накладывает жёсткие ограничения, что делает изменения в коде более сложными и менее безопасными. В большинстве случаев использование композиции и абстракции предпочтительнее, особенно при создании масштабируемых решений. Оптимальный подход – это комбинирование этих методов в зависимости от контекста, с фокусом на снижение связности и улучшение тестируемости кода.

Список использованных источников

1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.]. – Санкт-Петербург : Питер, 2022. – 366 с.
2. Мартин Р. Чистый код: создание, анализ и рефакторинг / Р. Мартин. – М.: Питер, 2021. – 464 с.
3. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. – М.: Питер, 2020. – 352 с.

УДК 004

РАЗРАБОТКА FULL-STACK ПРИЛОЖЕНИЙ НА NEXT.JS

Казаков В. Е., к.т.н., доц.
Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *В статье представлен обзор современного фреймворка для разработки веб-приложений. Представлено краткое описание основных особенностей данного фреймворка, позволяющих реализовывать full-stack web-приложение.*

Ключевые слова: full-stack, Next.JS, React, File Based Routing, SSG, SSR.

Next.js – это популярный фреймворк с открытым исходным кодом, построенный на основе React, который позволяет создавать серверные и статические веб-приложения более эффективно. Он расширяет возможности React, добавляя такие функции, как маршрутизация, генерация статического контента (SSG), серверный рендеринг (SSR), API-роуты и другие удобства для разработки.

Рассмотрим основные нововведения более подробно.

Серверный рендеринг означает, что HTML-страница генерируется на сервере при каждом запросе клиента. То есть браузер получает уже готовый HTML с содержимым страницы. При этом пользователю не нужно ждать выполнения JavaScript в своём браузере.

Процесс генерации происходит следующим образом: пользователь делает запрос к серверу; сервер выполняет логику (например, обращается к БД), генерирует HTML и отправляет его клиенту; клиент получает готовую страницу.

Плюсы:

– хорошее SEO – поисковики сразу видят содержимое;
– актуальные данные при каждом запросе;
– отличная производительность: страницы хранятся на CDN и загружаются очень быстро.

Минусы:

– высокая нагрузка на сервер при большом количестве запросов;
– медленнее, чем SSG, так как требуется обработка на лету.

Такой вид генерации применяется в сайтах с динамическим или часто меняющимся контентом; в приложениях, где важны актуальные данные (например, погода, акции, чаты). Способ хорошо работает для статичного или редко обновляемого контента (блоги, документация, портфолио).

Статическая генерация предусматривает генерацию HTML во время сборки проекта и хранится как статический файл, который пересылается клиенту без участия сервера.

При сборке Next.js-приложения предварительно генерирует HTML для всех указанных

страниц. Эти страницы хранятся как статические файлы. При запросе пользователь получает готовую страницу.

Плюсы:

– очень высокая производительность;
– отличное SEO;
– низкая нагрузка на сервер.

Минусы:

– требует пересборки для обновления контента;
– не подходит для сильно динамического контента.

Статическая генерация используется для создания редко изменяющихся страниц: портфолио, документация, landing pages.

Next.JS также поддерживает стандартный рендеринг React.JS на стороне клиента CSR (Client Side Rendering). Рендеринг происходит в браузере с помощью JavaScript.

Плюсы:

– простота реализации (стандартный React App);
– подходит для SPA (одностраничных приложений);
– легко обновлять UI без перезагрузки страницы.

Минусы:

– плохое SEO, если не используется SSR/SSG;
– первый экран может быть "пустым" до выполнения JS;
– зависит от скорости интернета пользователя.

Клиентская генерация используется для реализации пользовательского UI: внутренние админки, личные кабинеты, управление профилем/корзиной и т. п.

Комбинированный рендеринг (Hybrid Rendering) – это подход, при котором на одной странице могут использоваться разные методы рендеринга: например, часть контента генерируется серверно (SSR или SSG), а другая часть – динамически на клиенте (CSR). Обработка представлений в этом случае происходит следующим образом: сервер отправляет готовый HTML; браузер отображает его сразу; затем браузер загружает JavaScript; React «оживляет» DOM – добавляет обработчики событий, состояние, эффекты и т. д. Такой механизм называют Hydration, он превращает статическую разметку в живое приложение.

Такой подход позволяет получить преимущества SSR/SSG (SEO, быстрая загрузка первого экрана), и одновременно использовать CSR для интерактивных частей (например, формы, табы, фильтры и т. д.).

Ещё одной ключевой особенностью Next.JS является маршрутизация на основе файлов (File-based Routing). Next.JS позволяет создавать маршруты (URL-адреса), просто изменяя структуру размещения файлов с исходным кодом в папках проекта.

Этот подход упрощает управление маршрутами, так как не нужно вручную писать конфигурации для роутинга.

Next.JS предоставляет несколько схем описания роутов:

1. *Pages Router.* В классическом подходе маршруты создаются автоматически на основе структуры папки pages. Каждый .js файл в папке pages становится отдельным маршрутом.
2. *Dynamic Routing.* Для создания динамических маршрутов в названиях файлов используются квадратные скобки [param] (например, pages/blog/[slug].js). После вызова соответствующего скрипта этот параметр будет доступен для обработки.
3. *App Router.* Начиная с версии 13 Next.js, поддерживает новую систему маршрутизации, основанную на папке app/. Она использует аналогичный принцип, но имеет больше возможностей (например, параллельные и вложенные макеты).
4. *Обработка 404.* Если страница не найдена, Next.js автоматически показывает стандартную страницу 404. Чтобы кастомизировать её необходимо создать файл со специальным названием.
5. *API Routes.* Next.js также позволяет создавать бэкенд прямо в проекте с помощью API-маршрутов. Каждый файл из папки pages/api/ становится API-эндпоинтами, которые содержат серверную часть приложения.

Для формирования представлений Next.JS позволяет использовать технологию шаблонизации. Шаблонизация (Templating) – это процесс создания повторно используемых частей интерфейса, чтобы избежать дублирования кода и сделать разработку более структурированной и поддерживаемой. В Next.JS шаблонизация особенно важна при создании больших проектов со множеством страниц.

Помимо обычных средств шаблонизации, предоставляемых React, Next.JS предоставляет возможность шаблонизации, в том числе и с вложениями, с применением Pages Router. Для реализации этого механизма используется специальный файл layout.js, размещаемый в файловой структуре, обрабатываемой системой маршрутизации.

Обмен данными между серверной и клиентской частями в Next.JS – это ключевой аспект разработки приложений, особенно при использовании SSR (Server Side Rendering), SSG (Static Site Generation) или API-роуты. Next.JS предлагает механизм передачи данных на основе props, с помощью специальных функций getStaticProps и getServerSideProps, что позволяет соблюсти преемственность с архитектурой React-приложения. Кроме того, клиентские компоненты могут обмениваться данными с обработчиками запросов на стороне сервера (API Routes).

Дополнительно Next.js предоставляет широкие возможности для поддержки стилизации, что делает его гибким инструментом для разработки интерфейсов. Независимо от того, какой подход к стилям предпочитает использовать разработчик – CSS-модули, Tailwind CSS, SCSS или сторонние библиотеки – всё это можно легко использовать в проекте на Next.js.

Заключение

Next.js – это мощный и гибкий фреймворк, построенный на основе React, который предоставляет разработчикам широкие возможности для создания современных веб-приложений. Он сочетает в себе лучшие практики клиентской и серверной разработки, позволяя строить как статические, так и динамические приложения с высокой производительностью, отличной SEO-оптимизацией и богатой пользовательской интерфейсной логикой.

Список использованных источников

1. Сайт «react.dev» [Электронный ресурс]. – Режим доступа: https://react.dev/learn. – Дата доступа: 10.04.2025.
2. Сайт «nextjs.org» [Электронный ресурс]. – Режим доступа: https://nextjs.org/learn. – Дата доступа: 10.04.2025.
3. Next.js. Технология современной веб-разработки [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/auriga/articles/786912/. – Дата доступа : 10.04.2025.
4. Jack Herrington’s Set of articles [Электронный ресурс]. – Режим доступа: https://www.pronextjs.dev/articles. – Дата доступа: 01.05.2025.

УДК 539.21:535

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ МОДЕЛИРОВАНИЯ СПЕКТРОСКОПИЧЕСКИХ СВОЙСТВ ИОНА Pr^{3+} В ЛАЗЕРНЫХ МАТЕРИАЛАХ

Воронцова К. О., студ., Дунина Е. Б., к.ф.-м.н., доц., Корниенко А. А., д.ф.-м.н., проф.
Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *В статье представляется прикладное программное обеспечение для расчета параметров интенсивности Джадда–Офельта, используя экспериментальные данные из спектров поглощения материалов, легированных Pr^{3+}. Для определения параметров интенсивности используется процедура минимизация функционала ошибки, составленного из суммы квадратов отклонений вычисленных сил линий от соответствующих экспериментальных значений.*

Ключевые слова: лазерные материалы, ион празеодима, спектр поглощения, параметры интенсивности.