскільки деякі інструменти дають свій набір показників.

2.5.1 Ключові показники продуктивності

Існує безліч показників SEO для відстеження. Тим не менш, існує ряд очевидних підходів з використанням основних показників SEO, на яких необхідно зосередитись: органічні конверсії, органічний трафік, рейтинги перефразованих слів, авторитетність вебсайту, CTR. Відстеження правильних показників SEO – ключ до розуміння того, що працює, а що ні. Замість того, щоб губитися в даних, необхідно зосередитися на цифрах, які впливають на цілі.

Конверсії – це справжній доказ того, що розроблена SEO-оптимізація працює. Коли хтось надсилає форму запиту, робить покупку, підписується на вашу розсилку, завантажує електронну книгу – він здійснює конверсію. Це чітка ознака того, що контент і стратегія знаходять відгук у вашої аудиторії. Але не всі конверсії однакові. Для клієнта LeadGen studio успіх може означати генерування більшої кількості органічних лідів завдяки підпискам на форму. Ці ліди потім отримують цінний контент і пропозиції, рухаючись вниз по воронці продажів до того, щоб стати конвертованими лідами. Для сайтів електронної комерції все зосереджено на продажах і доході. Потрібно відстежувати кожну покупку, що надходить з органічного трафіку, щоб побачити рентабельність інвестицій (ROI). Налаштування відстеження конверсій електронної комерції в Google Analytics може здатися складним, але це важливо. Без нього розробник лише здогадується про те, наскільки добре працює SEO-оптимізація.

Авторитетність вебсайту схожа на онлайн-репутацію сайту — вона повідомляє пошуковим системам, наскільки надійним та релевантним є контент. Цей показник може впливати на те, наскільки високо сайт ранжується в результатах пошуку. Він формується кількома факторами, такими як якість ваших зворотних посилань та обсяг органічного трафіку, який залучається при цьому. Різні інструменти використовують різні показники для вимірювання цього. Semrush [47] має «Показник авторитетності», Moz використовує

«Авторитетність домену», а Ahrefs називає це «Рейтингом домену». Вищий бал зазвичай означає, що сайт сприймається пошуковими системами більш сприятливо, що може призвести до кращого рейтингу та більшого трафіку. Далі треба подивитися, що впливає на бал: чи надходять ваші зворотні посилання з авторитетних сайтів? Чи відповідає контент тому, що шукають користувачі? Регулярний моніторинг та розуміння цих факторів може допомогти удосконалити стратегію та з часом збільшити авторитет.

І нарешті, CTR показує, як часто люди натискають на посилання сайту, коли бачать його в результатах пошуку. Необхідні параметри та метрики для аналізу продуктивності та SEO є спільними для всіх додатків. Тому далі описано показники, які пов'язані з Lighthouse.

2.5.2 Метрики індексації та клікабельності

Core Web Vitals (CWV) - це сукупність метрик, які оцінюють досвід перебування відвідувачів на сторінці та визначають, наскільки їм зручно користуватися сайтом. Цілком нові CWB з'являться у всіх інструментах Google, які вимірюють швидкість, продуктивність та досвід сайту в новому звіті Web Vitals у Search Console. Тепер просто потрібно зрозуміти три основні метрики, щоб отримати уявлення про те, як працює сайт чи певні сторінки.

Основні метрики вебпоказників:

• Largest Contentful Paint (LCP): або скільки часу потрібно для завантаження найбільшого елемента контенту, який знаходиться у вікні перегляду.

• First Input Delay (FID), або затримка першого введення: або скільки часу потрібно браузеру, щоб відреагувати на взаємодію, яку вперше ініціював користувач (наприклад, натискання кнопки).

• Cumulative Layout Shift (CLS): або яка частина екрана залежить від руху, тобто чи стрибає щось на екрані?

Ці нові CWV використовують більш практичний підхід і ставлять користувацький досвід на перше місце. Інструменти відвідують ваш сайт через обмежене з'єднання на середньостатистичному пристрої, щоб імітувати те, що може відчувати реальний відвідувач у реальному світі. Замість того, щоб просто завантажувати ваш сайт, як це робили класичні інструменти для прискорення, ці інструменти на базі CWB перевіряють, як і коли він реагує на ввідні дані, і чи відбуваються якісь зміни після початкового завантаження. Вони знаходять точний момент, коли контент готовий до використання, тому ви можете спробувати оптимізувати його, коли він здається занадто повільним. Крім того, ви можете знайти проблеми, які заважають гарному досвіду роботи зі сторінкою.

Lighthouse не лише вимірює продуктивність, але й перевіряє SEO, різні рекомендації та доступність. Це повноцінний інструмент, який допомагає покращити сайт комплексно.

Для вимірювання продуктивності вашого додатка Lighthouse використовує наступні показники:

1. Перша метрика продуктивності Lighthouse — це First Contentful Paint (FCP) вимірює час, необхідний для відображення першої частини вмісту DOM після переходу на сторінку. Це включає зображення, елементи <canvas> та SVG, але виключає елементи всередині iframe.
2. Далі, Speed Index (SI) показує, як швидко контент візуально відображається на сторінці, він вимірюється шляхом захоплення відео завантаження сторінок.
3. Contentful Paint (LCP) вимірює час відображення самого великого зображення або текстового блоку в області перегляду Це одна з найважливіших нових метрик. Тут хороший показник означає, що користувачі сприймають ваш сайт як такий, що швидко завантажується.
4. Time to Interactive (TTI), — це час, необхідний для того, щоб сторінка стала повністю інтерактивною, незалежно від того, коли візуальний контент здається готовим. Сторінка може здаватися швидко

завантаженою, але потім виявити, що натискання деяких кнопок ще нічого не дає.

5. Total Blocking Time (TBT), загальний час блокування: TBT вимірює час між FCP та TTI, коли можуть виникати блокування, що перешкоджають реагуванню.
6. Cumulative Layout Shift (CLS) враховує кількість зміщень макета, що відбуваються під час повного завантаження сторінки. Щоразу, коли елемент перестрибує на екрані з кадру на кадр, це вважається зміщенням макета. CLS означає, що будь-який видимий елемент змінює своє становище без взаємодії з користувачем. Для хорошого досвіду користувача слід уникнути високого показника CLS.

Можна побачити, як розраховуються показники, перейшовши до Lighthouse Scoring Calculator. У звіті Lighthouse також представлені деякі можливості для покращення швидкості роботи мобільного сайту, зокрема, скільки часу вони зекономлять. До них належать зменшення кількості таблиць стилів, що блокують рендеринг, скриптів, що блокують рендеринг, правильне визначення розміру зображень та виправлення зображень поза екраном.

2.6 Висновок до другого розділу

У цьому розділі проаналізовано ефективність Next.js як фреймворку, що вирішує проблеми, що виникають через React.js, зокрема, в продуктивності, SEO та рівноправній вебдоступності. На основі розгляду структурних особливостей та можливостей імплементації проєктів на JavaScript проведено опис бібліотеки React та огляд сучасних бібліотек та фреймворків. Далі розглянуто різні методи та найкращі практики оптимізації продуктивності React-додатків, які допомагають розробникам створювати швидкий та адаптивний UI. Описано фреймворк Next від організації проєктів до шляхів оптимізації. Проведено порівняльний аналіз інструментів аналізу ефективності вебпроєктів. Показано способи моніторингу та аналізу SEO-показників Next.js.

Основні метрики оцінки продуктивності та SEO, які потрібно отримувати при порівняльному аналізі проєктів з використанням React і Next, викладено в заключному розділі.

Оптимізація продуктивності в React включає покращення швидкості рендерингу, зменшення непотрібних повторних рендерів та оптимізацію використання ресурсів для покращення загального користувацького досвіду. Важливо розуміти принципи процесу рендерингу в React, включаючи віртуальний DOM, узгодження та методи життєвого циклу компонентів. Для виявлення вузькіх місць продуктивності рекомендовано вибрати допоміжні інструментів профілювання, такі як React DevTools, Chrome DevToolsб або React Profiler, за допомогою якої здійснюється профілювання продуктивності React-додатків. Для запам'ятовування функціональних компонентів та запобігання непотрібним повторним рендерингам рекомендується використовувати React.memo(). Особливе значення для продуктивності рендерингу має оптимізація компонентів класу: для цього застосовується метод життєвого циклу shouldComponentUpdate. З метою оптимізації SEO найкраще використовувати статичну генерацію для сторінок, де вимоги не змінюються часто. Це робиться за допомогою getStaticProps та getStaticPaths в Next.js. Далі рекомендується використовувати рендеринг на сторонньому сервері сторінок, яким потрібні дані першого етапу аналізу.

Таким чином, у цьому розділі вибрано напрями оптимізації продуктивності та SEO вебдодатків на основі React та Next, визначено основні критерії для цього. На основі аналізу показників продуктивності обрано відповідні інструменти дослідження.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ТА SEO

В даному розділі визначено цілі, фазу планування експерименту, процес проведення експерименту і зразок експерименту. Також розглянуто деякі ідеї експерименту, пов'язані з плануванням процесу виконанням експерименту крок за кроком, тобто, як експерименти проводяться насправді. Далі описано різні методи та найкращі практики оптимізації продуктивності React-додатків, які допомагають розробникам створювати швидкий та адаптивний UI. Проведено порівняльний аналіз інструментів аналізу ефективності вебпроєктів. Показано способи моніторингу та аналізу SEO-показників Next.js. Проаналізовано експериментальні дані щодо основних метрик оцінки продуктивності та SEO, які було отримано при порівняльному аналізі тестових проєктів з використанням React і Next.

3.1 Постановка задачі та методологія експерименту

Це експериментальне дослідження спрямоване на прогнозування та визначення ефектів одного змінного експерименту шляхом керування кількома факторами, які можуть привести до зміни результатів. Головна ціль — використовувати експериментальний метод дослідження для вивчення широко використовуваних методів SEO для покращення індексації динамічних вебсторінок в основних пошукових системах. Другорядна ціль — порівняти методи SEO і побачити ефективність одного методу в порівнянні з іншими. Огляд літератури показує, що динамічні вебсайти зазвичай погано видимі в пошукових системах, тому що вони недружні до пошукових систем. Існує кілька міфів про видимість динамічних вебсайтів у пошукових системах; і пошукові системи постійно розвиваються, щоб зробити їх ефективними, і в цьому відношенні вносяться деякі поліпшення. Крім того, використання методів SEO для допомоги в індексації цих вебсторінок є оптимальною метою

експериментального дослідження. Ця дослідницька робота призводить до кращого розуміння; заповнити пробіл у дослідженнях, надати цікаві знання та виявити істотний вплив на сферу веброзробки та SEO.

3.1.1 Постановка мети й завдань

У експериментальному дослідженні важливо вибрати правильний суб'єкт, оскільки вибір суб'єкта безпосередньо пов'язані з узагальненням результатів, отриманих під час експерименту. Щоб спростити результати для обраної популяції, обраний суб'єкт повинен бути репрезентативним для цієї популяції. Вибір суб'єкта також відомий як вибірка із популяції. Існує два типи методів вибірки популяції: ймовірнісний та неймовірнісний. При ймовірності вибірки ймовірність вибору кожного суб'єкта відома, тоді як у випадку неймовірнісного вибору суб'єкта - невідома.

У цьому дослідженні використано метод ймовірнісної вибірки для вибору суб'єкта. Після цього проекспериментовано з деякими методами SEO (зокрема з дружніми URL-адресами), які задіяні в індексації динамічних вебсторінок у цільових пошукових системах. Оскільки мета досліджень полягає у порівняльному аналізі проєктів на React та Next, для цієї мети вирішено використовувати два динамічні вебсайти різного віку та розміри, завантажені на різні сервери. Основною причиною використання двох сайтів було те, щоб подивитися, чи враховують пошукові системи розмір та вік вебсайту при індексації динамічних URL-адрес. Вибір вебсторінок випадково здійснюється з вебсторінок, які проіндексовані пошуковими системами. Неможливо було випробувати всі методи SEO та повністю оптимізувати динамічний вебсайт через обмежений обсяг досліджень; тому вирішено обмежити дослідження методами, які допомагають індексувати динамічні URL-адреси. Щоб переконатися, що не проводиться індексація вебсайту, якому не вистачає базових методів SEO, були створені тестові застосунки, на яких можна отримати та порівняти релевантні показники. Ще одна причина вибрати

індексацію динамічних вебсторінок як об'єкт для вибірки — це вивчення найбільш спірних областей динамічного вебсайту (динамічних URL-адрес), зокрема таких, що використовують параметри та змінні сеансу у своїх URL-адресах, які вважаються проблемою для більшості пошукових систем.

Експериментальний дизайн - це план нагромадження експериментальних знань, т.е. знань, що ґрунтуються на аналізі дослідницьких даних. Він може бути практичним, коли забезпечує послідовне виконання події для покращення розуміння чи отримання кращої продуктивності. Упорядкування дизайну означає уважний вибір деяких експериментів, які мають виконуватися в контрольованих умовах.

3.1.2 Гіпотеза, змінні та індикатори ефективності

Планування експерименту – непросте завдання, оскільки для планування та розробки експериментів потрібна значна підготовка. Добре спланований експеримент гарантує, що експеримент буде проведено належним чином, а результат найкращим чином відобразить реальний світ. Гіпотеза – це пояснення явища, яке можна перевірити певним чином, який ідилічно доводить або спростовує гіпотезу. До перевірки гіпотези гіпотеза вважається істинною, і метою дослідника є методична перевірка терміну дії гіпотези. Зазвичай дослідники використовують нульову та альтернативну гіпотези на етапі формулювання гіпотези. Нульову гіпотезу можна описати як статистичну гіпотезу; нульова гіпотеза підтверджується для прийняття. Її також називають оригінальною гіпотезою. Будь-яка інша гіпотеза, крім нульової, відома як альтернативна гіпотеза. Коли нульова гіпотеза відхиляється, ми повинні прийняти альтернативну гіпотезу.

В експериментальних дослідженнях змінну можна описати як вимірювання або атрибут, значення якого може змінюватися протягом експерименту. Це допомагає визначити причину (незалежна змінна) та наслідок/результат (залежна змінна) в рамках експерименту. Неправильний

вибір змінних може негативно вплинути на результати експерименту. Тому дуже важливо вибрати правильні змінні. Існує дві добре відомі змінні, а саме незалежна та залежна. Незалежна змінна - це та, яка є причиною, маніпулює або впливає на результати емпіричного дослідження. І навпаки, залежні змінні залежать від поведінки обробки незалежних змінних. Залежні змінні спостерігаються та розглядаються для визначення впливу незалежних змінних. Іншими словами, залежні змінні - це результати дослідження.

У цьому експерименті було вибрано такі незалежні та залежні змінні:

• Незалежні змінні:

Методи SEO: незалежні методи SEO або набір методів SEO – це незалежні змінні, які використовуються для того, щоб допомогти динамічним вебсторінкам індексуватися в цільових пошукових системах.

• Залежні змінні:

Динамічне індексування вебсторінок: залежними змінними буде індексування динамічних вебсторінок у цільових пошукових системах.

У будь-якому дослідженні валідність результатів є фундаментальним питанням для дослідника. Усі дослідницькі варіанти дизайну, ймовірно, мають упередженість та загрози, які можуть вплинути на валідність результатів. Валідність – це ступінь, до якої експеримент виконує те, що він нібито перевіряє. Для точного застосування та розуміння результатів важливо, щоб експеримент був послідовно валідним. Спочатку дослідник класифікує два типи загроз валідності (внутрішні та зовнішні). Сучасні дослідження розширюються до чотирьох категорій: валідність статистичного висновку, зовнішня валідність, внутрішня валідність та валідність конструкту. У підрозділах описано ці загрози валідності, які можуть вплинути на обраний дослідницький дизайн цього експерименту.

Результати експерименту можуть бути неточними, оскільки щось інше вплинуло на результати експериментів. Внутрішні загрози валідності можуть бути факторами, які можуть впливати на незалежну змінну без відома дослідників і впливати на причину.

Статистична валідність висновків стосується питання, яке впливає на здатність робити висновки про зв'язок між оптимізацією та результатом. Статистичний висновок оцінює, чи процедури перевірки даних дають правильні результати.

Конструктивна валідність стосується узагальнення результатів експерименту до теорії або концепцій. У цьому експерименті результати реєструються з пошукових систем після застосування певних методів SEO; отже, ці результати можна використовувати для узагальнення ефективності методів SEO для індексації динамічних вебсайтів. Дизайн експерименту вибирається після розуміння рекомендацій та інтегрованих кроків, які використовуються для вибору відповідного дизайну експерименту, і враховуються загрози валідності, які можуть вплинути на дизайн дослідження експерименту.

3.1.3 Вибір інструментів і технічного середовища

React Developer Tool – це важливе розширення для браузера. Воно надає набір потужних функцій, які допомагають налагоджувати, перевіряти та оптимізувати React-додатки. Тому для аналізу будь-яких проєктів, створених на базі бібліотеки React і фреймворка на серверній стороні Next, це незамінний інструмент. Щоб підкреслити важливість цього інструменту, розглянемо сайт Netflix, який, як відомо, побудований на платформі Next.

Так, після відкриття сайту запускається DevTools в Chrome (рис. 3.1).

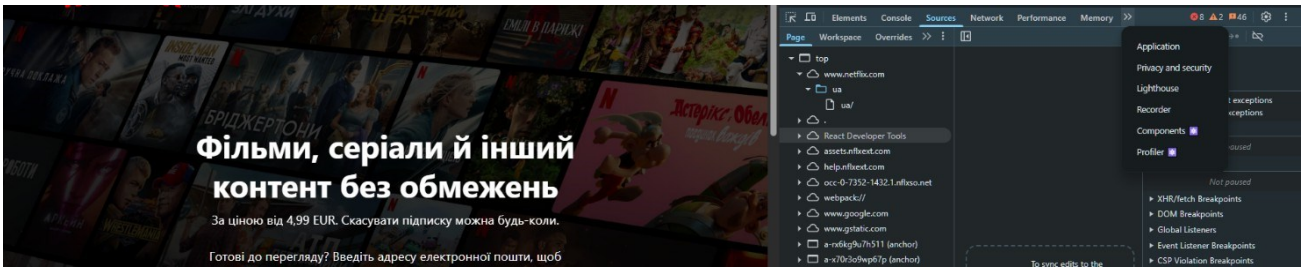

Рисунок 3.1 – Сторінка Netflix з інструментами DevTools

Таким чином, на цьому рисунку показано практично всі інструменти, які використовуються для проведення тестових випробувань. По-перше, це Lighthouse [42] — один із інструментів розробника в Google Chrome, який використовується для аудиту загальної продуктивності вебпрограми. Цей інструмент було розглянуто у розділі 2.4.1.

Після того як встановлюється React Developer Tools на комп'ютер, відкриваються інструменти розробника. Повинні бути дві вкладки, надані розширенням React: Компоненти (Components) та Профіль (Profiler). Спочатку розглядається вкладка Компоненти на рисунку 3.2.

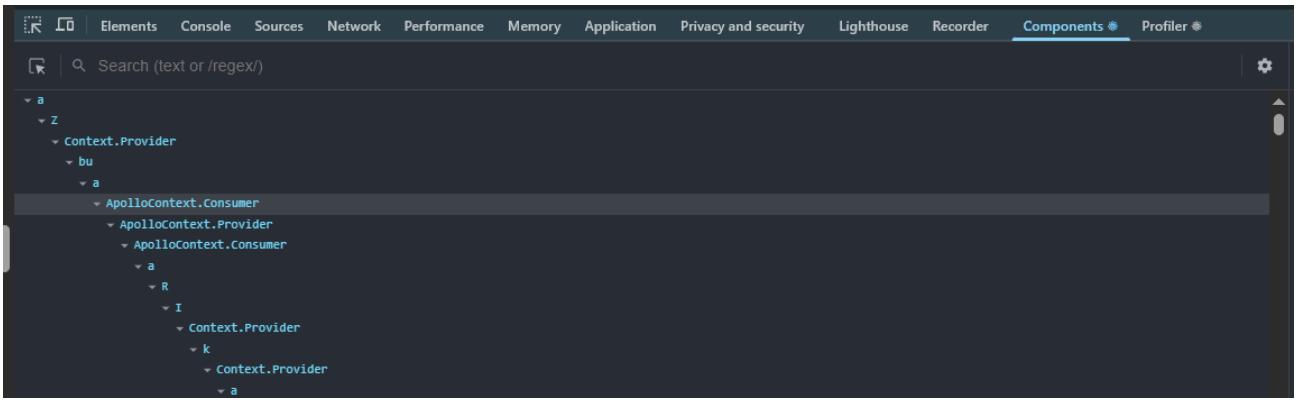

Рисунок 3.2 – Вкладка компоненти React Developer Tools

Зібравши додаток разом, його структура зрозуміла. Ігноруючи базові компоненти, що надаються фреймворком NextJS, все починається з компонента Page. Через деякі обгортки компонентів постачальника теми доходимо до того, що називається «LayoutComponent». Зазвичай, він має дочірні компоненти: Header (Заголовок), Menu (Меню), Speakers (Динаміки) та Footer (Нижній колонтитул). До речі, якщо навести курсор на будь-який з компонентів зі списку, відповідний компонент, що відображатиметься у браузері, буде виділено.

Профілювальник React Developer Tools [37] (дивиться розділ 2.4.1) надає інтерактивний метод для визначення причин виникнення певної проблеми продуктивності. Для профілювання програми записується сеанс браузера. Дані групуються за кожним комітом (рис. 3.3); висота комітів на відповідної панелі відповідає часу, необхідному для рендерингу. Коміти зі швидким часом

рендерингу можна приховати з налаштувань, щоб спростити процес аналізу [43].

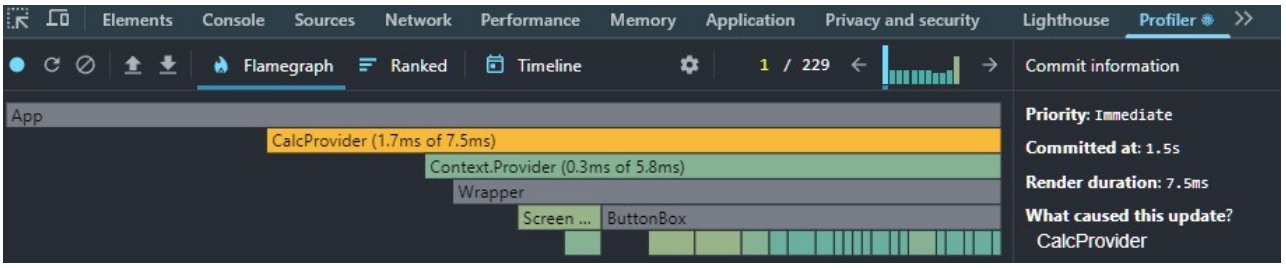

Рисунок 3.3 – Profiler, інформація після запису сеансу

За допомогою Lighthouse перевіряється якість сайту з метою підвищення його продуктивності. Lighthouse аналізує чотири показники: продуктивність, доступність, SEO та кращі практики (рис. 3.4). Для прогресивних вебдодатків додається п'ять параметрів — PWA. На рисунку 3.4 наведено результати аналізу сторінки Yahoo mail. React.js безперечно майбутнє SPA, особливо в поєднанні з архітектурою Flux. Yahoo! використовує Node.js протягом останніх кількох років, насправді початок 2010 - це приблизно той час, коли інженери Yahoo! почали грати з Node.js, задовго до того, як він став вважатися крутим мейнстримом або дійсно використовувався в якомусь висококласному масштабному середовищі.

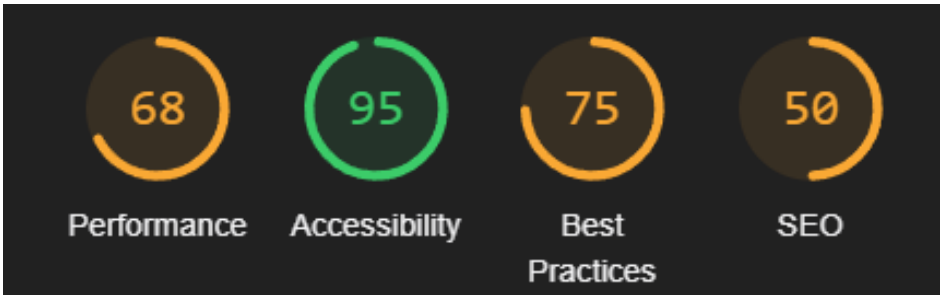

Рисунок 3.4 – Чотири показники Lighthouse аналізу Yahoo Mail

Performance – продуктивність. Аналізує швидкість завантаження сайту. На цю оцінку впливає час блокування, відтворення стилів, завантаження інтерактивних елементів, шрифтів та контенту. Progressive Web App -

Прогресивні web-додатки. Перевіряє, чи реєструє сайт Service Workers, чи працює офлайн, повертає помилку 200. Best Practices – найкращі практики. Перевіряє безпеку сайту та використання сучасних стандартів веброзробки. Оцінка залежить від того, чи використовується на сайті HTTPS, застарілі API, правильне кодування та інші параметри. Accessibility – доступність. перевіряє, чи всі користувачі можуть отримувати доступ до контенту і ефективно переміщатися по сайту. Ця оцінка залежить від зрозумілості та сприймання контенту, можливості керувати інтерфейсом та пересуватися вмістом без допомоги миші. SEO - оцінює відповідність сторінки порадам Google з пошукової оптимізації. Тут перевіряється використання метатегів, доступ до індексації та обходу роботами, наявність атрибутів alt у зображень, адаптованість до мобільних екранів та інші характеристики. Кожен параметр оцінюється за 100-бальною шкалою: що вище, то краще. Кожна група оцінок також має свій колір. Зелений виставляється за 90-100 балів, він показує, що з сайтом все добре. Помаранчевий можна отримати за 50-89 балів. Тобто із сайтом все добре, але можна зробити ще краще. Якщо оцінка нижче за 49 балів, вона стає червоною. Це означає, що над продуктивністю варто попрацювати.

У Yahoo! відкрито заявили про використання React у таких великих масштабах, тому можна очікувати, що в майбутньому він стане ще більш популярним. На рисунку 3.5 показані основні метрики продуктивності, що відображаються у Lighthouse Scoring Calculator. Розшифровка смислу кожного показника наведена в розділі 2.5.2.

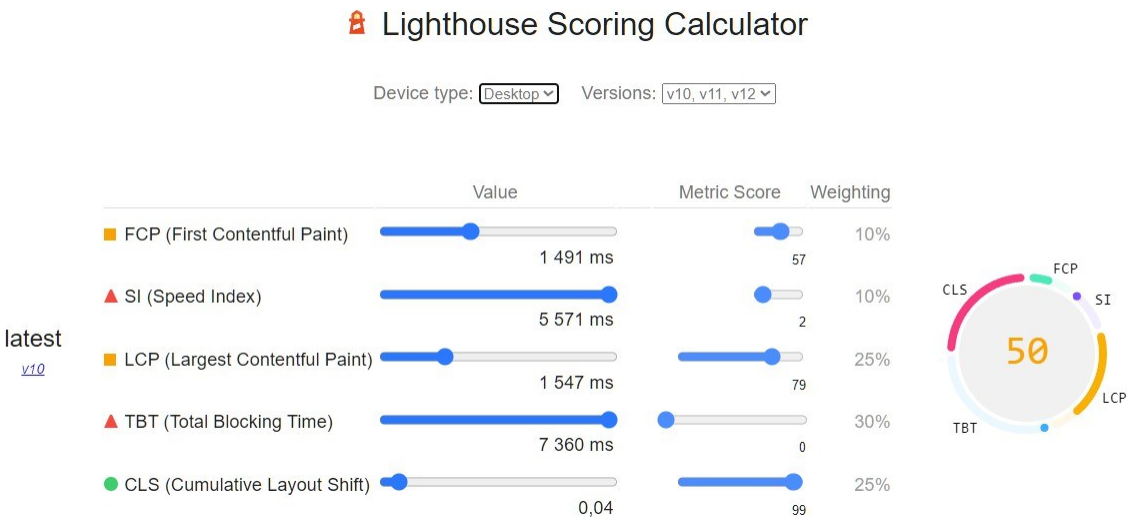

Рисунок 3.5 – Показники продуктивності в Lighthouse сайту Yahoo Mail

Таким чином, цих інструментів достатньо для отримання релевантних показників пов'язаних з роботою вебдодатку. Природно, якщо не акцентувати увагу на роботі React, необхідно задіяти весь спектр інструментів Google Analytics.

3.2 Базова конфігурація вебпроєкту для оптимізації

Перш ніж розглянути конкретні проєкти для перевірки ефективності базових конфігурацій, у цьому розділі буде детально досліджено популярність React.js та Next.js на основі доступних даних із різних джерел, що використовуються розробниками у щоденному процесі розробки. Вибрано Stack Overflow. Дані з цього джерела дають деяке уявлення про популярність фреймворків.

Stack Overflow – це онлайн-інструмент, де розробники можуть ставити запитання, пов’язані з програмуванням, відповідати на запити інших людей та знаходити рішення проблем, з якими вони стикаються під час програмування. Розробник повинен додавати теги, щоб допомогти іншим користувачам зрозуміти запит під час публікації запитання. Якщо задана відповідь вирішує проблему користувача, той, хто ставить запитання, може вибрати цю відповідь як прийнятну на цей запит. Будь-який користувач платформи може голосувати

за запити та рішення. Позитивні голоси називаються голосами «за», а негативні – голосами «проти», і вони вказують на те, наскільки корисними були запитання та відповідь для користувачів [40].

Stack Overflow — гарний спосіб виміряти популярність цих фреймворків. Щомісячно Stack Overflow відвідують понад 100 мільйонів людей. Процент питань, заданих щодо React.js і Next.js, показано на рисунку 3.6. З порівняння популярності видно, що більша увага приділяється проєктам React. Це торкається і проблем, пов'язаних з оптимізацією. Це не дивно, оскільки Next більш оптимізований як по продуктивному, так і з питань SEO.

Щоб оцінити різницю в продуктивності та швидкісному навантажувальному тесті між фреймворками, у кожному фреймворку побудовано простий клієнтський додаток для обчислень. Обидва додатки мають однаковий інтерфейс користувача, бізнес-логіку та дизайн.

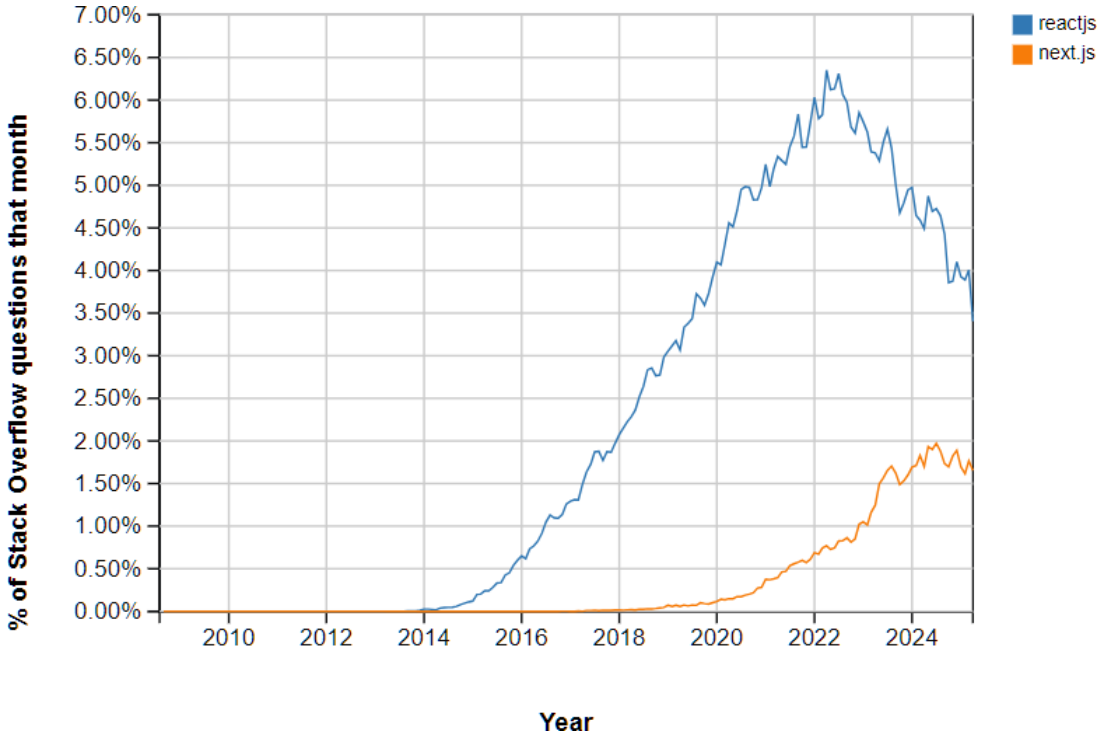

Рисунок 3.6 – Відсоток запитань, поставлених на Stack Overflow щодо React.js та Next.js з 2010 по 2025 рік [48]

Хоча Next.js є фреймворком React.js, налаштування застосунку в цих двох фреймворках відрізняється. У цьому проєкті було використано такі методи. Для налаштування фронтенду React.js проєкту використовується `npx create-react-app` та `npx create-next-app` для застосунку Next.js. У випадку застосунку Next.js, ім'я застосунку з'являється після введення вищевказаної команди. На відміну від Next.js, застосунок React.js вимагав окремого налаштування бекенду. З цієї причини в проєкті React.js сервер Node.js було створено на кореневому рівні в окремій папці під назвою бекенд. Команда npm init використовується для налаштування бекенду. Окрім ініціалізації Node.js у застосунку React.js, також встановлюється Express. Структура папок та файлів застосунків React.js та Next.js показана на рисунку 3.7. Ліворуч на рисунку 3.7 зображено структуру застосунку React.js, тоді як праворуч – структуру папок застосунку Next.js. Створення повноцінного вебзастосунку в React.js вимагає кількох налаштувань. З іншого боку, Next.js має повні можливості для створення повноцінного застосунку. Застосунок React.js поділено на фронтенд та бекенд частини.

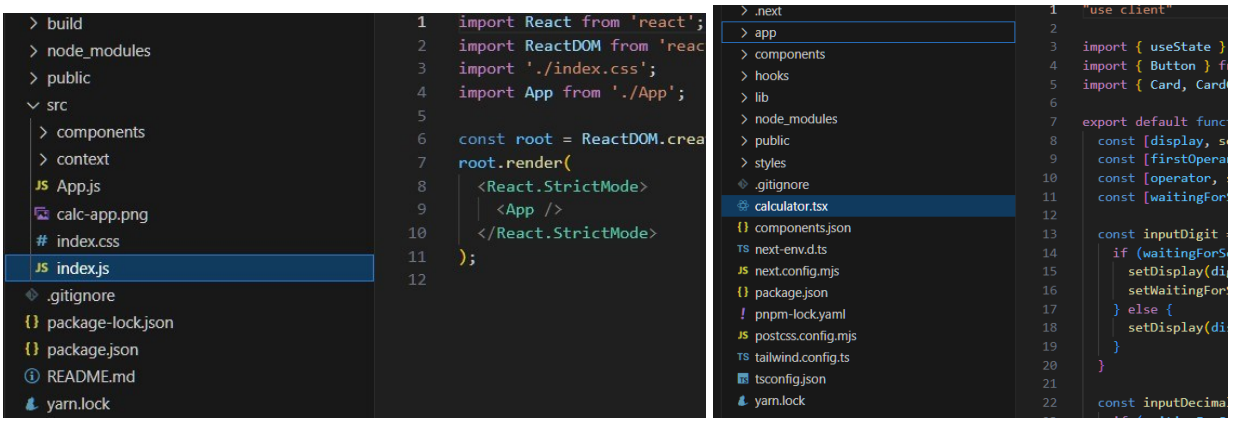

Рисунок 3.7 – Структура папок React.js ліворуч та Next.js-додатків праворуч

Стилізація та створення зручного інтерфейсу користувача для застосунку React.js виконується за допомогою react-bootstrap. React-Bootstrap — це інструментарій, який створює нативні елементи Bootstrap у чистому React.js. Навпаки, Material UI використовується для стилізації та побудови інтерфейсу

користувача застосунку Next.js. Material-UI [49] — це інструмент з компонентами React.js, що відповідає підходам Material Design від Google.

3.3 Впровадження оптимізації продуктивності

У сучасних вебдодатках продуктивність має вирішальне значення для забезпечення безперебійного досвіду користувача. Як було зазначено в попередніх розділах, React надає вбудований Profiler, який допомагає розробникам вимірювати та оптимізувати продуктивність рендерингу. У цьому розділі розглядається, як ефективно використовувати React Profiler та основні методи оптимізації продуктивності для розроблених програм. Якщо для "чистого" React використання Profiler відбувається органічно, то для проєкту в Next необхідне подальше налаштування.

Як було зазначено, React Profiler – це компонент, який допомагає виявляти вузькі місця продуктивності, вимірюючи час рендерингу для різних частин дерева компонентів React. Він збирає таку інформацію, як:

• Тривалість рендерингу: скільки часу потрібно компоненту для рендерингу

• Фази фіксації: час, потрібний для оновлення DOM

• Взаємодія: які зміни стану викликали повторні рендеринги

На першому етапі необхідно скоротити непотрібні повторні рендери, які суттєво уповільнюють роботу програм. Для цього використовуються відомі способи мемоїзації React: React.memo() для функціональних компонентів, useMemo() для дорогих обчислень, useCallback() для стабільних посилань на функції. Для оптимізації керування станами виключено непотрібні оновлення. Для цього використовується локальний стан, де це можливо. Для пакетного оновлення стану за допомогою useState або useReducer використовуються для програм ефективні бібліотеки станів, такі як, наприклад, Zustand або Recoil. Щоб віруалізувати списки, рендеринг яких може бути дорогим, застосовуються такі бібліотеки, як react-window та react-virtualized.

Використання Profiler у програмі React так:

Лістинг 3.1 – Підключення Profiler

```
import { Profiler } from "react";

function onRenderCallback(
  id,
  phase,
  actualDuration,
  baseDuration,
  startTime,
  commitTime,
  interactions
) {
  console.log(`Component ${id} rendered in ${actualDuration}ms.`);
}

function MyApp() {
  return (
    <Profiler id="MyComponent" onRender={onRenderCallback}>
      <MyComponent />
    </Profiler>
  );
}
```

Щоб мінімізувати маніпуляції з DOM, які суттєво знижують продуктивність, використовується об'єднання оновлень у пакети за допомогою паралельного режиму React та useTransition().

Таким чином, для вивчення продуктивності додатків, створених на React, інструмент Profiler дозволяє отримати в процесі запису сеансу практично всі показники, які використовуються для подальшої оптимізації (див. рис. 3.3). На основі цих показників можна оцінити ефективність змін, зроблених у коді. Звичайно, якщо слідувати ефективної стратегії оптимізації.

Тем не менш, Profiler стикається зі значними труднощами при використанні в проєктах Next, який працює на серверній стороні. Частично це пов'язано з тим, що аналізуються інші показники, якщо зрівняти з React. З іншої сторони, Next - фреймворк, побудований на базі React, тому зі своїми завданнями він хоче вирішити проблеми, які не вирішуються в початковій бібліотеці. Тому Next.js швидко став основним фреймворком для створення потужних, масштабованих вебдодатків. Його здатність обробляти рендеринг на

стороні сервера, генерацію статичних сайтів та маршрутизацію API робить його ідеальним вибором для великомасштабних проєктів. На рисунку 3.8 зображено сторінку Profiler для великомасштабного проєкту Next.js.

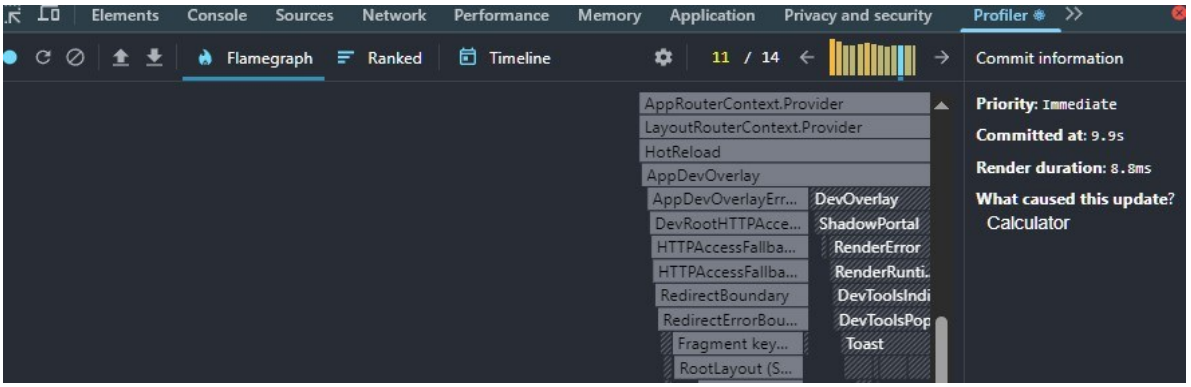

Рисунок 3.8 – Profiler у проєкті Next.js

Для оптимального аналізу слід дотримуватися таких правил: необхідно запустити Next.js в режимі розробки (npm run dev), найкраще використовувати React Developer Tools Profiler у браузері і лише після цього записати взаємодії та аналізувати шаблони рендерингу. Деякі подальші кроки проводяться так само, як і для оптимізації React, крім тих, які стосуються серверної сторони. Для оптимізації продуктивності щодо рендерингу на стороні сервера та стороні клієнта використовується SSR (getServerSideProps) для попередньої візуалізації сторінок з великим обсягом даних. Статична генерація (getStaticProps) використовується для сторінок, яким не потрібні часті оновлення, а відкладене завантаження (next/dynamic) лише завантаження компонентів лише за необхідності. Для більш глибокого аналізу сторінок Next.js використовується проміжне Next.js для скорочення надлишкових викликів API, а в деяких випадках реалізується паралельний режим, де це можливо.

Зазначимо, що завдяки інтеграції React Profiler з цими методами оптимізації тісний додаток Next.js запрацював більш плавно та ефективно.

3.4 SEO-оптимізація вебдодатку

React неймовірно потужний і гнучкий, але коли справа доходить до SSR, він відходить на другий план. Саме тут Next.JS яскраво сяє, пропонуючи SSR «з коробки», що є значним стимулом SEO. Справді, результати порівняння тестових показників у Chrome DevTools Lighthouse, які наводяться у наступному розділі, показують справедливість цього твердження.

Основна увага React ніколи не була зосереджена на SEO; він був розроблений як потужна бібліотека для створення користувацьких інтерфейсів. Хоча він чудово виконує цю роботу, коли справа доходить до SEO, він залишає бажати кращого. Відсутність рендерингу на стороні сервера в React робить його менш SEO-дружнім, оскільки пошуковим роботам важче сканувати та індексувати контент, що відображається на стороні клієнта. З іншого боку, Next.JS був створений з урахуванням SEO. Він забезпечує безперебійну функцію рендерингу на стороні сервера, яка гарантує повне відображення вмісту вашого вебсайту, перш ніж він потрапить до користувача, що полегшує сканування та індексацію вашого сайту пошуковими роботами, тим самим покращуючи рейтинг сайту в SEO. Це революційний процес у сучасному ландшафті веброзробки, де SEO є ключовим елементом онлайн-видимості та трафіку.

Ключові рецепти SEO для React.js багато в чому такі ж, як і для оптимізації продуктивності. А саме, ліниве завантаження та оптимізація часу завантаження. Для SEO ефективним є управління метаданими з використанням бібліотек, таких як react-helmet, для динамічного управління метатегами. Приклад коду для оптимізації SEO у додатку React наведено у лістингу 3.2.

Лістинг 3.2 – Динамічні метатеги з React Helmet Async

```
import { Helmet } from 'react-helmet-async';

const SEO = ({ title, description }) => (
  <Helmet>
    <title>{title}</title>
    <meta name="description" content={description} />
```

Продовження лістингу 3.2

```
        </Helmet>
      );
export default SEO;
```

Далі використовується компонент (див. листинг 3.3) для динамічного оновлення метатегів.

Лістинг 3.3 – Динамічні метатеги SSR з Next.js

```
import Head from 'next/head';
const Home = () => (
  <>
    <Head>
      <title>SEO Optimized Page</title>
        <meta name="description" content="This is an SEO-friendly
React page." />
    </Head>
    <h1>Welcome to My SEO Optimized Page</h1>
  </>
);
export default Home;
```

Використання плагіна Sitemap Generation сайту для Vite показує деяку ефективність для SEO (див. листинг 3.4)

Лістинг 3.4 – Динамічні метатеги SSR з Next.js

```
import { ViteSitemap } from 'vite-plugin-sitemap';
export default defineConfig({
  plugins: [
    ViteSitemap({
      baseUrl: 'https://yourdomain.com',
      routes: ['/home', '/about', '/contact'],
      generateRobotsTxt: true,
    }),
  ],
});
```

На жаль, робота з React-додатками додає до вже довгого списку проблем, які має перевірити технічний SEO-спеціаліст. Але завдяки таким фреймворкам, як Next.js, це робить роботу SEO-спеціаліста набагато простішою, ніж вона була раніше [50].

3.5 Аналіз результатів оптимізації

Хоча React Developer Tools зазвичай використовуються для налагодження та налаштування продуктивності, вони також можуть служити цінним ресурсом для аналізу даних, допомагаючи розробникам відстежувати тенденції, оптимізувати поведінку рендерингу та покращувати користувацький досвід.

Для оцінки продуктивності створено демонстраційні додатки, що містять набір даних. Дані надходять з API фіктивного постачальника даних. Продуктивність додатка аналізується шляхом взаємодії з інтерфейсом користувача, зокрема шляхом реалізації подій, пов'язаних з пошуком та кнопками.

Як пояснювалося в розділах теорії, ресурсномісткі обчислення в будь-якому сценарії мають вплив на продуктивність. Крім того, зміна стану батьківського компонента призводить до повторного рендерингу всіх його дочірніх компонентів і може призвести до уповільнення роботи програми, якщо компонент, що рендериться, є ресурсномістким.

Аналіз даних у розробці на React зосереджується на оцінці поведінки додатка, показників продуктивності та моделей взаємодії з користувачем. Представлено результати продуктивності впровадження useMemo, useCallback та лінивого завантаження для обробки повільних компонентів зі складними операціями та зменшення часу рендерингу та завантаження програми, як обговорювалося в цьому дослідженні. Знімки екрана результатів, надані Profiler, ілюструють кількість часу, витраченого компонентами на рендеринг, та причину рендерингу. Виконувані дії інтерфейсу користувача: пошук та видалення. Нижче наведено на рисунках 3.9 та 3.10 знімки екрана даних, наданих Profiler, що відображають продуктивність програми під час початкового рендерингу без оптимізації.

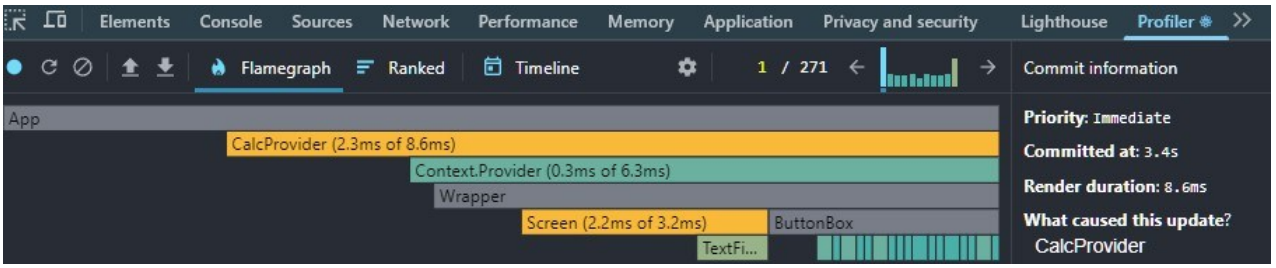

Рисунок 3.9 – Продуктивність програми під час рендерингу у проєкті React.js

На рисунку 3.10 показано ранжований вигляд Profiler, що вказує час, необхідний для рендерингу компонентів. Ця інформація чітко відображає покращення часу рендерингу. В розділі 2.2.2. були розглянуті програмні способи оптимізації продуктивності (лістинг 2.1 і 2.2). Далі відзначено, що для проєкту React такі методи практично не дають вкладу в підвищення продуктивності. Це перш за все пов'язано з невеликим розміром додатків. При цьому практично неможливо оцінити вклади в параметри продуктивності, які пов'язані з можливим масштабуванням. На вкладці Perfomance тестовий проєкт показав час LCP 0.78 с, який був класифікований як хороший, і відповідно час INP 40 мс.

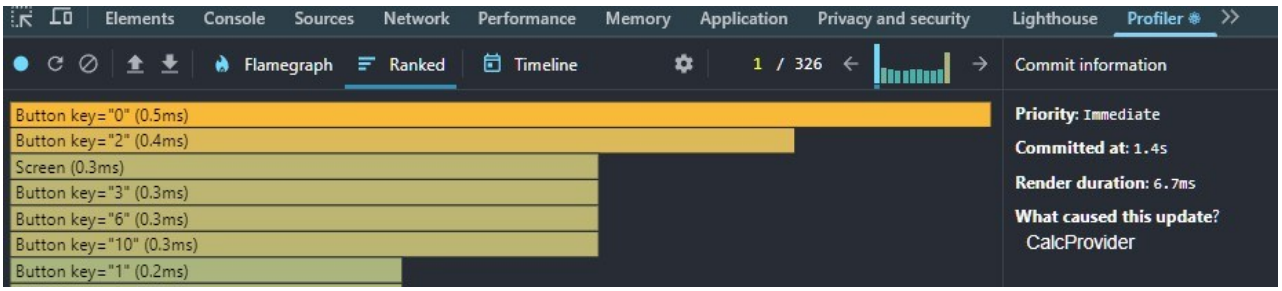

Рисунок 3.10 – Графік, що відображає причину відображення компонента

Показники в Lighthouse Scoring Calculator показано на рисунку 3.11.

Рисунок 3.11 – Показники Lighthouse у проєкті на React

На рисунку показано, що метрика FCP, яка вимірює час, необхідний для відображення першої частини вмісту DOM після переходу на сторінку, дорівнює 1568 мс, та займає 10% від всього часу. Далі SI – 2760 мс, що показує, як швидко контент візуально відображається на сторінці., він вимірюється шляхом захоплення відео завантаження сторінок.

Метрика LCP, яка вимірює час відображення самого великого зображення або текстового блоку в області перегляду, дорівнює 4425 мс. Слід зазначити, що це одна з найважливіших нових метрик. Тут хороший показник означає, що користувачі сприймають сайт як такий, що швидко завантажується. Для тестового проєкту цей показник незадовільний, програмні методи оптимізації не приводять к поліпшенню. Сумарний показник продуктивності дорівнює 60, а SEO, звичайно - 100, оскільки проєкт розгорнутий на локальному сервері [51].

Щоб оцінити можливості для більш глобальних проєктів, проводиться порівняння показників для програми та сайту https://www.pinterest.com/, в якому повністю задіяні React технології. Основні метрики зростають пропорційно до розміру проєкту: від 9000 до 15000 мс. Проте показник продуктивності падає до значення 26. Це показує, що у проєктах на React при масштабуванні ефективні показники значно падають. Тим не менш, SEO знову демонструє 100, що пов'язано з ефективності серверної частини на Next. Відомо, що yahoo mail також використовує React. Тим не менш, незважаючи на свій обсяг, значення метрик часу значно менше: від 3000 до 5000 мс, продуктивність - 65. Тим не

менш, SEO падає до 66. Звідси випливає, що хороша продуктивність вебдодатків може призводити до падіння SEO показників.

Для того щоб розглянути можливості проєктів у відношенні масштабування, проаналізовані за допомогою Profiler можливості багатовіконної програми. Приклад показників представлено на рисунку 3.12.

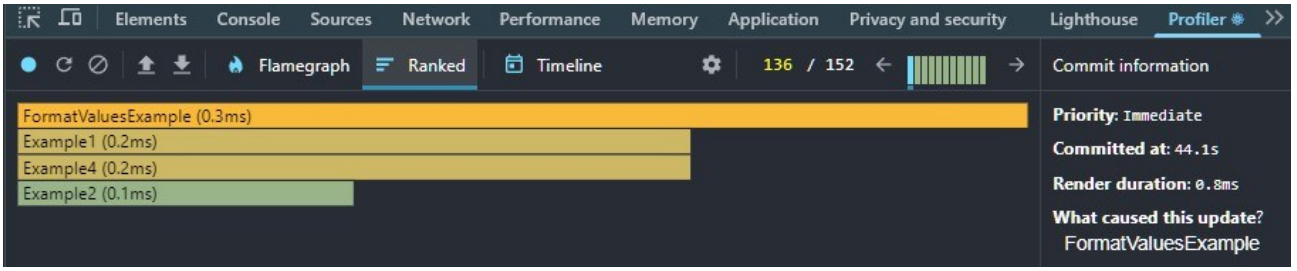

Рисунок 3.12 – Приклад показників Profiler у багатовіконної програми на React

Коли кількість вікон значно збільшується, показники падають навіть якщо інфромація вводиться в кілька вікон. Це пов'язано з тим, що рендеринг все одно проводиться по всіх вікнах, тому проста модернізація відгуку тільки на подію введення значно покращує показники продуктивності. Тим не менш, продуктивність та SEO багатокінного додатку значно падає, як показано на рисунку 3.13.

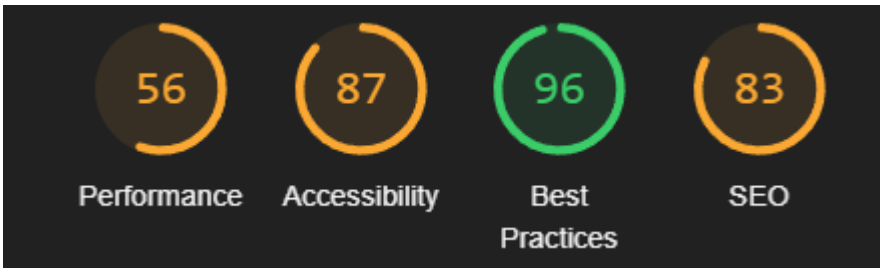

Рисунок 3.13 – Показники Lighthouse у багатовіконної програми на React

Використання з метою покращення SEO програмного засобу з лістингу 3.2 не призводить до значного покращення показників. Очевидно, це пов'язано з тим, що позитивний внесок від використання динамічних метатегів з React Helmet Async компенсується негативними наслідками від завантаження Helmet.

Природно, при масштабуванні цього не відбувається, тому показники SEO зростають, але незначно, оскільки програма не така велика, як, наприклад, pinterest.com.

React Developer Tools – це важливий інструмент для налагодження та перевірки компонентів React у застосунку Next.js. Вимірювання продуктивності вашого застосунку Next.js має вирішальне значення для оптимізації користувацького досвіду, рейтингу SEO та загальної ефективності. Загалом, існують додаткові метрики в проєктах Next.js, які забезпечують більш ретельне відстеження. Це час гідратації, яке вимірює, скільки часу потрібно, щоб сторінка стала інтерактивною. До цього додається час до першого байта (TTFB, Time to First Byte) – час, необхідний серверу для надсилання першого байта даних. Поряд із цим, існують ще додаткові інструменти для аналізу. Серед них може виділити Hook Next.js useReportWebVitals, який збирає дані Web Vitals. І аналітика Vercel, що пропонує автоматичне відстеження продуктивності [52].

Для аналізу використовується програма така ж, як і для React, але тільки використовує Next.js технологію. Програмні способи оптимізації та SEO засновані на кодах з лістингів 3.3 та 3.4, а також на підходах розроблених у розділі 2.

Таким чином, після запуску програми в режимі розробки (`npm run dev`) можна використовувати можливості React Developer Tools, як показано на рисунку 3.14.

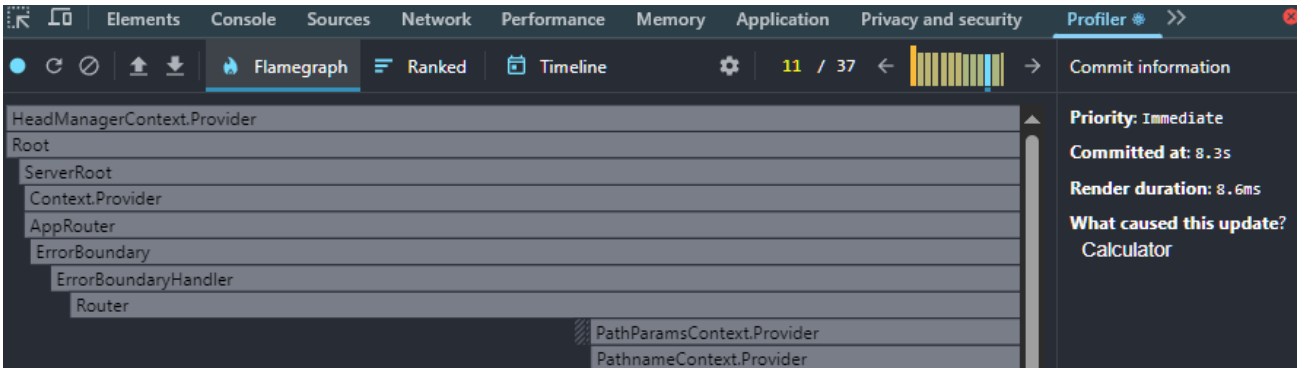

Рисунок 3.14 – Приклад показників Profiler у Next.js додатку

Програма запускається за 3.8 с. Очевидно, що в порівнянні з продуктом React цей час значно більше. Проэкт знаходиться за адресою для Local: http://localhost:3000, або Network: http://192.168.0.102:3000. Можна відзначити, що показник LCP дорівнює 5.29, що є досить поганим показником.

Далі проведено аналіз показників під час запису проєкту. Іншими словами, час виконання операцій, виходячи з якого неможливо оцінити ефективність оптимізації. Справді, існує дуже багато показників при тестуванні, які змінюються під час кожного завантаження. Тим не менш, після цього можна подивитися ще додатковий показник INP (рис. 3.15).

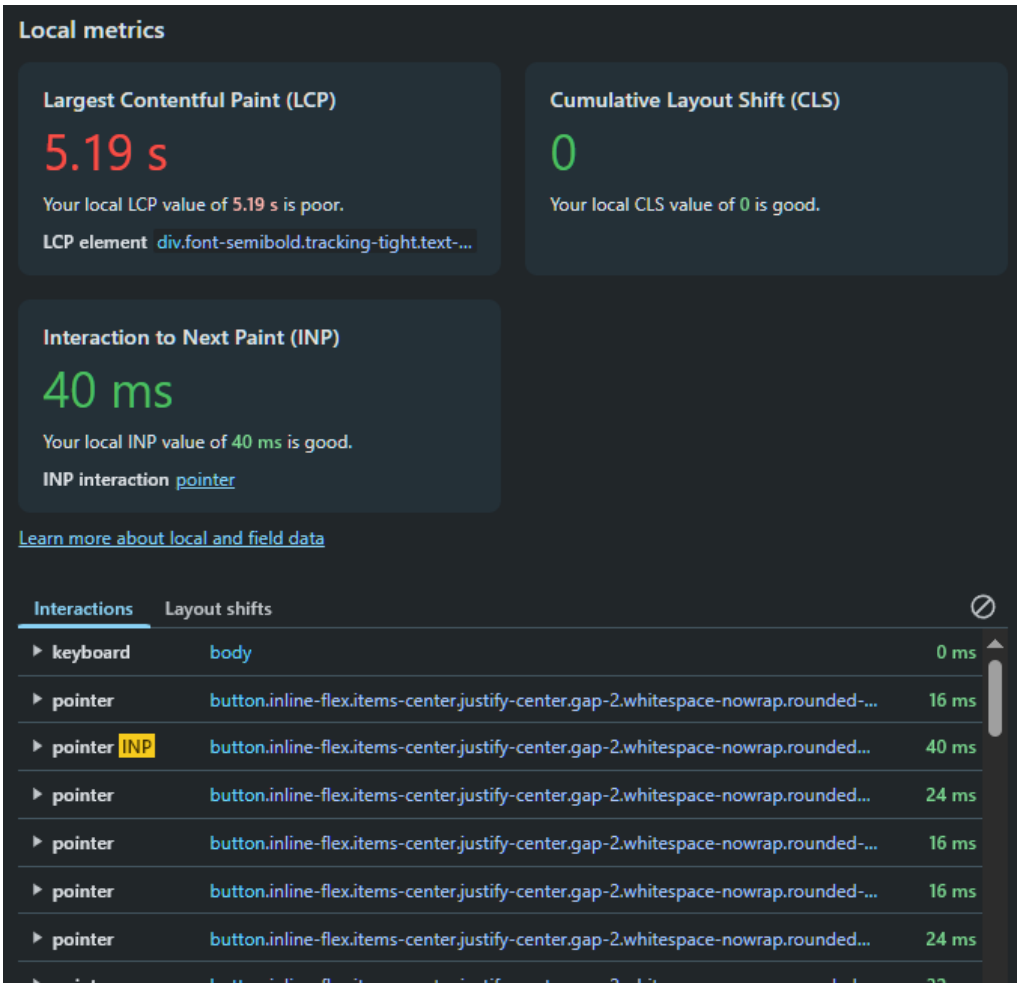

Рисунок 3.15 – Локальні метрики показників Profiler у Next.js додатку

Lighthouse - це потужний інструмент для аудиту та покращення продуктивності, доступності та SEO вебдодатків, включаючи ті, які створені за допомогою Next.js. Він запускає серію аудитів за наданою URL-адресою і

створює звіт, виділяючи області для покращення. Якщо використовується Next.js, доцільно інтегрувати Lighthouse у робочий процес для вимірювання основних вебпоказників та відповідної оптимізації вашого сайту. Рекомендується запускати Lighthouse в режимі інкогніто, щоб запобігти впливу сторонніх плагінів на результати. Крім того, використання таких інструментів, як Next.js Bundle Analyzer, може допомогти скоротити JavaScript, що не використовується, що є поширеною проблемою, зазначеною Lighthouse (рис. 3.16).

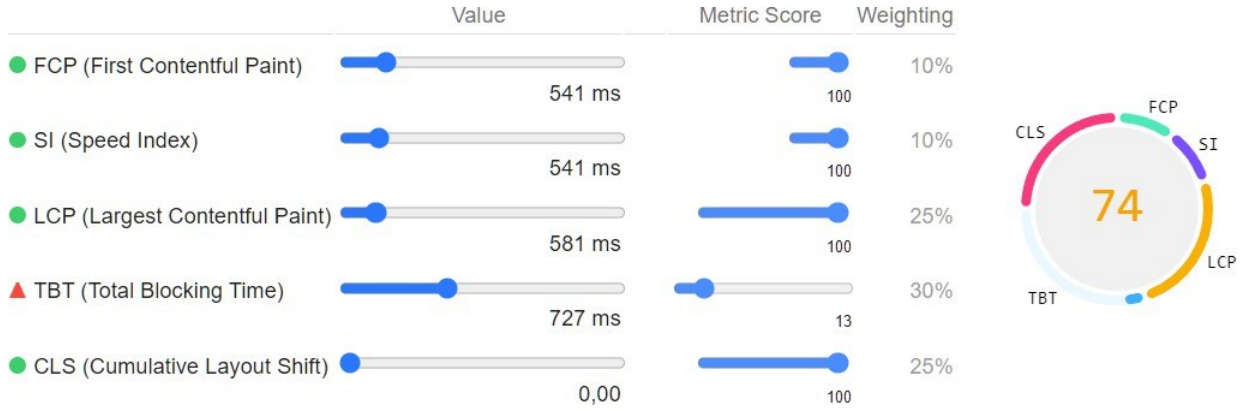

Рисунок 3.16 – Показники Lighthouse у проєкті на Next

Методи оптимізації пов'язані з такими конкретними діями. По-перше, для покращення часу завантаження були використані методи SSR або SSG. Далі перевірено дерево компонентів, налагоджено рендеринг як на стороні клієнта, так і на стороні сервера за допомогою посібника з налагодження Next.js. Але в основному використано програмні пропозиції щодо оптимізації, розглянуті у другому розділі та у лістингах 3.3 та 3.4.

Після модифікації коду час завантаження зменшився з 3.4 до 3.2 секунди. Цей показник не показує ефективність оптимізації, оскільки змінюється в залежності від сеансу. Інший показник «[Fast Refresh] done in 358ms» навіть трохи збільшився, якщо порівнювати з кодом до оптимізації.

Після такої оптимізації практично всі ключові метрики змінилися на краще (рис. 3.17). Тим не менш, зміни важко пов'язати з ефективності процесу оптимізації, оскільки сам проєкт досить маленький, з іншого боку,

масштабування просто проводиться клонуванням існуючих компонентів. Звичайно, для масштабних проєктів навіть такі зміни є дуже значущими.

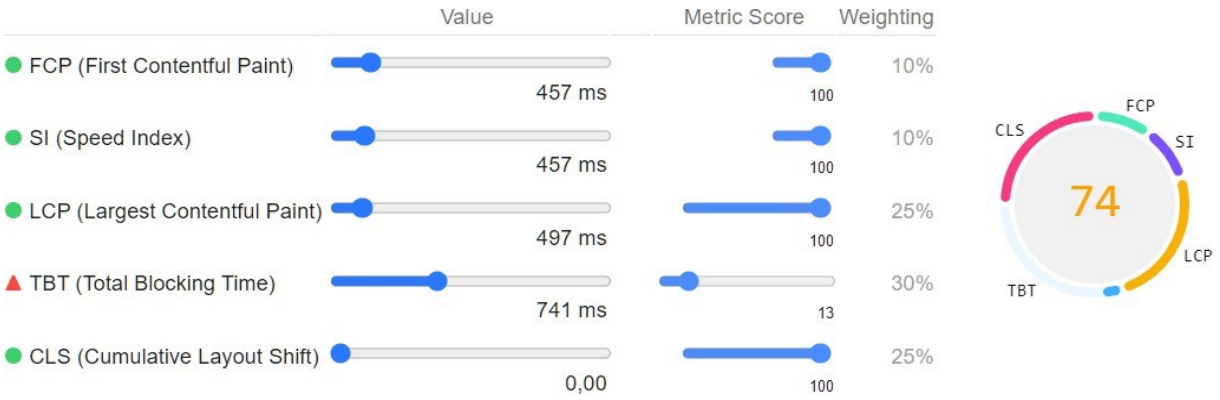

Рисунок 3.17 – Показники Lighthouse у оптимізованому проєкті на Next

У звіті Lighthouse також представлені деякі можливості для покращення швидкості роботи мобільного сайту, зокрема, скільки часу вони економлять. До них належать зменшення кількості таблиць стилів, що блокують рендеринг, скриптів, що блокують рендеринг, правильне визначення розміру зображень та виправлення зображень поза екраном.

3.6 Висновок до третього розділу

Проведено експериментальне дослідження спрямоване на прогнозування та визначення ефектів одного змінного експерименту шляхом керування кількома факторами, які можуть привести до зміни результатів. Головна ціль — використовувати експериментальний метод дослідження для вивчення широко використовуваних методів SEO для покращення індексації динамічних вебсторінок в основних пошукових системах. Другорядна ціль — порівняти методи SEO і побачити ефективність одного методу в порівнянні з іншими.

У експериментальному дослідженні важливо вибрати правильний суб'єкт, оскільки це безпосередньо пов'язано з узагальненням результатів, отриманих під час експерименту. Запропоновано, що результати реєструються з пошукових

систем після застосування певних методів SEO; отже, ці результати можна використовувати для узагальнення ефективності методів SEO для індексації динамічних вебсайтів. Дизайн експерименту вибирається після розуміння рекомендацій та інтегрованих кроків, які використовуються для вибору відповідного дизайну експерименту, і враховуються ці загрози валідності, які можуть вплинути на дизайн дослідження цього експерименту.

Далі було проведено вибір відповідних інструментів для дослідження та відповідного технічного середовища. По-перше, це React Developer Tool – це важливе розширення для браузера. Воно надає набір потужних функцій, які допомагають налагоджувати, перевіряти та оптимізувати React-додатки. Профілювальник React Developer Tools надає інтерактивний метод для визначення причин виникнення певної проблеми продуктивності. По-друге, Lighthouse — це автоматизований інструмент із відкритим вихідним кодом для вимірювання якості вебсторінок, розроблений Google.

Далі була запропонована базова конфігурація вебпроєкту для оптимізації. Подано порівняльний опис ефективності методів SEO оптимізації для проєктів React та Next. Проведено порівняльний аналіз, розроблено відповідні програми. На розроблених проєктах проведено експериментальне дослідження. Досліджено, фактично, один і той же проєкт, але тільки в React і в Next імплементаціях. Було проведено експерименти, отримано значення відповідних метрик. Аналіз даних у розробці на React зосереджено на оцінці поведінки додатка, показників продуктивності та ефективності взаємодії з користувачем. Представлено результати продуктивності впровадження алгоритмів useMemo, useCallback та лінивого завантаження для обробки повільних компонентів зі складними операціями для зменшення часу рендерингу та завантаження програми. Визначено, що для проєкту React такі методи практично не дають вкладу в підвищення продуктивності. Це перш за все пов'язано з невеликим розміром додатків. Але тестовий проєкт показав час LCP 0.78 с, який був класифікований як гарний, і відповідно час INP 40 мс.

Щоб оцінити можливості для більш глобальних проєктів, було проведено порівняльний аналіз показників для програми та сайтів, в яких повністю задіяно React технології. Так, для pinterest.com основні метрики зростають пропорційно до розміру проєкту: від 9000 до 15000 мс. Проте показник продуктивності падає до значення 26. Це показує, що у проєктах на React при масштабуванні ефективні показники значно падають. Тим не менш, SEO знову демонструє 100, що пов'язано з ефективності серверної частини на Next. Але для yahoo mail значення метрик часу значно менше: від 3000 до 5000 мс, продуктивність - 65. Тим не менш, SEO падає до 66. Звідси випливає, що гарна продуктивність вебдодатків може призводити до падіння SEO показників.

Далі для порівняльного аналізу використовується програма із застосуванням Next.js технології. Додаток запускається за 3.8 с., що в порівнянні з додатком React є гіршим показником. Можна відзначити, що показник LCP дорівнює 5.29, що є також поганим показником. За допомогою Lighthouse були протестовані та отримані релевантні показники. Для покращення часу завантаження були використані методи SSR або SSG. Далі перевірено дерево компонентів, налагоджено рендеринг як на стороні клієнта, так і на стороні сервера за допомогою посібника з налагодження Next.js. Але в основному використано програмні пропозиції щодо оптимізації. Після модифікації коду час завантаження зменшився з 3.4 до 3.2 секунди. Часи FCP та SI зменшилися з 541 до 457 мс. Важливий показник LCP значно зменшився після оптимізації. Але продуктивність та час TBT практично не змінилися.

Таким чином, доведено ефективність запропонованих методів оптимізації та SEO на невеликих додатках. Доведено, що для вивчення масштабованості треба використовувати проєкти із клонуванням деяких компонентів. Стверджується, що ефективність розроблених методів можна підтвердити тільки на масштабних проєктах, оскільки позитивні зміни показників після модернізації зовсім незначні.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Охорона праці в ІТ-компаніях має свої особливості, адже більшість працівників працюють в офісах або дистанційно, що створює специфічні виклики для забезпечення безпеки та здоров'я. Тому, від самого початку розробки потрібно враховувати нормативно-правові вимоги у сфері охорони праці, аби підсумкове рішення відповідало діючим державним стандартам та забезпечувало захист здоров’я працівників.

4.1 Фактори що впливають на функціональний стан користувачів комп’ютерів

У процесі роботи за комп’ютером організм людини піддається впливу низки факторів, що можуть призвести до зниження працездатності, втоми, розвитку хронічних захворювань. Знання та аналіз таких факторів дозволяє ефективно організувати робоче середовище та знизити професійні ризики (табл. 4.1).

Функціональний стан користувачів комп’ютерів значною мірою залежить від умов праці, організації робочого місця та психофізіологічних навантажень. Вчасне виявлення негативних факторів і впровадження профілактичних заходів дозволяє не лише зберегти здоров’я працівників, а й підвищити їхню ефективність та стійкість до професійних навантажень [54].

У контексті охорони праці користувачів комп’ютерної техніки доцільно звернути увагу на додаткові рекомендації міжнародних організацій охорони здоров’я, профілактичні методи підтримання здоров’я під час роботи за ПК, а також сучасні практики, що впроваджуються в ІТ-компаніях.

Згідно з рекомендаціями Всесвітньої організації охорони здоров’я (ВООЗ) та Міністерства охорони здоров’я України, тривалість безперервної роботи за комп’ютером не повинна перевищувати 2 годин поспіль [55]. Для запобігання перенавантаженню організму фахівці рекомендують робити короткі перерви

тривалістю 5–10 хвилин щогодини, що дозволяє уникати зорового та м’язового перевантаження, а також зменшує ризик виникнення професійних захворювань.

Таблиця 4.1 - Фактори, що впливають на функціональний стан користувачів комп’ютерів

№	Група факторів	Вплив на функціональний стан	Заходи профілактики
1	Зорове навантаження	Синдром комп’ютерного зору, погіршення гостроти зору	Дотримання правил 20-20-20, правильне освітлення, відстань до монітора 60–70 см
2	М’язово-опорно-рухове навантаження	Біль у шиї, попереку, ризик остеохондрозу	Ергономічне крісло, правильна постава, зміна пози кожні 40–60 хвилин
3	Монотонність діяльності	Втрата концентрації, психологічна втома	Варіювання задач, мікроперерви, зміна типу активності
4	Психоемоційне перенапруження	Стрес, емоційне вигорання, дратівливість	Психологічна підтримка, гнучкий графік, зони відпочинку
5	Порушення режиму праці	Перевтома, зниження продуктивності, порушення сну	Регулярні перерви, дотримання робочого графіку, автоматичні нагадування
6	Невідповідність мікроклімату	Головний біль, сухість слизових, підвищена втома	Температура 22–24°C, вологість 40–60%, провітрювання, зволожувачі
7	Недостатнє освітлення/відблиски	Швидка втома очей, головний біль	Регульоване освітлення, відсутність відблисків, антивідблискові екрани
8	Електромагнітні поля	Довготривалий вплив може мати кумулятивний ефект	Сертифіковане обладнання, заземлення, відповідність нормам електробезпеки
9	Шум та вібрація	Дратівливість, зниження уваги	Акустичне облаштування офісу, використання навушників або шумоізоляційних матеріалів
10	Надмірне інформаційне навантаження	Когнітивне перенавантаження, зниження швидкості прийняття рішень	Пріоритезація завдань, time-blocking, управління повідомленнями
11	Соціальна ізоляція	Втрата мотивації, депресивні настрої	Командна робота, регулярна комунікація, спільні перерви/активності

Одним із ефективних підходів до збереження здоров’я очей є правило 20-20-20: кожні 20 хвилин слід на 20 секунд відволікатися та дивитися на об’єкт, розташований на відстані не менше 20 футів (приблизно 6 метрів). Такий метод знижує навантаження на акомодаційний апарат ока [56]. Також

корисними є вправи для очей та шийно-комірцевої зони, що сприяють покращенню кровообігу та зменшенню напруги. Важливою складовою профілактики є тренінги з тайм-менеджменту, які дозволяють раціонально розподіляти час роботи, уникати надмірного інформаційного навантаження та запобігати емоційному вигоранню [57].

Сучасні IT-компанії активно використовують програмні нагадування про перерви, такі як Stretchly або EyeLeo, які автоматично сигналізують про необхідність зробити паузу в роботі. Фізичне робоче середовище також адаптується під потреби працівників: застосовуються standing desk (столи, за якими можна працювати стоячи), ергономічні клавіатури та миші з вертикальним хватом, що знижують ризик розвитку тунельного синдрому.

В багатьох сучасних організаціях реалізується концепція корпоративного добробуту (well-being), яка передбачає не лише безпечні умови праці, а й створення сприятливого психоемоційного клімату. Такі компанії впроваджують програми підтримки ментального здоров’я, організовують фізичну активність, надають доступ до здорового харчування, а також забезпечують гнучкі графіки роботи, що сприяє балансу між професійним і особистим життям [58].

Варто окремо розглянути поширене порушення — синдром комп’ютерного зору , який виникає внаслідок тривалої роботи за монітором.

Причинами синдрому комп’ютерного зору найчастіше є надмірна яскравість монітора, блискуча поверхня екрана, невірно підібрана відстань або кут огляду. Для зменшення впливу даного синдрому необхідно ретельно підбирати параметри робочого місця та використовувати захисні фільтри або монітори з антибліковим покриттям [59].

Тому, дотримання рекомендацій міжнародних організацій, використання адаптивного обладнання в сфері IT дозволяють значно знизити ризики для здоров’я користувачів , забезпечуючи ефективну та безпечну професійну діяльність.

У наш час важливо не лише фізичне, а й цифрове здоров’я користувача. Постійна присутність в інформаційному просторі, робота з великою кількістю

повідомлень, чатів, інтерфейсів призводить до явища, яке у психології позначають як "цифрова втома" або "інформаційне перевантаження". Це явище супроводжується розсіюванням уваги, тривожністю, зниженням мотивації та навіть професійним вигоранням. Для протидії цьому у багатьох компаніях запроваджується поняття цифрової гігієни, що включає обмеження часу, проведеного в онлайні, чергування роботи за ПК з офлайн-активностями, відключення сповіщень під час фокусної роботи (deep work) та використання софту для блокування зайвих відволікань [60].

Крім того, варто формувати корпоративну культуру безпеки, яка передбачає добровільну участь працівників у тренінгах і навчаннях, створення позитивної мотивації для дотримання правил, прозору комунікацію між керівництвом і персоналом щодо ризиків та визнання значущості кожного працівника як носія знань, відповідальності й ініціативи.

Таким чином, функціональний стан користувачів комп’ютерної техніки значною мірою залежить від організації умов праці, рівня психофізіологічного навантаження та дотримання гігієнічних норм. Виявлення та мінімізація шкідливих факторів дозволяють не лише зберегти здоров’я працівників, а й підвищити ефективність їхньої роботи. Сучасні підходи до охорони праці — від ергономіки робочого місця до практик цифрової гігієни — повинні інтегруватися у всі рівні організації, особливо в умовах інтенсивного використання комп’ютерних технологій.

4.2 Питання щодо охорони праці

На етапі розробки методів та алгоритмів оптимізації продуктивності та SEO в сучасних вебдодатках, створених на основі React.js та Next.js надзвичайно важливо забезпечити такі умови праці, за яких користувачі системи матимуть безпечне та продуктивне робоче середовище. У випадку даної кваліфікаційної роботи йдеться про покращення, яке дозволяє здійснювати розробки методів оптимізації продуктивності, які повинні

відповідати вимогам сучасних додатків та забезпечувати високий рівень коду та продуктивності.

Питання охорони праці та безпеки в надзвичайних ситуаціях у сучасній ІТ-сфері набувають усе більшої актуальності в контексті зростання цифрових навантажень, а також нових викликів, пов’язаних із кіберзагрозами, воєнними діями та техногенними ризиками. У рамках четвертого розділу було всебічно проаналізовано сукупність умов, вимог, ризиків і рішень, які формують безпечне середовище функціонування ІТ-фахівців.

Перш за все, слід наголосити, що охорона праці в ІТ-компаніях має свої специфічні риси, пов’язані не з фізичним виробництвом, а з інтелектуальною працею, інтенсивним використанням комп’ютерної техніки та тривалою роботою в статичному положенні.

Цілі охорони праці в ІТ-компаніях спрямовані на забезпечення комфортних та продуктивних умов роботи для співробітників [51]. Даний розділ включає визначення цілей і питань щодо охорони праці та безпеки в надзвичайних ситуаціях.

Крім того, врахування протипожежних та електробезпекових норм є критично важливим для запобігання нещасним випадкам, пов’язаним із використанням електрообладнання, систем живлення та інформаційно-комунікаційної інфраструктури. Для забезпечення безпечних і комфортних умов роботи варто врахувати основні положення «Правил охорони праці під час експлуатації електронно-обчислювальних машин» та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Ці нормативні документи визначають вимоги до приміщень, робочих місць, освітлення, шуму, вібрації, мікроклімату та ергономіки.[54]

У приміщенні необхідно забезпечити регулярне вологе прибирання для підтримання чистоти та зменшення накопичення пилу, який може негативно впливати на роботу електронного обладнання. Додатково, приміщення мають бути обладнані аптечками першої медичної допомоги. Для зменшення рівня

втоми працівників варто організувати кімнати відпочинку або зони психологічного розвантаження, де співробітники зможуть зробити перерву під час роботи.

Приміщення потрібно бути оснащене порошковим переносним вогнегасником відповідно до основних вимог пожежної безпеки приміщення, у яких знаходиться персональний комп'ютер. Підходи до засобів пожежогасіння є вільними.

У великих ІТ-компаніях поступово впроваджується стандартизована система управління охороною праці, що дозволяє поєднувати питання безпеки, охорони здоров'я та організаційної ефективності

Сучасні інформаційні технології відкривають нові можливості для цифровізації процесів охорони праці, тому уже сьогодні активно використовуються:

− смарт-системи моніторингу мікроклімату, які регулюють температуру, вологість і CO_2;

− системи розпізнавання облич для контролю доступу до критичних зон (серверних);

− аналітичні панелі (дашборди) для оцінки стану безпеки в реальному часі;

− віртуальні симулятори навчання охороні праці та НС, що дозволяють моделювати евакуацію, гасіння пожежі або ситуації кібератаки в інтерактивній формі [58].

У найближчому майбутньому набуде поширення застосування штучного інтелекту для прогнозування ризиків на основі поведінки користувача. Це дозволить не тільки попередити проблему, а й своєчасно вжити заходів — від простого нагадування до звернення до HR-служби.

Комплексний підхід до питань охорони праці та безпеки в надзвичайних ситуаціях не обмежується лише виконанням формальних вимог. У сучасному менеджменті все більшого значення набуває система управління ризиками, яка базується на принципах прогнозування, профілактики, безперервного моніторингу й постійного вдосконалення. У сфері ІТ це означає: регулярну

оцінку ймовірних загроз (наприклад, перегрів обладнання, відмова серверів, DDoS-атаки, нестача резервного живлення тощо). Побудову карт ризиків для різних сценаріїв надзвичайних подій, впровадження превентивних політик безпеки, що дозволяють уникати повторення критичних інцидентів та створення реєстрів інцидентів та аналітичних звітів, які слугують основою для навчання персоналу та вдосконалення протоколів.

4.3 Питання щодо безпеки в надзвичайних ситуаціях

Забезпечення пожежної безпеки в IT-компаніях має свої особливості, адже більшість працівників працюють з електронним обладнанням, яке може стати джерелом займання. Основні заходи для запобігання пожежам включають:

− Виявлення потенційних джерел займання, таких як перегрів електронних пристроїв або неправильне використання електропроводки.

− Регулярне технічне обслуговування обладнання, правильне зберігання легкозаймистих матеріалів та дотримання правил пожежної безпеки.

− Проведення інструктажів щодо дій у разі пожежі, ознайомлення з планом евакуації та використанням вогнегасників.

− Встановлення детекторів диму, тепла та полум'я для швидкого реагування на загрозу.

Для запобігання виникненню пожежі під час роботи з комп'ютерним обладнанням та допоміжними пристроями, слід дотримуватися наступних вимог:

− Використання негорючих або важкозаймистих матеріалів у робочому просторі;

− Зберігання горючих речовин у мінімально необхідній кількості, організація їх правильного розташування;

− Регулярне проведення провітрювання приміщення для підтримки концентрації горючих газів, пари, пилу та окисника поза межами пожежнонебезпечних значень;

− Використання справного електрообладнання, що відповідає Правилам улаштування електроустановок;

− Проведення регулярних оглядів електричних кабелів та пристроїв на предмет пошкоджень;

− Обладнання робочих місць засобами аварійного відключення живлення для запобігання перегріву або короткого замикання;

− Організація системи захисту від блискавок та електростатичного розряду, що є важливим для приміщень, у яких розміщується комп'ютерна техніка [61].

Усі робочі місця повинні бути оснащені системами протипожежного захисту та сигналізації, що забезпечує раннє виявлення загоряння та своєчасне реагування на надзвичайну ситуацію.

Окрім організаційних і технічних заходів пожежної безпеки, значну роль відіграє навчання працівників основам протипожежної безпеки та алгоритму дій у разі надзвичайних ситуацій, а саме організація інструктивної роботи, яка включає проведення вступного інструктажу, періодичні навчання та тренування з евакуації. Особливу увагу варто приділити ознайомленню з розташуванням засобів пожежогасіння, аварійних виходів та планів евакуації.

Під час воєнних загроз важливе значення має захист місць розташування серверного обладнання та забезпечення фізичного доступу до нього лише уповноваженому персоналу. За рекомендаціями експертів із цивільної безпеки, слід здійснювати періодичний аудит захищеності приміщень та дотримуватись вимог фортифікації (наприклад, зміцнення будівель, наявність укриттів, систем відеоспостереження). Додатково потрібен комплексний моніторинг стану мережевих ресурсів і систем розробки, щоб оперативно виявляти ознаки несанкціонованого доступу або кібератак. Іншим аспектом безпеки є підготовка користувачів. Вони повинні знати інструкції щодо дій під час сигналу тривоги, розуміти план евакуації, володіти базовими знаннями надання першої домедичної допомоги у разі поранень. Організація навчальних тренувань, інструктажів і поширення інформаційних матеріалів сприяє зниженню ризиків та мінімізації втрат [62].

У сфері інформаційних технологій безпека в надзвичайних ситуаціях охоплює ширший спектр ризиків, ніж лише пожежна небезпека. Сучасні загрози включають кіберінциденти, воєнні дії, техногенні аварії, відмову інфраструктури, витік даних або аварійне знеструмлення. У зв'язку з цим заходи безпеки мають бути комплексними, превентивними та адаптованими до специфіки діяльності ІТ-компаній.

1. Забезпечення пожежної безпеки. Пожежна безпека є однією з ключових складових. У приміщеннях, де розміщено комп'ютерне обладнання, основними ризиками є коротке замикання, перегрів пристроїв, використання несправної електропроводки або порушення правил зберігання легкозаймистих речовин. До ефективних превентивних заходів належать [63]:

− оцінка пожежних ризиків і щорічна ревізія технічного стану обладнання;

− обладнання приміщень детекторами диму, системами автоматичного оповіщення та аварійного освітлення;

− наявність доступних вогнегасників, аварійних кнопок відключення живлення, засобів блискавкозахисту;

− регулярне навчання персоналу, у тому числі тренування з евакуації.

2. Дії у випадку воєнної загрози. В умовах воєнного стану, характерного для України останніми роками, важливими є:

− встановлення чітких інструкцій дій у разі повітряної тривоги (зупинка роботи, збереження даних, переміщення до укриття);

− наявність укриттів або визначення найближчих безпечних місць;

− ознайомлення персоналу з маршрутами евакуації, маркування виходів, створення «тривожних наборів» (аптечки, документи, резервні носії, павербанки).

Важливо також забезпечити фізичну безпеку серверних приміщень, захист від вибухових хвиль, уламків та води (у разі гасіння пожежі чи прориву трубопроводів).

3. План безперервності діяльності. Для збереження критичних ІТ-сервісів у надзвичайних умовах компанії мають впроваджувати план безперервності бізнес-процесів, який включає:

− резервне копіювання даних (як локальне, так і хмарне);

− автоматичне перемикання на резервні канали зв’язку або сервери;

− забезпечення автономного електроживлення (дизель-генератори, джерела безперебійного живлення);

− план аварійного відновлення роботи.

4. Протидія кібератакам і цифровий захист. Кібератаки можуть супроводжуватися фізичними наслідками: блокування доступу до систем, шифрування даних, збій інфраструктури, саме тому важливо:

− застосовувати двофакторну автентифікацію, складні паролі, оновлене антивірусне ПЗ;

− впровадити системи виявлення вторгнень (IDS/IPS);

− обмежити доступ до критичних систем лише авторизованим користувачам;

− навчати персонал розпізнавати фішингові загрози та діяти відповідно до процедур кібербезпеки.

5. Перша домедична допомога, саме тому у приміщеннях компанії мають бути:

− аптечки з повним набором засобів для надання першої допомоги;

− персонал, навчений надавати першу допомогу у разі поранень, втрати свідомості, опіків;

− плакати або інструкції з алгоритмами дій, розміщені у видимих місцях.

6. Навчання персоналу та тренування, бо безпека персоналу напряму залежить від обізнаності з діями у надзвичайних ситуаціях. Саме тому необхідно регулярно проводити:

− вступний і періодичний інструктаж з охорони праці;

− евакуаційні тренування, у тому числі з імітацією реальних НС;

− розповсюдження інформаційних матеріалів (пам’яток, чек-листів, відеоінструкцій).

Таким чином, безпека в надзвичайних ситуаціях у сфері інформаційних технологій передбачає реалізацію багаторівневого підходу — від пожежної безпеки до кіберзахисту, від протидії воєнним загрозам до забезпечення безперервності бізнесу. Інтеграція організаційних, технічних і навчальних заходів забезпечує стійкість компанії до кризових ситуацій та створює умови для збереження життя, здоров’я працівників і критичної інфраструктури [65].

Окрім базових заходів протипожежної безпеки, в умовах воєнного стану, кіберактивності та частих інфраструктурних збоїв, питання безпеки в надзвичайних ситуаціях для IT-компаній потребують розширеного, міждисциплінарного підходу. Врахування новітніх викликів дозволяє не лише убезпечити персонал, а й забезпечити безперервність критичних бізнес-процесів.

1. Безпека при евакуації та збереження конфіденційної інформації, у разі раптової евакуації особливу увагу слід приділяти захисту корпоративної інформації:

− всі робочі місця мають бути обладнані функцією автоматичного блокування після 2–3 хвилин бездіяльності;

− у разі залишення робочого місця — застосовується шифрування даних та захист паролем;

− критично важливі сервери повинні мати режим швидкого збереження і завершення сесій із забезпеченням збереження стану системи.

2. Інклюзивна безпека для осіб з особливими потребами, безпека має бути доступною для всіх категорій працівників, тому евакуаційні маршрути повинні бути пристосовані для осіб з обмеженою мобільністю, у зоні укриття чи безпечного простору передбачаються спеціальні місця для інвалідів. Призначаються працівники-супроводжувачі, які забезпечують допомогу в евакуації та для осіб із вадами слуху або зору використовуються візуальні та голосові дублюючі сигнали.

3. Забезпечення умов для тривалого перебування в офісі чи укритті, у разі блокування евакуаційних шляхів або довготривалого перебування в офісі під час НС важливо: мати запас питної води, сухого пайка, аптечок, засобів особистої гігієни, забезпечити енергозабезпечення мобільних пристроїв (павербанки, інвертори) та використовувати портативні радіостанції або інтернет-незалежні джерела зв'язку.

4. Психологічна безпека та антистресові заходи, емоційна стабільність працівників напряму впливає на якість реагування в НС. Тому рекомендується організовувати психологічну підтримку, як очно, так і в онлайн-форматі. Створювати зони психологічного розвантаження, де можна усамітнитися чи перепочити та проводити антистресові тренінги з управління емоціями, особливо для команд лідерів і технічної підтримки.

5. Автоматизовані системи моніторингу та реагування, бо впровадження цифрових рішень значно підвищує оперативність реагування: системи відеоспостереження з розпізнаванням руху та автоматичним збереженням записів; розумні детектори пожежі, води, диму, які синхронізовані з внутрішніми сервісами оповіщення; внутрішні мобільні застосунки з push-сповіщеннями про НС, чек-ін системою, тривожними кнопками та GPS-локацією співробітників.

6. Визначення відповідальних осіб , де ефективність реагування залежить від чіткого розподілу ролей:

- особа, відповідальна за евакуацію;
- відповідальний за збереження даних і технічних систем;
- особа з навичками надання першої допомоги;
- координатор взаємодії з екстреними службами;
- особа, яка контролює облік присутніх у момент евакуації або укриття.

Для зручності матриця відповідальності може бути представлена у вигляді таблиці з ПІБ, посадою та коротким описом обов'язків, що зберігається в кожному кабінеті чи робочій зоні [65].

Отже, заходи безпеки в надзвичайних ситуаціях мають враховувати специфіку роботи в ІТ-секторі: наявність чутливої цифрової інформації, високотехнологічного обладнання та значного рівня емоційного навантаження. Інтеграція фізичної, цифрової, психологічної та організаційної безпеки дозволяє сформувати сталу систему реагування на будь-які виклики, зберегти персонал і забезпечити функціонування навіть в умовах форс-мажору.

4.3 Висновок до четвертого розділу

В рамках підрозділу було деталізовано ключові фактори, що впливають на функціональний стан користувачів комп’ютерів, серед яких — зорове та м’язове перенапруження, монотонність роботи, нестача рухової активності, порушення мікроклімату, погане освітлення, тривала дія електромагнітних полів, а також емоційне виснаження. Практичні заходи щодо мінімізації впливу цих факторів передбачають впровадження правил безперервної роботи (наприклад, 20-20-20), використання адаптивного обладнання, організацію зон відпочинку та доступ до психологічної підтримки.

Значну увагу приділено нормативно-правовому регулюванню охорони праці. Проведення вступних, первинних та періодичних інструктажів, регулярна атестація робочих місць, сертифікація обладнання та документальне забезпечення охорони праці є обов’язковими елементами функціонування будь-якої організації в ІТ-секторі.

Важливим напрямом є забезпечення пожежної безпеки, оскільки електронне обладнання в офісі може бути джерелом займання у разі порушення правил експлуатації. Було проаналізовано комплекс заходів з попередження пожеж: встановлення систем оповіщення, регулярне технічне обслуговування обладнання, наявність вогнегасників, створення умов для оперативної евакуації, а також застосування негорючих матеріалів у робочому просторі.

Особливої актуальності набуває безпека в умовах воєнного стану, яка включає фізичну захищеність серверного обладнання, наявність укриттів,

інструкції з евакуації та порядок дій у разі обстрілів чи повітряної тривоги. Працівники мають бути ознайомлені з планами евакуації, маршрутами до найближчих сховищ, основами домедичної допомоги. Забезпечення мінімального аварійного комплекту (вода, аптечка, павербанк, документи) є обов'язковою умовою адаптації компанії до кризових обставин.

Розглянуто також цифрову безпеку як невід'ємну складову безпеки IT-середовища. В умовах поширення кібератак, фішингу, зловмисного програмного забезпечення компанії повинні впроваджувати системи захисту доступу, багатофакторну автентифікацію, автоматичне блокування терміналів, шифрування даних, а також проводити навчання персоналу з інформаційної безпеки. Кіберінциденти можуть спричинити не лише втрату даних, а й порушення критичних бізнес-процесів, що підвищує значущість запровадження планів безперервності діяльності (BCP).

Значну увагу в розділі було приділено організаційним аспектам безпеки, включаючи визначення відповідальних осіб за окремі напрямки (евакуація, перша допомога, координація з рятувальними службами, цифровий захист). Запровадження матриці відповідальності, систем інформування та мобільних застосунків для оповіщення персоналу дозволяє суттєво скоротити час реагування у разі загрози.

У межах розділу також підкреслено важливість інклюзивного підходу до безпеки — створення умов евакуації для осіб з інвалідністю, адаптація сигналізації для людей із вадами слуху або зору, призначення асистентів. Безпека повинна охоплювати всіх працівників без винятку, що відповідає принципам сталого розвитку і соціальної відповідальності бізнесу.

Окремий акцент зроблено на психологічному аспекті безпеки. В умовах високої інтенсивності праці, дедлайнів, гібридної роботи та воєнної нестабільності працівники IT-компаній часто зазнають тривалого стресу, що може призвести до емоційного вигорання. Запровадження корпоративних програм well-being, доступ до психологів, створення кімнат відпочинку та

гнучких графіків роботи сприяють зниженню ризиків і формуванню здорового мікроклімату.

Таким чином, охорона праці та безпека в надзвичайних ситуаціях в IT-секторі повинні розглядатися як системний і міждисциплінарний процес, що об'єднує:

- технічні рішення (ергономіка, вентиляція, сигналізація, UPS);
- організаційні заходи (інструктажі, тренування, відповідальні особи);
- правові аспекти;
- цифрові інструменти (системи контролю, резервування, моніторинг);
- людський фактор (психологічна стійкість, обізнаність, навички допомоги).

Ігнорування навіть одного з цих компонентів створює додаткові ризики як для життя і здоров'я працівників, так і для безперервності операцій компанії.

Варто наголосити, що безпека в IT-компанії — це не лише зона відповідальності керівництва чи інженерних служб. Кожен працівник є активним учасником процесу формування безпечного середовища. Особистий рівень свідомості, дисципліна, готовність до навчання та допомоги іншим — критичні для ефективного функціонування всієї системи. Працівники мають знати інструкції з охорони праці й періодично проходити тестування на знання базових процедур та брати участь у навчаннях, що моделюють ситуації надзвичайного характеру. Виявляти ініціативу щодо вдосконалення безпекових протоколів та оперативно повідомляти про порушення або потенційні загрози (через внутрішні канали, анонімні форми тощо).

Безпека в IT-сфері — це не лише відповідність формальним нормам, а й культура, що ґрунтується на взаємній відповідальності, дотриманні етичних стандартів, технологічній обізнаності та адаптивності до змін. Її ефективність залежить від здатності поєднати технічні інструменти з гуманітарним підходом до людини — головного ресурсу сучасної цифрової економіки.

Висновки, викладені в даному розділі, мають прикладне значення й можуть бути використані для розробки внутрішніх політик безпеки

підприємств, покращення умов праці, оптимізації процедур реагування в НС, а також для формування корпоративної культури охорони праці.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр» описано основні принципи перегляду вебдодатків, розглянуто особливості та переваги підходу MVC (Model-View-Controller). Розглянуто переваги використання односторінкових додатків порівняно з багатосторінковими; розглядаються їх технологічні особливості. Проведено огляд сучасних методів спрямованих на покращення продуктивності сучасних вебдодатків, аналізуючи погляди до оптимізації як з боку загального підходу, так і із застосуванням методів пошукової оптимізації (SEO, search engine optimization). Показано, що Next.js, який побудовано на базі React, разом з іншими фреймворками, такими як Angular та Vue, змінили парадигму на більш функціональний підхід на основі використання концепції компонентів MVC для створення вебзастосунків. Тому у цьому компоненті представлення містить всю логіку (контролер) та дані (модель), необхідні для візуалізації в межах цього єдиного компонента. Обгрунтовано, що найкращим підходом до розробки вебдодатків є односторінкові застосунки (SPA, single-page application), що переносять до браузера прийоми роботи, які звичні для персональних додатків. Проаналізовано робочий процес SPA на основі видів рендерингу. Показано, що React.js спрощує створення красивого динамічного контенту, він все ще залишається односторінковим застосунком. Розглянуто наявні варіанти та останні тенденції, пов'язані з покращенням продуктивності вебдодатків. Для отримання метрик для порівняльного аналізу проєктів на основі React та Next запропоновано використовувати безкоштовні вебплатформи для аналізу та оптимізації вебсайтів з точки зору рейтингу пошукових систем.

В другому розділі кваліфікаційної роботи досліджено ефективність Next.js як фреймворку, що вирішує проблеми, що виникають через React.js, зокрема, в продуктивності, SEO та рівноправній вебдоступності. На основі розгляду структурних особливостей та можливостей імплементації проєктів на JavaScript викладається опис бібліотеки React. Описано огляд сучасних бібліотек та

фреймворків. Досліджено різні методи та найкращі практики оптимізації продуктивності React-додатків, які допомагають розробникам створювати швидкий та адаптивний UI. Розглянуто фреймворк Next від організації проєктів до шляхів оптимізації. Проведено порівняльний аналіз інструментів аналізу ефективності вебпроєктів. Показано способи моніторингу та аналізу SEO-показників Next.js. Описано основні метрики оцінки продуктивності та SEO, які потрібно отримувати при порівняльному аналізі проєктів з використанням React і Next.

В третьому розділі кваліфікаційної роботи визначено цілі, фази планування експерименту, процес проведення експерименту і зразок експерименту. Також запропоновано ідеї експерименту, пов'язані з плануванням процесу виконанням експерименту крок за кроком. Запропоновано методи та найкращі практики оптимізації продуктивності React-додатків, які допомагають розробникам створювати швидкий та адаптивний UI. Проведено порівняльний аналіз інструментів аналізу ефективності вебпроєктів. Представлено результати продуктивності впровадження алгоритмів useMemo, useCallback та лінивого завантаження для обробки повільних компонентів зі складними операціями для зменшення часу рендерингу та завантаження програми. Визначено, що для проєкту React такі методи практично не дають вкладу в підвищення продуктивності. Це перш за все пов'язано з невеликим розміром додатків. Але тестовий проєкт показав час LCP 0.78 с, який був класифікований як гарний, і відповідно час INP 40 мс. Для більш масштабованих додатків доведено, що гарна продуктивність вебдодатків може призводити до падіння SEO показників. Показано способи моніторингу та аналізу SEO-показників Next.js. Перевірено дерево компонентів, налагоджено рендеринг як на стороні клієнта, так і на стороні сервера, використано програмні пропозиції щодо оптимізації. Доведено, ці дії призвели до значного поліпшення основних показників оптимізації та SEO. Проаналізовано експериментальні дані щодо основних метрик оцінки продуктивності та SEO, які було отримано при порівняльному аналізі тестових проєктів з використанням React і Next. Доведено, що для вивчення

масштабованості треба використовувати проєкти із клонуванням деяких компонентів. Стверджується, що ефективність розроблених методів можна підтвердити тільки на масштабних проєктах, оскільки позитивні зміни показників після модернізації зовсім незначні. Доведено, що програмні рішення, запропоновані у цій роботі, щодо оптимізації продуктивності призводять до позитивних змін основних показників.

В четвертому розділі кваліфікаційної роботи розглянуто питання охорони праці та безпеки в надзвичайних ситуаціях у сучасній ІТ-сфері, які набувають усе більшої актуальності в контексті зростання цифрових навантажень, поширення гібридних форм праці, а також нових викликів, пов’язаних із кіберзагрозами, воєнними діями та техногенними ризиками. У рамках четвертого розділу було всебічно проаналізовано сукупність умов, вимог, ризиків і рішень, які формують безпечне середовище функціонування ІТ-фахівців.

Таким чином, запропоновані в цій роботі основні методи оптимізації продуктивності та SEO при побудові проєктів на React та Next мають важливе значення для проєктування та реалізації вебдодатків.

ПЕРЕЛІК ДЖЕРЕЛ

1. Berners-Lee, T. (1999). *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Harper San Francisco.
2. Understanding the Science Behind a High-Performing Web Application Architecture [Електронний ресурс] https://softteco.com/blog/web-application-archtecture-explained.
3. Fowler, M. (2012). *Patterns of enterprise application architecture*. Addison-Wesley.
4. Selfa, D. M., Carrillo, M., & Boone, M. D. R. (2006, March). A database and web application based on MVC architecture. In *16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06)* (pp. 48-48). IEEE.
5. How the Model View Controller Architecture Works – MVC Explained [Електронний ресурс] https://www.freecodecamp.org/news/model-view-architecture
6. Mikowski, M., & Powell, J. (2013). *Single page web applications: JavaScript end-to-end*. Simon and Schuster.
7. What is Single Page Application? 26 Apr, 2024 [Електронний ресурс] https://www.geeksforgeeks.org/what-is-single-page-application/
8. How to Create a React Single Page Application : [Електронний ресурс] https://programmers.io/blog/react-single-page-application/
9. Chinnathambi, K. (2018). *Learning React: a hands-on guide to building web applications using React and Redux*. Addison-Wesley Professional.
10. How to build single-page applications with Next.js [Електронний ресурс] https: //nextjs.org/docs/app/guides/single-page-applications
11. Welcome to the Next.js documentation! [Електронний ресурс] https://nextjs.org/docs
12. Vogel, L., & Springer, T. (2023, April). How streaming can improve the world (wide web). In *Companion Proceedings of the ACM Web Conference 2023* (pp. 140-143).

13. Pan, Y., White, J., Sun, Y., & Gray, J. (2015, May). Gray computing: An analysis of computing with background javascript tasks. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 1, pp. 167-177). IEEE.

14. Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2024). Overview of Web Application Performance Optimization Techniques. *arXiv preprint arXiv:2412.07892*.

15. Shethiya, A. S. (2025). Scalability and Performance Optimization in Web Application Development. *Integrated Journal of Science and Technology*, *2*(1).

16. Kleppmann, M., Wiggins, A., Van Hardenberg, P., & McGranaghan, M. (2019, October). Local-first software: you own your data, in spite of the cloud. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (pp. 154-178).

17. Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2023). The state of disappearing frameworks in 2023. *arXiv preprint arXiv:2309.04188*.

18. How to minify a next js app · Issue #3632. [Електронний ресурс] https://github.com/vercel/next.js/is sues/3632 (2018), [Accessed 18-02-2025]

19. Options: compress next.config.js. [Електронний ресурс] https://nextjs.org/docs/app/api-reference/next-config-js/compress (2024), [Accessed 8-03-2025]

20. Sellamuthu, K., Ranjithkumar, S., Kavitha, K., & Gowtham, S. (2022, April). On Page SEO techniques for better ranking in search engines. In *2022 8th International Conference on Smart Structures and Systems (ICSSS)* (pp. 01-06). IEEE.

21. Eslamdoust, A. (2022). An overview of search engine optimization techniques. *Future Generation of Communication and Internet of Things*, *1*(4), 52-57.

22. Van Looy, A. (2022). Search engine optimization. In *Social Media Management: Using Social Media as a Business Instrument* (pp. 125-146). Cham: Springer International Publishing.

23. Erdmann, A., Arilla, R., & Ponzoa, J. M. (2022). Search engine optimization: The long-term strategy of keyword choice. *Journal of Business Research*, *144*, 650-662.

24. Kowalczyk, K., & Szandala, T. (2024). Enhancing SEO in single-page web applications in contrast with multi-page applications. *IEEE Access*, *12*, 11597-11614.

25. Joshi, M. A., & Patel, P. (2018, December). Google page rank algorithm and it's updates. In *International Conference on Emerging Trends in Science, Engineering and Management, ICETSEM-2018*.

26. Roumeliotis, K. I., & Tselikas, N. D. (2022). An effective SEO techniques and technologies guide-map. *Journal of web engineering*, *21*(5), 1603-1649.

27. All-In-One SEO Software & Tools [Електронний ресурс] https://www.seobility.net/en/ [Accessed 2-04-2025]

28. PageSpeed Insights This site uses cookies from Google to deliver its services and to analyze traffic [Електронний ресурс] https://pagespeed.web.dev/ [Accessed 13-04-2025]

29. Run JavaScript Everywhere [Електронний ресурс] https://nodejs.org/en

30. Home • Angular [Електронний ресурс] https://angular.dev/ [Accessed 16-04-2025]

31. Vue.js - The Progressive JavaScript Framework [Електронний ресурс] https://vuejs.org/ [Accessed 23-04-2025]

32. Express - Node.js web application framework [Електронний ресурс] https://expressjs.com/ [Accessed 24-04-2025]

33. React Native · Learn once, write anywhere [Електронний ресурс] https://reactnative.dev/ [Accessed 25-04-2025]

34. Redux/ A JS library for predictable and maintainable global state management [Електронний ресурс] https://redux.js.org/ [Accessed 27-04-2025]

35. Create React App [Електронний ресурс] https://create-react-app.dev/ [Accessed 23-03-2025]

36. Запускайте JavaScript будь-де [Електронний ресурс] https://nodejs.org/uk [Accessed 3-04-2025]

37. React Developer Tools. Use React Developer Tools to inspect React components, edit props and state, and identify performance problems. [Електронний ресурс] https://react.dev/learn/react-developer-tools [Accessed 18-03-2025]

38. Chrome DevTools | Chrome for Developers [Електронний ресурс] https://developer.chrome.com/docs/devtools [Accessed 3-04-2025]

39. Дмитроца Л.П., Мушинська Г. Аналітика оптимізації чат-бота Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології "ТНТУ ім. Івана Пулюя. 2022. С.35

40. Особливості цифрового розвитку малого і середнього бізнесу України, країн Європи та G7 / Струтинська І. В. та ін. // Колективна монографія, Тернопіль. 2024. С. 411–427.

41. Дмитроца Л.П., Старицький О.Т. Оптимізація продуктивності та SEO при масштабуванні проєктів на основі React і Next.js Режим доступу: https://tntu.edu.ua/storage/pages/00001070/TNTU_Aktualni_zadachi_suchasnyh_tehnologiy_2024.pdf [дата звернення 01.03.2025]

42. Lighthouse /Introduction to Lighthouse [Електронний ресурс] https://developer.chrome.com/docs/lighthouse/overview/ [Accessed 8-04-2025]

43. Introducing the React Profiler [Електронний ресурс] https://legacy.reactjs.org/blog/2018/09/10/introducing-the-react-profiler.html [Accessed 17-04-2025]

44. Kozbur H. Numerical prediction of the strength of a thin-walled pipe loaded with internal pressure and axial tension taking into account its actual dimensions / Halyna Kozbur, Oleh Shkodzinsky, Lesia Dmytrotsa // Scientific Journal of TNTU. — Tern. : TNTU, 2020. — Vol 4. — No 100. — P. 11–19.

45. Start learning about Google Analytics [Електронний ресурс] https://developers.google.com/analytics [Accessed 11-04-2025]

46. What is Search Console? Why and how to use? [Електронний ресурс] https://seo.london/search-console/ [Accessed 23-04-2025]

47. SEO Toolkit [Електронний ресурс] https://www.semrush.com/seo/ [Accessed 30-04-2025]

48. Tag Trends Javascript Frameworks [Електронний ресурс] https://trends.stackoverflow.co/?tags=reactjs%2Cnext.js [Accessed 30-04-2025]

49. Ready to use Material Design components [Електронний ресурс] https://mui.com/material-ui/ [Accessed 30-04-2025]

50. Next.JS or React? Which is Better in SEO Optimization? [Електронний ресурс] https://www.codewalnut.com/learn/is-next-js-better-in-seo-optimization [Accessed 30-04-2025]

51. Iryna Strutynska, Lesia Dmytrotsa, Halyna Kozbur, Liliya Melnyk, Roman Sherstiuk. The Unification of Approaches to Measuring the Digital Maturity of Business Structures. ICTERI-2021, Vol I: Main Conference, PhD Symposium, Posters and Demonstrations, September 28 – October 2, 2021, Kherson, Ukraine, pp. CEUR Workshop Proceedings 3013, pp. 10-23. (ISSN: 1613-0073) https://ceur-ws.org/Vol 3013/20210010.pdf.

52. Iryna Strutynska, Lesia Dmytrotsa, Halyna Kozbur, Liliya Melnyk. The Digital Business Transformation Index Determining and Monitoring: Development of a National Online Platform. ITTAP'2021: 1nd International Workshop on Information Technologies: Theoretical and Applied Problems, November 16–18, 2021, Ternopil, Ukraine. CEUR Workshop Proceedings 3039, pp. 327-334 CEUR Workshop Proceedings (CEUR-WS.org) (ISSN: 1613-0073) https://ceur-ws.org/Vol 3039/short33.pdf

53. Стручок В.С. Техноекологія та цивільна безпека. Частина «Цивільна безпека»: навчальний посібник. Тернопіль: ФОП Паляниця В. А., 156 с. [Електронний ресурс] http://elartu.tntu.edu.ua/handle/lib/39424 [дата звернення 20.04.2025]

54. ВООЗ. *Ergonomic Guidelines for Healthy Computer Use*. WHO, 2020.

55. МОЗ України. *Рекомендації щодо організації праці користувачів ПК*. Київ, 2021.

56. American Optometric Association. *Computer Vision Syndrome* [Електронний ресурс]. https://www.aoa.org.

57. Covey, S. (2020). *The 7 Habits of Highly Effective People*. FranklinCovey.

58. Deloitte Insights. *2022 Global Human Capital Trends* [Електронний ресурс]. https://www2.deloitte.com.

59. Sheedy, J. E. (2005). *Computer Vision Syndrome: Evaluation and Treatment*. *J Behav Optom*.

60. Newport, C. (2016). *Deep Work: Rules for Focused Success in a Distracted World*. Grand Central Publishing.

61. Державні будівельні норми України. ДБН В.2.5-56:2014 «Системи протипожежного захисту» [Електронний ресурс]. https://online.budstandart.com/ua/catalog/doc-page?id_doc=59583.

62. National Institute of Standards and Technology (NIST). Contingency Planning Guide for Federal Information Systems. Special Publication 800-34 Rev. 1 [Електронний ресурс]. https://csrc.nist.gov/publications/detail/sp/800-34/rev-1/final

63. CERT-UA. Рекомендації з кіберзахисту в умовах воєнного стану [Електронний ресурс]. https://cert.gov.ua/article/3755359.

64. Міністерство охорони здоров'я України. Настанови з надання домедичної допомоги [Електронний ресурс]. https://moz.gov.ua/article/recommendation/nastanova-z-nadannja-domedichnoi-dopomogi.

65. Державні будівельні норми України. ДБН В.2.5-56:2014 «Системи протипожежного захисту» [Електронний ресурс]. https://online.budstandart.com/ua/catalog/doc-page?id_doc=59583.

ДОДАТКИ

Додаток А

Публікація в науковому виданні

Матеріали XIII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 11-12 грудня 2024 року

УДК 621.855

Л.П. Дмитроца, канд.техн.наук, доц.; О.Т. Старицький
(Тернопільський національний технічний університет)

ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ ТА SEO ПРИ МАСШТАБУВАННІ ПРОЄКТІВ НА ОСНОВІ REACT І NEXT.JS

L.P. Dmytrotsa Ph.D., Assoc. Prof.; O.T. Starytskyi

OPTIMIZING PERFORMANCE AND SEO WHEN SCALING PROJECTS BASED ON REACT AND NEXT.JS

Оптимізація продуктивності та SEO є надзвичайно важливими аспектами сучасної розробки веб-додатків, особливо з огляду на зростаючу конкуренцію в онлайн-просторі та зростання вимог користувачів до якості веб-ресурсів.

Метою роботи було провести аналіз доступних методів оптимізації продуктивності та SEO для проктів на основі React і Next, та визначити їх користь та доцільність для розробника.

Для аналізу методів було підібрано три сервіси, які визначають оптимізацію сайту по різних критеріях. Було визначено їх основні функції, переваги та недоліки (таблиця 1).

Таблиця 1 – Аналіз доступних застосунків

Характеристики	Google PageSpeed Insights	GTmetrix	Nitropack.io
Основні функції	- Аналіз продуктивності для мобільних і десктопних версій - Метрики Core Web Vitals - Рекомендації для покращення	- Детальний аналіз швидкості завантаження - Розбір розмірів і кількості ресурсів - Візуалізація Waterfall	- Автоматична оптимізація швидкості - Кешування і стиснення ресурсів - Інтеграція з популярними CMS
Переваги	-Безкоштовність - Інтеграція з екосистемою Google - Легкість використання	- Більше технічних деталей - Можливість налаштувати тестування (локація, браузер тощо) - Звіти у PDF	- Інтеграція без додаткового коду - Миттєве покращення метрик - Підходить для масштабних сайтів

Продовження таблиці 1.

Недоліки	- Обмежена деталізація - Дані можуть бути недостатньо точними для великих сайтів	- Безкоштовна версія має обмежені функції - Інтерфейс трохи складніший для новачків	- Платний сервіс - Менше контролю над ручною оптимізацією

Провівши порівняння та ознайомившись з функціоналом кожного з сервісів, ми обрали Google PageSpeed Insights. На основі даної платформи ми будемо проводити аналіз та тестувати сайт, а саме нагружати сторінки та пробувати нові методи оптимізації продуктивності, які існують в React та Next. Результати будуть зберігатись в таблицях та порівнюватись один з одним.

Рисунок 1 – Основні критерії для оптимізації

Висновок: досліджено наявні веб-аплікації та обрано оптимальний варіант для нашої роботи. На основі вибраного сервісу було обрано основні критерії для оптимізації. Обраний сервіс допоможе нам аналізувати та обирати кращі варіанти для покращення швидкості та розуміння сайту.

Література

1. React Performance Optimization Techniques Режим доступу: https://supertokens.com/blog/5-tips-for-optimizing-your-react-apps-performance. [дата звернення 12.11.2024].
2. Next.js SEO Optimization: Practical Techniques for Better Ranking. Режим доступу: https://nextjs.org/learn-pages-router/seo/introduction-to-seo. [дата звернення 12.11.2024].
3. Server-Side Rendering and Static Site Generation in Next.js. Режим доступу: https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation. [дата звернення 12.11.2024].
4. Implementing JSON-LD for Enhanced SEO in Next.js. Режим доступу: https://nextjs.org/docs/app/building-your-application/optimizing/metadata. [дата звернення 12.11.2024].
5. Lazy Loading and Code Splitting in React and Next.js. Режим доступу: https://dut.edu.ua/ua/lib/1/category/1137/view/1483. [дата звернення 12.11.2024].
6. Best Practices for Image Optimization in Next.js. Режим доступу: https://uploadcare.com/blog/image-optimization-in-nextjs/. [дата звернення 12.11.2024].
7. Using Google Analytics and Search Console with Next.js. Режим доступу: https://nextjs.org/docs/messages/next-script-for-ga. [дата звернення 12.11.2024].
8. Enhancing Mobile Responsiveness for Better SEO in Next.js. Режим доступу: https://nextjs.org/learn-pages-router/seo/web-performance/seo-impact. [дата звернення 12.11.2024].

Додаток Б

Лістинг коду розроблених тестових додатків

1. Лістинг коду React

```
import { useContext } from "react";
import { CalcContext } from '../context/CalcContext'

const getStyleName = btn => {
  const className = {
    '=': 'equals',
    'x': 'opt',
    '-': 'opt',
    '+': 'opt',
    '/': 'opt',
  }
  return className[btn]
}

const Button = ({ value }) => {
  const { calc, setCalc } = useContext(CalcContext);

  // User click comma
  const commaClick = () => {
    setCalc({
      ...calc,
      num: !calc.num.toString().includes('.') ? calc.num + value :
calc.num
    });
  }
  // User click C
  const resetClick = () => {
    setCalc({ sign: '', num: 0, res: 0 })
  }
  // User click number
  const handleClickButton = () => {
    const numberString = value.toString()

    let numberValue;
    if(numberString === '0' && calc.num === 0) {
      numberValue = "0"
    } else {
      numberValue = Number(calc.num + numberString)
    }

    setCalc({
      ...calc,
      num: numberValue
    })
  }
  // User click operation
  const signClick = () => {
```

```
  setCalc({
    sign: value,
    res: !calc.res && calc.num ? calc.num : calc.res,
    num: 0
  })
}
// User click equals
const equalsClick = () => {
  if(calc.res && calc.num) {
    const math = (a, b, sign) => {
      const result = {
        '+': (a, b) => a + b,
        '-': (a, b) => a - b,
        'x': (a, b) => a * b,
        '/': (a, b) => a / b,
      }
      return result[sign](a, b);
    }
    setCalc({
      res: math(calc.res, calc.num, calc.sign),
      sign: '',
      num: 0
    })
  }
}
// User click persen
const persenClick = () => {
  setCalc({
    num: (calc.num / 100),
    res: (calc.res / 100),
    sign: ''
  })
}
// User click invert button
const invertClick = () => {
  setCalc({
    num: calc.num ? calc.num * -1 : 0,
    res: calc.res ? calc.res * -1 : 0,
    sign: ''
  })
}

const handleBtnClick = () => {

  const results = {
    '.': commaClick,
    'C': resetClick,
    '/': signClick,
    'x': signClick,
    '-': signClick,
    '+': signClick,
    '=': equalsClick,
    '%': persenClick,
    '+-': invertClick
```

```
    }
    if(results[value]) {
      return results[value]()
    } else {
      return handleClickButton()
    }
  }

  return (
    <button onClick={handleBtnClick}
className={`${getStyleName(value)} button`}>{value}</button>
  )
}

export default Button
```

Warpper.js

```
import { useContext } from "react"
import { CalcContext } from "../context/CalcContext"
import { Textfit } from 'react-textfit';

const Screen = () => {
  const { calc } = useContext(CalcContext);

  return (
    <Textfit className="screen" max={70}  mode="single">{calc.num ? calc.num :
calc.res}</Textfit>
  )
}

export default Screen
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

1.Лістинг коду Next.js

```
"use client"

import { useState } from "react"
import { Button } from "@/components/ui/button"
import { Card, CardContent, CardHeader, CardTitle } from
"@/components/ui/card"
```

```
export default function Calculator() {
  const [display, setDisplay] = useState("0")
  const [firstOperand, setFirstOperand] = useState<number |
null>(null)
  const [operator, setOperator] = useState<string | null>(null)
  const [waitingForSecondOperand, setWaitingForSecondOperand] =
useState(false)

  const inputDigit = (digit: string) => {
    if (waitingForSecondOperand) {
      setDisplay(digit)
      setWaitingForSecondOperand(false)
    } else {
      setDisplay(display === "0" ? digit : display + digit)
    }
  }

  const inputDecimal = () => {
    if (waitingForSecondOperand) {
      setDisplay("0.")
      setWaitingForSecondOperand(false)
      return
    }

    if (!display.includes(".")) {
      setDisplay(display + ".")
    }
  }

  const clearDisplay = () => {
    setDisplay("0")
    setFirstOperand(null)
    setOperator(null)
    setWaitingForSecondOperand(false)
  }

  const handleOperator = (nextOperator: string) => {
    const inputValue = Number.parseFloat(display)

    if (firstOperand === null) {
      setFirstOperand(inputValue)
    } else if (operator) {
      const result = performCalculation(operator, firstOperand,
inputValue)
      setDisplay(String(result))
      setFirstOperand(result)
    }

    setWaitingForSecondOperand(true)
    setOperator(nextOperator)
  }
```

```
  const performCalculation = (op: string, first: number, second:
number) => {
    switch (op) {
      case "+":
        return first + second
      case "-":
        return first - second
      case "×":
        return first * second
      case "÷":
        return first / second
      default:
        return second
    }
  }

  const calculateResult = () => {
    if (firstOperand === null || operator === null) {
      return
    }

    const inputValue = Number.parseFloat(display)
    const result = performCalculation(operator, firstOperand,
inputValue)

    setDisplay(String(result))
    setFirstOperand(result)
    setOperator(null)
    setWaitingForSecondOperand(true)
  }

  const toggleSign = () => {
    setDisplay(String(-Number.parseFloat(display)))
  }

  const calculatePercentage = () => {
    const currentValue = Number.parseFloat(display)
    setDisplay(String(currentValue / 100))
  }

  return (
    <Card className="w-full max-w-md mx-auto shadow-lg">
      <CardHeader className="bg-slate-100 dark:bg-slate-800">
        <CardTitle className="text-right text-3xl font-mono p-2
h-16 overflow-hidden">{display}</CardTitle>
      </CardHeader>
      <CardContent className="p-4">
        <div className="grid grid-cols-4 gap-2">
          <Button
            variant="outline"
            onClick={clearDisplay}
            className="bg-slate-200 hover:bg-slate-300
dark:bg-slate-700 dark:hover:bg-slate-600"
          >
```

```
            C
          </Button>
          <Button
            variant="outline"
            onClick={toggleSign}
            className="bg-slate-200 hover:bg-slate-300
dark:bg-slate-700 dark:hover:bg-slate-600"
          >
            +/-
          </Button>
          <Button
            variant="outline"
            onClick={calculatePercentage}
            className="bg-slate-200 hover:bg-slate-300
dark:bg-slate-700 dark:hover:bg-slate-600"
          >
            %
          </Button>
          <Button
            variant="outline"
            onClick={() => handleOperator("÷")}
            className="bg-orange-400 hover:bg-orange-500
text-white"
          >
            ÷
          </Button>

          <Button variant="outline" onClick={() =>
inputDigit("7")}>
            7
          </Button>
          <Button variant="outline" onClick={() =>
inputDigit("8")}>
            8
          </Button>
          <Button variant="outline" onClick={() =>
inputDigit("9")}>
            9
          </Button>
          <Button
            variant="outline"
            onClick={() => handleOperator("×")}
            className="bg-orange-400 hover:bg-orange-500
text-white"
          >
            ×
          </Button>

          <Button variant="outline" onClick={() =>
inputDigit("4")}>
            4
          </Button>
          <Button variant="outline" onClick={() =>
inputDigit("5")}>
```

```
            5
          </Button>
          <Button variant="outline" onClick={() =>
inputDigit("6")}>
            6
          </Button>
          <Button
            variant="outline"
            onClick={() => handleOperator("-")}
            className="bg-orange-400 hover:bg-orange-500
text-white"
          >
            -
          </Button>

          <Button variant="outline" onClick={() =>
inputDigit("1")}>
            1
          </Button>
          <Button variant="outline" onClick={() =>
inputDigit("2")}>
            2
          </Button>
          <Button variant="outline" onClick={() =>
inputDigit("3")}>
            3
          </Button>
          <Button
            variant="outline"
            onClick={() => handleOperator("+")}
            className="bg-orange-400 hover:bg-orange-500
text-white"
          >
            +
          </Button>

          <Button variant="outline" onClick={() =>
inputDigit("0")} className="col-span-2">
            0
          </Button>
          <Button variant="outline" onClick={inputDecimal}>
            .
          </Button>
          <Button variant="outline" onClick={calculateResult}
className="bg-orange-400 hover:bg-orange-500 text-white">
            =
          </Button>
        </div>
      </CardContent>
    </Card>
  )
}
```