ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ/INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.60797/IRJ.2025.162.108

SERVER COMPONENTS В NEXT.JS 13: РЕВОЛЮЦИЯ В ПОДХОДАХ К РЕНДЕРИНГУ НА СЕРВЕРЕ И КЛИЕНТЕ

Научная статья

Шагилова Е.В.[1, *]

[1] Мордовский государственный университет им. Н.П. Огарева, Саранск, Российская Федерация

* Корреспондирующий автор (internet_mrsu[at]mail.ru)

Аннотация

Цель данной статьи — рассмотрение теоретических и практических аспектов Server Components, возможностей для повышения производительности и упрощения архитектуры веб-приложений. Статья анализирует Server Components в Next.js 13, их архитектурные особенности и роль в оптимизации процессов разработки веб-приложений. Особое внимание уделяется влиянию на производительность за счет сокращения клиентского бандла и улучшения пользовательского опыта. Рассматриваются принципы работы Server Components и их потенциал для создания высокопроизводительных, масштабируемых систем. Выявлены новые возможности для создания сложных и интерактивных веб-приложений. Проведен сравнительный анализ технологий рендеринга.

Ключевые слова: Server Components, Next.js 13, серверный рендеринг, клиентская интерактивность, оптимизация производительности, веб-разработка.

SERVER COMPONENTS IN NEXT.JS 13: A REVOLUTION IN APPROACHES TO RENDERING ON THE SERVER AND CLIENT

Research article

Shagilova E.V.[1, *]

[1] Mordovian State University named N.P. Ogarev, Saransk, Russian Federation

* Corresponding author (internet_mrsu[at]mail.ru)

Abstract

The aim of this article is to examine the theoretical and practical aspects of Server Components, opportunities for improving performance and simplifying web application architecture. The paper analyses Server Components in Next.js 13, their architectural traits and role in optimising web application development processes. Particular attention is paid to the impact on performance by reducing client bundle size and improving user experience. The principles of Server Components and their potential for creating high-performance, scalable systems are reviewed. New opportunities for creating complex and interactive web applications are identified. A comparative analysis of rendering technologies is conducted.

Keywords: Server Components, Next.js 13, server-side rendering, client-side interactivity, performance optimisation, web development.

Введение

Релиз Server Components (серверные компоненты) в версии Next.js 13 состоялся в 2022 году, открывая новые возможности для веб-разработки.

Server Components — это новый тип компонентов React, которые работают на сервере и возвращают скомпилированный JSX, отправляемый клиенту. Использование серверных компонентов даёт возможность более точно контролировать состояние приложения и переносить обработку информации на сервер. Эта технология, предлагая переработанную модель рендеринга, упрощает не только взаимодействие между серверной и клиентской частями, но и предоставляет возможность повысить эффективность работы приложений

Целью данной работы является исследование теоретических и практических аспектов технологии Server Components, а также их возможностей для повышения производительности и упрощения архитектуры веб-приложений. Особое внимание уделяется практической стороне вопроса: как меняется архитектура проекта, как перераспределяются вычисления между клиентом и сервером и в какой мере это отражается на скорости работы и удобстве разработки.

Для достижения поставленной цели решаются следующие задачи:

1. Рассмотреть принципы работы Server Components и понять, чем они отличаются от привычных подходов рендеринга в веб-разработке.
2. Проанализировать влияние серверных компонентов на реальную производительность: на скорость первого отображения, общий объём загружаемого клиентом JavaScript и поведение приложения под нагрузкой.
3. Сопоставить три подхода к рендерингу — CSR, SSR и Server Components — на основе экспериментальных замеров и единых критериев оценки.
4. Выяснить, как новая архитектура отражается на безопасности и SEO, а также на разделении логики между клиентом и сервером.
5. Определить ситуации, в которых Server Components дают наибольший практический эффект, и сферы разработки, где их применение выглядит наиболее оправданным.

Методы и принципы исследования

Исследования, посвящённые современным подходам к рендерингу веб-приложений, в последние годы сосредоточены вокруг поиска оптимального баланса между производительностью, интерактивностью и безопасностью. Традиционно противопоставлялись два подхода — client-side rendering (CSR) и server-side rendering (SSR), однако с развитием фреймворков нового поколения, таких как Next.js 13, появилась концепция Server Components, открывающая возможности гибридного рендеринга.

В материале А. Гусева «Что нового в Next.js 13?» описывается переход от традиционной системы маршрутизации к новой папке app/, появление серверных компонентов и директивы use client [1]. Статья даёт исторический контекст внедрения технологии и объясняет предпосылки перехода к гибридной модели рендеринга, когда часть логики выполняется на сервере, а интерактивность сохраняется на клиенте.

Анализ архитектурных преимуществ новой модели рендеринга представлен в публикации П. Андреева [2], где рассматриваются различия между классическим серверным рендерингом и концепцией Server Components. Автор обращает внимание на то, что новый подход устраняет ключевые ограничения SSR — прежде всего избыточную передачу клиентского JavaScript-кода и необходимость полной гидратации интерфейса. В работе делается вывод, что использование Server Components способствует более рациональному распределению вычислительной нагрузки и повышает отзывчивость пользовательского интерфейса.

В исследовании И. Козлова «SSR, SSG, ISR и Server Components: сравнительный анализ подходов к рендерингу» [3] проводится детальное сопоставление существующих стратегий отображения веб-контента по параметрам скорости, оптимизации и поисковой видимости. Автор демонстрирует, что гибридная архитектура Server Components сочетает преимущества серверного рендеринга с высокой интерактивностью клиентских компонентов, обеспечивая стабильную производительность даже при работе с динамическими данными. Методологически данное исследование близко к практическому анализу, представленному в данной статье, что позволяет рассматривать их в едином контексте эволюции современных подходов к веб-рендерингу.

Также необходимо отметить современное исследование Р.Ф. Гибадуллина и Д.А. Гашигуллина «Оптимизация асинхронных операций в .NET» [4], в котором сформулированы принципы оптимизации асинхронных операций. В работе делается вывод, что асинхронное программирование в .NET представляет собой мощный инструмент для повышения производительности и отзывчивости приложений.

Отдельного внимания заслуживают материалы российских профессиональных сообществ. На конференции HolyJS 2023 был представлен доклад «React Server Components» [5], в котором обсуждались практические аспекты применения Server Components при разработке крупных интерфейсов. В докладе подчёркивается, что новая архитектура позволяет снизить нагрузку на клиент и повысить безопасность за счёт серверной обработки данных.

Практические руководства, такие как локализованная документация Next.js (ru.nextjs.im) [6] и обучающие материалы В. Куликова (Frontend Union, 2024) [7] и V. Patel (International Journal of Computer Applications Technology and Research, 2023) [8], обеспечивают разработчиков инструментальной базой для внедрения Server Components и иллюстрируют архитектурные решения на реальных проектах.

Обобщая существующие исследования, можно выделить несколько тенденций:

1) Русскоязычные работы преимущественно фокусируются на обзорно-практическом описании технологии, её архитектурных особенностях и преимуществах, тогда как эмпирических исследований производительности и метрик (LCP, TTI, размер клиентского бандла) в условиях реальных нагрузок пока недостаточно.

2) Вопросы безопасности, распределения состояния и управления серверно-клиентской логикой в рамках Server Components остаются открытыми и требуют дальнейшего анализа.

Таким образом, существующая литература формирует прочную теоретическую и практическую основу для изучения Server Components в Next.js 13, однако оставляет пространство для дальнейших исследований, направленных на экспериментальную оценку производительности, анализ архитектурных компромиссов и разработку методик оптимизации гибридного рендеринга. Данная работа продолжает и расширяет эти направления, предлагая системный анализ влияния Server Components на эффективность и структуру современных веб-приложений.

Основные результаты

В Next.js 13 была представлена новая система маршрутизации, которая упрощает интеграцию Server Components в архитектуру веб-приложений. Маршрутизация использует папку app для организации маршрутов, которая имеет четкое разделение между компонентами, благодаря ей разработчик может легкоразделить серверы и клиентские компоненты [9]. Каждый компонент по умолчанию является серверным, то есть его код выполняется и остается на стороне сервера, если нужно взаимодействие с клиентом, то необходимо использовать директиву useclient в начале кода компонента, что позволит сделать его клиентским. Такой подход позволяет легко понимать, какой код выполняется на стороне клиента, а какой остается на стороне сервера [10].

Данная директива делает разработку интуитивно понятной, благодаря ей можно четко указать, где должен выполняться код, а также она совсем не требует какой-либо дополнительной настройки. Такой подход делает код более читаемым, что в свою очередь позволяет ускорить разработку.

Главным отличием между Server Components от SSR заключается в том, что, SSR работает на уровне всей страницы, а Server Components на уровне компонента, что и следует из его названия. Это позволяет комбинировать серверные и клиентские компоненты в дереве рендеринга по усмотрению разработчика.

Система App Router представляет гибкую организацию server-side и client-side частей кода, что упрощает разработку и поддержку приложения. Разработчики могут использовать различные подходы для разных компонентов UI, выбирая между Server и Client Components в зависимости от нуждаемости конкретной части приложения [11]. Этот подход позволяет намного легче поддерживать баланс между производительностью и интерактивностью, а также, улучшать пользовательский опыт.

Server Components открывают новые возможности для оптимизации работы с данными. Благодаря выполнению запросов к базам данных непосредственно на сервере, можно избежать передачи избыточной информации на клиентскую сторону. Это особенно актуально для приложений, работающих с большими объемами данных, где каждая миллисекунда загрузки имеет значение. Например, можно получить только необходимые поля из базы данных и сразу же отрендерить их в Server Component, отправляя клиенту уже готовый HTML.

Еще одним важным преимуществом Server Components является улучшенная безопасность. Поскольку конфиденциальные данные и логика обработки находятся на сервере, риск их утечки на клиентскую сторону значительно снижается. Это позволяет безопасно работать с секретными ключами, токенами доступа и другими чувствительными данными, не опасаясь, что они попадут в руки злоумышленников.

Использование Server Components также способствует улучшению SEO. Благодаря тому, что контент генерируется на сервере и сразу доступен поисковым роботам, индексация страниц происходит быстрее и эффективнее. Это особенно важно для сайтов, стремящихся занять высокие позиции в результатах поиска.

В целом, Server Components представляют собой мощный инструмент для создания высокопроизводительных, безопасных и SEO-оптимизированных веб-приложений. Они позволяют значительно улучшить пользовательский опыт, снизить нагрузку на клиентскую сторону и упростить архитектуру приложения.

Обсуждение

Технология Server Components позволяет оптимизировать производительность, ускоряя начальную загрузку и весь процесс работы приложения, открывая множество возможностей для оптимизации производительности веб-приложений и улучшения пользовательского опыта. Перераспределение вычислительной нагрузки на сервер позволяет существенно уменьшить размер клиентского бандла. Это особенно важно для пользователей с довольно слабым устройством, например, мобильным телефоном, а также тех, для кого интернет-соединение не самое быстрое. Уменьшение передаваемых данных на клиентскую сторону напрямую влияет на время загрузки приложения, а это значит, повышает производительность и сокращает вероятность задержек в его отображении. К тому же, Server Components взаимодействуют с сервером и ускоряют генерацию контента. Так как сервер выполняет все вычисления и принимает ваши запросы заранее, то на экране клиента видны данные сразу, это сокращает время ожидания при загрузке страницы. При этом улучшаются также важные системные метрики, такие как Core Web Vitals (метрики веб-страниц), в которых заметны Largest Contentful Paint (LCP) и First Contentful Paint (FCP) в первую очередь.

Отдельные элементы интерфейса можно создавать на стороне сервера для того, чтобы добиться высокой производительности приложения, а другие элементы сделать интерактивными и добавить к ним клиентскую логику. Такой подход позволяет создавать и поддерживать динамические пользовательские интерфейсы, при этом скрывая бизнес логику от пользователя на сервере.

Все это очень удобно при разработке сложных интерфейсов, таких как например, панель управления или же инструменты аналитики, в которых важна высокая производительность приложения, но при этом оно должно оставаться интерактивным, а также показывать динамический контент [12].

Сравнительный анализ технологий рендеринга

Веб-приложения в наши дни требуют оптимизации рендеринга, для того чтобы понять, какая технология наиболее перспективная на данный момент, был проведён сравнительный анализ три ключевых подхода (рис. 1).

Анализ проводился в одинаковых условиях, объектом тестирования служило веб приложение на Next.js 13, с идентичной функциональностью, для измерений использовались:

· Lighthouse — оценка FCP (First Contentful Paint), TTI (Time to Interactive).

· k6 — нагрузочное тестирование (запросов в секунду, RPS).

· Webpack Bundle Analyzer — анализ объема клиентского кода.

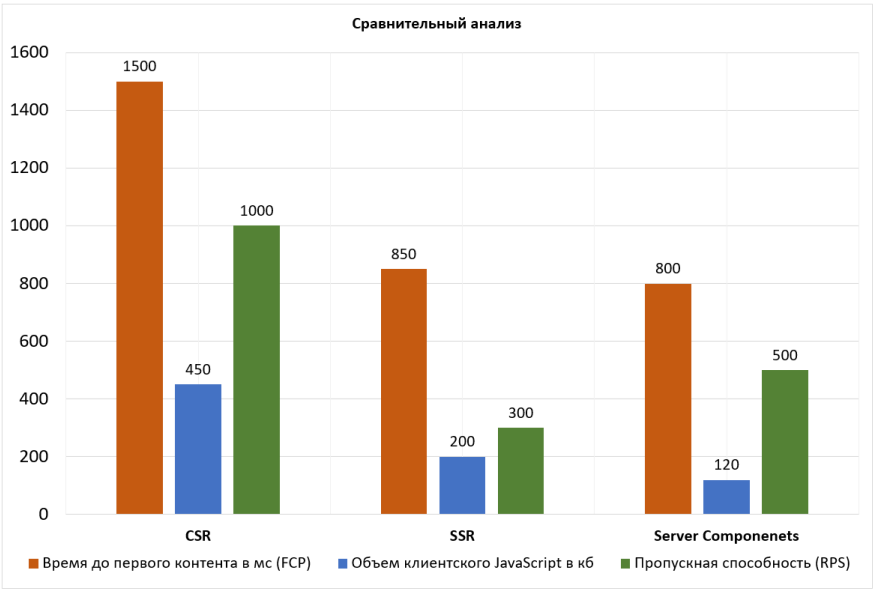

Рисунок 1 - Гистограмма сравнительного анализа
DOI: https://doi.org/10.60797/IRJ.2025.162.108.1

Из гистограммы видно, что CSR имеет самый плохой показателя FCP(~1500 мс), так как клиенту передается JS код и браузер блокирует интерактивность до его выполнения. Также видно, что объем, передаваемый JS, является максимальным среди этих технологий (~450 КБ), так как весь рендеринг выполняется на стороне клиента. Пропускная способность (до 1000 RPS) однако выше других технологий из-за того, что вся нагрузка происходит на стороне клиента, а сервер в свою очередь только передает статичные файлы, не тратя свои ресурсы на рендер.

Идеальные кейсы использования этой технологии — это SPA с высокой интерактивностью, где не так важны SEO показатели.

SSR, в свою очередь, имеет средний показатель FCP (~850 мс). Это лучше, чем у CSR, но все ещё хуже, чем у Server Components. При использовании SSR HTML генерируется на сервере, но клиент ждет гидрации.

Гидрация – это процесс, при котором клиентский JavaScript добавляет интерактивность к статическому HTML, полученному с серверной части (SSR), например, добавляя разные обработчики событий.

Объем передаваемого кода на клиент также имеет средний показатель (~200кб), лучше, чем CSR, но все также хуже, чем у Server Components, и тут опять требуется полная гидрация интерфейса. Пропускная способность RPS самая низкая из трех представленных технологий (~300), так как вся нагрузка происходит на стороне сервера при рендеринге каждой страницы.

Идеальные кейсы использования технологий — это SEO-критичные проекты с частыми обновлениями, например, новостные сайты.

У Server Components показатель FCP самый высокий (~800 мс) из представленных методов рендера, он близко к SSR, но выигрывает за счёт того, что не Server Components обходится без полной гидрации интерфейса. Также показатель объема передаваемого JavaScript кода (~120 КБ) лучший среди представленных технологий за счёт того, что передается на клиент только код для интерактивных компонентов. Показатель пропускной способность RPS (~500) средний, так как сервер рендерит только часть страницы.

Идеальные кейсы использования SC — это большие динамические приложения с тяжелыми данными, например, админ-панели, аналитика, где важны свежие данные и быстрое время интерактивности.

ServerComponents являются наиболее перспективной технологией на данный момент так как они устраняют недостатки SSR и CSR.

Заключение

Server Components в Next.js 13 на данный момент являются самой актуальной и перспективной технологией, устраняющей недостатки других. Идеально подходит для создания современных высоконагруженных и масштабируемых веб-приложений, минимизируя при этом использование JavaScript на стороне клиента, а также, соблюдая баланс между серверной обработкой и интерактивностью на стороне клиента. Также Server Components позволяют сосредоточиться над функциональностью и удобством использования, не обращая на специфику сервера или клиента, давая возможность беспрепятственно интегрировать логику между сервером и клиентом.

Благодаря Server Components, разработчики могут оптимизировать процесс получения данных, выполняя запросы к базам данных и API непосредственно на сервере. Это не только повышает безопасность, предотвращая утечку конфиденциальной информации на клиентскую сторону, но и снижает нагрузку на клиентское устройство, поскольку ему не нужно выполнять сложные вычисления или запросы данных.

Server Components открывают новые возможности для создания сложных и интерактивных веб-приложений. Разработчики могут использовать их для предварительной обработки данных, рендеринга сложных UI-компонентов и выполнения других серверных операций, которые ранее требовали значительных усилий и ресурсов.

Новизна исследования заключается в том, что Server Components рассматриваются не только как очередное нововведение в Next.js, но и как важный шаг в сторону пересмотра привычной модели фронтенд-разработки. В работе соединён теоретический анализ технологии с практическими экспериментами, в ходе которых оцениваются реальные метрики поведения приложения. Дополнительный элемент новизны состоит в оценке технологии с точки зрения перераспределения вычислительной нагрузки и особенностей передачи данных между сервером и клиентом — аспектов, которые редко рассматриваются комплексно.

Полученные результаты отличаются оригинальностью по нескольким направлениям:

1) Исследование показывает, как именно Server Components меняют структуру веб-приложения на практике: что переносится на сервер, что остаётся клиенту, и как это влияет на использование ресурсов.

2) Экспериментальный сравнительный анализ позволяет не теоретически, а на реальных измерениях показать разницу между CSR, SSR и Server Components — в скорости загрузки, объёме скриптов и устойчивости под нагрузкой.

3) Работа предлагает собственный набор рекомендаций по выбору стратегии рендеринга в зависимости от типа проекта, особенностей данных и приоритетных метрик.

4) Исследование подчёркивает архитектурные преимущества Server Components для сложных интерфейсов, где важно объединить высокую интерактивность, безопасность и производительность.

Таким образом, внедрение Server Components в экосистему Next.js 13 позволяет объединить преимущества серверной и клиентской обработки, достигая оптимального соотношения между скоростью работы, безопасностью и удобством сопровождения приложений. Использование серверных компонентов способствует более рациональному распределению вычислительных ресурсов и снижает зависимость от клиентского оборудования, что особенно актуально для сложных и нагруженных систем.

Применение Server Components не только изменяет архитектурные принципы построения интерфейсов, но и облегчает сам процесс разработки: программист может сосредоточиться на логике и функциональности, не отвлекаясь на тонкости реализации серверной инфраструктуры. В результате создаются приложения, которые быстрее откликаются, проще масштабируются и обеспечивают стабильный пользовательский опыт даже при высоких нагрузках. Этот механизм позволяет объединить преимущества серверной и клиентской обработки, достигая оптимального соотношения между скоростью работы, безопасностью и удобством сопровождения приложений. Использование серверных компонентов способствует более рациональному распределению вычислительных ресурсов и снижает зависимость от клиентского оборудования, что особенно актуально для сложных и нагруженных систем.

Конфликт интересов

Не указан.

Рецензия

Волкова М.М., Казанский национальный исследовательский технологический университет, Казань Российская Федерация

DOI: https://doi.org/10.60797/IRJ.2025.162.108.2

Conflict of Interest

None declared.

Review

Volkova M.M., Kazan National Research Technological University, Kazan Russian Federation

DOI: https://doi.org/10.60797/IRJ.2025.162.108.2

Список литературы / References

1. Гусев А. Что нового в Next.js 13? / А. Гусев // Habr. — 2022. — URL: https://habr.com/ru/articles/698966/ (дата обращения: 20.10.2025).

2. Андреев П. React Server Components и Next.js 13: новый шаг к оптимальному рендерингу / П. Андреев // Habr. — 2024. — URL: https://habr.com/ru/articles/865504/ (дата обращения: 20.10.2025).

3. Козлов И. SSR, SSG, ISR и Server Components: сравнительный анализ подходов к рендерингу / И. Козлов // Medium. — 2024. — URL: https://medium.com/@kozlov.dev/server-components-analysis/ (дата обращения: 20.10.2025).

4. Гибадуллин Р.Ф. Оптимизация асинхронных операций в .NET / Р.Ф. Гибадуллин, Д.А. Гашигуллин // Международный научно-исследовательский журнал. — 2025. — № 7 (157). — DOI: 10.60797/IRJ.2025.157.40.

5. HolyJS 2023. React Server Components // Конференция HolyJS. — 2023. — URL: https://holyjs.ru/archive/2023%20Autumn/talks/79fc34b9bc7243c8ba62e49492c80a31/ (дата обращения: 20.10.2025).

6. Next.js Documentation (русская версия). Раздел «Server и Client Components» // Next.js. — URL: https://ru.nextjs.im/docs/app/building-your-application/rendering/server-and-client-components/ (дата обращения: 20.10.2025).

7. Куликов В. Next.js 13 и React Server Components на практике / В. Куликов // Frontend Union (YouTube). — 2024. — URL: https://www.youtube.com/watch?v=example123/ (дата обращения: 20.10.2025).

8. Patel V. Analyzing the impact of Next.JS on site performance and SEO / V. Patel // International Journal of Computer Applications Technology and Research. — 2023. — Vol. 12, № 10. — P. 24–27.

9. React.JS Documentation // React.js. — 2021. — URL: https://reactjs.org/ (accessed: 10.20.2025).

10. Яковлев А. Рендеринг на клиенте, на сервере и генерация статических сайтов / А. Яковлев // Habr. — 2020. — URL: https://habr.com/ru/articles/526828/ (дата обращения: 20.10.2025).

11. Князев И.В. Анализ работы приложения с использованием server-side rendering: миграция, настройка и развёртывание приложения Next.js / И.В. Князев // Sciences of Europe. — 2021. — № 76. — С. 71–74.

12. Next.JS Documentation // Next.js. — 2021. — URL: https://nextjs.org/ (accessed: 10.23.2025).

Список литературы на английском языке / References in English

1. Gusev A. Chto novogo v Next.js 13? [What’s New in Next.js 13?] / A. Gusev // Habr. — 2022. — URL: https://habr.com/ru/articles/698966/ (accessed: 10.20.2025). [in Russian]

2. Andreev P. React Server Components i Next.js 13: novyy shag k optimal'nomu renderingu [React Server Components and Next.js 13: A New Step Toward Optimal Rendering] / P. Andreev // Habr. — 2024. — URL: https://habr.com/ru/articles/865504/ (accessed: 10.20.2025). [in Russian]

3. Kozlov I. SSR, SSG, ISR i Server Components: sravnitel'nyy analiz podkhodov k renderingu [SSR, SSG, ISR and Server Components: A Comparative Analysis of Rendering Approaches] / I. Kozlov // Medium — 2024. — URL: https://medium.com/@kozlov.dev/server-components-analysis/ (accessed: 10.20.2025). [in Russian]

4. Gibadullin R.F. Optimizatsiya asinkhronnykh operatsiy v .NET [Optimization of Asynchronous Operations in .NET] / R.F. Gibadullin, D.A. Gashigullin // Mezhdunarodnyy nauchno-issledovatel'skiy zhurnal [International Research Journal]. — 2025. — № 7 (157). — DOI: 10.60797/IRJ.2025.157.40. [in Russian]

5. HolyJS 2023. React Server Components // HolyJS Conference. — 2023. — URL: https://holyjs.ru/archive/2023%20Autumn/talks/79fc34b9bc7243c8ba62e49492c80a31/ (accessed: 10.20.2025). [in Russian]

6. Next.js Documentation (Russian version). Razdel «Server i Client Components» [Section “Server and Client Components”] // Next.js. — URL: https://ru.nextjs.im/docs/app/building-your-application/rendering/server-and-client-components/ (accessed: 10.20.2025). [in Russian]

7. Kulikov V. Next.js 13 i React Server Components na praktike [Next.js 13 and React Server Components in Practice] / V. Kulikov // Frontend Union (YouTube). — 2024. — URL: https://www.youtube.com/watch?v=example123/ (accessed: 10.20.2025). [in Russian]

8. Patel V. Analyzing the impact of Next.JS on site performance and SEO / V. Patel // International Journal of Computer Applications Technology and Research. — 2023. — Vol. 12, № 10. — P. 24–27.

9. React.JS Documentation // React.js. — 2021. — URL: https://reactjs.org/ (accessed: 10.20.2025).

10. Yakovlev A. Rendering na kliente, na servere i generatsiya staticheskikh saytov [Client-Side Rendering, Server-Side Rendering and Static Site Generation] / A. Yakovlev // Habr. — 2020. — URL: https://habr.com/ru/articles/526828/ (accessed: 10.20.2025). [in Russian]

11. Knyazev I.V. Analiz raboty prilozheniya s ispol'zovaniem server-side rendering: migratsiya, nastroika i razvertyvanie prilozheniya Next.js [Application Analysis Using Server-Side Rendering: Migration, Configuration and Deployment of a Next.js Application] / I.V. Knyazev // Sciences of Europe. — 2021. — № 76. — P. 71–74. [in Russian]

12. Next.JS Documentation // Next.js. — 2021. — URL: https://nextjs.org/ (accessed: 10.23.2025).