Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження та створення веб-застосунків з використанням Server-Side-rendering на основі React та Next.js
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-6

Мезенцев М.А.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник проф. Четвериков Г.Г.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мезенцеву Максиму Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження та створення веб-застосунків з використанням Server-Side-rendering на основі React та Next.js»

Затверджена наказом по університету від «29» березня 2024 р. №250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 12.06.2024 р.

3. Вихідні дані до роботи: *технології розробки веб-застосунків, методи веб-інтеграції, основи серверного рендерингу в React та Next.js*

4. Перелік питань, що потрібно опрацювати в роботі: *аналіз існуючих методів рендерінгу, розробка застосунків з використанням React та Next.js, вибір методу рендерінгу, швидкість рендерінгу*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	23.01 – 14.02.24	*виконано*
2	Аналіз та вибір API для дослідження	15.02 – 24.02.24	*виконано*
3	Аналіз та моделювання предметної області	17.02 – 28.02.24	*виконано*
4	Планування експериментів	25.02 – 28.02.24	*виконано*
5	Програмна реалізація кожного з обраних для дослідження API	25.02 – 01.04.24	*виконано*
6	Експериментальні дослідження	02.04 – 20.04.24	*виконано*
7	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.04 – 23.04.24	*виконано*
8	Написання та оформлення статті та тез доповіді	17.04 – 23.04.24	*виконано*
9	Підготовка пояснювальної записки	01.04 – 26.04.24	*виконано*
10	Підготовка презентації та доповіді	26.04 – 02.05.24	*виконано*
11	Нормоконтроль	03.05 – 08.05.24	*виконано*
12	Рецензування	08.05 – 14.05.24	*виконано*
13	Занесення диплома в електронний архів	15.05.2024	*виконано*
14	Попередній захист	15.05.2024	*виконано*
15	Допуск до захисту у зав. кафедри	18.05.2024	*виконано*

Дата видачі завдання 20 січня 2024 р.

Студент _____ Мезенцев М.А.
(підпис)

Керівник кваліфікаційної роботи _____ проф. Четвериков Г.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи, 41 с., 7 рис., 10 джерел

ВЕБ-ЗАСТОСУНКИ, КОМПОНЕНТИ, ШВИДКІСТЬ РЕНДЕРІНГУ, RENDERING, SERVER-SIDE.

Об'єктом дослідження є розробка та оптимізація веб-застосунків з використанням Server-side rendering на базі технологій React та Next.js.

Метою роботи є створення ефективних компонентів та програмного забезпечення для побудови веб-додатків будь-якої складності з використанням Server-side rendering та визначення швидкості завантаження сторінки в залежності від метода рендерінгу.

У процесі розробки використовуються сучасні методи розробки, бібліотеки та мова програмування JavaScript.

В результаті дослідження виявлено найбільш швидкий метод рендерінгу під конкретні задачі бізнесу.

WEB APPLICATIONS, COMPONENTS, RENDERING SPEED, RENDERING, SERVER-SIDE.

The subject of the research is the development and optimization of web applications using Server-side Rendering based on React and Next.js technologies.

The aim of the work is to create efficient components and software for building web applications of any complexity using Server-side Rendering and to determine the page load speed depending on the rendering method.

The development process uses modern development methods, libraries, and the JavaScript programming language.

As a result of the research, the fastest rendering method for specific business tasks was identified.

Я, Мезенцев Максим Андрійович, студент гр. ІПЗм-22-6, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя робота на тему «Дослідження та створення веб-застосунків з використанням Server-Side-rendering на основі React та Next.js», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

JS – JavaScript

NPM – Node Package Manager

SSG – Static Site Generation

SSR – Server-Side Rendering

TS – TypeScript

CSR – Client Side Rendering

ВСТУП

У світі веб-розробки швидко розвиваються технології, і вибір правильних інструментів грає ключову роль у створенні продуктивних та ефективних веб-застосунків. Дослідження використання React та Next.js з фокусом на Server-side Rendering стає об'єктом цієї роботи.

Один із головних аспектів дослідження – це вивчення різних видів рендеринга, таких як Client-side Rendering (CSR), Server-side Rendering (SSR) та Static Site Generation (SSG). Кожен з цих підходів має свої переваги та використовується залежно від вимог до продуктивності та SEO-оптимізації.

Мета даного дослідження – пошук кращого рішення при виборі рендерінгу в залежності від потреб бізнесу. Дослідження включає в себе сучасні технології та бібліотеки необхідні для створення сучасних веб-сайтів.

У процесі дослідження використовується бібліотеки React та Next.js. Ці інструменти призначені для програмної реалізації веб-застосунків із застосуванням Server-side Rendering різної складності.

Результати дослідження дозволять визначити оптимальні технології для створення веб-застосунків із застосуванням SSR на базі React та Next.js. Висновки також обговорюватимуть перспективи розвитку цієї області та можливості впровадження отриманих результатів у реальних проектах.

За останні роки відбулося істотне зростання інтересу до використання Server-side Rendering (SSR) та Static Site Generation (SSG) у веб-розробці, зокрема, на базі популярних бібліотек та фреймворків, таких як React та Next.js. Ці технології дозволяють оптимізувати продуктивність веб-застосунків та покращувати їхню SEO-оптимізацію. Відповідно, важливо розглядати їхнє використання у контексті створення сучасних веб-додатків.

Для досягнення поставленої мети передбачено вирішення таких завдань:

– аналіз сучасних підходів до рендерингу: Огляд та порівняння різних видів рендерингу для визначення їхніх переваг та недоліків;

- вивчення технологій React та Next.js: Аналіз основ та можливостей цих фреймворків для реалізації SSR та SSG;
- проектування та реалізація прикладних веб-застосунків: Розробка практичних проектів для ефективного вивчення і випробування вивчених концепцій;
- оптимізація та аналіз результатів: Вдосконалення розроблених додатків та вивчення їхньої продуктивності.

В ході дослідження очікується отримання глибокого розуміння процесу розробки веб-застосунків із використанням SSR та SSG. Окрема увага буде приділена виявленню можливостей оптимізації та вдосконаленню продуктивності додатків на основі здобутих знань.

1 РОЛЬ REACT ТА NEXT.JS У РОЗРОБЦІ SSR ВЕБ-ЗАСТОСУНКІВ

1.1 Використання React та Next.js

Сучасний веб-розвиток вимагає від розробників використання ефективних стратегій для покращення продуктивності та взаємодії користувача. Один з ключових інструментів у цьому контексті – це Server-Side Rendering (SSR) [1], особливо коли розглядається його застосування в синергії з React [2] та Next.js [3]. У цій статті ми розглянемо різні аспекти використання SSR у розробці веб-застосунків, зосередившись на його реалізації з використанням бібліотек React та фреймворка Next.js.

Сутність Server-Side Rendering полягає в тому, що SSR представляє собою метод, при якому генерація HTML відбувається на сервері, а не на клієнті, що дозволяє досягти більш ефективного відображення контенту. Цей підхід в основному спрямований на зменшення часу завантаження сторінки та підвищення відгукливості інтерфейсу.

Роль React в контексті SSR в тому, що React, як одна з провідних JavaScript-бібліотек [4], активно застосовується в SSR. Його архітектурні особливості забезпечують можливість серверного рендерингу, що дозволяє ефективно створювати динамічні користувацькі інтерфейси та поліпшувати загальну продуктивність веб-додатків.

Next.js є кращим фреймворком для SSR та став популярним вибором при розробці веб-додатків з використанням SSR. Його інтеграція з React спрощує реалізацію серверного рендерингу та надає розробникам інструментарій для ефективного управління процесом.

SSR активно використовується в великих компаніях, зокера в Facebook, який впроваджує SSR з метою поліпшення кількох ключових аспектів. По-перше, зменшення часу завантаження сторінки суттєво підвищує задоволення користувачів, особливо при роботі з обширним обсягом контенту в соціальній мережі. Во-вторих, використання SSR сприяє оптимізації для пошукових систем, що важливо для привертання додаткового трафіку.

Переваги та виклики використання SSR включають в собі поліпшений SEO, зменшення навантаження на клієнтську сторону та більш ефективне управління станом. Однак існують виклики, такі як серверні витрати ресурсів і необхідність уважного управління кешуванням.

SSR порівнюється з іншими методами, такими як інкрементальний рендерінг та статичний генеративний рендерінг, щоб виявити його переваги в різних сценаріях.

Сміло можно стверджувати, що SSR в розробці веб-додатків, особливо при використанні бібліотеки React і фреймворка Next.js, є стратегічним кроком для забезпечення високої продуктивності та конкурентоспроможності. Досвід Facebook підкреслює важливість використання SSR в контексті сучасних тенденцій веб-розробки і необхідність пошуку оптимальних рішень для поліпшення користувацького досвіду та збільшення прибутку компанії.

1.2 Переваги SSR

Однією з ключових переваг використання SSR є поліпшення індексації сторінок пошуковими системами. Це особливо важливо для веб-додатків, які покладаються на органічний трафік. Працюючи з React та Next.js у контексті SSR, ви підвищуєте шанси на високий рейтинг у пошукових системах, що допомагає залучити більше аудиторії.

Використання SSR відкриває можливості для розробки ефективних стратегій кешування, які дозволяють зберігати готовий контент на сервері та швидко його віддавати при наступних запитаннях. Це дозволяє покращити час завантаження сторінок та знизити навантаження на сервер.

При впровадженні SSR, розробники стикаються із викликами, такими як необхідність вирішення проблем серверних витрат ресурсів та підтримка стану додатку. Використання React та Next.js дозволяє вирішувати ці проблеми шляхом ефективного управління станом і ресурсами.

За ріками SSR не втрачає своєї актуальності і, навпаки, зазнає постійного розвитку. За допомогою React та Next.js, SSR стає ще більш доступним та

потужним інструментом для розробників. Прогнозується, що у майбутньому він продовжить займати центральне місце у розробці веб-додатків, забезпечуючи ефективний та швидкий користувацький досвід.

1.3 Недоліки SSR

SSR (Server-side-rendering) не є актуальним для всіх компаній та для всіх задач. В нього також є свої мінуси, через які великі компанії можуть відмовитись від його використання. На сьогоднішній день важко навести конкретні приклади великих компаній, які відмовилися від Server-Side Rendering (SSR), оскільки така інформація може бути конфіденційною та не завжди загальнодоступною.

Такі рішення зазвичай пов'язані зі специфікою і потребами кожного окремого проекту та його архітектурою. Однак можна розглянути загальні тенденції та аргументи, які можуть впливати на вибір інших методів рендерингу.

Netflix: Хоча немає конкретних підтверджень, що Netflix відмовився від SSR, але вони відомі своєю великою увагою до високої продуктивності та швидкості завантаження.

У своєму веб-інтерфейсі Netflix активно використовує асинхронний рендеринг та інші техніки для забезпечення швидкого відгуку.

Twitter також володіє динамічним та постійно змінюючимся контентом. Хоча конкретні деталі їхньої архітектури не завжди відкриті для громадськості, Twitter відомий своїми заходами щодо оптимізації завантаження та інтерактивності.

LinkedIn також може служити прикладом використання альтернативних підходів. Специфіка соціальної мережі може вимагати акценту на швидкість завантаження та динамічний зміст, що може бути досягнуто за допомогою асинхронного рендерингу.

Серед іншого, може бути декілька причин, чому компанії можуть відмовитись від підходу SSR.

Як публічно відомо, компанії з великим об'ємом динамічного та часто змінюючого контенту можуть обрати увагу на асинхронний рендеринг або інші

методи, які дозволяють більше контролю над взаємодією користувача та зменшенням часу на завантаження сторінок.

Також, у випадку, коли компанії, які ставлять перед собою завдання забезпечити високу продуктивність та миттєвий відгук для користувачів, можуть відмовитися від SSR на користь методів, що дозволяють швидке відтворення вмісту на клієнтському боці.

Потрібно також враховувати і специфіку веб-додатку, бо індивідуальні вимоги конкретного веб-додатку можуть впливати на вибір методу рендерингу. Наприклад, додатки зі складними інтерактивними компонентами можуть здобути від асинхронного рендерингу чи інших підходів.

1.4 Бібліотека для SSR – Next.js

Next.js – це фреймворк для розробки веб-додатків, який з'явився завдяки команді Vercel (раніше відомої як ZEIT). Його історія бере початок у 2016 році, коли було виявлено потребу в більш простому та ефективному інструменті для створення універсальних веб-додатків на базі бібліотеки React. Next.js став відповіддю на це виклик, забезпечуючи розробникам ряд інструментів для зручної та продуктивної роботи.

Цей фреймворк використовується для широкого спектру завдань у сфері веб-розробки. Він ідеально підходить для створення різноманітних веб-додатків, включаючи статичні сайти, лендінги, веб-додатки та блоги. Next.js дозволяє розробникам зосередитися на функціональності своїх додатків, забезпечуючи при цьому оптимізований та ефективний робочий процес.

Однією з ключових переваг Next.js є його підтримка Server-Side Rendering (SSR). Цей підхід дозволяє виконувати рендеринг сторінок на сервері, що поліпшує SEO та забезпечує швидкий ініціальний завантаження веб-додатків. Крім того, Next.js надає можливість використовувати Static Site Generation (SSG) для створення статичних сторінок, що дозволяє значно покращити продуктивність та швидкість завантаження.

Підтримка TypeScript [5] вбудована у фреймворк, що дозволяє розробникам створювати типізовані веб-додатки та забезпечує високу робочу ефективність. Маршрутизація на рівні файлів робить процес створення маршрутів інтуїтивно зрозумілим та зручним.

Next.js володіє розширеними засобами управління даними та API, такими як Redux [6] та SWR (Stale-While-Revalidate), що полегшує взаємодію з сервером та керування станом додатків.

Ця бібліотека стала важливим інструментом для розробників, які вирішили використовувати React у своїх проектах. Його зручний та продуктивний інтерфейс дозволяє швидко створювати ефективні, масштабовані та високопродуктивні веб-додатки на основі сучасних технологій.

Next.js – це не просто фреймворк, а повноцінне середовище для розробки веб-додатків, яке зробило суттєвий внесок у сферу веб-розробки. Його гнучкість і зручність здобули широке визнання серед розробників, а ряд вбудованих можливостей робить його потужним інструментом для будь-якого проекту.

Next.js – це потужний інструмент для розробки веб-додатків, який виділяється своєю гнучкістю та зручністю використання. Це не просто фреймворк, а ціле екосистемне рішення, яке дозволяє розробникам ефективно вирішувати різноманітні завдання.

Коли мова йде про використання Next.js, цей інструмент проявляє себе на різних фронтах веб-розробки. Він ідеально підходить для створення статичних сайтів та лендінгів, де акцент робиться на продуктивності та ефективному управлінні контентом. За допомогою можливостей Server-Side Rendering (SSR) та будівництва на React, Next.js також стає відмінним вибором для створення високопродуктивних та динамічних веб-додатків. Завдяки своїй простоті та потужності, Next.js легко забезпечує розробку блогів та сайтів із багатим контентом, де ключовими є SEO та динамічність.

Одним із ключових переваг Next.js є Server-Side Rendering (SSR), який дозволяє виконувати рендеринг сторінок на сервері. Це не лише поліпшує оптимізацію для пошукових систем, але й забезпечує ефективне взаємодію з

користувачами. Статична генерація сайту (Static Site Generation, SSG) дає можливість створювати статичні сторінки, що підвищує продуктивність та забезпечує швидке завантаження контенту.

Однак, незважаючи на всі переваги, важливо підкреслити, що кожен інструмент має свої особливості. Next.js надає розробникам інтеграцію зі стандартними засобами управління станом та API, такими як Redux та SWR, а також підтримує TypeScript, що робить його потужним і гнучким інструментом.

Отже, Next.js стає невід'ємним компаньйоном для розробників, які використовують React у своїх проектах. З його допомогою вони можуть швидко та ефективно створювати високопродуктивні веб-додатки, готові до викликів сучасного веб-розвитку.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РЕНДЕРІНГУ У СТВОРЕННІ ВЕБ-ЗАСТОСУНКІВ НА ОСНОВІ REACT ТА NEXT.JS

2.1 Введення в Server-Side Rendering

Server-Side Rendering (рендеринг на стороні сервера) є стратегією веб-розробки, в якій HTML [7] сторінок генерується на сервері, а не на клієнтському браузері. Це протиставлення традиційному підходу, в якому весь код відображення, включаючи HTML, CSS та JavaScript, генерується браузером клієнта.

Основні принципи Server-Side Rendering полягають в тому, що попереднє рендеринг на сервері (Pre-rendering): SSR передбачає генерацію вмісту сторінок на сервері перед тим, як вони будуть відправлені браузеру. Це означає, що клієнт отримує готовий HTML-код і може відображати сторінку негайно.

SEO-оптимізація: одна з ключових переваг SSR полягає в поліпшенні індексації веб-сторінок пошуковими системами. Так як контент вже включений у HTML на сервері, пошукові роботи легше і швидше індексують сторінку.

Швидкий перший байт (Fast Time to First Byte): стратегія SSR дозволяє зменшити час отримання першого байта сторінки. Готовий HTML надсилається швидко, що поліпшує час завантаження сторінок.

Є багато переваг використання SSR, наприклад SEO-оптимізація [8], яка є кращою індексацією веб-сторінок пошуковими системами.

Прискорюється відображення, зменшується часс до отримання першого контенту для користувача.

Сумісність з соціальними мережами, завдяки оптимальний вивіду контенту для публікації у соціальних мережах.

Зменшення навантаження на клієнтський браузер завдяки спрощення завдання браузера, оскільки велика частина роботи виконується на сервері.

Недоліки SSR:

– збільшений обсяг запитів на сервер: завантаження сервера може збільшитися, особливо при великому обсязі запитів;

– складніше управління станом: обробка стану додатка може виявитися складнішою через переходні стани при рендерингу на сервері.

Server-Side Rendering є потужним інструментом для розробників, особливо тих, хто прагне поліпшити продуктивність та користувацький досвід веб-застосунків. Однак вибір між SSR та іншими стратегіями рендерингу повинен бути зроблений враховуючи конкретні потреби проекту.

2.2 Роль React у SSR

React, бібліотека розробки інтерфейсів від Facebook, стала справжнім героєм у світі SSR, де вона розкриває свої найкращі якості для створення ефективних та швидких веб-застосунків. Розглянемо глибше, як React допомагає у реалізації цього заходу.

React дозволяє розробникам створювати динамічні та інтерактивні компоненти інтерфейсу, які можуть реагувати на події та зміни стану. Це ідеально підходить для впровадження на сервері, де потрібно створювати HTML-код із певним станом, який може змінюватися з часом.

Один з ключових аспектів React – ефективне управління станом компонентів. При SSR важливо мати можливість обробляти стан як на сервері, так і на клієнтському боці. React дозволяє створювати компоненти, які можуть легко синхронізувати свій стан між сервером і клієнтом.

React може використовуватися для рендерингу компонентів на сервері, забезпечуючи HTML-код, готовий до відправлення клієнту. Це полегшує виведення початкового вмісту сторінки без чекання завантаження JavaScript.

Завдяки React, розробники можуть створювати загальний код, який використовується як на сервері, так і на клієнтському боці. Це робить SSR більш ефективним та спрощує роботу з кодом, оскільки немає потреби у подвійному кодуванні.

Оскільки рендеринг відбувається на сервері, HTML-код, який генерується React, містить всю необхідну інформацію для пошукових систем. Це поліпшує SEO-показники веб-сторінок.

Використання React у SSR сприяє загальній продуктивності веб-застосунків, дозволяючи швидше завантаження та кращий досвід користувача.

Розгортаючи всі ці можливості, React стає надійним партнером у впровадженні SSR, роблячи веб-розробку ефективнішою та користувачський досвід ще кращим.

2.3 Next.js та його роль у розробці SSR застосунків

У світі веб-розробки одним із захоплюючих напрямків стає створення веб-додатків з використанням Server-Side Rendering (SSR). React та Next.js виступають як невід'ємні союзники у цьому цікавому процесі.

Next.js – це фреймворк для React, який надає додаткові інструменти для роботи з SSR. Він "збудовано за замовчуванням" для підтримки SSR, роблячи його ідеальним вибором для розробників, які хочуть швидко створити SSR веб-застосунок.

У контексті SSR, React використовується для створення компонентів, які можуть рендеритися як на серверному, так і на клієнтському боці. Це дозволяє ефективно управляти станом та працювати зі сторінкою ще до завантаження JavaScript на клієнтському пристрої.

Next.js надає додаткові можливості, такі як генерація статичних та динамічних сторінок, потужна система маршрутизації, оптимізація завантаження, робота з серверними функціями, автоматичне кешування та гнучкі налаштування конфігурації.

Next.js стає невід'ємним інструментом для розробників React, які хочуть впровадити SSR у свої веб-додатки. Його зручність, продуктивність та розширені можливості конфігурації роблять його ідеальним вибором для ефективної розробки SSR веб-додатків.

2.4 Переваги використання React та Next.js у розробці SSR веб-застосунків

Використання React та Next.js у розробці веб-застосунків із Server-Side Rendering (SSR) – це справжня перевага для розробників. Це як керма, яка

спрямовує твій корабель по водяних шляхах ефективності та високоякісного користувацького досвіду.

React, створений Facebook, став однією з ключових технологій для розробки інтерфейсів. У світі SSR він використовується для створення компонентів, які без зусиль рендеряться на сервері та на клієнтському боці. Це означає ефективне управління станом та можливість взаємодії зі сторінкою ще до завантаження JavaScript на клієнтський пристрій.

Next.js – це наступний етап у розвитку, вбудований фреймворк для React, який відзначається підтримкою SSR "з коробки". Це означає, що ти можеш зосередитися на розробці, не турбуючись про складнощі SSR, оскільки вони вже враховані.

Основна перевага полягає в швидкості завантаження сторінок. React сприяє ефективності, а Next.js надає оптимізації для зниження часу завантаження. І це не просто про швидкість, але і про SEO.

Вміст SSR відправляється у вигляді готового HTML, що полегшує роботу пошукових систем. З можливістю визначити статичні шляхи в Next.js, процес індексації стає ще простішим.

Окрім того, React дозволяє ефективно керувати станом компонентів, навіть на серверному боці, і Next.js робить розробку простою та зручною. Ці інструменти не просто допомагають у розробці, вони перетворюють її на захоплюючу подорож до високоякісних SSR веб-застосунків.

2.5 Процес роботи з SSR у Next.js

Поглибимося в процес роботи з Server-Side Rendering (SSR) у фреймворку Next.js, де кожен етап виявляється ключовим аспектом оптимізації та високопродуктивної розробки.

На першому етапі визначається функція getServerSideProps, що є основною будівельною одиницею SSR в Next.js. Ця функція дозволяє визначити асинхронні дані, які будуть необхідні для рендерингу сторінки. При кожному запиті на

сторінку сервер викликає getServerSideProps, надаючи можливість динамічно створювати вміст на сервері перед його відправленням клієнту.

Ключовим моментом є те, що дані, отримані через getServerSideProps, передаються компоненті для рендерингу, забезпечуючи гнучкість та контроль над вмістом сторінки. Ця концепція властива також для параметрів шляху, які можна використовувати для створення динамічних URL-адрес.

Наступним кроком є серверна обробка та генерація HTML, який надсилається клієнту. Суттєво, що використовується той самий код компоненти, який працює на клієнтському боці. Це спрощує завдання браузера, оскільки він отримує готовий до відображення HTML-контент.

Оптимізація проявляється у відправці лише необхідних для сторінки даних, що сприяє швидкості завантаження та кращому використанню ресурсів. При цьому Next.js активно застосовує стратегії, що дозволяють завантажувати лише необхідні дані для певного запиту.

Отже, використання SSR у Next.js визначається не лише високою продуктивністю, але й гнучкістю розробки, забезпечуючи ефективне використання ресурсів та збереження швидкості відгуку сторінок.

2.6 Рендеринг на стороні клієнта

Рендеринг на стороні клієнта (Client-Side Rendering, CSR) є важливою концепцією у сучасному веб-розробці, особливо коли мова йде про використання бібліотеки React та фреймворка Next.js. Цей підхід призначений для покращення динамічності та інтерактивності веб-додатків, проте він також має свої переваги та виклики.

Однією з ключових переваг CSR є здатність створювати інтерактивні інтерфейси, які реагують на користувацькі дії миттєво, без необхідності повторного завантаження сторінки. Це робить взаємодію з додатком більш природною та зручною для користувача.

Використання асинхронних запитів у CSR дозволяє динамічно завантажувати дані та ресурси, не блокуючи основний потік виконання. Це сприяє

збереженню відгукливості додатка навіть при обробці великого обсягу інформації.

Однак важливо враховувати, що перше завантаження сторінки при використанні CSR може бути повільним, особливо при наявності значного обсягу JavaScript-коду. Це може вплинути на загальний час завантаження та спричинити менш задоволенних користувачів.

У плані пошукової оптимізації CSR має певні виклики. Пошукові роботи не завжди ефективно індексують контент, який генерується динамічно за допомогою JavaScript. Це може вплинути на SEO-параметри та видимість в пошукових результатах.

Також, важливо враховувати залежність від JavaScript. Якщо користувач вимикає JavaScript або використовує застарілий браузер, може виникнути проблема з доступністю функціоналу, який використовує CSR.

У React для реалізації CSR використовується бібліотека react-dom. Це дозволяє забезпечити високий рівень динамічності та ефективність веб-додатків, але розробникам важливо уважно вибирати підхід з урахуванням особливостей конкретного проекту та його цільової аудиторії.

2.7 Генерація статичних сайтів

Генерація статичних сайтів (Static Site Generation, SSG) [9] – це стратегія рендерингу веб-сторінок, яка передбачає генерацію статичних HTML-файлів на етапі збірки додатка, перед їх розміщенням на сервері. Цей підхід здатний значно поліпшити продуктивність та ефективність веб-додатків, особливо в контексті сайтів з чіткою структурою та невеликою динамікою.

Основна перевага SSG полягає в тому, що відвідувачі отримують статичні HTML-файли, що покращує час завантаження сторінок. Це особливо важливо для користувачів з поганим Інтернет-з'єднанням або які відвідують сайт з великою кількістю контенту.

SSG часто використовується для створення блогів, сайтів-візитівок, або будь-яких сайтів, де контент майже не змінюється або змінюється рідко. Важливо

зауважити, що при використанні SSG динаміка сайту обробляється на етапі збірки, а не на стороні сервера чи клієнта.

Однак, SSG має свої обмеження, особливо для додатків із складною динамікою та великою кількістю персоналізації. Якщо контент часто змінюється або залежить від користувацьких дій, SSG може виявитися менш практичним.

Засоби, такі як Next.js, надають можливість використовувати SSG разом із Server-Side Rendering (SSR) чи Client-Side Rendering (CSR) в одному додатку, тим самим комбінуючи переваги різних стратегій рендерингу в залежності від конкретних потреб проекту.

3 ВИБІР МЕТОДУ РЕНДЕРІНГА ПІД КОНКРЕТНІ ЗАДАЧІ

При виборі методу рендерінга для конкретних задач важливо враховувати різні аспекти, щоб забезпечити ефективність та оптимальність веб-додатку. Одним із ключових критеріїв є тип контенту, який ви працюєте.

Якщо ваш веб-сайт або додаток складається переважно зі статичного контенту, Static Site Generation (SSG) може бути оптимальним вибором. Це дозволяє попередньо генерувати статичні сторінки на етапі збору проекту, що призводить до швидшого завантаження та поліпшення продуктивності, особливо на стороні клієнта.

Якщо ваш вміст частково або повністю динамічний, Server-Side Rendering (SSR) може стати важливим інструментом. Використання SSR дозволяє генерувати сторінки на сервері для кожного запиту, забезпечуючи оновлення контенту в реальному часі та взаємодію користувача з динамічним вмістом.

У випадках, коли важлива швидкість завантаження та відгукливість, Client-Side Rendering (CSR) може бути вибором, оскільки він завантажує базову HTML-структуру та потім запускає JavaScript для відображення вмісту. Це особливо ефективно для веб-додатків, які широко використовують взаємодію користувача без перезавантаження сторінок.

Важливо також враховувати SEO-параметри. Якщо оптимізація для пошукових систем є ключовою, SSR та SSG можуть бути перевагою, оскільки вони надають стабільний та індексований контент для пошукових роботів.

Таким чином, вибір методу рендерінга повинен базуватися на конкретних потребах вашого проекту, типі вмісту та вимогах щодо продуктивності та SEO. Оптимальний підхід може варіюватися в залежності від конкретних умов та задач.

Обираючи метод SSR для мого веб-додатку, я керувався кількома ключовими факторами, що відіграють важливу роль у досягненні конкретних цілей та вимог проекту.

По-перше, з урахуванням природи мого вмісту та його динамічності, SSR здавався оптимальним рішенням. Цей метод дозволяє генерувати сторінки на

сервері для кожного запиту, що особливо важливо при необхідності надавати користувачам актуальний та оновлюваний контент.

Далі, враховуючи важливість SEO для мого веб-додатку, обрання SSR було стратегічним рішенням. Здатність генерувати HTML на сервері допомагає забезпечити стабільний та індексований вміст для пошукових систем, що позитивно впливає на позиції в пошукових результатах.

У реалізації SSR для мого проекту я збираюся використовувати бібліотеку React та фреймворк Next.js. React, завдяки своїй архітектурі, надає ефективні засоби для серверного рендерингу, а Next.js спростить цей процес та забезпечить необхідний інструментарій для управління SSR.

Планується створення серверних компонентів за допомогою Next.js, які будуть відповідати за генерацію контенту на сервері. Такий підхід дозволить забезпечити оптимальну продуктивність та швидке завантаження сторінок для користувачів.

Враховуючи вищеописані фактори та стратегії реалізації, обрання SSR для мого веб-додатку є обгрунтованим та спрямованим на досягнення максимальної ефективності та задоволення потреб користувачів.

Додатково, обрання SSR має на меті забезпечити користувачам високу відгукливість та швидкість веб-додатку. Завдяки можливості генерування контенту на сервері, користувачі отримують вже готові сторінки, що значно прискорює час завантаження та поліпшує загальний користувацький досвід.

Також, обрання SSR спрямоване на забезпечення легкості управління станом додатку. Під час використання SSR з React та Next.js, можна ефективно управляти станом інтерфейсу на сервері, що дозволяє запобігти зайванню логіки на стороні клієнта та покращити масштабованість проекту.

Додатково, SSR може полегшити завдання з підтримки індексації контенту для соціальних мереж та покращити можливості поділу посилань. Створення повноцінних HTML-сторінок на сервері дозволяє забезпечити коректне відображення та роботу мета-тегів для соціальних мереж, що важливо для поділу контенту та привертання нових користувачів.

Таким чином, вибір SSR для мого веб-додатку націлений на забезпечення швидкості, відзивчивості та оптимального управління станом, з урахуванням важливості SEO та соціальної взаємодії.

Server-Side Rendering (SSR) є ключовою стратегією веб-розробки, яка надає безліч переваг у створенні ефективних та продуктивних веб-додатків. Розглянемо докладніше кілька аспектів, які підтримують обрання SSR для мого веб-додатку.

Важливою перевагою використання SSR є його вплив на процес індексації веб-додатку пошуковими системами. Оскільки весь контент генерується на сервері та передається як готовий HTML, пошукові роботи легко аналізують та індексують вміст. Це може позитивно впливати на позиції в результатах пошуку, поліпшуючи видимість веб-додатку для користувачів.

Ще однією вагомою перевагою є можливість оптимізації сайту для підвищення продуктивності. Генерація сторінок на сервері дозволяє ефективніше використовувати ресурси сервера та зменшує обсяг передаваних даних. Відправка клієнту вже готового HTML сприяє швидкішому завантаженню сторінок та покращує загальний досвід користувача.

SSR також розкриває нові можливості для інтеграції зі сторонніми сервісами та додатками. Сервіси аналітики, маркетингові інструменти чи соціальні платформи можуть легше взаємодіяти з веб-додатком через доступний HTML. Це розширює можливості функціональності та сприяє розширенню додатку за межі базового функціоналу.

Вигляд створеної екосистеми при розгортанні Next.js пакетів (див. рис. 3.1)

Таким чином, мій вибір на користь SSR базується на комплексі вигід, спрямованих на поліпшення продуктивності, SEO, швидкості завантаження та зручності інтеграції з різноманітними сервісами. Це стратегічний крок у напрямку створення високоефективного веб-додатку, що забезпечить задоволення користувачів та конкурентоспроможність на ринку.

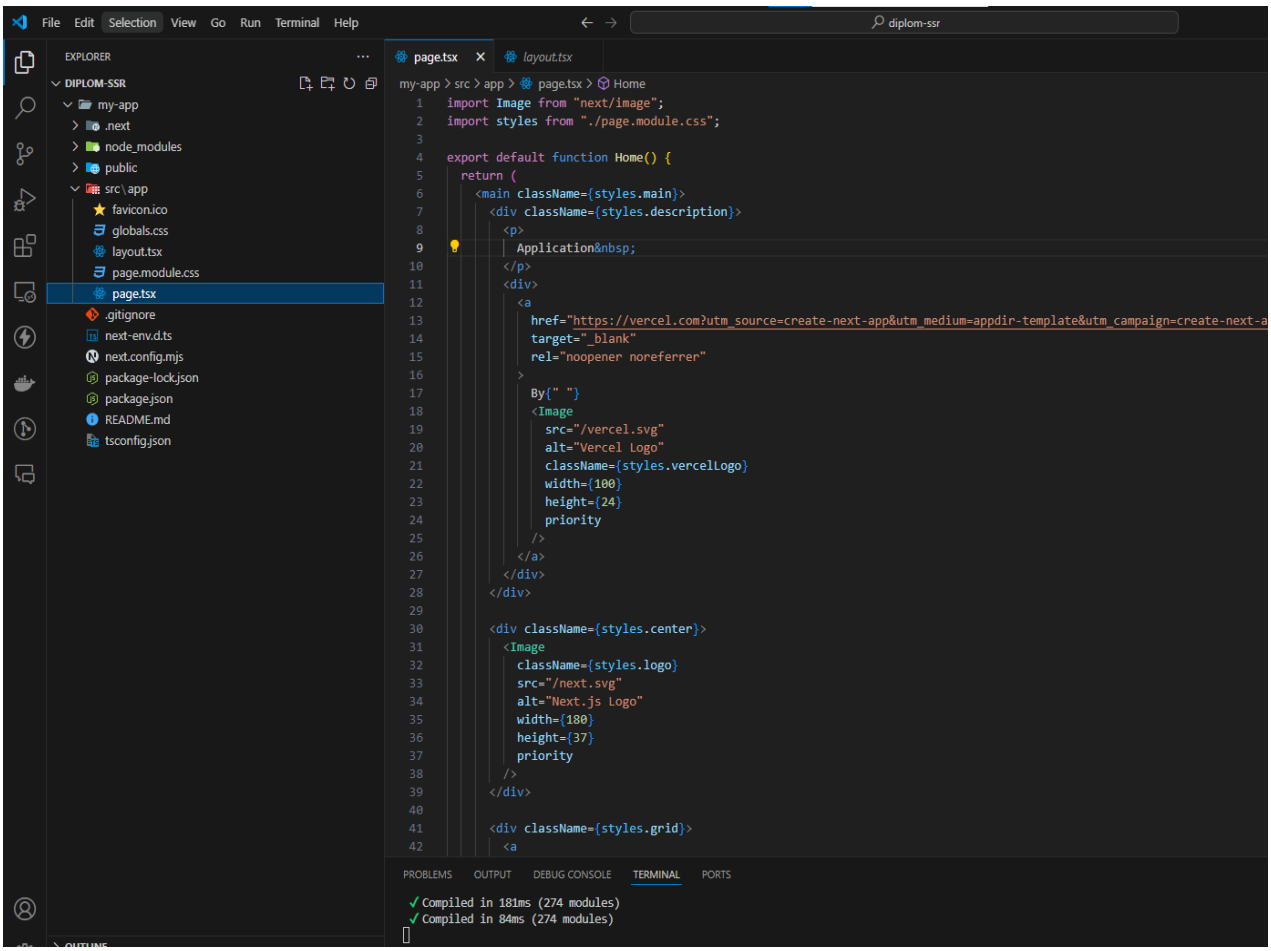

Рисунок 3.1 – Вигляд нового Next.js додатку

4 ОПИС РОЗРОБКИ ВЕБ-САЙТУ З ВИКОРИСТАННЯМ SSR

У цьому розділі буде розглянуто створення веб-сайту з використанням мови програмування JS або TS, яка є фаворитом для веб-розробки.

4.1 Опис розробки з використанням сучасних технологій

Створення веб-сайту з використанням Server-Side Rendering (SSR) є захоплюючим та важливим процесом у сучасній розробці веб-додатків. SSR дозволяє генерувати HTML на сервері, покращуючи час завантаження сторінок та забезпечуючи більш ефективне взаємодію з користувачами. Ось опис етапів розробки веб-сайту з використанням SSR:

Перший крок – вибір технологій для розробки. У контексті SSR часто використовуються фреймворки та бібліотеки, такі як React та Next.js. Вони надають зручний інструментарій для роботи з SSR.

Створення нового проекту та його налаштування для використання SSR. Це може включати встановлення необхідних пакетів та конфігурацію проекту для відповідності вимогам SSR.

Розробка React-компонентів, які будуть використовуватися на стороні клієнта та на сервері. Це може включати створення реюзабельних UI-елементів та компонентів для конкретної бізнес-логіки.

Для полегшення завантаження статичних сторінок, які не залежать від динамічних даних, можна використовувати статичний генерацій. Це дозволяє попередньо згенерувати HTML для певних сторінок під час збірки.

Настройка серверної частини для обробки запитів та виконання SSR. Додатково, необхідно розробити механізми для передачі даних з сервера до клієнта, забезпечуючи повноцінний SSR.

Проведення оптимізації для покращення продуктивності, такої як кешування, стиснення ресурсів та інші оптимізаційні методи. Також важливо виконати тестування для переконання, що веб-сайт працює ефективно та коректно на різних пристроях та браузерах.

Тестування [10] веб-додатків із Server-Side Rendering (SSR) є важливою частиною розробки, спрямованою на забезпечення надійності та ефективності веб-сайту. Процес тестування включає різноманітні аспекти, які враховують особливості SSR, починаючи з модульного тестування React-компонентів, розробники переконуються, що кожен компонент працює коректно та відображає очікувані результати. Це важливо, оскільки компоненти використовуються як на стороні клієнта, так і на сервері. Також, оскільки однією з переваг SSR є покращений пошуковий оптимізації (SEO), важливо виконати тестування для визначення, як добре сторінки індексуються пошуковими системами.

Після завершення розробки та тестування, веб-сайт готовий до розгортання на сервері. Важливо вибрати надійне середовище для розгортання, таке як хмарні платформи або власний сервер.

Після розгортання важливо забезпечити регулярне оновлення та підтримку веб-сайту. Це може включати в себе виправлення помилок, оновлення залежностей та розширення функціональності.

Такий підхід до розробки веб-сайту з використанням SSR дозволяє створити продуктивний та швидкий веб-додаток, який задовольнить потреби користувачів та володарів бізнесу.

JavaScript та TypeScript є двома основними мовами програмування, які використовуються в розробці веб-додатків з використанням бібліотеки React та фреймворка Next.js.

Next.js, як фреймворк для React, також використовує JavaScript як основну мову. Всі плюси та можливості JavaScript доступні для розробників Next.js. Фреймворк розширює функціонал React, додаючи можливості серверного рендерингу, статичного генерування та інші, що допомагає оптимізувати продуктивність веб-додатків.

Використання TypeScript в React та Next.js має кілька вагомих переваг. По-перше, статична типізація TypeScript дозволяє виявляти помилки на етапі розробки, що сприяє створенню більш безпечного та надійного коду. Це особливо

корисно при розробці великих та складних проектів, де точність та стабільність є важливими аспектами.

Додатково, TypeScript забезпечує підвищену читабельність коду та активну підтримку в інтегрованих середовищах розробки (IDE). За допомогою статичної типізації, розробники отримують деталізовані підказки та підтримку автодоповнення, що полегшує роботу та підвищує ефективність у процесі кодування.

Підвищення продуктивності є ще однією перевагою використання TypeScript. Зменшення кількості помилок завдяки визначенню типів сприяє швидшій розробці та редагуванню коду. Розробники можуть більш впевнено вносити зміни та розширювати функціонал, знаючи, що система типів виявить потенційні проблеми.

Нарешті, TypeScript полегшує масштабування великих проектів. Визначення структури даних та інтерфейсів дозволяє покращити організацію коду, забезпечуючи зрозумілість та легкість управління проектом у майбутньому. Це особливо важливо в сучасному веб-розвитку, де великі та розширені додатки стають все більш поширеними.

4.2 Створення веб-додатку з використанням SSR

Для створення веб-додатку використовуються завчасно створені пакети від інших розробників або компаній. Це значно спрощує процес розробки, оскільки багато типових задач вже вирішені й доступні у вигляді готових рішень. Наприклад, завдяки open-source підходу, такі компанії-гіганти, як Facebook, надають можливість безкоштовно користуватися своїми розробками, такими як бібліотека React або фреймворк Next.js. Ці інструменти є потужними та добре задокументованими, що робить їх дуже популярними серед розробників у всьому світі.

React, розроблений Facebook, є бібліотекою JavaScript для створення користувацьких інтерфейсів. Його ключова особливість — компонентний підхід, що дозволяє створювати багаторазові компоненти, які можна використовувати в

різних частинах програми. Це значно полегшує розробку та підтримку великих додатків. React також забезпечує високу продуктивність завдяки віртуальному DOM, що мінімізує операції з реальним DOM і, таким чином, покращує швидкодію.

Next.js, який також підтримується і розвивається за участю Facebook, є фреймворком для створення серверних рендерних додатків на основі React. Next.js пропонує такі можливості, як автоматичне розділення коду, генерація статичних сайтів, серверний рендеринг та багато інших. Це робить його відмінним вибором для створення як простих веб-сайтів, так і складних веб-додатків. За допомогою Next.js розробники можуть швидко створювати високопродуктивні та SEO-оптимізовані додатки.

Використання таких інструментів значно спрощує процес розробки веб-додатків. Розробники можуть зосередитися на логіці та функціональності своїх програм, не витрачаючи час на вирішення типових задач, які вже вирішені в цих бібліотеках та фреймворках. Крім того, велика спільнота користувачів та розробників означає, що для цих інструментів існує багато навчальних матеріалів, прикладів коду, а також готових рішень для різних задач.

Таким чином, використання завчасно створених пакетів і бібліотек значно прискорює процес розробки веб-додатків, дозволяючи зосередитися на унікальних аспектах проекту. В результаті розробники можуть швидше створювати якісні продукти, використовуючи найкращі практики та інструменти, розроблені спільнотою.

Для створення додатку Next.js, використаємо завчасно створені open-source пакети, та завантажимо їх за допомогою npm (див. рис. 4.1)

Щоб розпочати роботу зі створення веб-додатку за допомогою Next.js, необхідно встановити необхідні пакети та налаштувати проект. Спочатку переконаємося, що у нас встановлений Node.js і npm (Node Package Manager). Якщо вони ще не встановлені, завантажте і встановіть їх з офіційного сайту Node.js.

```
PS C:\Users\maxme\diplom-ssr> npx create-next-app@latest
>>
Need to install the following packages:
create-next-app@14.2.3
Ok to proceed? (y) y
√ What is your project named? ... my-app
√ Would you like to use TypeScript? ... No / Yes
√ Would you like to use ESLint? ... No / Yes
√ Would you like to use Tailwind CSS? ... No / Yes
√ Would you like to use `src/` directory? ... No / Yes
√ Would you like to use App Router? (recommended) ... No / Yes
√ Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in C:\Users\maxme\diplom-ssr\my-app.

Using npm.

Initializing project with template: app


Installing dependencies:
- react
- react-dom
- next
```

Рисунок 4.1 – Створення Next.js додатку через npm

Після встановлення Node.js і npm, відкриємо командний рядок або термінал і скористаємося командою для створення нового проекту на базі Next.js. Команда дозволяє швидко налаштувати новий проект зі стандартними налаштуваннями. Для цього створимо новий додаток і надамо йому назву, наприклад, "my-app".

У процесі створення проекту Next.js буде згенеровано всі необхідні файли та структури для нашого додатку. Це включає файли конфігурації, початкові сторінки, компоненти та інші необхідні ресурси.

Таким чином, ми встановили пакет Next.js і обрали необхідні налаштування для нашого проекту. Тепер ми готові запустити наш додаток в режимі розробки, щоб побачити його в дії. Для цього перейдемо в папку нашого проекту і запустимо сервер розробки.

Після запуску серверу розробки ми побачимо повідомлення в терміналі про те, що сервер працює і наш додаток доступний за адресою http://localhost:3000.

Відкриймо веб-браузер і перейдемо за цією адресою. Ми побачимо початкову сторінку нашого Next.js додатку.

На цьому етапі ми маємо базовий шаблон Next.js додатку, який можна використовувати як відправну точку для подальшої розробки. Ми можемо почати додавати нові сторінки, компоненти та функціонал, адаптуючи додаток до наших потреб.

Наприклад, щоб створити нову сторінку, додамо новий файл у папку pages. Припустимо, ми хочемо створити сторінку «About Us». Для цього створимо файл about.js у папці pages, де визначимо простий React-компонент, який буде відображати заголовок та текст на сторінці «About Us». Після збереження цього файлу ми можемо відвідати сторінку за адресою http://localhost:3000/about, і побачимо наш новий контент.

Next.js також підтримує динамічні маршрути, що дозволяє створювати сторінки з параметрами у URL. Наприклад, можна створити сторінки для окремих статей блогу, користувачів або продуктів, використовуючи шаблонні файли та динамічні маршрути.

Крім того, Next.js надає інструменти для серверного рендерингу, статичної генерації сайтів, оптимізації продуктивності та багато іншого. Ці функції роблять Next.js потужним інструментом для розробки сучасних веб-додатків, що можуть масштабуватись і задовольняти потреби будь-якого проекту.

Отже, ми розглянули основні кроки для створення та запуску Next.js додатку, включаючи встановлення необхідних пакетів, налаштування проекту та запуск сервера розробки. Тепер ми можемо приступати до розробки і додавання нового функціоналу, використовуючи потужні можливості Next.js для створення сучасних веб-додатків.

Таким чином, завдяки використанню інструментів і бібліотек, які надаються Next.js, ми можемо швидко створити повноцінний веб-додаток з усіма необхідними функціями. Це значно спрощує процес розробки, зменшуючи час на налаштування та дозволяючи зосередитися на створенні унікального контенту та функціоналу. Next.js також забезпечує високу продуктивність і оптимізацію для пошукових систем, що є важливим аспектом для сучасних веб-додатків.

Завдяки великій спільноті користувачів та розробників, Next.js має багато навчальних матеріалів, прикладів коду та готових рішень для різних задач. Це робить його відмінним вибором для розробників будь-якого рівня, від новачків до професіоналів.

Запуск нашого додатку на локальному сервері дає нам можливість тестувати і бачити результати змін в режимі реального часу. Це дуже зручно для розробки, оскільки дозволяє швидко виявляти і виправляти помилки, а також експериментувати з новими функціями і дизайном. Рисунок 4.2 демонструє, як виглядає ця сторінка у веб-браузері при запуску серверу на локальному хості.

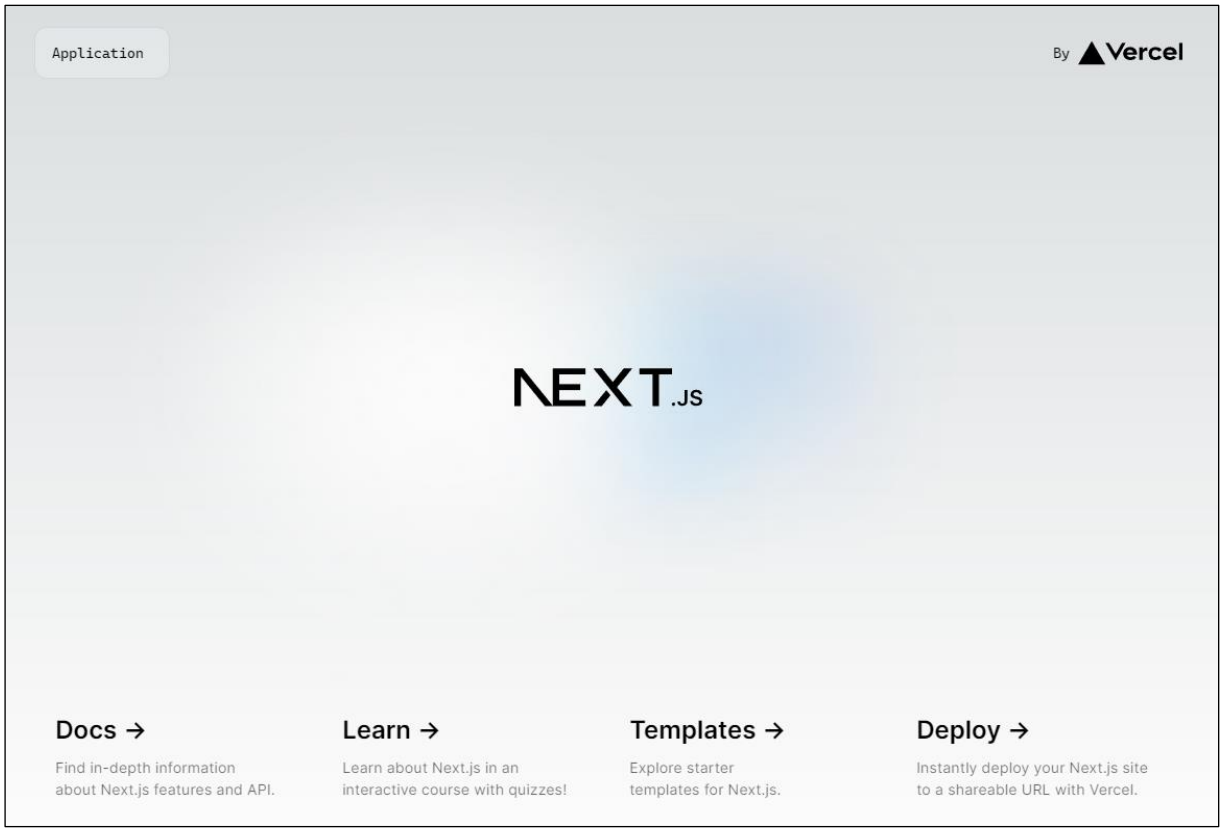

Рисунок 4.2 – Результат створеного додатку

Втім, це тільки початок. На даному етапі наш застосунок запускається на клієнті, тобто на комп'ютері користувача, у форматі SSG (Static Site Generation). Це означає, що всі сторінки нашого додатку генеруються заздалегідь під час побудови і зберігаються як статичні HTML-файли. Такий підхід має свої переваги, включаючи швидкість завантаження сторінок і кращу SEO-оптимізацію, оскільки контент вже доступний для індексації пошуковими системами.

Проте, в деяких випадках нам може знадобитися динамічне завантаження даних або рендеринг сторінок на сервері, залежно від запиту користувача. У таких випадках ми можемо змінити підхід створення компонентів, використовуючи інші методи рендерингу, які пропонує Next.js.

Нижче продемонстровано стандартний підхід CSR (Client Side Generation) (див. рис. 4.3)

```
export default function Csr() {
  const [serverTime, setServerTime] = useState("");

  useEffect(() => {
    const fetchServerTime = async () => {
      const response = await fetch("/api/time");
      const data = await response.json();
      setServerTime(data.serverTime);
    };

    fetchServerTime();
  }, []);

  return (
    <>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className={`${styles.main} ${inter.className}`}>
        <div className={styles.description}>
          <p>
            This is CSR page
          </p>
          <div>
```

Рисунок 4.3 – Підхід CSR у компоненті Next.js

Нам потрібно це змінити на підхід SSR, тож змінимо функції отримання серверних пропсів (див. рис. 4.4).

У цьому випадку функція getServerSideProps викликається на кожен запит до сторінки, забезпечуючи завжди актуальні дані, що можуть змінюватися динамічно. Це підходить для сторінок, які потребують актуалізації контенту при

кожному запиті, наприклад, для дашбордів, профілів користувачів або сторінок з динамічним контентом.

```
export async function getServerSideProps() {
  // Fetch server time
  const serverTime = new Date().toString();


  return {
    props: {
      serverTime,
    },
  };
}

export default function Home({ serverTime }) {
  return (
    <>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      <main className={`${styles.main} ${inter.className}`}>
        <div className={styles.description}>
          <p>
            this is SSR page

          </p>
          <div>
            <a
              href="https://vercel.com?utm_source=create-next-app&utm_medium=default-template&utm_campaign=create-next-app"
              target="_blank"
              rel="noopener noreferrer"
            >
              By{" "}
              <Image
```

Рисунок 4.4 – Підхід SSR у компоненті Next.js

Завдяки цьому підходу, сторінка завжди відображає найсвіжішу інформацію, що може бути критично важливо для різних типів застосунків. Наприклад, в адміністративних панелях і дашбордах, де постійно оновлюються дані, такі як статистика, метрики або стан системи, важливо мати можливість оперативно відображати поточну інформацію.

У випадку з профілями користувачів, getServerSideProps забезпечує завантаження найновіших даних про користувача, його активність та інші релевантні дані, що можуть змінюватися в режимі реального часу. Це робить взаємодію з сайтом більш інтерактивною та персоналізованою.

Також, на сторінках з динамічним контентом, таких як новини, блоги або коментарі, важливо показувати користувачам найактуальнішу інформацію без

необхідності ручного оновлення сторінки. getServerSideProps дозволяє автоматично завантажувати новий контент під час кожного запиту, забезпечуючи безперервний потік інформації.

Крім того, цей підхід корисний для сайтів, які інтегруються з зовнішніми API або базами даних, де дані часто змінюються. Відтак, кожен запит до сторінки може включати в себе найсвіжіші результати від зовнішніх сервісів, підвищуючи загальну ефективність та релевантність відображуваного контенту.

4.3 Швидкість рендерінгу в залежності від вибіру SSR

Якщо ми порівняємо швидкість завантаження сторінки з контентом залежно від вибору методу рендерінгу, то побачимо, що рендерінг на сервері є більш оптимізованим та швидким з точки зору часу завантаження для кінцевого користувача. Це пояснюється тим, що серверна обробка дозволяє підготувати весь необхідний контент до відправки клієнту, скорочуючи час, який користувач витрачає на очікування завантаження і обробки даних на своєму пристрої.

Проте, варто враховувати, що такий підхід перекладає значне навантаження на сервер, особливо при великій кількості одночасних користувачів. В той час, як статичне генерація (SSG) створює сторінки заздалегідь, зберігаючи їх у вигляді готових HTML-файлів, що дозволяє значно зменшити навантаження на сервер під час кожного запиту, серверний рендерінг (SSR) виконує повний цикл обробки даних та формування сторінки при кожному запиті.

З одного боку, SSR забезпечує більш актуальний і динамічний контент, що особливо важливо для додатків з високою частотою оновлення даних, таких як інформаційні панелі, новинні портали або сторінки з персоналізованим контентом. З іншого боку, SSG є більш ефективним з точки зору ресурсів сервера, оскільки більша частина обробки відбувається на етапі збірки, а не під час кожного запиту.

Щоб збалансувати переваги обох підходів, можна використовувати гібридні методи рендерінгу. Наприклад, поєднання SSG з ISR (Incremental Static Regeneration) дозволяє оновлювати статичні сторінки періодично, забезпечуючи

свіжість контенту без значного збільшення навантаження на сервер. Інший варіант – використовувати CSR (Client-Side Rendering) для менш критичних частин контенту, дозволяючи серверу зосередитись на обробці найбільш важливих даних. Давайте подивимось на швидкість рендерінгу сторінки з SSR (див. рис. 4.5)

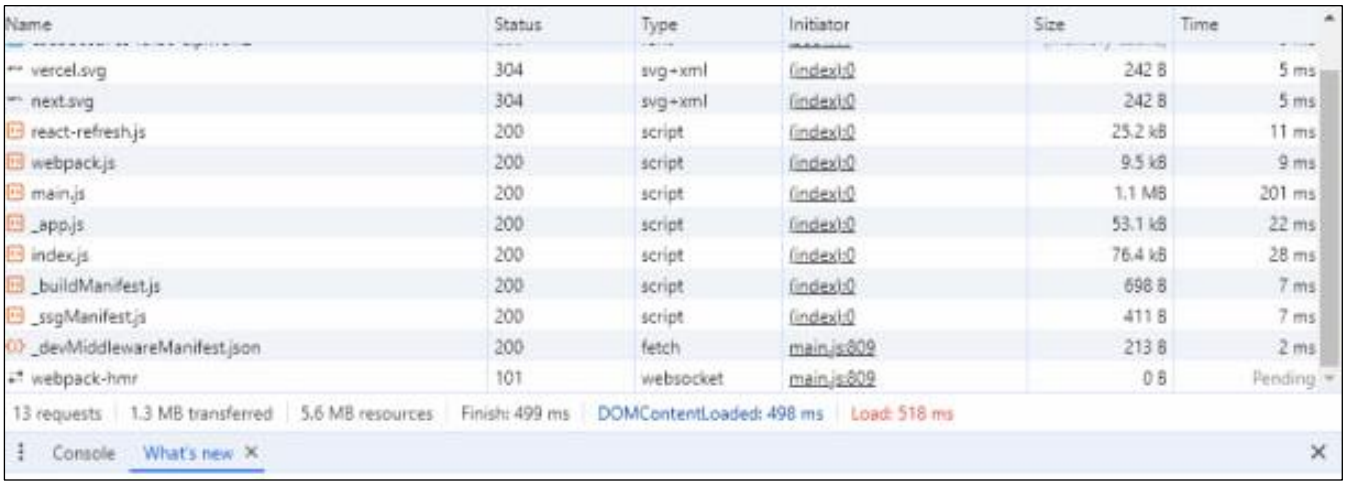

Рисунок 4.5 – Результат швидкості рендерінгу сторінки з SSR

Як можемо побачити, за 499 мілісекунд, сторінка була завантажена. Це досить швидкий результат. Якщо порівняти це з CSR (див. рис. 4.6), то можемо побачити значну різницю в швидкості.

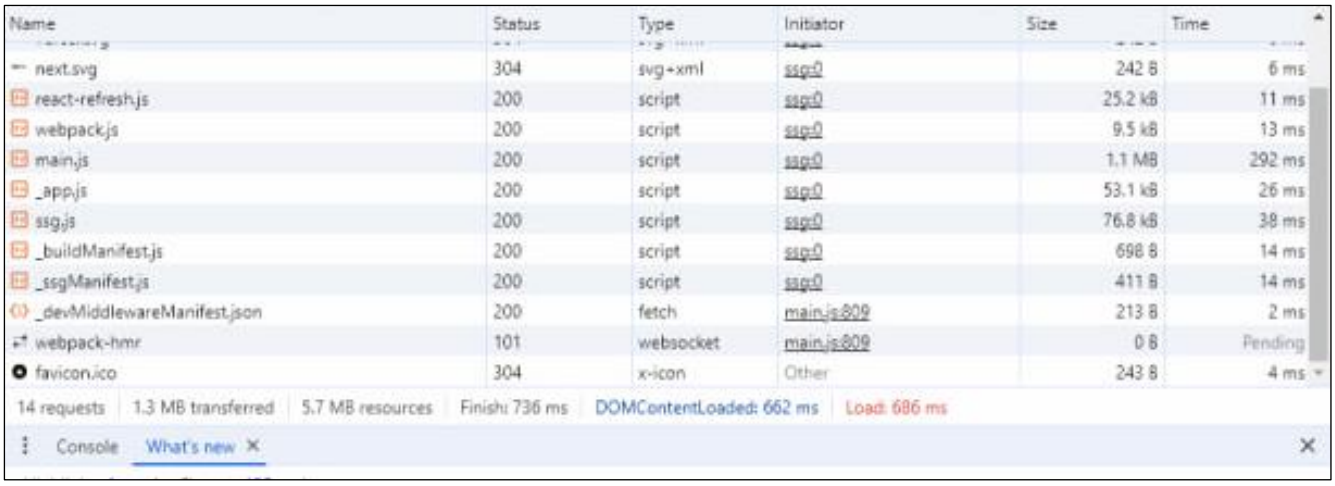

Рисунок 4.6 – Результат швидкості рендерінгу сторінки з CSR

Тож, навіть на прикладі з невеликим навантаженням, серверний рендерінг суттєво швидший, ніж клієнтський. Сервер готує весь необхідний контент до

відправки клієнту, що зменшує час завантаження сторінки для кінцевого користувача. Коли наш додаток стає більш складним і включає в себе різні складні обчислення, навантаження на клієнта значно зростає при клієнтському рендерінгу.

У випадках, коли додаток виконує багато важких обчислень або обробляє великі обсяги даних, серверний рендерінг стає ще більш виправданим. Сервер, як правило, має більше ресурсів для обробки складних задач порівняно з кінцевими користувачами, чиї пристрої можуть бути менш потужними. Це означає, що, виконавши ці обчислення на сервері, ми можемо забезпечити більш плавний і швидкий досвід для користувачів, оскільки їх пристрої отримують вже підготовлені дані, а не виконують важкі операції самостійно.

ВИСНОВКИ

Дослідження та створення веб-застосунків з використанням Server-Side Rendering (SSR) на основі React та Next.js виявилося стратегічним кроком для досягнення високої продуктивності, швидкості завантаження та ефективного управління станом. Аналіз використання SSR у веб-розробці, зосереджений на реалізації з використанням бібліотек React та фреймворка Next.js, видокремив ряд важливих аспектів та переваг.

SSR визначається як метод, при якому генерація HTML відбувається на сервері, сприяючи зменшенню часу завантаження сторінки та підвищенню відгукливості інтерфейсу. У контексті SSR роль React, як провідної JavaScript-бібліотеки, виявляється важливою, забезпечуючи серверний рендеринг та створення динамічних інтерфейсів.

Next.js, як популярний фреймворк для SSR, взаємодіє з React, спрощуючи реалізацію серверного рендерингу та надаючи ефективний інструментарій для управління процесом розробки. Багато великих компаній, включаючи Facebook, обирають використання SSR для покращення часу завантаження та оптимізації для пошукових систем.

Переваги використання SSR включають покращений SEO, зменшення навантаження на клієнтську сторону та краще управління станом. Однак існують виклики, такі як серверні витрати ресурсів і необхідність уважного управління кешуванням.

Застосування SSR також позитивно впливає на індексацію веб-додатку пошуковими системами та відкриває можливості для підтримки сторонніх сервісів та інтеграції з зовнішніми додатками.

Розгляд використання SSR у веб-розробці на основі React та Next.js дозволяє виокремити кілька ключових аспектів, що впливають на вибір цього методу рендерингу. Перевагою SSR є поліпшення індексації веб-додатків пошуковими системами. Через те, що генерація HTML відбувається на сервері, весь контент доступний для пошукових роботів, що може позитивно впливати на рейтинг та висновки в результатах пошуку.

Окрім цього, важливим аспектом є можливість оптимізації сайту для підвищення продуктивності. Генерація сторінок на сервері дозволяє ефективніше використовувати ресурси сервера та зменшує обсяг передаваних даних, оскільки клієнту відправляється вже готовий HTML. Це сприяє швидшому завантаженню сторінок та кращій відзивчивості веб-додатку.

Крім того, використання SSR відкриває можливості для підтримки сторонніх сервісів та інтеграції з зовнішніми додатками. Аналітичні інструменти, маркетингові платформи та соціальні мережі можуть легше взаємодіяти з веб-додатком, який використовує SSR, завдяки доступності HTML.

Таким чином, обрання SSR для веб-додатку базується на комплексі переваг, спрямованих на поліпшення SEO, збільшення продуктивності, підтримку інтеграції та кращий взаємозв'язок з пошуковими системами. Цей підхід стає ключовим для створення швидких, ефективних та конкурентоспроможних веб-додатків з використанням React та Next.js

Отже, обрання SSR у контексті розробки веб-додатків є обґрунтованим стратегічним рішенням для забезпечення високої продуктивності, поліпшення користувацького досвіду та конкурентоспроможності.

В ході дослідження було продемонстровано, що серверній підхід рендерінгу є більш швидким та оптимальним для більшості задач бізнесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Michele Riva. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for productio / Publisher: Packt Publishing, 2022, – 366 p.

2. Stoyan Stefanov. React: Up & Running: Building Web Applications / Publisher: O'Reilly Media, 2016, – 219 p.

3. Douglas Crockford. The Good Parts: The Good Parts / Publisher: Yahoo Press, 2020, – 170 p.

4. Yakov Fain, Anton Moiseev. TypeScript Quickly / Publisher: Manning, 2020 – 350 p.

5. Marc Garreau, Will Faurot. Redux in Action / Publisher: Manning, 2018 – 312 p.

6. Jon Duckett. HTML and CSS: Design and Build Websites / Publisher: John Wiley & Sons, 2011 – 490 p.

7. Peter Kent. Search Engine Optimization for Dummies / Publisher: For Dummies, 2012, – 456 p.

8. Gatsby.js vs. Next.js — Which is the Best React Framework? (+ SSG/SSR Rendering Option) / URL: https://serinryu.medium.com/gatsby-js-vs-next-js-which-is-the-best-react-framework-ssg-ssr-rendering-option-55672aa6e321 (дата звернення: 10.05.2024).

9. JavaScript Testing Best Practices to Follow / URL: https://www.browserstack.com/guide/javascript-testing-best-practices (дата звернення: 13.05.2024).

10. Gerardus Blokdyk. Web Application Testing A Complete Guide – 2023 Edition / Publisher: The Art of Service - Web Application Testing Publishing, 2023, – 313 p.