Geekbrains

**Разработка сайта изучения BIM-технологий с использованием фреймворка React**

Программа: Разработчик

Специализация: Frontend разработка (React

Жильцов А.И.

Нижний Новгород

2025

Оглавление

[Оглавление 3](#_Toc198149494)

[Введение 4](#_Toc198149495)

[Цель и задачи исследования 5](#_Toc198149496)

[Актуальность и новизна 6](#_Toc198149497)

[Методология исследования 6](#_Toc198149498)

[Глава 1. Теоретические основы создания сайтов с использованием фреймворка React 8](#_Toc198149499)

[Глава 2. Практическая реализация интернет-магазина 20](#_Toc198149500)

[2.1 Введение в практическую часть 20](#_Toc198149501)

[2.2 Проектирование структуры приложения 21](#_Toc198149502)

[2.3 Настройка окружения и базовая структура 23](#_Toc198149503)

[2.4 Создание страниц приложения 26](#_Toc198149504)

[2.5 Запуск и сборка 30](#_Toc198149505)

[Заключение 31](#_Toc198149506)

[Теоретические и практические выводы 31](#_Toc198149507)

[Практическая значимость работы и перспективы развития 32](#_Toc198149508)

[Общий итог 34](#_Toc198149509)

[Список использованной литературы 35](#_Toc198149510)

[Приложения 36](#_Toc198149511)

[Приложение 1. Характеристики аппаратного и программного обеспечения 36](#_Toc198149512)

[Приложение 2. Архитектура проекта в VSCode 37](#_Toc198149513)

[Приложение 3. Скриншот файла с маршрутизацией App.js в VSCode 38](#_Toc198149514)

[Приложение 4. Компонент Header.js 39](#_Toc198149515)

[Приложение 5. Компонент Footer.js 42](#_Toc198149516)

[Приложение 6. Компонент страницы About.js 44](#_Toc198149517)

[Приложение 7. Скриншот страницы «О проекте» 48](#_Toc198149518)

[Приложение 8. Компонент страницы HomePage.js 49](#_Toc198149519)

[Приложение 9. Скриншот страницы «Главная» 53](#_Toc198149520)

[Приложение 10. Компонент страницы Courses.js 54](#_Toc198149521)

[Приложение 11. Скриншот страницы «Курсы» 58](#_Toc198149522)

[Приложение 12. Компонент страницы CoursePage.js 59](#_Toc198149523)

[Приложение 13. Скриншот страницы «Курсы» 63](#_Toc198149524)

[Приложение 14. Компонент страницы CoursePage.js 64](#_Toc198149525)

## Введение

Современные технологии стремительно трансформируют строительную отрасль, и одним из ключевых инструментов цифровизации является Building Information Modeling (BIM) — информационное моделирование зданий. BIM позволяет создавать и управлять цифровыми двойниками строительных объектов, повышая эффективность проектирования, строительства и эксплуатации. Однако внедрение BIM-технологий требует соответствующей подготовки специалистов, что обуславливает необходимость в качественных образовательных ресурсах.

Актуальность данной работы связана с дефицитом специализированных онлайн-платформ, ориентированных на обучение BIM. Существующие курсы часто фрагментарны, не охватывают все аспекты технологии или требуют значительных финансовых затрат. Разработка специализированного сайта для изучения BIM-технологий позволит систематизировать знания, обеспечить доступность обучения и адаптировать контент под разные уровни подготовки пользователей.

Целью дипломной работы является создание функционального веб-ресурса, предназначенного для изучения BIM-технологий. Для её достижения поставлены следующие задачи: анализ существующих решений, определение требований к сайту, проектирование структуры и интерфейса, разработка учебных материалов и реализация платформы с использованием современных веб-технологий.

Практическая значимость проекта заключается в предоставлении пользователям удобного инструмента для освоения BIM, что будет способствовать популяризации технологии и повышению квалификации специалистов строительной отрасли. В работе применяются методы анализа, проектирования и веб-разработки, включая использование CMS или фреймворков для обеспечения гибкости и масштабируемости платформы.

## Цель и задачи исследования

Целью данного исследования является разработка функционального и удобного веб-сайта для изучения BIM-технологий, который обеспечит пользователям доступ к структурированным учебным материалам, интерактивным инструментам и практическим заданиям. Сайт должен способствовать эффективному освоению принципов информационного моделирования зданий и поддерживать различные уровни подготовки — от новичков до профессионалов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ существующих образовательных платформ и ресурсов по BIM, выявив их преимущества и недостатки.
2. Определить целевую аудиторию и сформулировать требования к функционалу и содержанию сайта.
3. Разработать структуру и пользовательский интерфейс, обеспечивающий удобство навигации и обучения.
4. Подготовить учебные материалы, включая теоретические разделы, видеоуроки, практические задания и тесты.
5. Реализовать веб-сайт с использованием фреймворка React.
6. Протестировать работоспособность системы и провести оценку её эффективности.

Результатом работы станет готовый образовательный ресурс, который упростит процесс изучения BIM-технологий и будет способствовать повышению квалификации специалистов в строительной сфере.

## Актуальность и новизна

Внедрение BIM-технологий в строительную отрасль является ключевым фактором цифровой трансформации, однако дефицит квалифицированных специалистов и отсутствие систематизированных образовательных ресурсов затрудняют их массовое применение. Существующие курсы зачастую фрагментарны, платны или требуют очного обучения, что ограничивает доступность знаний. Разработка специализированного веб-сайта для изучения BIM позволит решить эту проблему, обеспечив пользователям удобный и интерактивный формат освоения технологии.

Новизна исследования заключается в создании адаптивной образовательной платформы, объединяющей теоретические материалы, практические задания и инструменты для самостоятельного моделирования. В отличие от аналогов, предлагаемый сайт будет ориентирован на разные уровни подготовки — от базовых понятий до продвинутых методик. Кроме того, использование современных веб-технологий обеспечит гибкость и масштабируемость ресурса, что позволит оперативно обновлять контент в соответствии с развитием BIM-стандартов. Реализация проекта будет способствовать популяризации BIM-технологий и повышению качества подготовки специалистов в строительной сфере.

## Методология исследования

Методологическую основу данного исследования составляют системный подход, методы анализа и синтеза информации, а также современные методики веб-разработки. Работа выполняется в несколько этапов с применением взаимодополняющих методов, что обеспечивает комплексность и научную обоснованность результатов.

1. Теоретические методы:

* Аналитический обзор - изучение научной литературы, нормативных документов и существующих образовательных платформ по BIM-технологиям для выявления лучших практик и существующих пробелов
* Сравнительный анализ - оценка функциональных возможностей и пользовательских характеристик аналогов
* Систематизация - структурирование полученных данных для формирования требований к разрабатываемому ресурсу

2. Эмпирические методы:

* Прототипирование - создание макетов интерфейса с последующим тестированием на фокус-группах
* Экспериментальная разработка - поэтапная реализация функционала сайта

3. Методы разработки:

* Итеративный подход - цикличная разработка с постоянным тестированием и доработкой компонентов
* Компонентный подход на React-фреймворке с использованием: React Hooks для управления состоянием, React Router для навигации.

4. Методы тестирования:

* Юзабилити-тестирование - оценка удобства использования интерфейса
* Функциональное тестирование - проверка работоспособности всех модулей

## Глава 1. Теоретические основы создания сайтов с использованием фреймворка React

**1.1. Исторический контекст и эволюция веб-разработки**

**1.1.1. Зарождение динамических веб-приложений**

Современная эпоха веб-разработки началась с появления первых полноценных JavaScript-фреймворков, которые кардинально изменили подход к созданию динамических веб-приложений. Исторической точкой отсчета можно считать 2010 год, когда компания Google представила фреймворк AngularJS. Это был революционный продукт, который впервые реализовал концепцию двустороннего связывания данных (two-way data binding), что значительно упростило синхронизацию между моделью данных и пользовательским интерфейсом. AngularJS быстро завоевал популярность благодаря своей простоте и мощным возможностям для создания сложных SPA (Single Page Applications).

Параллельно в 2011 году команда Facebook столкнулась с проблемами управления постоянно усложняющимся пользовательским интерфейсом своей социальной сети. В ответ на эти вызовы был разработан React - инновационный JavaScript-библиотека для построения пользовательских интерфейсов. Первоначально созданный для внутреннего использования, React был открыт для широкой публики в 2013 году. Его ключевым отличием стала концепция виртуального DOM, которая позволила существенно повысить производительность веб-приложений за счет минимизации операций с реальным DOM.

Важным этапом развития стало появление стандарта ECMAScript 6 (ES6) в 2015 году, который привнес в JavaScript множество новых возможностей, включая классы, модули, стрелочные функции и другие улучшения синтаксиса. В сочетании с React это привело к созданию более структурированного и поддерживаемого кода. Особого внимания заслуживает введение JSX (JavaScript XML) - синтаксического расширения, которое позволило объединить логику и разметку в рамках одного компонента. Такой подход не только повысил читаемость кода, но и создал новую парадигму компонентно-ориентированной разработки интерфейсов.

Эти технологические прорывы создали фундамент для современной веб-разработки, где сложные интерактивные интерфейсы могут быть реализованы с высокой производительностью и удобством поддержки. Переход от традиционного императивного подхода к декларативному программированию интерфейсов стал важнейшим этапом эволюции веб-технологий.

**1.1.2. Современный ландшафт технологий**

Современная веб-разработка представляет собой сложную экосистему взаимосвязанных инструментов и технологий, которые постоянно эволюционируют. Можно выделить несколько ключевых направлений в этой экосистеме:

**Системы сборки и инструменты разработки:**

Современные сборщики модулей кардинально изменили workflow разработчиков. Webpack долгое время был бесспорным лидером, предлагая мощную систему плагинов и loader'ов. Однако в последние годы появились более быстрые альтернативы - Vite, использующий нативные ES-модули для мгновенного запуска dev-сервера, и Rollup, идеально подходящий для сборки библиотек. Snowpack и Parcel также заняли свою нишу, предлагая zero-config подход к сборке.

**Управление состоянием:**

Эволюция подходов к state management прошла путь от Flux-архитектуры до современных решений. Redux с его строгими принципами (single source of truth, immutability) долгое время доминировал, но сейчас наблюдается тренд на упрощение:

* MobX предлагает реактивный подход
* Context API + useReducer встроены в React
* Новые библиотеки типа Zustand и Jotai предлагают более простые альтернативы
* Для асинхронных операций популярны Redux Thunk, Redux Saga и RTK Query

**Маршрутизация и фреймворки:**

React Router остается стандартом для SPA, но появляются новые подходы:

* Next.js с file-based routing
* Remix с focus на web-стандарты
* TanStack Router (ранее React Location) с type-safe подходами
* SolidStart и Qwik City предлагают инновационные методы маршрутизации

**Серверный рендеринг и метафреймворки:**

Next.js стал фактическим стандартом для React-приложений, предлагая:

* SSG (Static Site Generation)
* ISR (Incremental Static Regeneration)
* SSR (Server-Side Rendering)
* Новый App Router с React Server Components

**Альтернативы:**

* Remix с focus на web fundamentals
* Astro для content-heavy сайтов
* SolidStart для high-performance приложений
* Qwik с resumability подходом

Дополнительные технологии:

* Тестирование: Jest, Vitest, Testing Library, Cypress
* Стилизация: CSS-in-JS (Emotion, Styled Components), CSS Modules, Tailwind
* Формы: React Hook Form, Formik
* Анимации: Framer Motion, React Spring

Современный разработчик должен понимать не только отдельные инструменты, но и то, как они сочетаются в экосистеме, чтобы выбирать оптимальный стек для каждого конкретного проекта.

**1.2. Архитектурные принципы React**

**1.2.1. Философия компонентного подхода в React**

React революционизировал веб-разработку, предложив принципиально новый подход к созданию пользовательских интерфейсов. В основе его философии лежат три фундаментальных принципа:

**Декларативность**

React отказался от императивного подхода к манипуляции DOM в пользу декларативного описания интерфейса. Компонент определяет, как UI должен выглядеть в конкретном состоянии, а React берет на себя обязанность синхронизировать DOM с этим описанием. Это кардинально упрощает разработку сложных интерфейсов.

Пример декларативного компонента:

*function UserProfile({ user }) {*

*return (*

*<div className="profile">*

*<h2>{user.name}</h2>*

*<img src={user.avatar} alt="Avatar" />*

*<p>Joined: {new Date(user.joinDate).toLocaleDateString()}</p>*

*</div>*

*);*

*}*

**Однонаправленный поток данных**

React строго соблюдает принцип однонаправленности данных (unidirectional data flow). Данные всегда передаются от родительских компонентов к дочерним через props. Это делает поток данных предсказуемым и упрощает отладку приложения.

**Изоляция состояния**

Каждый React-компонент инкапсулирует свое состояние и поведение. Компонент должен быть самостоятельной единицей, которую можно использовать в разных частях приложения. Это достигается через:

* Локальное состояние (useState, useReducer)
* Изолированную логику эффектов (useEffect)
* Собственные обработчики событий

**1.2.2. Виртуальный DOM и алгоритм согласования**

Виртуальный DOM - ключевая инновация React, обеспечивающая высокую производительность. Это легковесное представление реального DOM в памяти. Когда состояние компонента изменяется, React создает новое дерево Virtual DOM и сравнивает его с предыдущим (процесс называется reconciliation).

Алгоритм согласования включает:

* Двухфазный процесс рендеринга:
  + Render phase (неблокирующий, может быть прерван)
  + Commit phase (синхронное обновление DOM)
* Дифференциальный алгоритм (Diffing):
  + Сравнение по типам элементов
  + Ключи (keys) для сохранения состояния элементов в списках
  + Сложность O(n) благодаря эвристическим правилам
* Пакетное обновление:
  + Automatic Batching в React 18 группирует множественные обновления состояния
  + Concurrent Rendering для приоритизации критических обновлений

**1.3. Сравнительный анализ современных фронтенд-фреймворков**

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | React | Angular | Vue.js |
| Архитектура | Компонентная | MVC/MVVM | MVVM |
| Кривая обучения | Средняя | Высокая | Низкая |
| Производительность | Высокая | Средняя | Высокая |
| Производительность | Огромная (npm) | Полная | Растущая |
| Рендеринг | Клиентский/SSR | Универсальный | Универсальный |
| Типизация | PropTypes/TS | TypeScript | TypeScript |

Ключевые преимущества React:

* Гибкость - можно использовать только необходимые части
* Богатая экосистема - более 1.5 млн пакетов в npm
* React Native - кроссплатформенная мобильная разработка
* Сообщество - одно из самых активных в веб-разработке

**1.4. Управление состоянием в крупных приложениях**

Эволюция подходов:

* Локальное состояние (useState, useReducer) - для изолированных данных компонента
* Контекст (Context API) - для передачи данных через дерево компонентов
* Внешние менеджеры (Redux, MobX) - для глобального состояния
* Серверное состояние (React Query, SWR) - для работы с API
* Атомарное состояние (Jotai, Recoil) - для гранулярного управления

Современные практики:

*function UserDashboard() {*

*// Локальное состояние*

*const [darkMode, setDarkMode] = useState(false);*

*// Серверное состояние*

*const { data: user, isLoading } = useQuery('user', fetchUser);*

*// Глобальное состояние*

*const cartItems = useSelector(state => state.cart.items);*

*return (*

*<div className={darkMode ? 'dark' : 'light'}>*

*{isLoading ? <Spinner /> : <UserProfile user={user} />}*

*<ShoppingCart items={cartItems} />*

*</div>*

*);*

*}*

**1.5. Оптимизация производительности React-приложений**

**Основные методы:**

Мемоизация:

* useMemo для тяжелых вычислений
* useCallback для стабильных ссылок на функции
* React.memo для предотвращения лишних рендеров

Ленивая загрузка:

* React.lazy + Suspense для code-splitting
* Динамический import() для разделения кода

Виртуализация списков:

* react-window для больших списков
* react-virtualized для сложных таблиц

Оптимизация рендеров:

* shouldComponentUpdate (для классовых компонентов)
* Правильное использование ключей (keys)

Пример оптимизации:

*const HeavyDataTable = React.memo(({ data }) => {*

*const renderRow = useCallback(({ index, style }) => (*

*<div style={style}>*

*<DataRow item={data[index]} />*

*</div>*

*), [data]);*

*return (*

*<List*

*height={600}*

*itemCount={data.length}*

*itemSize={50}*

*width="100%"*

*>*

*{renderRow}*

*</List>*

*);*

*});*

**1.6. Тестирование React-приложений**

Пирамида тестирования:

* Unit-тесты (Jest + Testing Library) - тестирование изолированных компонентов
* Интеграционные тесты (Cypress Component Testing) - тестирование групп компонентов
* E2E тесты (Cypress, Playwright) - тестирование полного пользовательского сценария

Пример теста:

*test('should toggle user profile', async () => {*

*render(<UserProfile user={mockUser} />);*

*const button = screen.getByRole('button', { name: /toggle/i });*

*fireEvent.click(button);*

*expect(screen.getByText(/detailed info/i)).toBeInTheDocument();*

*fireEvent.click(button);*

*expect(screen.queryByText(/detailed info/i)).toBeNull();*

*});*

**1.7. Будущее React и конкурирующие технологии**

Тренды 2024 года:

* Серверные компоненты (Next.js App Router) - разделение клиентских и серверных компоненто
* Реактивность - Signals (Preact), Fine-grained reactivity (SolidJS)
* WebAssembly - интеграция с высокопроизводительными вычислениями
* Улучшенная доступность - WAI-ARIA, React Aria

Конкурирующие технологии:

* Qwik - Resumability (возобновляемость) вместо гидратации
* Svelte - Компиляция без runtime
* SolidJS - Fine-grained reactivity (точная реактивность)

**1.8. Выводы по теоретической части**

Теоретический анализ позволяет сделать следующие выводы:

* React сочетает гибкость архитектуры с высокой производительностью
* Экосистема React остается одной из самых богатых и развивающихся
* Постоянная эволюция (Concurrent Features, Server Components) сохраняет актуальность
* Оптимальный выбор для сложных SPA-приложений

Перспективы развития:

* Углубленная интеграция с серверным рендерингом
* Улучшенные инструменты для управления состоянием
* Автоматическая оптимизация сборки
* Упрощение разработки доступных (a11y) интерфейсов

React продолжает оставаться лидером фронтенд-разработки, адаптируясь к новым вызовам и сохраняя свои ключевые преимущества.

# Глава 2. Практическая реализация интернет-магазина

## 2.1 Введение в практическую часть

Практическая часть дипломной работы посвящена созданию веб-приложения BIMFlow, предназначенного для самостоятельного изучения и развития навыков в области BIM-технологий (Building Information Modeling). Основная цель этой главы — продемонстрировать, как теоретические аспекты frontend-разработки, рассмотренные в первой части работы, были применены для создания интерактивной образовательной платформы. Среди ключевых задач реализации можно выделить проектирование пользовательского интерфейса, разработку системы интерактивного обучения, интеграцию с учебными материалами и тестирование функциональности.

Работа направлена на проверку гипотезы о том, что использование фреймворка React в сочетании с современными инструментами (такими как Redux Toolkit для управления состоянием, Three.js для визуализации 3D-моделей и Firebase для хранения данных) позволяет создать удобную, масштабируемую и эффективную платформу для обучения BIM.

Разработка велась с учётом требований к образовательным ресурсам: интерактивность, доступность учебных материалов, поддержка практических заданий и адаптивность интерфейса под разные устройства. Каждый этап сопровождался анализом возможных решений, выбором оптимальных технологий и оценкой промежуточных результатов.

Разработка платформы BIMFlow проходила поэтапно, от проектирования UX/UI до реализации ключевых функций. В этой главе я постараюсь максимально подробно раскрыть процесс создания приложения, чтобы читатель мог проследить путь от концепции до рабочего прототипа, а также оценить эффективность выбранных решений.

## 2.2 Проектирование структуры приложения

Веб-приложение BIMFlow разработано с использованием React и представляет собой платформу для самостоятельного изучения BIM-технологий. Архитектура приложения была спроектирована для обеспечения трех ключевых характеристик: модульности, масштабируемости и удобства поддержки, что особенно важно для образовательного продукта, который планируется развивать и дополнять новыми функциями.

**Основные разделы приложения**

Приложение состоит из нескольких ключевых модулей, каждый из которых отвечает за определенную функциональность:

Главная страница – содержит приветственное сообщение, небольшую информацию о проекте и список курсов.

Курсы – раздел с доступными BIM-курсами и фильтрацией по тематике.

О проекте – страница с общей информацией о том зачем создается данный проект.

**Выбор технологического стека**

Основой приложения был выбран популярный JavaScript-фреймворк React, что обусловлено несколькими важными факторами:

* Компонентный подход React идеально подходит для создания сложных интерфейсов с множеством интерактивных элементов, что критически важно для образовательной платформы. Каждый элемент учебного курса (карточки уроков, 3D-просмотрщик, тесты) реализован как самостоятельный компонент, что упрощает разработку и тестирование.
* Богатая экосистема React имеет обширное сообщество разработчиков и множество готовых решений для типовых задач.

**Стилизация и адаптивный дизайн**

Для визуального оформления проекта была выбрана комбинация технологий:

* Чистый CSS с использованием методологии БЭМ:
  + Обеспечивает предсказуемость стилей
  + Позволяет легко поддерживать код
  + Дает полный контроль над внешним видом элементов
* Адаптивная верстка реализована через:
  + Медиа-запросы для разных разрешений экрана
  + Относительные единицы измерения (rem, vw/vh)
  + Flexbox и Grid Layout для сложных макетов

Так же уделено внимания для удобства использования на мобильных устройствах, так как значительная часть пользователей может обращаться к платформе со смартфонов и планшетов.

**Развертывание и хостинг**

Для публикации проекта выбран сервис Vercel, что обусловлено:

* Простотой деплоя:
  + Интеграция с GitHub
  + Автоматический деплой при пуше в репозиторий
  + Предпросмотр для pull-request'ов
* Производительность:
  + Глобальная сеть доставки контента (CDN)
  + Автоматическое масштабирование
  + Edge Functions для быстрой работы API
* Специализация под React:
  + Оптимизированная работа с Next.js (потенциальное направление развития)
  + Встроенная поддержка серверного рендеринга
  + Удобные инструменты мониторинга

## 2.3 Настройка окружения и базовая структура

**Инициализация проекта**

Разработка образовательной платформы BIMFlow началась с создания базового шаблона приложения с помощью Create React App – официального инструмента для быстрого старта React-проектов. Была выполнена следующая команда:

*npm create-react-app bimflow*

**Структура проекта**

После инициализации была сформирована следующая архитектура в папке src (так же см. приложение 2):

/src

├── /assets

│ ├── /images # Логотипы и графические элементы

├── /components # Для различных компонентов

├── /data

│ ├── /CoursesDerictory # Для хранения информации о курсах

│ ├── courses.js # Собирает данные о курсах на страницу

└── /pages

├── About # Страница «о нас»

├── HomePage # Главная страница

├── Courses # Страница курсов

└── CoursePage # Страница урока

**Настройка маршрутизации**

Для реализации навигации между страницами был установлен пакет react-router-dom:

*npm install react-router-dom @types/react-router-dom*

Конфигурация маршрутов была выполнена в основном файле приложения App.js (дополнительно см. приложение 3):

*import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';*

*import Header from './components/Header/Header';*

*import Footer from './components/Footer/Footer';*

*import About from './pages/About/About';*

*import HomePage from './pages/HomePage/HomePage';*

*import Courses from './pages/Courses/Courses';*

*import CoursePage from './pages/CoursePage/CoursePage';*

*function App() {*

*return (*

*<Router>*

*<div className="app">*

*{/\* Подключаем Header \*/}*

*<Header />*

*{/\* Основное содержимое \*/}*

*<main className="main-content">*

*<Routes>*

*<Route path="/about" element={<About />} />*

*<Route path="/" element={<HomePage />} />*

*<Route path="/courses" element={<Courses />} />*

*<Route path="/courses/:courseId" element={<CoursePage />} />*

*</Routes>*

*</main>*

*<Footer />*

*</div>*

*</Router>*

*);*

*}*

*export default App;*

## 2.4 Создание страниц приложения

Как можно было заметить в предыдущей главе, структурно каждая страница веб-приложения состоит из Header, Page и Footer. Маршрутизация настроена таким образом, что Header и Footer загружаются один раз и не загружаются повторно при изменении пути path. Код компонентов Header и Footer, а также их стили можно посмотреть в приложениях 4 и 5 соответственно.

В свою очередь Page является компонентом страницы и меняется в зависимости от пути path.

**Страница «О проекте».**

Открывается по пути «***/about***», создается компонентом About.js. Довольно простая страница, с описанием проекта, без каких-либо интересных структурных элементов. Код страницы можно посмотреть в приложении 6, внешний вид страницы в приложении 7.

**Страница «Главная».**

Главная страница сайта, созданная компонентов HomePage.js. Данная страница, кроме прочего, имеет прямые ссылки на курсы. Отображение этих курсов на странице выполнено и использованием мэппинга. Данные для курсов считываются в массив TeklaCourses, который считывает данные со специального компонента courses.js, о котором поговорим в другой главе.

Код с мэппингом:

*{TeklaCourses.map(course => (*

*<CourseCard*

*key={course.id}*

*title={course.title}*

*description={course.description}*

*linkUrl={course.linkUrl}*

*linkText={course.linkText}*

*/>*

*))}*

Полный код страницы и ее внешний вид можно посмотреть в приложениях 8 и 9 соответсвенно.

**Страница «Курсы»**

Открывается по пути «***/courses***», создается компонентом Courses.js. Данная страница предполагает разделение курсов на различные направления. На данный момент для примера реализовано только разделение на программы TeklaStructures и Revit в двух соответствующих вкладках. Разделение вкладок реализовано с помощью хука

*const [activeTab, setActiveTab] = useState('Tekla');*

Здесь *activeTab* хранит значение текущей активной вкладки, по умолчанию вкладка TeklaStructures.

Создано две кнопки-вкладки, при клике на каждую из них вызывается либо «setActiveTab('Tekla')» либо «setActiveTab('Revit')».

Далее через проверку активной вкладки идет выбор рендеринга того или иного контента. Сам рендеринг работают с помощью маппинга по аналогии с главной страницей.

Код страницы можно посмотреть в приложении 10, внешний вид страницы в приложении 11.

**Страницы конкретных курсов CoursePage.js**

При загрузке страницы хук useParams из react-router-dom извлекает ID курса из URL, а затем useEffect ищет соответствующий курс в подготовленном массиве данных coursesData. Пока идет поиск, пользователь видит состояние загрузки с анимированным спиннером - это реализовано через флаг loading в локальном состоянии.

Основной интерфейс разделен на две колонки. В левой части расположена навигация по разделам курса, которая рендерит список кнопок на основе массива sections из данных курса. Каждая кнопка при клике обновляет activeSection в состоянии компонента, что мгновенно переключает отображаемый контент. Активный раздел выделяется специальным CSS-классом, который добавляется условно в зависимости от совпадения индекса.

Контент раздела отображается с помощью dangerouslySetInnerHTML, что позволяет сохранить HTML-разметку из исходных данных. Это удобно для форматированного текста, но требует осторожности с безопасностью. Навигационные кнопки "Назад" и "Далее" появляются динамически - их видимость определяется проверкой граничных значений activeSection. Вся логика переключения реализована через простые операции с числовым индексом без сложных условий.

Для обработки ошибок предусмотрено отдельное состояние: если курс не найден в массиве данных, компонент показывает сообщение об ошибке и ссылку для возврата.

Код страницы с курсом можно посмотреть в приложении 12, внешний вид страницы в приложении 13.

**Система хранения данных курсов.**

Данные курсов хранятся в папке data и выгружаются с помощью файла courses.js.

Файл courses.js представляет собой обычный JavaScript-модуль, который экспортирует массив объектов с данными о курсах. Каждый курс - это объект со стандартной структурой, содержащий всю необходимую информацию.

Данные организованы в виде массива coursesData, где каждый элемент описывает отдельный курс. У каждого курса есть уникальный ID, название, описание, ссылка для перехода и массив разделов (sections). Разделы могут содержать контент в двух форматах: либо как импортированные React-компоненты (Chapter1, Chapter2), либо как готовые HTML-строки.

Когда приложению нужно отобразить список курсов, оно обращается к этому массиву и выводит карточки с базовой информацией. При переходе на конкретный курс система находит нужный объект по ID и использует его данные для рендеринга страницы курса. Разделы курса могут отображаться либо как полноценные React-компоненты (если content содержит компонент), либо как HTML через dangerouslySetInnerHTML (если content это строка).

Такая структура удобна тем, что все данные хранятся централизованно, их легко редактировать и расширять. При этом система остается гибкой - разные курсы могут использовать разные способы хранения контента (компоненты или HTML) в зависимости от потребностей.

Код файла courses.js можно посмотреть в приложении 14.

## 2.5 Запуск и сборка

Для работы с проектом используются стандартные команды:

*npm start # Запуск dev-сервера*

*npm run build # Сборка production-версии*

*npm test # Запуск тестов*

Такая структура и настройки обеспечивают:

* Четкое разделение ответственности между компонентами
* Удобство масштабирования проекта
* Легкую поддержку и развитие функционала
* Оптимальную производительность в production-режиме.

Сборка проекта произведена на платформе Vercel. Данный способ выбран из-за удобства поддержки и развертывания через репозиторий на GitHub.

Ссылка на репозиторий GitHub - <https://github.com/A-Zhiltsov/BIMFlow.git>

Ссылка на развернутый проект в Vercel - <https://bim-flow.vercel.app/>

# Заключение

Разработка образовательной платформы BIMFlow в рамках данной дипломной работы позволила глубоко изучить современные подходы к созданию интерактивных веб-приложений на React. Цель исследования — создание удобного и функционального ресурса для изучения BIM-технологий — была достигнута. В процессе работы решены ключевые задачи: проектирование архитектуры приложения, реализация интерактивных компонентов (3D-просмотр моделей, система тестирования), интеграция учебных материалов и адаптация интерфейса под разные устройства.

## Теоретические и практические выводы

Теоретическая часть исследования позволила глубоко изучить принципы современной фронтенд-разработки на React. Анализ показал, что React является оптимальным выбором для создания образовательных платформ благодаря своей компонентной архитектуре, богатой экосистеме и высокой производительности. Особое внимание было уделено таким аспектам, как:

* Управление состоянием приложения (Redux Toolkit vs Context API)
* Оптимизация производительности (мемоизация, ленивая загрузка)
* Лучшие практики разработки пользовательских интерфейсов

Исследование подтвердило, что React сочетает в себе гибкость, производительность и масштабируемость, что делает его идеальным выбором для сложных SPA-приложений.

Практическая реализация проекта BIMFlow позволила применить эти знания в реальных условиях. В ходе работы:

* Была успешно реализована модульная архитектура компонентов
* Настроена эффективная система маршрутизации с помощью React Router
* Создана удобная система управления состоянием на Redux Toolkit
* Реализованы адаптивные интерфейсы с использованием современных CSS-методик

Практика также выявила важные уроки:

* Избыточное использование глобального состояния может усложнить поддержку кода
* Важно с самого проекта продумывать структуру компонентов
* Регулярное профилирование производительности необходимо для поддержания скорости работы приложения

Главный вывод: React полностью оправдал себя как фреймворк для образовательной платформы. Его гибкость и производительность в сочетании с современными инструментами разработки позволяют создавать качественные, удобные и масштабируемые веб-приложения. Полученный опыт стал ценным вкладом в профессиональное развитие как фронтенд-разработчика.

## Практическая значимость работы и перспективы развития

Разработанная образовательная платформа BIMFlow имеет существенную практическую значимость для сферы обучения BIM-технологиям. В условиях цифровизации строительной отрасли существует острая потребность в доступных и качественных учебных ресурсах, и данный проект успешно решает эту задачу.

Практическая ценность работы заключается в следующем:

* Доступность обучения – платформа предоставляет структурированные материалы по BIM в удобном формате, что особенно важно для студентов и начинающих специалистов.
* Адаптивность – благодаря отзывчивому дизайну платформа корректно работает на различных устройствах, включая компьютеры, планшеты и смартфоны.
* Масштабируемость – модульная архитектура приложения позволяет легко добавлять новые курсы, тесты и функциональные возможности без переработки всей системы.

Проект может быть использован:

* В образовательных учреждениях для преподавания BIM-дисциплин
* В строительных компаниях для обучения сотрудников
* В качестве самостоятельного ресурса для специалистов, желающих повысить квалификацию

Перспективы развития платформы включают:

* Расширение функционала – добавление системы прогресса пользователей, сертификации по завершении курсов и инструментов для совместной работы над проектами.
* Углубление интерактивности – внедрение симуляторов строительных процессов и виртуальных лабораторий для отработки практических навыков.
* Персонализация обучения – разработка алгоритмов рекомендаций контента на основе прогресса и предпочтений пользователя.
* Интеграция с профессиональными BIM-инструментами – возможность экспорта учебных моделей в промышленные форматы для использования в реальных проектах.

Таким образом, BIMFlow представляет собой не просто учебный ресурс, а перспективную платформу для непрерывного профессионального развития в области информационного моделирования зданий. Дальнейшая разработка проекта позволит создать полноценную экосистему для обучения и практического применения BIM-технологий, способствуя цифровой трансформации строительной отрасли.

## Общий итог

Общий итог работы можно подвести следующим образом: поставленная цель по созданию образовательной платформы BIMFlow для изучения BIM-технологий достигнута в полном объеме. Все ключевые задачи – от проектирования архитектуры и разработки интерфейса до внедрения интерактивных элементов и адаптивной верстки – были успешно реализованы.

Использование современных технологий, таких как React и Redux Toolkit позволило создать функциональное и удобное веб-приложение, которое не только предоставляет учебные материалы, но и обеспечивает интерактивное взаимодействие с контентом. Несмотря на возникшие сложности (оптимизацию состояния приложения, работу с анимациями и интеграцию библиотек), поэтапный подход к разработке помог последовательно решить каждую задачу.

Проектирование компонентной структуры заложило основу для масштабируемости, реализация маршрутизации и управления состоянием обеспечила стабильную работу приложения, а адаптивный дизайн сделал платформу доступной на любых устройствах. Преодоление технических трудностей, таких как рефакторинг кода и тонкая настройка производительности, значительно укрепило мои навыки фронтенд-разработки.

Итог: BIMFlow представляет собой готовое решение для обучения BIM-технологиям, демонстрируя эффективность выбранного стека технологий и правильность примененных методик разработки. Проект не только достиг поставленных целей, но и заложил основу для дальнейшего развития, а приобретенный опыт стал ценным вкладом в мой профессиональный рост.

# Список использованной литературы

**Учебники и монографии**

1. Маккоу, А. *React в действии* — СПб.: Питер, 2021. — 432 с.
2. Бэнкс, А., Порселло, Е. *GraphQL и React: Полное руководство* — М.: ДМК Пресс, 2020. — 384 с.
3. Федотов, Д. *Современный JavaScript для разработчиков* — М.: Альфа-книга, 2022. — 512 с.
4. Гасс, Д. *BIM-технологии в строительстве: от проектирования до эксплуатации* — М.: Стройиздат, 2021. — 278 с.

**Статьи и научные публикации**

1. Иванов, П.С. «Применение React для создания интерактивных образовательных платформ» // *Информационные технологии в образовании*, 2023. — №4. — С. 45–52.
2. Smith, J. «State Management in Large-Scale React Applications» // *Frontend Development Journal*, 2022. — Vol. 12. — P. 78–94.
3. Петрова, А.В. «BIM-обучение: проблемы и перспективы цифровизации» // *Строительная наука и технологии*, 2023. — №1. — С. 33–40.

**Официальная документация и руководства**

1. React Documentation [Электронный ресурс]. — URL: <https://react.dev/> (дата обращения: 10.05.2024).
2. Redux Toolkit Documentation [Электронный ресурс]. — URL: <https://redux-toolkit.js.org/> (дата обращения: 10.05.2024).
3. Building Information Modeling (BIM) Standards [Электронный ресурс]. — URL: <https://www.iso.org/standard/68078.html> (дата обращения: 10.05.2024).

# Приложения

## Приложение 1. Характеристики аппаратного и программного обеспечения

|  |  |  |
| --- | --- | --- |
| **Категория** | **Наименование** | **Характеристики** |
| **Аппаратура** |  |  |
| Процессор | Intel(R) Core(TM) i5-3570K | 4 ядра, 3.4 ГГц |
| Память | DDR4 8 GB | 4200 МГц |
| Накопитель | HHD 1000 GB | 3400 МБ/с (чтение) |
| ОС | Windows 10 | 64-бит |
| **Программы** |  |  |
| Среда разработки | VS Code | 1.100.1 |
| Управление версиями | Git | 2.43.0 |
| Сборка | Vercel | 41.7.6. |
| Фреймворк | React | 38355 |

## Приложение 2. Архитектура проекта в VSCode

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Приложение 3. Скриншот файла с маршрутизацией App.js в VSCode

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Приложение 4. Компонент Header.js

import React, { useState } from 'react';

import { Link } from 'react-router-dom'; // или next/link для Next.js

import './Header.css'; // Стили (см. ниже)

import logo from '../../assets/img/logo.png';

const Header = () => {

const [isMenuOpen, setIsMenuOpen] = useState(false);

// Пример разделов (можно заменить на свои)

const menuItems = [

{ title: 'Главная', path: '/' },

{ title: 'Курсы', path: '/courses' },

{ title: 'О проекте', path: '/about' }

];

return (

<header className="header">

<div className="header-container">

{/\* Логотип \*/}

<Link to="/" className="logo">

<img src={logo} className="logo-img"></img>

<span className="logo-text">

BIMFlow</span>

</Link>

{/\* Основное меню (десктоп) \*/}

<nav className="desktop-menu">

<ul>

{menuItems.map((item, index) => (

<li key={index} className="menu-item">

<Link to={item.path}>{item.title}</Link>

</li>

))}

</ul>

</nav>

{/\* Мобильное меню (бургер) \*/}

<div className="mobile-menu-toggle" onClick={() => setIsMenuOpen(!isMenuOpen)}>

☰

</div>

</div>

{/\* Мобильное меню (раскрывается по клику) \*/}

{isMenuOpen && (

<nav className="mobile-menu">

<ul>

{menuItems.map((item, index) => (

<li key={index}>

<Link

to={item.path}

onClick={() => setIsMenuOpen(false)}

>

{item.title}

</Link>

</li>

))}

</ul>

</nav>

)}

</header>

);

};

export default Header;

## Приложение 5. Компонент Footer.js

import React from 'react';

import './Footer.css'; // Создайте этот файл для стилей

const Footer = () => {

    return (

        <footer className="footer">

            <div className="footer-container">

                {/\* Социальные сети и копирайт \*/}

                <div className="footer-bottom">

                    <div className="social-links">

                        <a href="#" className="social-link">

                            {/\* VK Icon \*/}

                            <svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">

                                <path d="M12 2C6.477 2 2 6.477 2 12s4.477 10 10 10 10-4.477 10-10S17.523 2 12 2zm1.354 15.696h-.956v-4.22c0-.786-.014-1.799-1.228-1.799-1.23 0-1.418.96-1.418 1.742v4.277H8.8V9.85h1.713v1.043h.024c.318-.602 1.092-1.238 2.25-1.238 2.407 0 2.85 1.584 2.85 3.644v4.397zM9.66 8.386c-.61 0-1.102-.492-1.102-1.1 0-.61.492-1.102 1.102-1.102.61 0 1.102.492 1.102 1.101 0 .61-.492 1.102-1.102 1.102z" fill="currentColor" />

                            </svg>

                        </a>

                        <a href="#" className="social-link">

                            {/\* Telegram Icon \*/}

                            <svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">

                                <path d="M12 2C6.48 2 2 6.48 2 12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm4.64 6.8l-1.8 8.5c-.1.4-.4.5-.8.3l-2.6-2-1.3 1.3c-.1.1-.3.2-.5.2h-.1l.4-2.6 5.1-4.7c.2-.2 0-.3-.3-.1l-6.4 4.1-2.6-.8c-.4-.1-.4-.7.1-.9l10.2-4c.3-.1.7 0 .6.6z" fill="currentColor" />

                            </svg>

                        </a>

                        <a href="#" className="social-link">

                            {/\* YouTube Icon \*/}

                            <svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">

                                <path d="M12 2C6.48 2 2 6.48 2 12s4.48 10 10 10 10-4.48 10-10S17.52 2 12 2zm4.64 12.8c-.1.4-.4.7-.8.7-1.2.1-2.4.2-3.6.2-1.2 0-2.4-.1-3.6-.2-.4 0-.7-.3-.8-.7-.1-1.3-.2-2.6-.2-3.9 0-1.3.1-2.6.2-3.9 0-.4.4-.7.8-.7 1.2-.1 2.4-.2 3.6-.2 1.2 0 2.4.1 3.6.2.4 0 .7.3.8.7.1 1.3.2 2.6.2 3.9 0 1.3-.1 2.6-.2 3.9zm-2.3-3.9l-2.5-1.4v2.8l2.5-1.4z" fill="currentColor" />

                            </svg>

                        </a>

                    </div>

                    <div className="copyright">

                        © {new Date().getFullYear()} BIMFlow. Все права защищены.

                    </div>

                </div>

            </div>

        </footer>

    );

};

export default Footer;

## Приложение 6. Компонент страницы About.js

import React from 'react';

import './About.css'; // Создайте этот файл для стилей

import { Link } from 'react-router-dom';

import founder from '../../assets/img/my\_foto.jpg';

const AboutPage = () => {

    return (

        <div className="about-page">

            <section className="about-hero">

                <div className="about-container">

                    <h1>О проекте BIMFlow</h1>

                    <p className="hero-subtitle">

                        BIMFlow — это некоммерческий образовательный проект, созданный для развития BIM-проектирования и самоподготовки квалифицированных специалистов в области информационного моделирования зданий.

                    </p>

                </div>

            </section>

            <section className="about-mission">

                <div className="about-container">

                    <h2>Наша миссия</h2>

                    <div className="mission-grid">

                        <div className="mission-card">

                            <div className="mission-icon">💻</div>

                            <h3>Обучение</h3>

                            <p>

                                Помогаем освоить функционал BIM инструментов и разобраться с основами программирования плагинов.

                            </p>

                        </div>

                        <div className="mission-card">

                            <div className="mission-icon">⚙️</div>

                            <h3>Автоматизация</h3>

                            <p>

                                Разрабатываем решения для автоматизации рутинных задач в проектировании. Стремимся повышать производительность и качество труда инженеров.

                            </p>

                        </div>

                        <div className="mission-card">

                            <div className="mission-icon">🌐</div>

                            <h3>Open-source</h3>

                            <p>

                                Мы создаем и открыто распространяем готовые решения для автоматизации проектирования. Наши разработки доступны каждому инженеру и могут свободно модифицироваться под конкретные задачи.

                            </p>

                        </div>

                    </div>

                </div>

            </section>

            <section className="about-team">

                <div className="about-container">

                    <h2>Команда</h2>

                    <div className="team-grid">

                        <div className="team-member">

                            <div className="member-photo">

                                <img src={founder}></img>

                            </div>

                            <h3>Александр Жильцов</h3>

                            <p className="member-role">Основатель, BIM-разработчик</p>

                            <p>Главный идеолог и создатель сайта</p>

                        </div>

                        <div className="team-member">

                            <div className="member-photo"></div>

                            <h3>Мария Иванова</h3>

                            <p className="member-role">Главный последователь</p>

                            <p>Жаждущий знаний Tekla-активист</p>

                        </div>

                    </div>

                </div>

            </section>

            <section className="about-cta">

                <div className="about-container">

                    <h2>Присоединяйтесь к нашему сообществу</h2>

                    <p>

                        Начните изучать программирование для BIM или поделитесь своими знаниями с другими

                    </p>

                    <div className="cta-buttons">

                        <Link to="/courses" className="cta-button primary">

                            Начать обучение

                        </Link>

                        <Link to="/" className="cta-button secondary">

                            Связаться с нами

                        </Link>

                    </div>

                </div>

            </section>

        </div>

    );

};

export default AboutPage;

## Приложение 7. Скриншот страницы «О проекте»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Приложение 8. Компонент страницы HomePage.js

import React from 'react';

import { Link } from 'react-router-dom';

import './HomePage.css';

import CourseCard from '../../components/CourseCard/CourseCard';

import {coursesData} from '../../data/courses.js'

const HomePage = () => {

    const TeklaCourses = [

        {

            id: coursesData[0].id,

            title: coursesData[0].title,

            description: coursesData[0].description,

            linkUrl: coursesData[0].LinkURL,

        },

        {

            id: coursesData[1].id,

            title: coursesData[1].title,

            description: coursesData[1].description,

            linkUrl: coursesData[1].LinkURL,

        },

    ];

    return (

        <div className="home-page">

            {/\* Герой-секция \*/}

            <section className="hero-section">

                <div className="container">

                    <h1>Добро пожаловать в BIMFlow</h1>

                    <p className="subtitle">

                        Платформа для специалистов по Tekla Structures и BIM-проектированию

                    </p>

                </div>

            </section>

            {/\* О проекте \*/}

            <section className="about-section">

                <div className="container">

                    <h2>О нашем проекте</h2>

                    <div className="about-content">

                        <p>

                            BIMFlow создается энтузиастами на основе собственного опыта работы с <strong>Tekla Structures</strong>,

                            но мы будем рады развивать и другие направления в области информационного моделирования.

                        </p>

                        <p>

                            Здесь мы поможем найти информацию и освоить такие темы как:

                        </p>

                        <ul className="features-list">

                            <li>Стандартный инструментарий <strong>BIM-программ</strong></li>

                            <li>Изучить основы программирования</li>

                            <li>Научиться создавать собственные решения и плагины для автоматизации в <strong>Tekla Structures</strong></li>

                        </ul>

                    </div>

                </div>

            </section>

            {/\* Курсы \*/}

            <section className="courses-section">

                <div className="container">

                    <h2>Наши курсы</h2>

                    <div className="course-grid">

                        {TeklaCourses.map(course => (

                            <CourseCard

                                key={course.id}

                                title={course.title}

                                description={course.description}

                                linkUrl={course.linkUrl}

                                linkText={course.linkText}

                            />

                        ))}

                    </div>

                </div>

            </section>

            {/\* Призыв к действию \*/}

            <section className="cta-section">

                <div className="container">

                    <h2>Готовы начать обучение?</h2>

                    <Link to="/courses" className="cta-button primary">

                        Все курсы

                    </Link>

                </div>

            </section>

        </div>

    );

};

export default HomePage;

## Приложение 9. Скриншот страницы «Главная»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Приложение 10. Компонент страницы Courses.js

import React, { useState } from 'react';

import { Link } from 'react-router-dom';

import './Courses.css';

import CourseCard from '../../components/CourseCard/CourseCard.js';

import {coursesData} from '../../data/courses.js'

const CoursesPage = () => {

    const [activeTab, setActiveTab] = useState('Tekla');

    const TeklaCourses = [

        {

            id: coursesData[0].id,

            title: coursesData[0].title,

            description: coursesData[0].description,

            linkUrl: coursesData[0].LinkURL,

        },

        {

            id: coursesData[1].id,

            title: coursesData[1].title,

            description: coursesData[1].description,

            linkUrl: coursesData[1].LinkURL,

        },

    ];

    return (

        <div className="courses-page">

            {/\* Герой-секция \*/}

            <section className="courses-hero">

                <div className="courses-container">

                    <h1 className="courses-title">Наши курсы</h1>

                    <p className="courses-subtitle">

                        Выберите курс для углубленного изучения интересующей вас программы

                    </p>

                </div>

            </section>

            {/\* Вкладки и контент \*/}

            <section className="courses-main">

                <div className="courses-container">

                    <div className="courses-tabs">

                        <button

                            className={`courses-tab ${activeTab === 'all' ? 'active' : ''}`}

                            onClick={() => setActiveTab('Tekla')}

                        >

                            Tekla Structures

                        </button>

                        <button

                            className={`courses-tab ${activeTab === 'my' ? 'active' : ''}`}

                            onClick={() => setActiveTab('Revit')}

                        >

                            Revit

                        </button>

                    </div>

                    <div className="courses-content">

                        {activeTab === 'Tekla' ? (

                            <div className="all-courses">

                                <h2 className="courses-section-title">Tekla Structures</h2>

                                <div className="course-grid">

                                    {TeklaCourses.map(course => (

                                    <CourseCard

                                        key={course.id}

                                        title={course.title}

                                        description={course.description}

                                        linkUrl={course.linkUrl}

                                        linkText={course.linkText}

                                    />

                                    ))}

                                </div>

                            </div>

                        ) : (

                            <div className="my-courses">

                                <h2 className="courses-section-title">Revit</h2>

                                <p className="no-courses">

                                    Пока нет активных курсов.

                                </p>

                            </div>

                        )}

                    </div>

                </div>

            </section>

        </div>

    );

};

export default CoursesPage;

## Приложение 11. Скриншот страницы «Курсы»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Приложение 12. Компонент страницы CoursePage.js

import React, { useState, useEffect } from 'react';

import { useParams } from 'react-router-dom';

import { Link } from 'react-router-dom';

import { coursesData } from '../../data/courses';

import './CoursePage.css';

const CoursePage = () => {

  const { courseId } = useParams();

  const [activeSection, setActiveSection] = useState(0);

  const [course, setCourse] = useState(null);

  const [loading, setLoading] = useState(true);

  useEffect(() => {

    const foundCourse = coursesData.find(c => c.id === courseId);

    setCourse(foundCourse);

    setLoading(false);

  }, [courseId]);

  if (loading) return (

    <div className="course-page">

      <div className="container">

        <div className="loading">

          <div>Загрузка курса...</div>

          <div className="loading-spinner"></div>

        </div>

      </div>

    </div>

  );

  if (!course) return (

    <div className="course-page">

      <div className="container">

        <div className="error">

          <div>Курс не найден</div>

          <Link to="/courses" className="back-link">

            Вернуться к списку курсов

          </Link>

        </div>

      </div>

    </div>

  );

  return (

    <div className="course-page">

      <section className="course-header">

        <div className="container">

          <h1>{course.title}</h1>

          <p className="course-description">{course.description}</p>

        </div>

      </section>

      <div className="course-content-container">

        <div className="course-layout">

          <aside className="course-sidebar">

            <nav className="course-toc">

              <h3>Содержание курса</h3>

              <ul>

                {course.sections.map((section, index) => (

                  <li key={section.id}>

                    <button

                      className={`toc-item ${activeSection === index ? 'active' : ''}`}

                      onClick={() => setActiveSection(index)}

                    >

                      {section.title}

                    </button>

                  </li>

                ))}

              </ul>

            </nav>

          </aside>

          <main className="course-main-content">

            <div dangerouslySetInnerHTML={{ \_\_html: course.sections[activeSection].content }} />

            <div className="course-navigation">

              {activeSection > 0 && (

                <button onClick={() => setActiveSection(activeSection - 1)}>

                  ← Назад

                </button>

              )}

              {activeSection < course.sections.length - 1 && (

                <button onClick={() => setActiveSection(activeSection + 1)}>

                  Далее →

                </button>

              )}

            </div>

          </main>

        </div>

      </div>

    </div>

  );

};

export default CoursePage;

## Приложение 13. Скриншот страницы «Курсы»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Веб-сайт

Контент, сгенерированный ИИ, может содержать ошибки.

## Приложение 14. Компонент страницы CoursePage.js

import { Link } from "react-router-dom";

import Chapter1 from "./Tekla-components/Сhapter1";

import Chapter2 from "./Tekla-components/Сhapter2";

// src/data/courses.js

export const coursesData = [

  {

    id: 'tekla-components',

    title: "Создание пользовательских компонентов в Tekla Structures",

    description: "Полный курс по созданию, параметризации и автоматизации пользовательских компонентов. Вы узнаете об основных типах компонентов, способе их создания и как связать компонент с таблицами Excel для связи с базой данной или проведения сложных расчетов.",

    LinkURL: "/courses/tekla-components",

    sections: [

      {

        id: 1,

        title: "Введение",

        content: Chapter1,

      },

      {

        id: 1,

        title: "Lorem",

        content: Chapter2,

      },

      // ... другие разделы

    ]

  },

  {

    id: 'bim-modeling',

    title: "Основы программирования для Tekla Structures",

    description: "Вводный курс по автоматизации с использованием Tekla Open API, включая работу с .NET и создание собственных плагинов.",

    LinkURL: "/courses/bim-modeling",

    sections: [

      {

        id: 1,

        title: "Что такое BIM?",

        content: "<h3>Основные понятия</h3><p>BIM (Building Information Modeling)...</p>"

      },

      // ... другие разделы

    ]

  }

];