Geekbrains

**Разработка веб-приложения для онлайн-магазина техники**

**в условиях отсутствия медиаконтента на момент разработки**

Программа: Frontend разработка (React)

Специализация разработчик-программист

Алексеев Михаил Сергеевич

Москва

2024 год

**Содержание**

Введение

Глава 1. Основы создания веб-приложения (сайта) на React.

1.1 Веб-сайт как ключевой элемент цифровой среды: определение, структура и функции

1.2 Специализация программиста по созданию сайтов: навыки, технологии и профессиональные перспективы.

1.3 Особенность и преимущества создания приложения на React: современные подходы и возможности

1.4 Инструменты React для создания веб-страниц

1.5 Технология гипертекста — язык HTML

Глава 2. Подготовка к созданию веб-сайта

2.1 Определение требований заказчика и функциональности сайта

2.2 Сбор дополнительной информации

2.3 Создание спецификации проекта и согласование с заказчиком

2.4 Структурирование проекта

2.5 Разработка компонентов

2.6 Реализация маршрутизации

2.7 Интеграция с API

2.8 Обеспечение поддержки и обновления сайта

Глава 3. Создание веб-сайта

3.1 Создание макета

3.2 Определение модулей сайта (главная, страницы категорий товаров, поиск, авторизация, корзина)

3.3 Верстка через Flexbox: Оптимальное расположение модулей и компонентов

3.4 Добавление логики модулей и интеграция с API

3.5 Финальная верстка: Наложение стилей и улучшение пользовательского интерфейса.

3.6 Финальные тесты

3.7 Сдача продукта

Заключение

Список используемой литературы

Приложения

# Введение

Темой дипломного проекта выбрана «Разработка веб-приложения для онлайн-магазина техники, в условиях отсутствия медиаконтента на момент разработки».

Целью проекта является разработка веб-сайта для заказчика в условиях ограниченных сроков запуска бизнеса. Решает проблему заведомого создания сайта до наличия готовой к реализации продукции или при отсутствии готового медиаконтента конкретных товаров, но необходимости подготовки основы сайта в данных условиях. Фото товара могут быть подгружены позже, а заказчик успеет соответствовать гайдлайну запуска бизнеса.

Задачи:

1. Изучить литературу по программированию веб-сайтов.
2. Ознакомиться с основными принципами создания веб-сайта
3. Рассмотреть основные этапы создания веб-сайтов.
4. Описать задачу по созданию веб-сайта в заданных условиях
5. Создать веб-сайт для заказчика на основе React , с имеющимся объемом информации
6. Проверить соответствие готового сайта поставленной задаче заказчика

Инструменты:

* Power Poin: для создания макета;
* Node.js: Для установки зависимостей и работы с сервером;
* npm или Yarn: Для управления пакетами и зависимостями проекта;
* Create React App: Для быстрой настройки начального проекта на React;
* React Router: Для реализации маршрутизации и навигации по страницам приложения;
* Redux или Context API: Для управления состоянием приложения, особенно если оно большое и сложное;
* Styled Components или CSS Modules: Для стилизации компонентов с использованием CSS-in-JS или модульного подхода;
* Git и GitHub: Для контроля версий проекта и совместной работы;

Состав команды: Алексеев Михаил Сергеевич (Программист).

**Глава 1. Основы создания веб-приложения (сайта) на React**

**1.1 Веб-сайт как ключевой элемент цифровой среды: определение, структура и функции**

В век развития информационных технологий веб-сайты имеют высокое значение в цифровом пространстве, используемые в различных целях.

Под веб-сайтом мы подразумеваем совокупность взаимосвязанных веб-страниц, опубликованных в открытом доступе в сети Интернет. Страницы сайта объединены общим доменным именем. Используется для демонстрации пользователям информации или для развлечения.

Веб сайты могут представлять из себя как простые статичные страницы, связки страниц или сложные динамические платформы, которые взаимодействуют с пользователем в режиме реального времени.

Пример классификации веб-сайтов по целям и выполняемым задачам:

- информационные;

- коммерческие;

- образовательные;

- развлекательные и социальные платформы.

Структура веб-сайта включает несколько основных компонентов, такие как: заголовки, навигацию (меню), блоки контента, сам контент. В результате комбинации должен получаться удобный интерфейс для пользователей, позволяющий легко ориентироваться по сайту и получать необходимую информацию / реализовывать необходимые действия.

Среди функций сайта можно выделить такие примеры как: предоставление информации, возможность взаимодействия с пользователями через формы обратной связи, функции онлайн-магазинов, ведение блогов и платформы для обмена знаниями.

Кроме того, веб-сайты являются важным инструментом маркетинга и продвижения брендов, позволяя компаниям расширять свое присутствие в интернете и привлекать целевую аудиторию.

Таким образом, веб-сайт является неотъемлемой частью нашей повседневной жизни, выступая связующим элементом между пользователями и информацией. Осознание его структуры и функций помогает более эффективно использовать этот инструмент в различных сферах деятельности.

**1.2 Специализация программиста по созданию сайтов: навыки, технологии и профессиональные перспективы.**

Специализация программиста по разработке веб-сайтов приобрела высокую значимость с увеличением числа онлайн-платформ и сервисов. Компаниям, которые хотят развиваться в ногу со временем нужны квалифицированные специалисты, способные использовать свой опыт и навыки в создании и поддержании представленности их бизнеса на веб-ресурсах.

Одним из основных аспектов работы программиста является знание языков программирования, используемых для создания веб-сайтов. Наиболее популярными являются HTML, CSS и JavaScript. HTML (HyperText Markup Language) служит основой для формирования структуры веб-страниц, позволяя разработчикам организовывать контент и определять его форматирование. CSS (Cascading Style Sheets) отвечает за внешний вид и стилизацию страниц, помогая создавать привлекательный и удобный интерфейс. JavaScript добавляет интерактивные элементы на сайт, что позволяет реализовывать динамические функции, такие как формы и анимации.

Современные веб-технологии также предполагают использование различных фреймворков и библиотек, которые значительно упрощают процесс разработки. Например, библиотеки React и Vue.js позволяют создавать сложные пользовательские интерфейсы, а фреймворки Angular помогают организовывать код и обеспечивать его поддержку. Знание этих инструментов становится необходимым для программистов, стремящихся оставаться конкурентоспособными на рынке труда.

Кроме технических навыков, программисту по созданию сайтов также необходимо знать основы веб-дизайна и пользовательского опыта (UX). Понимание принципов дизайна и умение создавать удобные интерфейсы имеют решающее значение для успешной разработки веб-проектов. Эффективный UX-дизайн помогает удовлетворять потребности пользователей, что способствует повышению конверсии и популярности сайта.

Также важно иметь представление о серверной разработке и работе с базами данных. Программисты, занимающиеся созданием сайтов, часто используют серверные языки программирования, такие как PHP, Python или Ruby, а также системы управления базами данных, такие как MySQL или PostgreSQL. Эти навыки позволяют им разрабатывать полнофункциональные веб-приложения, которые могут обрабатывать пользовательские запросы и взаимодействовать с хранимыми данными.

Не менее значимым является знание основ SEO (Search Engine Optimization) — оптимизации для поисковых систем. Понимание принципов SEO помогает программистам разрабатывать сайты, которые лучше индексируются и занимают более высокие позиции в поисковых системах, что, в свою очередь, способствует привлечению большего числа пользователей.

С точки зрения карьерных перспектив, программисты, специализирующиеся на создании сайтов, имеют широкий выбор путей развития. Они могут работать как в крупных компаниях, так и в небольших стартапах, заниматься фрилансом или открывать собственные веб-студии. С ростом цифровизации и увеличением числа интернет-проектов спрос на квалифицированных специалистов будет только расти. Программисты также могут развивать свои навыки в смежных областях, таких как разработка мобильных приложений, кибербезопасность или анализ данных.

**1.3 Особенность и преимущества создания приложения на React: современные подходы и возможности**

Библиотека React, созданная Facebook, стала очень популярной среди разработчиков веб-приложений. С момента своего появления она завоевала признание благодаря своей гибкости, высокой производительности и возможности создания масштабируемых приложений. В этом тексте я хотел бы поделиться своими мыслями о ключевых особенностях и преимуществах разработки сайтов на React, а также современными подходами, которые делают эту библиотеку такой привлекательной для разработчиков.

Одной из самых заметных особенностей React является его компонентный подход. Приложения на React строятся из независимых и переиспользуемых компонентов. Это значит, что сложные пользовательские интерфейсы можно разбивать на более простые части, что значительно упрощает процесс разработки и поддержания кода. Каждый компонент может иметь свои собственные состояния и свойства, что делает их гибкими и динамичными. Такой подход не только улучшает организацию кода, но и помогает легче находить и исправлять ошибки.

Еще одной важной особенностью React является виртуальный DOM. В отличие от традиционных методов, где изменения в интерфейсе приводят к полному обновлению реального DOM, React использует виртуальное представление DOM для оптимизации производительности. Когда происходит обновление компонента, React сначала обновляет виртуальный DOM, а затем сравнивает его с реальным DOM. Это позволяет минимизировать количество операций, необходимых для обновления интерфейса, что значительно ускоряет работу приложений и улучшает пользовательский опыт.

Преимущества использования React также включают активное сообщество и широкий набор инструментов и библиотек. Благодаря большому количеству разработчиков, работающих с React, существует множество ресурсов, документации и обучающих материалов, что облегчает освоение этой технологии для новичков. Кроме того, экосистема React включает множество библиотек, таких как Redux для управления состоянием и React Router для маршрутизации, которые помогают разработчикам создавать более сложные и функциональные приложения.

Также стоит отметить, что React поддерживает концепцию одностороннего потока данных. Это означает, что данные передаются от родительских компонентов к дочерним, что упрощает управление состоянием приложения. Такой подход позволяет легче отслеживать изменения и предотвращает возможные проблемы, связанные с синхронизацией состояния. Это значительно упрощает отладку и тестирование приложений, так как разработчики могут быть уверены, что изменения в состоянии происходят в предсказуемом порядке.

Современные подходы к разработке на React также включают использование функциональных компонентов и хуков. Функциональные компоненты, в отличие от классовых, предлагают более простой и лаконичный синтаксис, что делает код более читаемым и удобным для работы. Хуки позволяют добавлять состояние и другие функции React в функциональные компоненты, что дает разработчикам возможность использовать преимущества React без необходимости создания классовых компонентов. Это делает разработку более удобной и понятной.

Кроме того, React хорошо интегрируется с другими технологиями и фреймворками. Например, разработчики могут легко использовать React вместе с серверными языками, такими как Node.js, или интегрировать его с популярными системами управления контентом (CMS). Это делает React универсальным инструментом для создания как простых, так и сложных веб-приложений.

В итоге, React предлагает разработчикам множество преимуществ и возможностей для создания современных веб-сайтов. Его компонентный подход, виртуальный DOM, активное сообщество и поддержка одностороннего потока данных делают его одним из самых привлекательных инструментов для веб-разработки. Учитывая современные подходы, такие как использование функциональных компонентов и хуков, React продолжает оставаться на переднем крае технологий, позволяя разработчикам создавать высококачественные и производительные веб-приложения.

**1.4 Инструменты React для создания веб-страниц**

Прежде всего, основным инструментом является сама библиотека React. Она предоставляет разработчикам средства для создания компонентов, которые могут быть использованы повторно, что упрощает как разработку, так и поддержку кода. React имеет удобный интерфейс для работы с состоянием компонентов, а также для обработки событий и управления жизненным циклом этих компонентов.

Для управления состоянием в более сложных приложениях часто используется библиотека Redux. Она позволяет централизовать управление состоянием и облегчает взаимодействие между различными компонентами. Redux предлагает мощные инструменты для отслеживания изменений состояния, что особенно полезно в крупных приложениях с множеством взаимодействий.

Еще одним важным инструментом является React Router. Эта библиотека позволяет реализовать маршрутизацию в приложениях на React, что упрощает переходы между различными страницами и компонентами. React Router особенно полезен для создания одностраничных приложений (SPA), где необходимо быстро загружать новые данные без перезагрузки страницы.

С точки зрения стилизации компонентов на React существует множество подходов. Одним из самых популярных является библиотека styled-components, которая позволяет писать CSS прямо в JavaScript. Такой подход делает стилизацию более модульной и удобной, так как стили привязываются к компонентам. Также есть CSS-in-JS решения, такие как Emotion, которые предлагают аналогичные возможности.

Для быстрого старта новых проектов на React разработчики могут воспользоваться утилитой Create React App. Этот инструмент позволяет быстро создать новый проект, автоматически настраивая все необходимые зависимости и конфигурации. Create React App включает в себя такие инструменты, как Webpack и Babel, которые отвечают за сборку и транспиляцию кода.

Для тестирования компонентов на React часто используется библиотека Jest, разработанная Facebook. Jest предоставляет широкие возможности для написания тестов и включает функции для тестирования компонентов, проверки их состояния и взаимодействий. В сочетании с React Testing Library разработчики могут создавать надежные тесты, которые проверяют функциональность компонентов в реальных сценариях.

Еще одним полезным инструментом для работы с API и выполнения асинхронных запросов является Axios — популярная библиотека для выполнения HTTP-запросов. Axios предлагает простой интерфейс для работы с API, поддерживает промисы и позволяет удобно обрабатывать запросы и ответы.

Кроме того, для отладки и разработки приложений на React полезно использовать встроенные инструменты разработчика в браузерах, например, Chrome DevTools. С их помощью разработчики могут отслеживать производительность приложений, анализировать состояние компонентов и выявлять потенциальные проблемы.

В итоге, экосистема React предлагает множество инструментов, которые делают процесс создания веб-страниц более удобным и эффективным. Использование таких инструментов, как Redux для управления состоянием, React Router для маршрутизации, styled-components для стилизации, а также Jest для тестирования, позволяет разработчикам создавать качественные и масштабируемые приложения. Эти инструменты помогают упростить разработку, повысить производительность и улучшить взаимодействие с пользователями.

**1.5 Технология гипертекста — язык HTML**

HTML, или HyperText Markup Language, — это основной язык разметки, который используется для создания веб-страниц. Он является основой технологии гипертекста и позволяет разработчикам структурировать и представлять информацию в интернете. Благодаря HTML мы можем организовывать текст, изображения, ссылки и другие элементы, что делает его незаменимым инструментом в веб-разработке.

Основная функция HTML заключается в разметке содержимого документа. Разработчики используют HTML для создания различных элементов, таких как заголовки, параграфы, списки и таблицы. Каждый элемент в HTML создается с помощью тегов, которые заключаются в угловые скобки. Например, тег <h1> обозначает заголовок первого уровня, а тег <p> — абзац текста.

Одной из самых важных особенностей HTML является возможность создания гиперссылок. С помощью тега <a> разработчики могут связывать разные веб-страницы и ресурсы, что позволяет пользователям легко перемещаться между страницами и находить нужную информацию. Гиперссылки могут вести как на страницы одного сайта, так и на внешние ресурсы, что значительно расширяет возможности навигации.

HTML также поддерживает встраивание мультимедийных элементов, таких как изображения, видео и аудио. Например, тег <img> используется для добавления изображений, а теги <video> и <audio> — для встраивания мультимедийного контента. Это делает веб-страницы более привлекательными и интерактивными, что помогает удерживать интерес пользователей.

С течением времени язык HTML развивался, и сейчас мы используем HTML5, который предлагает новые возможности для разработчиков. В HTML5 появились семантические элементы, такие как <header>, <footer>, <article> и <section>, которые позволяют лучше структурировать контент и делают его более доступным как для пользователей, так и для поисковых систем.

HTML также хорошо работает в связке с каскадными таблицами стилей (CSS), которые помогают управлять внешним видом страниц. С помощью CSS разработчики могут изменять шрифты, цвета, отступы и другие визуальные аспекты элементов, что позволяет создавать привлекательный и уникальный дизайн.

В заключение, HTML является основой веб-разработки и важной технологией гипертекста. Его простота и гибкость делают его доступным для изучения, как для новичков, так и для опытных разработчиков. Используя HTML, можно создавать структурированные, интерактивные и мультимедийные веб-страницы, которые соответствуют современным требованиям пользователей.

**Глава 2. Подготовка к созданию веб-сайта**

**2.1 Определение требований заказчика и функциональности сайта**

Для определения требований заказчика к функциональности сайта на первоначальном этапе удобно использовать бриф с перечнем вопросов, которые позволят выстроить взаимопонимание. Брифы имеют свою особенность ввиду общий целей сайта (коммерческий, информационный и т.д.)

Бриф включает следующие вопросы (минимальный перечень):

- Описание целей сайта

- Описание целевой аудитории

- Выбор функций сайта (например: авторизация пользователя, поисковые функции, интернет-магазин)

- Референсы (примеры сайтов, которые нравятся и почему)

- Какой контент необходимо разместить на сайте

Пример заполненного брифа [1] представлен в Приложении.

После получения брифа необходимо изучить конкурентную среду и подготовить спецификацию проекта для описания конечного результата, на основе критериев которого мы поймем, что сайт соответствует ожиданиям заказчика.

**2.2 Сбор дополнительной информации**

Конкурентный анализ является важным этапом разработки веб-сайтов, особенно для программистов, занимающихся версткой новых проектов. Он позволяет не только понять текущие тренды в дизайне и функциональности, но и выявить сильные и слабые стороны конкурентов. В данной работе мы рассмотрим основные шаги и методы, которые помогут программисту провести эффективный конкурентный анализ перед началом верстки.

* **Определение целей анализа.** Перед тем как приступить к исследованию, важно четко определить цели конкурентного анализа. Это могут быть: понимание целевой аудитории конкурентов; выявление уникальных предложений и преимуществ; анализ пользовательского опыта (UX) на сайтах конкурентов, оценка структуры, дизайна и функциональности – что больше подходит непосредственно для задачи программиста.
* **Поиск сайтов конкурентов через поисковые системы.** Поиск по ключевым словам, связанным с проектом. Выбрать 2-4 сайта среди первых, представленных в результатах.
* **Анализ сайтов через специализированные сервисы.** SimilarWeb, SEMrush и Ahrefs предоставляют детальную информацию о трафике, ключевых словах и стратегиях SEO.
* **Анализ пользовательского опыта (UX).** Оценка навигации и доступности информации. Оценка дизайна и визуальных элементов. Проверка скорости загрузки страниц и отображения на мобильных устройствах.
* **Выявление ключевых особенностей и трендов.** На основе собранной информации выделение ключевых особенностей, которые делают сайты конкурентов успешными. Это может включать: уникальный контент и предложения; инновационные функциональные возможности.
* **Оценка аспектов для программиста.**, такие как: дизайн (цветовая палитра, шрифты); структура (количество страниц, разделы); функциональность (формы обратной связи, интеграции).

По итогу создается таблица, в которой сравниваются различные аспекты сайтов конкурентов.

Пример таблицы представлен в Приложении [2].

На основе проведенного анализа сформировываются выводы, которые помогут в процессе верстки.

Конкурентный анализ помогает программисту не только лучше понять рынок, но и разработать более эффективный и привлекательный продукт для конечного пользователя. Правильный подход к анализу позволит избежать распространенных ошибок и создать качественный сайт, соответствующий современным требованиям и ожиданиям пользователей.

**2.3 Создание спецификации проекта и согласование с заказчиком**

Создание спецификации проекта сайта является важным этапом в процессе разработки, обеспечивающим четкое понимание целей, требований и ожиданий всех участников проекта. Спецификация служит основой для дальнейшей работы над сайтом и играет ключевую роль в согласовании между разработчиком и заказчиком. В данной главе мы рассмотрим основные этапы создания спецификации, а также важность ее согласования с заказчиком.

На начальном этапе необходимо определить цели, которые должен достичь сайт. Это может включать:

- Увеличение продаж или лидов.

- Повышение узнаваемости бренда.

- Улучшение пользовательского опыта.

Четкое понимание целей поможет формулировать задачи, которые необходимо решить в рамках проекта.

Сбор требований – это процесс, в ходе которого разработчик взаимодействует с заказчиком, чтобы понять его ожидания и потребности. Важно задать открытые вопросы и внимательно выслушать ответы. Ключевые аспекты, которые следует обсудить:

- Функциональность сайта (например, наличие интернет-магазина, блога, форм для обратной связи).

- Дизайн и пользовательский интерфейс (предпочтения по стилю, цветам, структуре).

- Технические требования (выбор платформы, хостинга, интеграции с другими системами).

На основе собранных данных разрабатывается спецификация проекта. Этот документ должен быть структурированным и содержать следующие разделы:

- Введение, описывающее проект и его цели.

- Функциональные требования, описывающие, что сайт должен делать.

- Нефункциональные требования, такие как производительность, безопасность и доступность.

- Описание дизайна, включая макеты и прототипы.

- Этапы разработки и сроки выполнения.

После создания спецификации необходимо провести ее обсуждение с заказчиком. Важно обеспечить понимание всех аспектов документа и убедиться, что он соответствует ожиданиям. На этом этапе могут возникнуть следующие действия:

- Презентация спецификации, в ходе которой разработчик объясняет каждый раздел и отвечает на вопросы.

- Обсуждение и внесение правок на основе отзывов заказчика.

- Подписание спецификации, что подтверждает согласие сторон с содержанием документа.

Спецификация проекта должна быть живым документом, который может изменяться в процессе разработки. Важно установить процесс управления изменениями, который включает:

- Регулярные встречи с заказчиком для обсуждения хода работ и возможных корректировок.

- Оформление всех изменений в спецификации, чтобы избежать недоразумений.

Создание спецификации проекта сайта и ее согласование с заказчиком – это ключевые этапы, которые определяют успех всего проекта. Четко сформулированные требования и открытое взаимодействие с заказчиком помогают избежать множества проблем и недопонимания в процессе разработки. В результате, грамотно составленная спецификация становится основой для успешного завершения проекта и достижения поставленных целей.

Пример спецификации представлен в Приложении [3].

2.3 Прототипирование сайта

Прототипирование веб-сайта позволяет визуализировать и протестировать основные идеи и концепции перед их реализацией. Данная глава посвящена процессу прототипирования сайта по продаже техники, который включает в себя создание и оценку различных версий дизайна, функциональности и пользовательского интерфейса.

Целью прототипирования является выявление и уточнение требований к сайту на ранних этапах разработки. Это позволяет сократить время и затраты на изменения в будущем. Основные задачи прототипирования включают:

- **Определение структуры сайта:** Установление главных разделов и подсайтов, которые будут включены в проект.

- **Создание пользовательских сценариев:** Определение ключевых сценариев взаимодействия пользователей с сайтом, что помогает понять, как они будут перемещаться по страницам и использовать функционал.

- **Визуализация интерфейса:** Создание эскизов или макетов страниц, которые отображают расположение элементов интерфейса, таких как кнопки, формы, меню и изображения товаров.

- **Получение обратной связи:** Сбор мнений пользователей и заинтересованных сторон о прототипах для дальнейшего улучшения и доработки.

Для прототипирования сайта используются различные инструменты и методы, позволяющие создавать как низкоуровневые, так и высокоуровневые прототипы. К ним относятся:

- **Эскизы и бумажные прототипы:** Начальный этап, на котором создавались простые эскизы страниц на бумаге. Этот метод позволяет быстро генерировать идеи и вносить изменения на ранних стадиях.

- **Wireframes:** Создание каркасных моделей (wireframes) с помощью инструментов, таких как Figma или Adobe XD. Wireframes помогают визуализировать структуру сайта и расположение основных элементов без детальной проработки дизайна.

- **Интерактивные прототипы:** Разработка интерактивных прототипов, которые позволяют пользователям взаимодействовать с интерфейсом. Это помогает выявить потенциальные проблемы в навигации и функциональности до начала разработки. Инструменты, такие как InVision и Marvel, были использованы для создания интерактивных прототипов.

Прототипирование сайта по продаже техники проходило в несколько этапов:

* + - * **Исследование и анализ:** На этом этапе была проведена работа по изучению конкурентов и анализу потребностей целевой аудитории. Это помогло определить ключевые функции и элементы, которые должны быть реализованы на сайте.
      * **Создание низкоуровневых прототипов:** Сначала были разработаны эскизы и wireframes для основных страниц сайта, таких как главная страница, категория товаров и страница товара. Эти прототипы фокусировались на структуре и навигации.
      * **Разработка высокоуровневых прототипов:** На основе полученной обратной связи от пользователей и заинтересованных сторон были созданы более детализированные интерактивные прототипы. Эти прототипы включали в себя элементы дизайна, такие как цветовая палитра, шрифты и изображения.
      * **Тестирование и итерации:** Прототипы были протестированы среди потенциальных пользователей, что позволило выявить недостатки и получить предложения по улучшению. На основе собранной информации были внесены изменения и доработки.

В результате процесса прототипирования были достигнуты следующие результаты:

- **Улучшение навигации:** Прототипы позволили выявить возможные проблемы в навигации сайта, что привело к более логичной и удобной структуре.

- **Оптимизация пользовательского опыта:** Полученная обратная связь помогла определить, какие элементы интерфейса наиболее удобны для пользователей, что улучшило общий пользовательский опыт.

- **Согласование требований:** Прототипы помогли согласовать требования и ожидания всех заинтересованных сторон, что снизило риск недопонимания на этапе разработки.

Прототипирование сайта является критически важным этапом, позволяющим визуализировать и тестировать идеи до начала разработки.

Этот процесс помогает сократить время и затраты на внесение изменений, улучшает пользовательский опыт и позволяет создать сайт, соответствующий потребностям целевой аудитории. В следующей главе будет рассмотрен процесс разработки компонентов сайта на основе созданных прототипов.

**2.4 Структурирование проекта**

Структурирование сайта является ключевым этапом, который следует за прототипированием. Этот процесс включает в себя организацию и иерархию страниц, а также определение логики навигации и взаимодействия между различными компонентами. В данной главе будет рассмотрен процесс структурирования сайта по продаже техники, включая создание схемы сайта, описание основных разделов и их взаимосвязей.

Целью структурирования сайта является создание логичной и интуитивно понятной структуры, которая обеспечит пользователям удобный доступ к информации и функционалу. Основные задачи, которые необходимо решить на этом этапе, включают:

* + - * Определение основных разделов: Выделение ключевых разделов сайта, которые должны быть представлены пользователям.
      * Создание иерархии страниц: Установление иерархии страниц и подстраниц, чтобы обеспечить логичное и последовательное движение по сайту.
      * Разработка навигационной структуры: Проектирование навигационного меню и других элементов, которые помогут пользователям легко находить нужную информацию.
      * Оптимизация пользовательского опыта: Обеспечение удобства и простоты в использовании сайта для конечных пользователей.

На этом этапе разрабатывается схема сайта, представляющая собой графическое изображение структуры и иерархии страниц. Схема сайта включает следующие основные разделы:

- Главная страница: Центральный узел, откуда пользователи могут перейти к другим разделам.

- Каталог товаров: Раздел, содержащий список всех доступных товаров, сгруппированных по категориям.

- Страница товара: Индивидуальная страница для каждого товара, где представлена подробная информация, включая описание, характеристики, изображения и кнопки для добавления в корзину.

- Корзина: Раздел, где пользователи могут просмотреть добавленные товары, изменить их количество или удалить.

- Оформление заказа: Страница, на которой пользователи могут ввести свои данные и подтвердить заказ.

- Контакты и поддержка: Раздел с контактной информацией, часто задаваемыми вопросами и формой обратной связи.

Схему сайта можно создавать с помощью инструментов визуализации, таких как Lucidchart или Miro, что позволиляет наглядно представить взаимосвязи между разделами.

Навигационная структура сайта разрабатывается на основе созданной схемы и включает в себя:

- Главное меню: Навигационное меню, доступное на всех страницах, которое содержит ссылки на основные разделы сайта, такие как "Главная", "Каталог товаров", "Корзина", "Контакты" и "О нас".

- Боковые панели: Для улучшения навигации в разделе каталога товаров были добавлены боковые панели с фильтрами по категориям, ценам и характеристикам.

- Хлебные крошки: Введение хлебных крошек на страницах товара и в корзине, что позволяет пользователям легко возвращаться к предыдущим уровням навигации.

Структурирование сайта также включает в себя установление иерархии страниц. Это помогает определить, какие страницы являются основными, а какие подстраницами. Например:

- **Главная страница**

- **Каталог товаров**

- **Категория 1**

- **Товар 1**

- **Товар 2**

- **Категория 2**

- **Товар 3**

- **Товар 4**

- **Корзина**

- **Оформление заказа**

- **Контакты**

Такое структурирование позволяет пользователям быстро ориентироваться в содержимом сайта и находить нужную информацию.

В процессе структурирования сайта особое внимание уделяется оптимизации пользовательского опыта.Логичное расположение элементов: Размещение элементов интерфейса и навигации в соответствии с ожиданиями пользователей. Упрощение навигации: Минимизация количества кликов, необходимых для доступа к нужной информации. Тестирование структуры: Проведение тестов с реальными пользователями для оценки удобства навигации и выявления возможных проблем.

Структурирование сайта является важным этапом, который обеспечивает логичную организацию страниц и удобную навигацию. Создание схемы сайта, определение иерархии страниц и разработка навигационной структуры позволяют пользователям легко находить необходимую информацию и улучшает общий пользовательский опыт. В следующей главе будет рассмотрен процесс разработки компонентов сайта на основе созданной структуры.

**2.5 Разработка компонентов**

Разработка компонентов следует за структурированием. На этом этапе программист реализует функциональные элементы интерфейса, опираясь на ранее разработанную структуру и прототипы. В данной главе будет рассмотрен процесс разработки компонентов, включая выбор технологий, создание базовых и сложных компонентов, интеграцию с API, а также стилизацию интерфейса.

Процесс разработки начался с создания базовых компонентов, которые будут использоваться на разных страницах сайта. К ним относятся:

- Кнопки: Разработка многоразового компонента кнопки с заданными стилями и поведением.

- Карточки товаров: Компонент, отображающий информацию о каждом товаре, включая изображение, название и цену.

- Формы: Компоненты для ввода данных, такие как форма оформления заказа и форма для обратной связи. Формы включают валидацию с помощью библиотек, таких как Formik.

После создания базовых компонентов программист переходит к разработке более сложных элементов, которые требуют взаимодействия с состоянием и API:

- Корзина: Компонент, который отображает выбранные товары и позволяет пользователям изменять количество или удалять товары. Он использует состояние для управления содержимым корзины:

- Интерактивные элементы: Реализация функций фильтрации и сортировки товаров в каталоге, используя состояние и обработчики событий.

Стилизация компонентов может быть выполнена с использованием технологии «styled-components или CSS Modules». Программист создает отдельные стили для каждого компонента, что позволиляет достичь хорошей организации кода и избежать конфликтов стилей.

Каждый компонент проходит тестирование для проверки его функциональности и юзабилити. Программист использует библиотеки, такие как Jest и React Testing Library, для написания тестов, которые обеспечивают надежность и стабильность работы компонентов.

Разработка компонентов является ключевым этапом в создании сайта по продаже техники. Этот процесс включает в себя создание базовых и сложных компонентов, интеграцию с API и стилизацию интерфейса. Правильная реализация компонентов обеспечивает функциональность и удобство использования сайта, что в конечном итоге способствует положительному пользовательскому опыту. В следующей главе будет рассмотрен процесс реализации маршрутизации на сайте, который позволит пользователям легко перемещаться между страницами.

**2.6 Реализация маршрутизации**

Реализация маршрутизации веб-сайта на React обеспечивает навигацию между различными страницами и компонентами приложения. В данной главе будет рассмотрен процесс настройки маршрутизации на сайте по продаже техники с использованием библиотеки **React Router.** Представлены основные принципы маршрутизации, создание маршрутов и обработку параметров, а также навигацию и управление состоянием маршрутов.

Первым шагом в реализации маршрутизации является установка библиотеки React Router.

После установки импортируются необходимые компоненты из библиотеки в главный файл приложения (обычно это `App.js`), где будет происходить настройка маршрутов.

В React Router используются несколько ключевых компонентов для реализации маршрутизации:

- **BrowserRouter:** Оборачивает приложение и позволяет использовать маршрутизацию в приложении.

- **Route:** Определяет маршруты, связывая URL с определенным компонентом.

- **Switch:** Оборачивает несколько маршрутов и отображает только первый подходящий маршрут.

Каждый маршрут связывает определенный путь URL с компонентом, который будет отображаться при переходе по этому пути. Например, маршрут для страницы каталога товаров имеет путь /products, а для страницы товара — /products/:id, где :id является параметром, который позволяет динамически подставлять идентификатор товара.

На этом этапе также создаются компоненты для каждой из страниц, таких как Home, Products, ProductDetail, Cart и NotFound. Эти компоненты содержат логику и интерфейс, которые будут отображаться пользователям.

Для страниц, где требуется отображение конкретного товара, используется параметр маршрута. В компоненте ProductDetail можно получить ID товара из URL с помощью хука useParams.

Для удобства пользователей важно реализовать навигацию между страницами. Это можно сделать с помощью компонента Link, который позволяет создать ссылки на другие маршруты.

Важно обеспечить обработку случаев, когда пользователь пытается получить доступ к несуществующему маршруту. Для этого в маршрутах добавляется последний маршрут, который будет отображаться, если ни один из предыдущих не совпадает. Компонент NotFound может содержать сообщение о том, что страница не найдена, и ссылку для возврата на главную страницу.

После завершения настройки маршрутизации программист проводит тестирование, чтобы убедиться, что все маршруты работают корректно. Это включает в себя:

- Проверку всех маршрутов на предмет их правильной работы.

- Тестирование переходов между страницами и отображения компонентов.

- Убедиться, что параметры маршрутов корректно передаются и обрабатываются.

Реализация маршрутизации является важным этапом в разработке сайта по продаже техники на React, обеспечивая удобное перемещение между страницами и доступ к различному контенту. Правильная настройка маршрутов, обработка параметров и навигации способствует улучшению пользовательского опыта. В следующей главе будет рассмотрен процесс интеграции с API, который позволит сайту получать и отображать данные о товарах.

**2.7 Интеграция с API**

Интеграция с API является критически важным этапом в разработке веб-сайта по продаже техники. Она позволяет сайту взаимодействовать с внешними источниками данных, обеспечивая динамическое получение информации о товарах, пользователях и других ресурсах. В этой подглаве будет рассмотрен процесс интеграции с API, включая настройку запросов, обработку ответов и управление состоянием приложения.

Для интеграции сайта выбирается подходящее API, которое предоставляет данные о товарах, их характеристиках, ценах и наличии. Важно, чтобы API поддерживал необходимые методы запросов и предоставлял данные в формате, удобном для обработки, например, в формате JSON.

Для выполнения HTTP-запросов к API на сайте используется библиотека Axios, которая облегчает работу с асинхронными запросами. Сначала необходимо установить Axios.

После настройки запросов важно правильно обрабатывать ответы от API. Это включает в себя проверку статуса ответа и обработку ошибок. Например, можно использовать конструкцию try-catch для обработки исключений.

Полученные данные интегрируются в компоненты сайта. Например, в компоненте Products можно вызвать функцию для получения списка товаров и обновить состояние компонента.

Интеграция с API также требует обновления состояния приложения в ответ на изменения данных. Например, при добавлении товара в корзину или изменении его количества, необходимо обновить состояние как компонента, так и глобального состояния, если используется Redux или Context API.

После завершения интеграции с API следует провести тестирование, чтобы убедиться, что все запросы выполняются корректно, данные отображаются правильно, а ошибки обрабатываются должным образом. Важно проверить, как приложение реагирует на различные сценарии, такие как отсутствие интернет-соединения или ошибки на стороне сервера.

Правильная настройка запросов, обработка ответов и управление состоянием приложения обеспечивают надежность и стабильность работы сайта. В следующей главе будет рассмотрен процесс тестирования и отладки приложения, что поможет выявить и устранить возможные проблемы перед запуском.

**2.8 Обеспечение поддержки и обновления сайта**

Обеспечение поддержки и обновления сайта включает в себя регулярное обслуживание, исправление ошибок, обновление функционала и контента, а также адаптацию сайта к изменяющимся требованиям пользователей и технологическим стандартам. В данной подглаве будут рассмотрены основные аспекты, связанные с поддержкой и обновлением сайта по продаже техники, разработанного с использованием React.

Регулярное обслуживание сайта включает в себя мониторинг его работы, анализ производительности и безопасности, а также обновление библиотек и фреймворков. Важно следить за следующими аспектами:

- Обновление библиотек и фреймворков: Сайты, построенные на React, требуют своевременного обновления зависимостей. Использование npm или yarn позволяет легко обновлять пакеты и получать последние функции и исправления безопасности.

- Мониторинг производительности: Использование инструментов, таких как React Profiler и Lighthouse, позволяет отслеживать производительность компонентов и выявлять узкие места, влияющие на пользовательский опыт.

- Проверка безопасности: Регулярный аудит безопасности, включая использование инструментов для статического анализа кода, помогает выявлять уязвимости и предотвращать атаки.

Ошибки и баги могут возникать в любом приложении на React, и важно оперативно реагировать на них. Для этого рекомендуется:

- Система мониторинга: Внедрение систем мониторинга, таких как Sentry или LogRocket, позволяет отслеживать ошибки в реальном времени и получать уведомления о их возникновении.

- Обратная связь от пользователей: Создание форм обратной связи или использование специализированных инструментов позволяет пользователям сообщать о проблемах и недочетах, что помогает быстрее их устранять.

- Регулярные тесты: Проведение юнит-тестирования с использованием таких библиотек, как Jest и React Testing Library, помогает заранее выявлять и устранять возможные ошибки в коде.

Актуальность контента является ключевым фактором успешной работы сайта. Регулярные обновления помогают поддерживать интерес пользователей и повышают SEO-оптимизацию. Основные действия включают:

- Добавление новых товаров: Регулярное обновление ассортимента и добавление новых товаров в каталог позволяют привлекать новых клиентов и удерживать существующих.

- Обновление информации: Проверка актуальности информации о товарах, ценах и акциях, а также обновление статей и описаний на сайте.

- Создание нового контента: Публикация статей, новостей или блогов, связанных с продуктами или услугами, может повысить интерес к сайту и улучшить его видимость в поисковых системах.

Технологический мир постоянно меняется, и веб-сайты на React должны адаптироваться к новым условиям. Это может включать:

- Мобильная оптимизация: Обеспечение адаптивного дизайна с использованием таких инструментов, как CSS Grid и Flexbox, а также использование медиазапросов для оптимизации отображения на мобильных устройствах.

- Изменение функционала: Внедрение новых функций и улучшений на основе отзывов пользователей и анализа их поведения на сайте.

- Соблюдение стандартов: Адаптация сайта к новым стандартам веб-разработки, включая доступность и соответствие требованиям GDPR или других нормативных актов.

Поддержка и обновление сайта также требуют наличия качественной документации и обучения для команды разработчиков. Важно:

- Создание документации: Подробная документация по коду, архитектуре приложения и процессам обновления помогает новым членам команды быстро войти в курс дела.

- Обучение команды: Регулярные тренинги и семинары по новым технологиям и методам разработки на React помогают команде оставаться в курсе последних тенденций.

Регулярное обслуживание, исправление ошибок, обновление контента и адаптация к изменениям позволяют поддерживать сайт в актуальном состоянии и обеспечивать положительный пользовательский опыт. Важно помнить, что успешный сайт требует постоянного внимания и усилий со стороны команды разработчиков, что в конечном итоге приводит к его успешному функционированию и росту.

**Глава 3. Создание веб-сайта**

**3.1 Создание макета**

Создание макета сайта определяет внешний вид и функциональность. Для этого я провел конкурентный анализ сайтов конкурентов и учел спецификацию, согласованную с клиентом.

Первым шагом в создании макета было проведение конкурентного анализа. Я изучил несколько популярных интернет-магазинов, специализирующихся на продаже техники. Основное внимание уделено следующим аспектам:

- **Структура и навигация:** как организованы разделы и категории товаров, а также как осуществляется навигация по сайту.

- **Дизайн и пользовательский интерфейс:** цветовые схемы, шрифты, размещение элементов и общая эстетику.

- **Функциональность:** какие функции предлагают конкуренты, такие как фильтры, сортировка, сравнение товаров и системы отзывов.

Параллельно с конкурентным анализом проводилась работа над спецификацией, для согласования с заказчиком.

Следующим этапом реализован макет. Для этого я использовался PowerPoint**.** В макете отражены:

- **Структура:** каркас сайта, с определением основных разделов и навигации главной страницы, каталога, страницы товара, корзины.

- **Дизайн компонентов:** кнопки, карточки товаров, формы и слайдер, интуитивно понятных и эстетически привлекательных.

После завершения макета проведено обсуждение с клиентом для получения обратной связи. Коррективы не потребовались.

Макет позволил получить ясное представление сайта..

**3.2 Определение модулей сайта.**

Определение модулей было одним из ключевых этапов в создании сайта, так как правильная архитектура модулей позволяет упростить разработку, тестирование и его поддержку.

При проектировании сайта я выделил несколько ключевых разделов и функциональных модулей, которые обеспечивают пользователям полный спектр услуг. Основные модули, которые я определил, включают:

- **Главная страница:** Этот модуль представляет общее представление магазина и включает баннеры, популярные товары и новинки.

- **Каталог товаров:** Модуль, который отображает список доступных категорий товаров с вложенными внутри несколькими товарами.

- **Страница товара:** Этот модуль предоставляет детальную информацию о конкретном товаре, включая описание и характеристики.

- **Корзина:** Модуль, который позволяет пользователям просматривать добавленные товары и переходить к оформлению заказа.

- **Авторизация:** Модуль, который позволяет пользователям авторизоваться на сайте под своим профилем.

- **Поиск:** Модуль, который обеспечивает возможность быстрого поиска товаров по ключевым словам.

Каждый из этих модулей я разбил на более мелкие функциональные компоненты, которые обеспечивают выполнение конкретных задач. Рассмотрим основные функциональные модули подробнее.

Для главной страницы я создал несколько компонентов:

- **Слайдер:** Компонент, который отображает самые популярные группы продуктов.

- **Поиск по группам товарам:** Компонент, позволяющий пользователям быстро находить нужные товары по группе.

- **Список товаров:** Компонент, отвечающий за отображение карточек товаров с краткой информацией (изображение, название, цена).

- **Пагинация:** Компонент, который управляет навигацией между страницами каталога.

**- Конвертер валют:** модуль позволяющий оперативно узнать актуальный курс альтернативной валюты на товар.

В процессе разработки страницы товара я выделил следующие компоненты:

- **Изображение товара:** Компонент для отображения изображений товара в высоком качестве.

- **Описание и характеристики:** Компоненты, которые предоставляют подробную информацию о товаре.

Для модуля корзины я реализовал:

- **Список товаров в корзине:** Компонент, отображающий все добавленные товары с возможностью изменения количества и удаления.

- **Итоговая стоимость:** Компонент, вычисляющий общую сумму заказа в 2х валютах.

- **Кнопка оформления:** Компонент, позволяющий перейти к завершению покупки.

Для модуля поиска я реализовал:

- **Поле ввода:** Компонент, в который пользователи вводят ключевые слова для поиска.

- **Список результатов:** Компонент, отображающий найденные товары в реальном времени.

Для организации кода и упрощения разработки использовался компонентный подход, создавая отдельные файлы для каждого компонента и соответствующих стилей.

Правильная архитектура модулей и компонентов обеспечила удобство разработки сайта Структурирование функциональных модулей позволило создать гибкую и масштабируемую платформу.

**3.3 Верстка через Flexbox: Оптимальное расположение модулей и компонентов**

В процессе разработки было решено использовать Flexbox для создания гибких и адаптивных макетов, использование которого значительно упростило работу с различными элементами интерфейса.

Flexbox позволил легко управлять расположением элементов внутри контейнера, а также обеспечивает адаптивность и отзывчивость дизайна. Я изучил основные свойства Flexbox, такие как display: flex, flex-direction, justify-content и align-items, что дало мне необходимую базу для реализации макета.

При создании структуры компонентов для сайта я определил, какие элементы будут использовать Flexbox.

Каждая карточка товара была обернута в flex-контейнер, что позволяло легко управлять их расположением и адаптировать их под разные размеры экрана. Flex-wrap позволил автоматический переход карточек на новую строку при уменьшении ширины окна браузера.

Для модуля корзины был использовн Flexbox для выравнивания элементов формы, таких как поля ввода и кнопки. Это обеспечивало аккуратное и логичное расположение элементов.

Одним из ключевых преимуществ использования Flexbox стало создание адаптивного дизайна. Были настроены медиа-запросы, чтобы изменить свойства Flexbox в зависимости от ширины экрана.

После верстки было проведено тестирование на различных устройствах и браузерах, чтобы убедиться, что все элементы отображаются корректно и адаптивно. При изменении размеров окна браузера - все макеты элемента сохраняют свое расположение и функциональность.

**3.4 Добавление логики модулей и интеграция с API**

Этап добавления логики модулей и интеграции сайта с API стал решающим для обеспечения динамического взаимодействия пользователей с сайтом и позволил реализовать основные функциональные возможности, такие как получение данных о товарах, оформление заказов и управление пользовательскими сессиями.

Перед тем как приступить к интеграции с API, была определена логика каждого модуля сайта. Это включало в себя понимание того, как модули будут взаимодействовать друг с другом и какие данные они будут использовать. Основные модули, которые я выделил для логики, включают:

Модуль каталога отвечает за отображение списка товаров, фильтрацию и сортировку. Было определено, что он будет получать данные о товарах из API и обновлять состояние приложения при изменениях.

Модуль корзины позволяет пользователям добавлять, изменять количество и удалять товары из корзины. Эта информация также синхронизируется с локальным хранилищем для сохранения состояния между сессиями.

Модуль оформления заказа требует логики для сбора данных о пользователе и выбранных товарах.

Функция получения данных о товарах из API вызывалась в эффекте useEffect на странице каталога, что позволило загружать данные при первом рендере компонента.

После добавления логики модулей и интеграции с API было проведено тестирование всех функций на корректность загрузки данных о товарах, работы функциональности корзины и оформления заказа. В процессе отладки были использованы инструменты разработчика браузера для отслеживания состояния и выполнения запросов.

Процесс добавления логики модулей и интеграции с API позволил обеспечить динамическое взаимодействие пользователей с сайтом и реализовать основные функции, которые делают сайт удобным и функциональным.

**3.5 Финальная верстка: Наложение стилей и улучшение пользовательского интерфейса.**

На этапе финальной верстки был сделан акцент на наложении стилей и улучшении пользовательского интерфейса. Этот процесс был направлен на создание эстетически привлекательного и интуитивно понятного дизайна, который позволит пользователям легко взаимодействовать с сайтом.

Для стилизации компонентов было использовано **Styled Components,** что позволило применять CSS прямо в JavaScript-коде и создавать компоненты с инкапсулированными стилями. Это облегчило управление стилями и сделало их более модульными. Кроме того, были использованы **CSS-модули** для некоторых компонентов, что обеспечивало изоляцию стилей и предотвращало конфликты имен.

Определение цветовой схемы соответствало тематике интернет-магазина техники. Сочетание нейтральных и ярких цветов позволило выделить ключевые элементы интерфейса, такие как кнопки и ссылки. Шрифты были подобраны с учетом читаемости и современного стиля, что способствовало созданию приятного визуального восприятия.

Основные цвета.

Фон: Темный графитовый (#1E1E2E) или угольно-черный (#121212).

Основной текст: Светло-серый (#E0E0E0) для хорошей читаемости.

Акцентные элементы (кнопки, ссылки): Ярко-синий (#3A82F7) или электрический голубой (#4C91F7).

Дополнительные акценты: Лаймовый зеленый (#00FF7F) или оранжевый (#FF9500) для выделения ключевых элементов.

Границы и второстепенные элементы: Серый (#555555) или слегка приглушенный металлический (#808080).

Стилизация основных компонентов реализована, используя подход Styled Components.

Каждая карточка была оформлена с добавлением теней и закругленных углов, чтобы сделать их более привлекательными. Я добавил эффекты при наведении, подчеркивающие интерактивность элементов.

Созданы стильные кнопки с плавными переходами и эффектами при наведении. Это делало их более заметными и привлекательными для пользователей.

Особое внимание уделено стилизации формы, чтобы она была интуитивно понятной и удобной для пользователей. Поля ввода были оформлены с учетом современного дизайна, добавлены подсказки и сообщения об ошибках.

Одним из ключевых аспектов финальной верстки стало обеспечение адаптивности интерфейса. Я использовал медиа-запросы в CSS для изменения стилей в зависимости от размеров экрана, чтобы компоненты автоматически перестраивались, обеспечивая удобство использования на мобильных устройствах и планшетах.

Наложение стилей и улучшение пользовательского опыта способствовали созданию удобной платформы для покупателей.

**3.6 Финальные тесты**

На этапе финального тестирования были проверены все функциональные возможности сайта, чтобы убедиться в его стабильности и надежности перед запуском. Этот процесс включал как автоматизированные, так и ручные тесты, направленные на выявление возможных ошибок и недочетов.

Для автоматизации тестирования использована библиотека **Jest** вместе с **React Testing Library.** Это позволило создавать тесты для компонентов и функциональности приложения, проверяя их поведение в различных сценариях.

Я начал с написания тестов для основных компонентов, таких как карточки товаров, кнопки и формы. Тесты проверяли, правильно ли рендерятся компоненты, отображаются ли данные и срабатывают ли события при взаимодействии пользователя.

Также реализованы тесты для проверки логики, связанной с корзиной и оформлением заказа.

Помимо автоматизированных проедено ручное тестирование для проверки пользовательского интерфейса и общего пользовательского опыта.

Проверены основные функции, такие как навигация по страницам, фильтрация и сортировка товаров, добавление товаров в корзину и оформление заказа. Все действия выполняются без ошибок и интерфейс реагирует на взаимодействия пользователя.

Сайт протестирован на различных устройствах и разрешениях экранов, интерфейс корректно отображается как на мобильных устройствах, так и на настольных компьютерах.

Использованы инструменты разработчика браузера для тестирования производительности. Анализ времени загрузки страниц и отзывчивости интерфейса помог выявить узкие места в производительности.

Для получения дополнительной информации о работе сайта я организовал сессию тестирования стороннего пользователя. Он взаимодействовал с сайтом и дал комментарии. Это позволило выявить некоторые моменты, требующие улучшения.

На основе результатов был составлен список выявленных ошибок и недочетов. Я приоритизировал их и приступил к исправлению, тестируя каждое изменение, чтобы убедиться, что оно не вызвало новых проблем.

По итогу данный этап позволил убедиться в надежности и стабильности сайта, а также улучшить пользовательский опыт. Теперь, когда все тесты пройдены и недочеты устранены, интернет-магазин готов к согласованию с заказчиком.

**3.7 Сдача продукта**

Процесс сдачи включал в себя несколько ключевых шагов, направленных на уверенность заказчика в качестве и функциональности продукта.

Во время встречи была проведена демонстрация интернет-магазина. Подробно рассказано о каждом модуле, включая каталог товаров, корзину и процесс оформления заказа. Заказчик мог увидеть, как работает сайт, и оценить его интерфейс и пользовательский опыт.

После демонстрации на сессии для вопросов и ответов заказчик задавал вопросы о возможности дополнения модуля с конвертацией валюты и доработки дизайна.

Промежуточная сдача интернет-магазина техники заказчику прошла успешно. Задачи приняты к реализации.

# Заключение

В ходе выполнения дипломной работы была изучена теоретическая часть основ создания веб-сайтов на React и достигнута ключевая практическая задача по создания готового продукта.

Создание веб-сайтов на React обладает рядом преимуществ, таких как высокая производительность, возможность создания компонентов и простота в управлении состоянием сайта. Практическое исследование подтвердило, что с использованием React можно эффективно разрабатывать интерактивные интерфейсы, которые удовлетворяют требованиям пользователей.

В результате была разработана структура сайта, включающая основные модули, такие как страницы категорий товаров, корзина и система поиска, что значительно упростило процесс навигации для пользователей.

Проведенное исследование позволило выявить и реализовать основные этапы создания промежуточного и функционального веб-сайта, в условиях отсутствия на момент разработки товара для реализации, включая определение и выполнение требований заказчика, прототипирование и интеграцию с API. Результаты показали, что несмотря на отсутствие медиаконтента, возможно создать функциональный и привлекательный сайт, который можно доработать в будущем. Важным итогом стало успешное тестирование всех функций сайта, что подтвердило его работоспособность и соответствие требованиям заказчика.

Практическая значимость данной работы заключается в том, что она предоставляет эффективное решение для компаний, которым необходимо запустить онлайн-магазин в короткие сроки, даже при отсутствии готового контента. Рекомендации по дальнейшим действиям включают активное использование собранного функционала и последующее добавление медиаконтента по мере его готовности. Планы на дальнейшие исследования могут включать изучение методов оптимизации производительности приложения и анализ пользовательского опыта.

В процессе работы была достигнута цель проекта — разработка веб-сайта для заказчика, который отвечает современным требованиям и способен функционировать в условиях отсутствия медиаконтента. Все задачи, поставленные в начале работы, были успешно выполнены, что подтверждает актуальность выбранной темы. Гипотеза о возможности создания полноценного интернет-магазина без готового контента была доказана.

В качестве рекомендаций по совершенствованию объекта исследования можно выделить необходимость внедрения системы управления контентом (CMS) для более удобного добавления и редактирования товаров. Кроме того, стоит рассмотреть возможность интеграции инструментов аналитики для отслеживания поведения пользователей и улучшения функционала сайта на основе полученных данных. Также полезно будет исследовать возможности внедрения технологий адаптивного дизайна для оптимизации пользовательского опыта на мобильных устройствах.

Таким образом, данная работа не только продемонстрировала возможности разработки веб-приложений с использованием React, но и предложила практическое решение для бизнеса, стремящегося к быстрому выходу на рынок в условиях ограниченных ресурсов.

**Список используемой литературы**

1. Барабанов, А. В. Основы разработки веб-приложений. Москва: БХВ-Петербург, 2020. 256 с.

2. Гусев, И. А. React: создание пользовательских интерфейсов. Санкт-Петербург: Питер, 2019. 320 с.

3. Дьяков, С. В. Веб-разработка на JavaScript. Екатеринбург: Уральское издательство, 2021. 180 с.

4. Кузнецов, П. А. Программирование на React. Москва: Вильямс, 2022. 295 с.

5. Лисов, И. В. "Методы тестирования веб-приложений." Журнал информационных технологий 15, номер 2 (2021): 45-53.

6. Никифоров, Е. В. Основы проектирования веб-сайтов. Новосибирск: Сибирское университетское издательство, 2018. 210 с.

7. Петров, А. С. "Интеграция API в веб-приложения." Вестник программирования 10, номер 1 (2020): 12-18.

8. Тихонов, С. А. "Прототипирование веб-приложений." Информационные технологии и системы 7, номер 3 (2022): 99-105.

9. React. "Документация по React." Доступно по ссылке: https://reactjs.org/docs/getting-started.html. (Дата обращения: 15.03.2023).

10. MDN Web Docs. "HTML: Основы." Доступно по ссылке: https://developer.mozilla.org/ru/docs/Web/HTML. (Дата обращения: 10.03.2023).

11. W3Schools. "CSS Tutorial." Доступно по ссылке: https://www.w3schools.com/css/. (Дата обращения: 12.03.2023).

12. Stack Overflow. "Вопросы и ответы по веб-разработке." Доступно по ссылке: https://stackoverflow.com/. (Дата обращения: 05.03.2023).

# Приложение №1

«Пример брифа заказчика»

|  |  |
| --- | --- |
| Информация о бренде | |
| Название | Х |
| Краткое описание, особенности | Поставщик техники из Китая проверенного и надежного товара, аналога известным брендам, но по выгодной стоимости |
| Цели сайта | создать интернет-магазин для продажи техник |
| Целевая аудитория | |
| Возраст, пол | молодые люди и взрослые в возрасте от 18 до 45 лет |
| Интересы | интересующиеся технологиями, студенты, работающие профессионалы и семьи, которые ищут качественную технику по доступным ценам |
| Функции сайта | |
| Оставьте из предложенного приоритетные, остальные удалите | - Авторизация пользователя: возможность создания аккаунта для удобства покупок.  - Поисковые функции: мощный поиск по товарам с фильтрацией по категориям, характеристикам и ценам.  - Интернет-магазин: возможность добавления товаров в корзину и оформления заказов. |
| Контент | |
| Что необходимо разместить на сайте? | Категории товаров (компьютеры, планшеты, смартфоны, флешки, SSD), карточки товаров, характеристики товаров, описание, цены. |
| Создание, продвижение, поддержка | |
| Каков ваш бюджет на разработку сайт | Мы планируем выделить бюджет в размере 100,000 рублей на начальную разработку сайта |
| Каковы ваши сроки на разработку сайта | запустить сайт в течение 3 месяцев, чтобы успеть к началу продаж |
| Как вы планируете продвигать сай | планируем использовать SEO, контекстную рекламу и активное присутствие в социальных сетях для привлечения клиентов |
| Как вы планируете поддерживать и обновлять сайт после его запуска | Интересует поддержка |
| Референсы | |
| Ссылки на примеры сайтов, которые нравятся / сайты конкурентов | [www.techstore.com](http://www.techstore.com/) интуитивно понятный интерфейс.  [www.electroshop.ru](http://www.electroshop.ru/) — красивый дизайн  [www.gadgetworld.com](http://www.gadgetworld.com/) |
| Дополнительная информация | |
| Укажите то, что посчитаете нужным для выстраивания взаимопонимания | Техника на этапе логистики, есть информация о кол-ве и категориях, фото отсутствуют. Необходимо начать подготовку сайта сейчас, чтобы потом быстро подгрузить недостающую информацию и начать продвижение без пауз. |

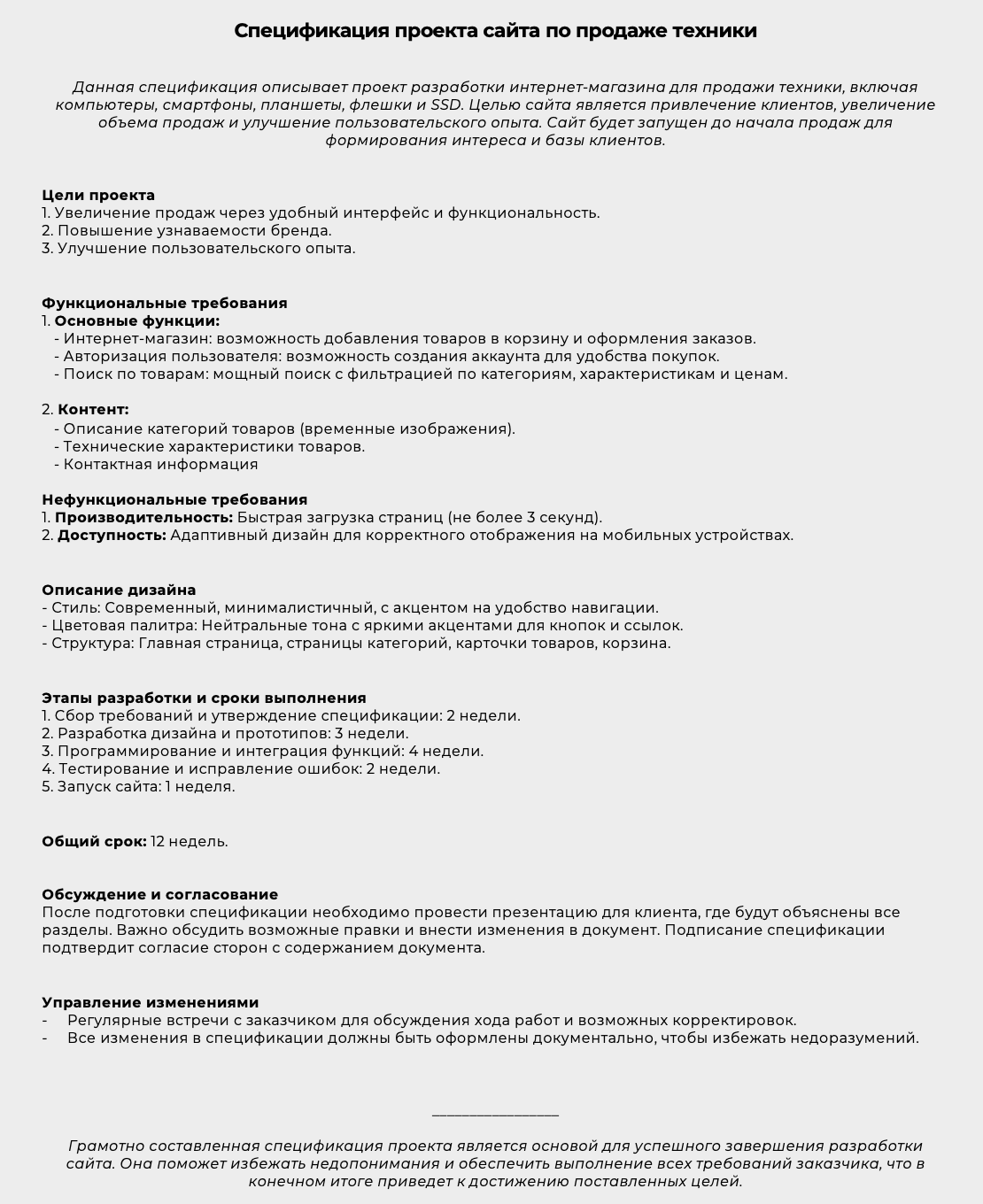
# Приложение №2

«Пример конкурентного анализа»

|  |  |  |  |
| --- | --- | --- | --- |
| Результаты конкурентного анализа | | | |
| Сайты и аспекты | **techstore.com** | **electroshop.ru** | **gadgetworld.com** |
| **Цели сайта** | Привлечение клиентов через удобный интерфейс | Увеличение продаж через простоту навигации | Создание сообщества вокруг технологий |
| **Целевая аудитория** | Технологически подкованные пользователи | Широкая аудитория, ищущая доступные цены | Любители гаджетов и новинок |
| **Дизайн** | Современный, минималистичный | Яркий, привлекательный | с акцентом на визуализацию |
| **Навигация** | Удобная, с четкими категориями | Простая, но требует улучшения | Интуитивно понятная, с фильтрами |
| **Функциональность** | Поиск по характеристикам, корзина | Быстрый процесс оформления заказа | Возможность оставлять отзывы и комментарии |
| **Контент** | Подробные описания товаров, блоги | Ограниченное количество статей | Много статей и новостей о продуктах |
| **SEO и продвижение** | Хорошая оптимизация, активное использование SEO | Средняя оптимизация, требует улучшений | Сильная SEO-стратегия, активно продвигается |
| **Скорость загрузки** | Быстрая загрузка страниц | Загрузка средняя, можно оптимизировать | Быстрая загрузка, хорошо оптимизирована |
| **Мобильная версия** | Адаптивный дизайн, удобен на мобильных устройствах | Мобильная версия требует доработки | Полностью адаптивный сайт |
| **Уникальные предложения** | Широкий ассортимент и качественный сервис | |Конкурентные цены и скидки | Акцент на отзывы и сообщество |
| Вывод | | | |
| Конкурентный анализ показал, что каждый из сайтов имеет свои сильные и слабые стороны. Сайт A выделяется современным дизайном и удобной навигацией, что делает его привлекательным для пользователей. Сайт B предлагает конкурентные цены, но требует улучшений в области навигации и SEO. Сайт C выделяется своим контентом и активным сообществом, однако также имеет свои недостатки. | | | |

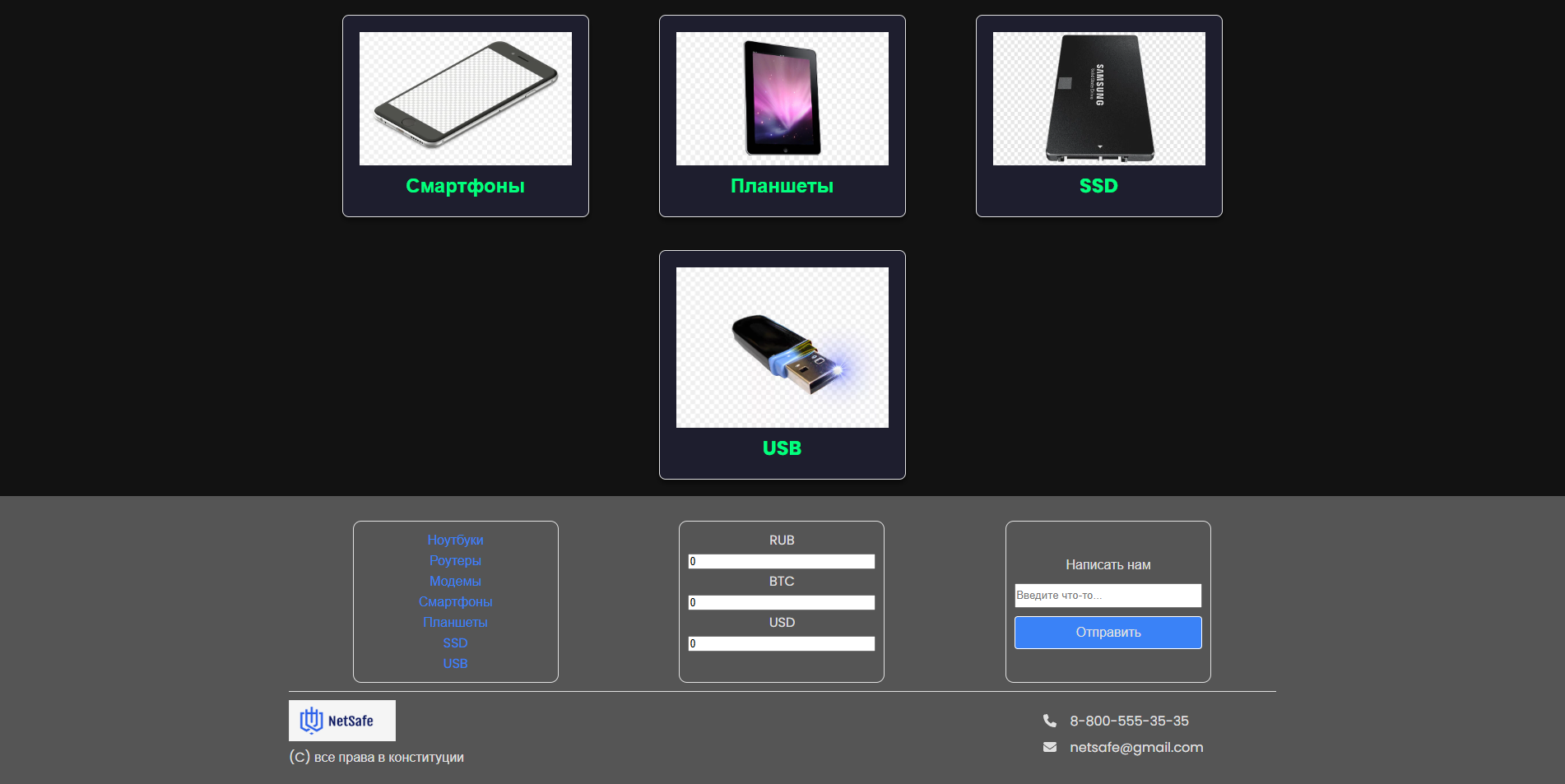
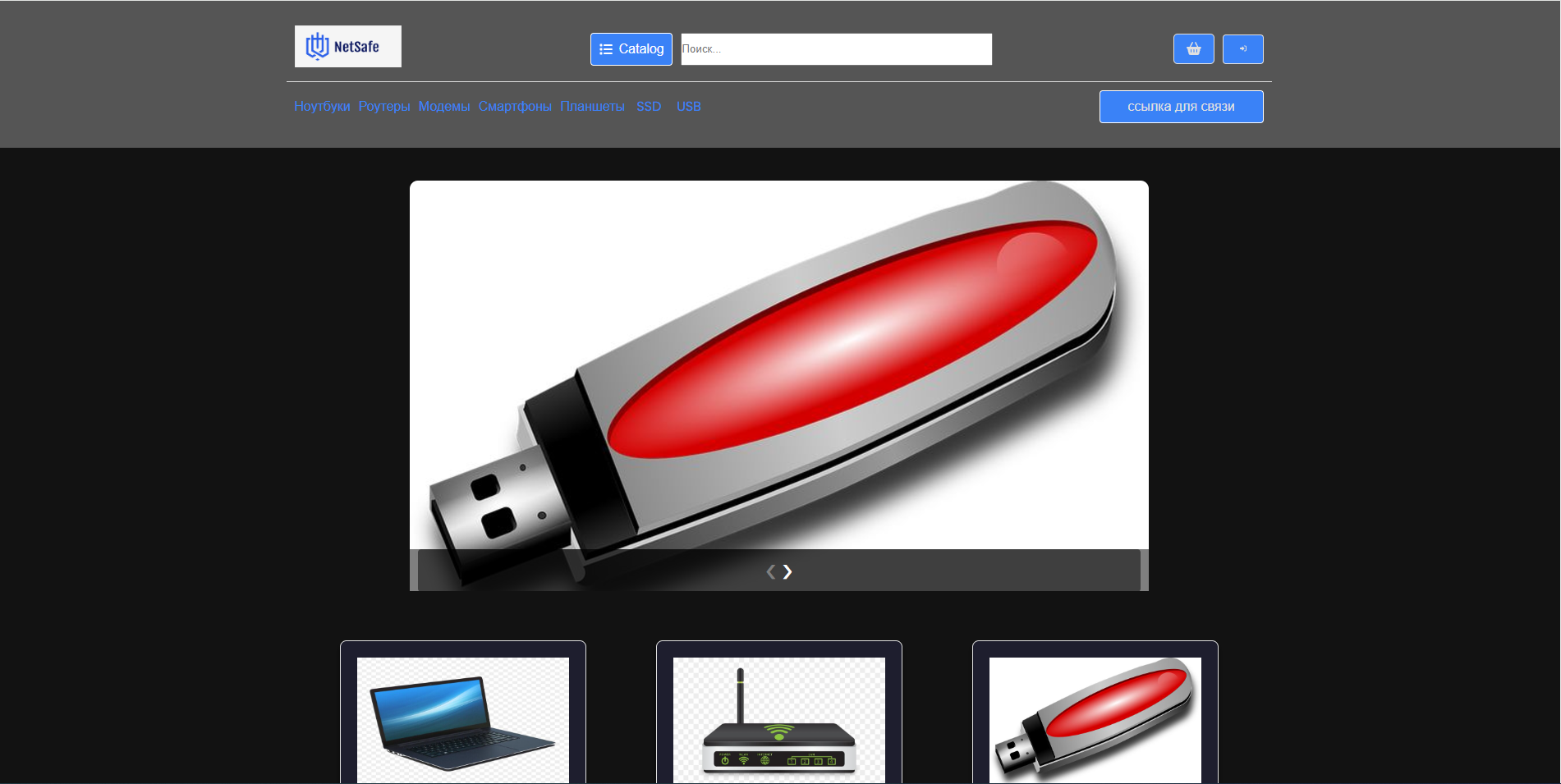
# Приложение №3

«Пример спецификации»



# Приложение №4

«Скриншоты функционала»



# 

# 

# Приложение №5

«Код»

**Add.jsx**

import { useContext } from "react";

import { BasketContext } from "../../context"; // Import the BasketContext

import { addToBasket } from "../util/basketLocalStorage";

import "./Add.style.css";

export function Add({ id }) {

// Получение контекста корзины через useContext

const basketValue = useContext(BasketContext);

// Проверка, находится ли товар с данным id в корзине

const isInBasket = basketValue.products.includes(id);

// Функция для обработки клика по кнопке "Добавить"

const handleAddClick = () => {

// Обновление состояния корзины, добавление нового товара

basketValue.addProduct([...basketValue.products, id]);

// Дополнительное действие для добавления товара в корзину (например, запись в локальное хранилище)

addToBasket(id);

};

// Рендер кнопки "Добавить", если товар еще не в корзине

return (

!isInBasket && (

<button

className="button\_\_add"

type="button"

onClick={handleAddClick} // При клике вызывается handleAddClick

>

Добавить

</button>

)

);

}

**Add.style.css**

.button\_\_add {

max-width: 100px;

padding: 5px;

background-color: blueviolet;

color: #e0e0e0;

}

.button\_\_add:hover {

background-color: #0056b3;

}

**Card.jsx**

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

import { useContext } from "react";

import { BasketContext } from "../../context";  // Контекст корзины

import { Add } from "../Add/Add";  // Импортируем кнопку Add

import { Remove } from "../Remove/Remove";  // Импортируем кнопку Remove

import { CurrencyContext } from "../../context";

import "./Card.style.css";

export function Card({ text, imageSrc, id, showRemoveButton, price, isGroup, url }) {

  // Получение данных корзины через контекст

  const basketValue = useContext(BasketContext);

  // Проверка, содержится ли товар с данным id в корзине

  const isInBasket = basketValue.products.includes(id);

  // Получение текущей валюты через контекст

  const currency = useContext(CurrencyContext);

  return (

    <div className="card">

      {/\* Ссылка на страницу товара \*/}

      <Link to={url}>

        {/\* Изображение товара, если оно передано \*/}

        {imageSrc && <img src={imageSrc} alt={text} />}

        {/\* Название товара \*/}

        <h2>{text}</h2>

        {/\* Отображение цены в рублях и биткоинах (если указана цена) \*/}

        {price ? <p>{price} ₽ / {(price \* currency.rub.btc).toFixed(6)} ฿ </p> : null}

      </Link>

      {/\* Если товар в корзине, показываем кнопку "Удалить", если:

          - передан showRemoveButton

          - товар не является частью группы (isGroup === false) \*/}

      {isInBasket && showRemoveButton && !isGroup ? <Remove id={id} /> : null}

      {/\* Если товар в корзине или он принадлежит группе, кнопка "Добавить" не отображается \*/}

      {isInBasket || isGroup ? null : <Add id={id} />}

    </div>

  );

}

**Card.style.css**

.card {

    max-width: 300px;

    display: inline-block;

    transition: transform 0.3s ease;

}

.card:hover{

    transform: scale(1.1);

}

**CatalogButton.jsx**

import { useState, useEffect } from "react";

import { faList } from "@fortawesome/free-solid-svg-icons";

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import "./CatalogButton.style.css";

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

import { products } from "../../data";

export const CatalogButton = () => {

  // Состояние для отслеживания, открыто ли модальное окно

  const [isModalOpen, setIsModalOpen] = useState(false);

  // Хук useEffect, добавляющий обработчик кликов по документу

  useEffect(() => {

    // При клике в любом месте документа закрывает модальное окно

    window.document.addEventListener('click', () => {

      setIsModalOpen(false);

    });

    // Удаление обработчика при размонтировании компонента

    return () => {

      window.document.removeEventListener('click', () => {

        setIsModalOpen(false);

      });

    };

  }, []);

  // Функция для переключения состояния модального окна

  const toggleModal = (e) => {

    // Останавливаем всплытие события, чтобы клик внутри компонента

    // не закрыл модальное окно

    e.stopPropagation();

    // Переключаем состояние (открыть/закрыть)

    setIsModalOpen((prev) => !prev);

  };

  return (

    <div className="header\_\_button\_\_catalog">

      {/\* Кнопка для открытия/закрытия модального окна \*/}

      <button className="catalog-button" onClick={toggleModal}>

        <FontAwesomeIcon icon={faList} />

        Catalog

      </button>

      {/\* Модальное окно, которое отображается только если isModalOpen === true \*/}

      {isModalOpen && (

        <div className="modal" onClick={toggleModal}>

          {/\* Контент модального окна \*/}

          <div className="modal-content">

            <h3>Список:</h3>

            <ul>

              {/\* Отображение списка товаров \*/}

              {products.map((product) => (

                <li key={product.id} className="item\_\_list">

                  <Link to={"/product/" + product.id}>{product.text}</Link>

                </li>

              ))}

            </ul>

          </div>

        </div>

      )}

    </div>

  );

};

**CatalogButton.style.jsx**

.catalog-button {

  max-width: 100px;

  padding: 10px 10px;

  background-color: #3a82f7;

  color: white;

  border-radius: 4px;

  cursor: pointer;

  font-size: 16px;

  display: flex;

  align-items: center;

  gap: 8px;

  transition: background-color 0.3s ease;

  border-color: 1px solid #e0e0e0;

}

.catalog-button:hover {

  background-color: #0056b3;

}

.modal {

  display: block;

  position: absolute;

  top: 80px;

  left: 40%;

  transform: translateX(-50%);

  background-color: rgba(0, 0, 0, 0.7);

  border-radius: 5px;

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

  width: 300px;

  z-index: 1;

}

.modal-content {

  padding: 20px;

  border-radius: 8px;

  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

  width: 300px;

  text-align: center;

}

.modal-content h3 {

  margin-bottom: 10px;

}

.modal-content ul {

  list-style-type: none;

  padding: 0;

  margin: 0;

}

.modal-content li {

  margin-bottom: 8px;

}

@keyframes fadeIn {

  from {

    opacity: 0;

    transform: scale(0.9);

  }

  to {

    opacity: 1;

    transform: scale(1);

  }

}

**Exchange.jsx**

import { useState, useRef, useEffect, useContext } from "react";

import { CurrencyContext } from "../../context";

import "../../page/Layout/Layout.style.css";

export function Exchange() {

  // Ссылки на элементы input для управления их значениями напрямую

  const rubRef = useRef(null);

  const btcRef = useRef(null);

  const usdRef = useRef(null);

  // Состояние для хранения значений RUB, BTC, USD

  const [rub, setRub] = useState(0);

  const [btc, setBtc] = useState(0);

  const [usd, setUsd] = useState(0);

  // Получение текущих курсов валют через контекст

  const currency = useContext(CurrencyContext);

  // Эффекты для синхронизации значений input с состоянием

  useEffect(() => {

    rubRef.current.value = rub.toString(); // Устанавливает значение поля RUB

  }, [rub]);

  useEffect(() => {

    btcRef.current.value = btc.toString(); // Устанавливает значение поля BTC

  }, [btc]);

  useEffect(() => {

    usdRef.current.value = usd.toString(); // Устанавливает значение поля USD

  }, [usd]);

  return (

    <div className="footer\_\_up\_\_rules">

      {/\* Блок для ввода RUB \*/}

      <div className="rub\_\_change">

        <label>RUB</label>

        <input

          type="text"

          id="rub"

          ref={rubRef} // Привязка к useRef

          onChange={(e) => {

            const num = Number(e.target.value); // Преобразование значения в число

            if (!isNaN(num)) {

              setRub(num); // Обновление состояния RUB

              setBtc(num \* currency.rub.btc); // Пересчет BTC

              setUsd(num \* currency.rub.usd); // Пересчет USD

            }

          }}

        />

      </div>

      {/\* Блок для ввода BTC \*/}

      <div className="btc\_\_change">

        <label>BTC</label>

        <input

          type="text"

          id="btc"

          ref={btcRef} // Привязка к useRef

          onChange={(e) => {

            const num = Number(e.target.value); // Преобразование значения в число

            if (!isNaN(num)) {

              setBtc(num); // Обновление состояния BTC

              setRub(num \* currency.btc.rub); // Пересчет RUB

              setUsd(num \* currency.btc.usd); // Пересчет USD

            }

          }}

        />

      </div>

      {/\* Блок для ввода USD \*/}

      <div className="usd\_\_change">

        <label>USD</label>

        <input

          type="text"

          id="usd"

          ref={usdRef} // Привязка к useRef

          onChange={(e) => {

            const num = Number(e.target.value); // Преобразование значения в число

            if (!isNaN(num)) {

              setUsd(num); // Обновление состояния USD

              setBtc(num \* currency.usd.btc); // Пересчет BTC

              setRub(num \* currency.usd.rub); // Пересчет RUB

            }

          }}

        />

      </div>

    </div>

  );

}

**GetCurrency.js**

export async function getCurrency() {

    // Выполнение HTTP-запроса к API для получения данных о курсах обмена

    const responce = await fetch("https://api.coingecko.com/api/v3/exchange\_rates");

    // Преобразование ответа в JSON

    const body = await responce.json();

    // Извлечение нужных курсов (RUB, USD, BTC) из ответа

    const { rub, usd, btc } = body.rates;

    // Возвращаем объект с пересчитанными курсами

    return {

      // Курс BTC относительно RUB и USD

      btc: {

        rub: rub.value, // 1 BTC → RUB

        usd: usd.value  // 1 BTC → USD

      },

      // Курс RUB относительно BTC и USD

      rub: {

        btc: 1 / rub.value,        // 1 RUB → BTC (обратное значение)

        usd: usd.value / rub.value // 1 RUB → USD (через соотношение USD/RUB)

      },

      // Курс USD относительно BTC и RUB

      usd: {

        btc: 1 / usd.value,        // 1 USD → BTC (обратное значение)

        rub: rub.value / usd.value // 1 USD → RUB (через соотношение RUB/USD)

      }

    };

  }

**Footer.jsx**

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

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import { faPhone, faEnvelope } from "@fortawesome/free-solid-svg-icons";

import { Exchange } from "../../component/Exchange";

import { products } from "../../data";

export const Footer = () => {

  return (

    <footer className="footer center">

      {/\* Верхняя часть футера \*/}

      <div className="footer\_\_up">

        {/\* Блок каталога товаров \*/}

        <div className="footer\_\_up\_\_catalog">

          <ul>

            {/\* Отображение списка товаров \*/}

            {products.map((product) => (

              <li key={product.id} className="item\_\_list">

                {/\* Ссылки на страницы товаров \*/}

                <Link to={"/product/" + product.id}>{product.text}</Link>

              </li>

            ))}

          </ul>

        </div>

        {/\* Блок обмена валют (компонент Exchange) \*/}

        <Exchange />

        {/\* Форма обратной связи \*/}

        <form className="footer\_\_up\_\_contact">

          <div className="footer\_\_up\_\_contact\_\_text">Написать нам</div>

          {/\* Поле для ввода сообщения \*/}

          <input

            className="footer\_\_input"

            type="text"

            id="myInput"

            name="myInput"

            placeholder="Введите что-то..."

          ></input>

          {/\* Кнопка отправки сообщения \*/}

          <button className="footer\_\_up\_\_button">Отправить</button>

        </form>

      </div>

      {/\* Нижняя часть футера \*/}

      <div className="footer\_\_down">

        {/\* Левая часть нижнего блока \*/}

        <div className="footer\_\_down\_\_left">

          {/\* Логотип \*/}

          <div className="logo">

            <img src='/img/logo.png' alt="logo" />

          </div>

          {/\* Информация о правах \*/}

          <div className="footer\_\_down\_\_rule">(C) все права в конституции</div>

        </div>

        {/\* Правая часть нижнего блока \*/}

        <div className="footer\_\_down\_\_right">

          {/\* Телефонный номер \*/}

          <div className="telephone">

            <FontAwesomeIcon className="pnone\_\_icon" icon={faPhone} />

            8-800-555-35-35

          </div>

          {/\* Электронная почта \*/}

          <div className="mail">

            <FontAwesomeIcon className="pnone\_\_icon" icon={faEnvelope} />

            netsafe@gmail.com

          </div>

        </div>

      </div>

    </footer>

  );

};

**Header.jsx**

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

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import { faBasketShopping } from "@fortawesome/free-solid-svg-icons";

import { CatalogButton } from "../CatalogButton/CatalogButton";

import { Search } from "../Search";

import { UserAuth } from "../UserAuth";

import { products } from "../../data";

export const Header = ({ basketState }) => {

  return (

    <header className="header center">

      {/\* Верхняя часть шапки сайта \*/}

      <div className="header\_\_up">

        <div className="header\_\_up\_\_list">

          {/\* Логотип с ссылкой на главную страницу \*/}

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

            <img src="/img/logo.png" alt="logo" />

          </Link>

          <div className="header\_\_block1">

            {/\* Кнопка для открытия каталога товаров \*/}

            <CatalogButton />

            {/\* Компонент поиска на сайте \*/}

            <Search />

          </div>

          <div className="header\_\_block2">

            {/\* Ссылка на корзину покупок \*/}

            <Link to={"/basket"} className="basket">

              {/\* Если в корзине есть товары, отображается их количество \*/}

              {basketState.length > 0 && (

                <span className="basket\_\_number">{basketState.length}</span>

              )}

              {/\* Иконка корзины \*/}

              <FontAwesomeIcon

                className="svg\_\_basket"

                icon={faBasketShopping}

              />

            </Link>

            {/\* Компонент для авторизации пользователя \*/}

            <div className="header\_\_login">

              <UserAuth />

            </div>

          </div>

        </div>

      </div>

      {/\* Нижняя часть шапки сайта \*/}

      <div className="header\_\_down">

        {/\* Список с категориями или разделами товаров \*/}

        <ul className="header\_\_down\_\_list">

          {products.map((product) => (

            <li key={product.id} className="item\_\_list">

              {/\* Ссылка на страницу конкретного товара или категории \*/}

              <Link to={"/product/" + product.id}>{product.text}</Link>

            </li>

          ))}

        </ul>

        {/\* Кнопка для отображения ссылки на связь \*/}

        <button className="header\_\_down\_\_link">

          <Link to="#">

            <p className="header\_\_button\_\_text">ссылка для связи</p>

          </Link>

        </button>

      </div>

    </header>

  );

};

**Remove.jsx**

import { useContext } from "react";

import { BasketContext } from "../../context";  // Импортируем контекст корзины

import { removeFromBasket } from "../util/basketLocalStorage";  // Функция для удаления из локального хранилища

import './Remove.style.css'

export function Remove({ id }) {

  const basketValue = useContext(BasketContext);  // Получаем контекст корзины

  // Проверяем, есть ли продукт в корзине

  const isInBasket = basketValue.products.includes(id);

  const handleRemoveClick = () => {

    // Функция для удаления товара из корзины

    const updatedProducts = basketValue.products.filter(productId => productId !== id);

    basketValue.addProduct(updatedProducts);  // Обновляем корзину в контексте

    removeFromBasket(id);  // Удаляем из локального хранилища

  };

  // Отображаем кнопку Remove только если продукт есть в корзине

  return (

    isInBasket && (

      <button

        className="button\_\_remove"

        type="button"

        onClick={handleRemoveClick}

      >

        Отменить

      </button>

    )

  );

}

**Remove.style.jsx**

.button\_\_remove{

    max-width: 100px;

    padding: 5px;

    background-color: red;

    color: white;

}

**Search.jsx**

import { useContext } from "react";

import { SearchContext } from "../../context";

import "./Search.style.css";

export function Search({}) {

  // Получение контекста поиска (объект с функцией управления поисковым состоянием)

  const search = useContext(SearchContext);

  return (

    <div className="header\_\_search">

      {/\* Поле ввода для поиска \*/}

      <input

        type="text"

        id="myInput"

        name="myInput"

        placeholder="Поиск..." // Подсказка в поле ввода

        onChange={(e) => {

          // Обновление значения поиска при изменении текста

          search.onChange(e.target.value);

        }}

      ></input>

    </div>

  );

}

**Slider.jsx**

import { useState, useEffect } from 'react';

import './Slider.style.css'

const images = [

  '/img/modem\_slider.jpg',

  '/img/router\_slider.png',

  '/img/smart\_slider.png'

];

export function Slider() {

  // Состояние для хранения текущего индекса изображения

  const [currentIndex, setCurrentIndex] = useState(0);

  // Функция для перехода к следующему слайду

  const prevSlide = () => {

    setCurrentIndex((prevIndex) => (prevIndex + 1) % images.length);

  };

  // Функция для перехода к предыдущему слайду

  const nextSlide = () => {

    setCurrentIndex((prevIndex) =>

      prevIndex === 0 ? images.length - 1 : prevIndex - 1

    );

  };

  // Автоматическая смена слайдов каждые 3 секунды

  useEffect(() => {

    const interval = setInterval(nextSlide, 3000); // Вызывает `nextSlide` каждые 3000 мс

    return () => clearInterval(interval); // Очищает интервал при размонтировании компонента

  }, []);

  return (

    <div className="slider">

      {/\* Отображение текущего изображения \*/}

      <img

        src={images[currentIndex]} // URL текущего изображения

        alt={`Slide ${currentIndex + 1}`} // Описание для текущего слайда

        className="slide"

      />

      {/\* Кнопка для перехода к следующему слайду \*/}

      <button className="slider\_\_button next" onClick={nextSlide}>❮</button>

      {/\* Кнопка для перехода к предыдущему слайду \*/}

      <button className="slider\_\_button prev" onClick={prevSlide}>❯</button>

    </div>

  );

}

**Slider.style.jsx**

.slider {

    position: relative;

    max-width: 900px;

    max-height: 500px;

    margin: 0 auto;

    overflow: hidden;

    margin-bottom: 40px;

    margin-top: 40px;

  }

  .slide {

    max-width: 100%;

    height: 100%;

    border-radius: 10px;

    transition: opacity 0.5s ease;

  }

  .slider\_\_button {

    position: absolute;

    top: 95%;

    transform: translateY(-50%);

    background-color: rgba(0, 0, 0, 0.5);

    color: #fff;

    border: none;

    padding: 10px;

    cursor: pointer;

    font-size: 24px;

  }

  .slider\_\_button.prev {

    left: 10px;

  }

  .slider\_\_button.next {

    right: 10px;

  }

**UserAuth.jsx**

import { UserContext } from "../../context";

import { useContext, useState, useRef } from "react";

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import { faRightToBracket } from "@fortawesome/free-solid-svg-icons";

import { setUser } from "../util/userLocalStorage";

export function UserAuth() {

  // Получение контекста пользователя (данные о текущем пользователе и функции управления)

  const userContext = useContext(UserContext);

  const [isModelShow, setIsModelShow] = useState(false); // Состояние отображения модального окна

  const loginRef = useRef(); // Референс для поля ввода логина

  const passRef = useRef(); // Референс для поля ввода пароля

  // Если пользователь авторизован, показываем кнопку выхода

  if (userContext.user) {

    return (

      <button

        className="action\_\_btn"

        onClick={() => {

          userContext.setUser(null); // Очистка данных пользователя

          setUser(null); // Обновление состояния (опционально, если используется глобальный стейт)

        }}

      >

        {userContext.user} {/\* Отображение имени пользователя \*/}

      </button>

    );

  }

  // Если пользователь не авторизован, показываем кнопку входа

  return (

    <div>

      <button

        className="action\_\_btn"

        onClick={() => {

          setIsModelShow(!isModelShow); // Переключение отображения модального окна

        }}

      >

        <FontAwesomeIcon icon={faRightToBracket} /> {/\* Иконка входа \*/}

      </button>

      {/\* Модальное окно для ввода логина и пароля \*/}

      {isModelShow && (

        <div className="login">

          <div className="login\_\_input">

            <p>Введите Логин</p>

            <div className="input\_\_folder">

              {/\* Поле ввода логина \*/}

              <input

                className="input\_\_place"

                type="text"

                placeholder="Логин..."

                ref={loginRef}

              ></input>

            </div>

            <p>Введите Пароль</p>

            <div className="input\_\_folder">

              {/\* Поле ввода пароля \*/}

              <input

                className="input\_\_place"

                type="text"

                placeholder="Пароль..."

                ref={passRef}

              ></input>

            </div>

            <div className="button\_\_input">

              {/\* Кнопка "Войти" \*/}

              <button

                className="button\_\_login"

                onClick={() => {

                  const login = loginRef.current.value; // Получение значения логина

                  // const pass = passRef.current.value // Пароль, если понадобится

                  userContext.setUser(login); // Установка имени пользователя в контексте

                  setUser(login); // Локальное обновление (опционально)

                }}

              >

                Войти

              </button>

              {/\* Кнопка "Отмена" \*/}

              <button

                className="button\_\_cancel"

                onClick={() => {

                  setIsModelShow(false); // Скрыть модальное окно

                }}

              >

                Отмена

              </button>

            </div>

          </div>

        </div>

      )}

    </div>

  );

}

**basketLocalSTorage.jsx**

// Ключ для хранения корзины в localStorage

const BASKET\_KEY = 'STORE\_BASKET'

// Получаем список товаров из localStorage

const goodList = localStorage.getItem(BASKET\_KEY)

// Если корзина пуста (товаров нет), создаём пустой массив в localStorage

if (!goodList) {

    localStorage.setItem(BASKET\_KEY, JSON.stringify([]))

}

/\*\*

 \* Функция для добавления товара в корзину.

 \*/

export function addToBasket(good) {

    // Получаем текущие товары в корзине

    const goods = getFromBasket()

    // Добавляем новый товар в корзину

    goods.push(good)

    // Записываем обновленную корзину в localStorage

    // В данной версии функция setToBasket используется для записи в localStorage

    setToBasket(goods)

}

/\*\*

 \* Функция для получения товаров из корзины.

 \*/

export function getFromBasket() {

    // Получаем строку JSON из localStorage и преобразуем её в массив

    const goodList = localStorage.getItem(BASKET\_KEY)

    return JSON.parse(goodList) // Преобразуем строку JSON в массив

}

/\*\*

 \* Функция для удаления товара из корзины.

 \*/

export function removeFromBasket(id) {

   // Получаем текущие товары из корзины

   const goods = getFromBasket()

   // Фильтруем корзину, удаляя товар с данным ID

   const updatedBasket = goods.filter(productId => productId !== id);

   // Записываем обновленную корзину в localStorage

   setToBasket(updatedBasket)

}

/\*\*

 \* Функция для записи обновленного списка товаров в корзине в localStorage.

 \*/

export function setToBasket(goods) {

    // Сохраняем обновленный список товаров в localStorage

    localStorage.setItem(BASKET\_KEY, JSON.stringify(goods))

}

**userLocalStorage.js**

// Ключ для хранения данных пользователя в localStorage

const USER\_KEY = 'STORE\_USER'

/\*\*

 \* Функция для получения данных о пользователе из localStorage.

 \* Если данных нет, возвращается null.

 \* Имя пользователя, если оно есть в localStorage, иначе null.

 \*/

export function getUser(){

    // Получаем значение из localStorage по ключу USER\_KEY. Если оно отсутствует, возвращаем null.

    return localStorage.getItem(USER\_KEY) || null;

}

/\*\*

 \* Функция для сохранения или удаления данных о пользователе в localStorage.

 \* Если передан аргумент userArg, данные о пользователе сохраняются.

 \* Если аргумент равен null или undefined, данные удаляются из localStorage.

 \* userArg - Имя пользователя, которое сохраняется в localStorage.

 \*/

export function setUser(userArg){

    if(userArg){

        // Если userArg задан, сохраняем данные о пользователе в localStorage

        localStorage.setItem(USER\_KEY, userArg)

    }else{

        // Если userArg равен null или undefined, удаляем данные о пользователе из localStorage

        localStorage.removeItem(USER\_KEY)

    }

}

**Basket.jsx**

import { items } from "../../data";

import { BasketContext } from "../../context.js";

import { useContext, useState } from "react";

import { Card } from "../../component/index.js";

import { CurrencyContext } from "../../context.js";

import { removeFromBasket } from "../../component/util/basketLocalStorage.js";

export function Basket() {

  // Извлекаем данные о корзине и валюте из контекста

  const basket = useContext(BasketContext);

  const currency = useContext(CurrencyContext);

  // Суммируем стоимость всех товаров в корзине

  const sum = basket.products.reduce((sum, id) => {

    const item = items[id];  // Ищем товар по его id

    return sum + item.price;  // Складываем цену товара к общей сумме

  }, 0);

  // Обработчик для покупки товаров в корзине

  const buyHandler = () => {

    // Для каждого товара в корзине вызываем функцию удаления товара из корзины

    basket.products.forEach(productId => removeFromBasket(productId));

    // Очищаем корзину в контексте

    basket.addProduct([]);

    // Обновляем состояние, что покупка была оформлена

    setItemBuyed(true);

  };

  // Состояние для отображения сообщения о том, что покупка была оформлена

  const [itemBuyed, setItemBuyed] = useState(false);

  return (

    <div className="basket\_\_flex">

      <h2>Корзина</h2>

      {/\* Карточки товаров в корзине \*/}

      <div className="basket\_\_card">

        {basket.products.map((id) => {

          return (

            <Card

              key={id}  // Уникальный ключ для каждого элемента списка

              id={id}  // Передаем ID товара в компонент Card

              text={items[id].text}  // Название товара

              imageSrc={items[id].imageSrc}  // Изображение товара

              price={items[id].price}  // Цена товара

              showRemoveButton={true}  // Показываем кнопку удаления

              url={"/item/" + id}  // Ссылка на страницу товара

            />

          );

        })}

      </div>

      {/\* Если покупка была оформлена, показываем сообщение \*/}

      {itemBuyed ? <h2>Покупка оформлена</h2>: null }

      {/\* Если корзина не пуста, показываем блок с итоговой стоимостью и кнопкой оплаты \*/}

      {basket.products.length > 0 ? (

        <div className="endPrice">

          {/\* Выводим итоговую цену в рублях и биткойнах \*/}

          Итого: {sum} ₽ / {(sum \* currency.rub.btc).toFixed(6)} ฿

          {/\* Кнопка для оплаты \*/}

          <button className="endPrice\_\_button" onClick={buyHandler}>Оплатить</button>

        </div>

      ) : null}

    </div>

  );

}

**ItemPage.jsx**

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

import { items } from "../../data";

export function ItemPage() {

  // Получаем параметр itemId из URL, используя useParams.

  const { itemId } = useParams();

  // Извлекаем товар из массива или объекта items по id.

  const item = items[itemId];

  // Рендерим страницу товара, используя полученные данные.

  return (

    <div className="items">

      <div className="">{item.description}</div> {/\* Описание товара \*/}

      <div className="">{item.text}</div> {/\* Название товара \*/}

      <div className="">{item.price} руб.</div> {/\* Цена товара в рублях \*/}

    </div>

  );

}

**Layout.jsx**

import { useState } from "react";

import { BasketContext, SearchContext, UserContext } from "../../context";

import { getFromBasket } from "../../component/util/basketLocalStorage";

import { getUser } from "../../component/util/userLocalStorage";

import { Header } from "../../component/Header/Header";

import { Footer } from "../../component/Footer/Footer";

import "./Layout.style.css";

// Компонент Layout, который оборачивает основную часть приложения,

// предоставляя контексты для корзины, поиска и пользователя

export const Layout = ({ children }) => {

  // Инициализация состояния корзины, поискового запроса и состояния пользователя

  // Данные корзины и пользователя берутся из локального хранилища (или дефолтных значений)

  const [basketState, setBasketState] = useState(getFromBasket());  // Состояние корзины

  const [searchState, setSearchState] = useState("");  // Состояние поискового запроса

  const [userState, setUserState] = useState(getUser());  // Состояние пользователя

  return (

    // Провайдер контекста для пользователя

    <UserContext.Provider value={{ user: userState, setUser: setUserState }}>

      {/\* Провайдер контекста для корзины \*/}

      <BasketContext.Provider

        value={{ products: basketState, addProduct: setBasketState }}

      >

        {/\* Провайдер контекста для поискового запроса \*/}

        <SearchContext.Provider

          value={{ search: searchState, onChange: setSearchState }}

        >

          <div>

            {/\* Заголовок приложения, отображающий корзину или другую информацию \*/}

            <Header basketState={basketState} />

            {/\* Основной контент страницы, дочерние компоненты будут отображаться здесь \*/}

            <main className="main center">{children}</main>

            {/\* Нижняя часть страницы с информацией о компании и контактах \*/}

            <Footer />

          </div>

        </SearchContext.Provider>

      </BasketContext.Provider>

    </UserContext.Provider>

  );

};

**Layout.style.css**

\* {

  width: 100%;

  margin: 0;

  padding: 0;

  box-sizing: border-box;

}

:root {

  --background-color: #121212; /\* Основной фон \*/

  --text-color: #e0e0e0; /\* Цвет текста \*/

  --accent-color: #3a82f7; /\* Основной акцент \*/

  --accent-hover: #4c91f7; /\* При наведении \*/

  --secondary-color: #555555; /\* Границы и второстепенные элементы \*/

  --highlight-color: #00ff7f; /\* Яркие выделения \*/

}

header {

  background-color: var(--secondary-color);

  padding: 20px;

}

footer {

  background-color: var(--secondary-color);

  color: var(--text-color);

  padding: 20px;

  text-align: center;

}

.card {

  background-color: #1e1e2e; /\* Графитовый цвет \*/

  color: var(--text-color);

  border: 1px solid var(--text-color);

  border-radius: 8px;

  padding: 20px;

  margin: 20px;

  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.5);

}

.card h2 {

  color: var(--highlight-color);

}

.card button {

  margin-top: 10px;

}

button {

  background-color: var(--accent-color);

  color: var(--text-color);

  border: none;

  border-radius: 4px;

  padding: 10px 20px;

  cursor: pointer;

  font-size: 16px;

  border: 1px solid white;

}

a {

  color: var(--accent-color);

  text-decoration: none;

  transition: color 0.3s ease;

}

body {

  background-color: var(--background-color);

  color: var(--text-color);

  /\* font-family: Arial, sans-serif; \*/

  margin: 0;

  padding: 0;

  font-family: "Poppins", sans-serif;

}

.center {

  width: 100%;

  padding-left: calc(50% - 600px);

  padding-right: calc(50% - 600px);

}

.header {

  /\* max-width: 1140px; \*/

  /\* display: flex; \*/

  justify-content: space-around;

  /\* height: 100px; \*/

  align-items: center;

  /\* border: 1px solid; \*/

  flex-direction: column;

  display: inline-block;

  /\* background-color: aqua; \*/

}

.header\_\_up {

  display: flex;

  width: 100%;

  border-bottom: 1px solid;

  align-items: center;

  padding: 10px;

  flex-wrap: wrap;

}

.logo {

  max-width: 130px;

  flex: 1 1 calc(50% - 10px);

}

.header\_\_block1 {

  display: flex;

  align-items: center;

  justify-content: center;

  max-width: 500px;

  flex: 1 1 100%;

  /\* order: 1; \*/

}

.header\_\_block2 {

  display: flex;

  align-items: center;

  max-width: 110px;

  gap: 10px;

  flex: 1 1 calc(50% - 10px);

}

.header\_\_up\_\_list {

  display: flex;

  align-items: center;

  justify-content: space-between;

  flex-wrap: wrap;

  gap: 10px;

}

.catalog\_\_list {

  max-width: 300px;

}

.header\_\_down {

  display: flex;

  align-items: center;

  padding: 10px;

  justify-content: space-between;

  flex-wrap: wrap;

  gap: 10px;

}

.header\_\_down\_\_list {

  display: flex;

  max-width: 500px;

  gap: 10px;

  /\* flex-wrap: wrap; \*/

  text-align: center;

}

.header\_\_down\_\_link {

  max-width: 200px;

  color: white;

  &:hover {

    background-color: #0056b3;

  }

}

.header\_\_search {

  display: flex;

  max-width: 400px;

  height: 40px;

}

ul {

  list-style: none;

}

.loop {

  padding-top: 8px;

  padding-bottom: 8px;

  max-width: 50px;

  border: 1px solid;

}

.basket {

  padding-top: 5px;

  padding-bottom: 5px;

  max-width: 50px;

  border: 1px solid #e0e0e0;

  position: relative;

  background-color: #3a82f7;

  color: #e0e0e0;

  border-radius: 5px;

  &:hover {

    background-color: #0056b3;

  }

}

.basket\_\_flex {

  display: flex;

  flex-wrap: wrap;

  text-align: center;

}

.endPrice {

  display: flex;

  align-items: center;

  justify-content: center;

  gap: 10px;

  padding-bottom: 20px;

}

.endPrice\_\_button {

  max-width: 115px;

  &:hover {

    background-color: #0056b3;

  }

}

.basket\_\_number {

  display: flex;

  justify-content: space-evenly;

  color: rgb(175, 13, 13);

  position: absolute;

  padding-left: 25px;

}

.header\_\_login {

  max-width: 50px;

  position: relative;

}

.login {

  display: block;

  position: absolute;

  top: 40px;

  left: 50%;

  transform: translateX(-50%);

  background-color: rgba(0, 0, 0, 0.7);

  /\* color: white; \*/

  padding: 15px;

  border-radius: 5px;

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

  width: 300px;

  /\* text-align: center; \*/

  /\* opacity: 0; \*/

  transition: opacity 0.3s ease-in-out;

  z-index: 1;

}

.action\_\_btn {

  padding-top: 8px;

  padding-bottom: 8px;

  max-width: 50px;

  border: 1px solid;

  &:hover {

    background-color: #0056b3;

  }

}

.product\_\_items {

  display: flex;

  text-align: center;

  flex-wrap: wrap;

  flex-direction: column;

}

.input\_\_place {

  max-width: 190px;

  height: 30px;

  margin-bottom: 20px;

  margin-top: 5px;

}

.input\_\_folder {

  display: flex;

  align-items: baseline;

}

.button\_\_input {

  display: flex;

  max-width: 100px;

  gap: 5px;

}

.button\_\_login {

  border: 1px solid;

}

.button\_\_cancel {

  border: 1px solid;

}

.main {

  display: flex;

  flex-direction: column;

}

.header\_\_button\_\_catalog {

  max-width: 120px;

  height: 40px;

  display: flex;

  align-items: center;

  padding: 10px;

}

.header\_\_button\_\_text {

  color: #e0e0e0;

}

.item\_\_list {

  transition: transform 0.3s ease; /\* Плавный переход \*/

}

.item\_\_list:hover {

  transform: scale(1.1);

}

.footer {

  /\* max-width: 1140px; \*/

  /\* display: flex; \*/

  justify-content: space-around;

  /\* height: 100px; \*/

  align-items: center;

  /\* border: 1px solid; \*/

  flex-direction: column;

  /\* background-color: aqua; \*/

}

.footer\_\_up {

  display: flex;

  border-bottom: 1px solid;

  padding: 10px;

  justify-content: space-around;

  flex-wrap: wrap;

  gap: 10px;

}

.footer\_\_up\_\_button {

  transition: background-color 0.3s ease;

}

.footer\_\_up\_\_button:hover {

  background-color: #0056b3;

  /\* transform: scale(1.1); \*/

}

.footer\_\_up\_\_catalog {

  border: 1px solid;

  border-radius: 10px;

  padding: 10px;

  max-width: 250px;

  /\* background-color: white; \*/

}

.footer\_\_up\_\_rules {

  border: 1px solid;

  border-radius: 10px;

  padding: 10px;

  max-width: 250px;

  /\* background-color: white; \*/

}

.footer\_\_up\_\_contact {

  border: 1px solid;

  border-radius: 10px;

  padding: 10px;

  max-width: 250px;

  /\* background-color: white; \*/

  display: flex;

  flex-direction: column;

  justify-content: center;

  gap: 10px;

}

.footer\_\_input {

  height: 30px;

}

.footer\_\_down {

  display: flex;

  align-items: center;

  justify-content: space-between;

  margin-top: 10px;

}

.footer\_\_down\_\_left {

  display: flex;

  flex-direction: column;

  max-width: 250px;

}

.footer\_\_down\_\_rule {

  width: 214px;

}

.footer\_\_down\_\_right {

  display: flex;

  flex-wrap: wrap;

  max-width: 300px;

}

.telephone {

  display: flex;

  align-items: center;

}

.pnone\_\_icon {

  padding-top: 8px;

  padding-bottom: 8px;

  max-width: 50px;

}

.mail {

  display: flex;

  align-items: center;

}

@media (min-width: 768px) and (max-width: 1024px) {

  .center {

    padding-left: 16px;

    padding-right: 16px;

  }

  .logo {

    flex: 1 1 calc(50% - 10px);

    text-align: center;

  }

  .header\_\_block2 {

    flex: 1 1 calc(50% - 10px);

    text-align: center;

  }

  .header\_\_block1 {

    flex: 1 1 100%;

    text-align: center;

  }

}

@media (min-width: 320px) and (max-width: 767.99px) {

  .center {

    padding-left: 8px;

    padding-right: 8px;

  }

  .header\_\_down {

    display: flex;

    justify-content: center;

  }

  .header\_\_down\_\_list {

    display: flex;

    flex-wrap: wrap;

  }

  .footer\_\_down {

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

  }

  .footer\_\_down\_\_left {

    display: flex;

    flex-direction: column;

    align-items: center;

  }

  .logo {

    flex: 1 1 calc(50% - 10px);

    text-align: center;

  }

  .header\_\_block2 {

    flex: 1 1 calc(50% - 10px);

    text-align: center;

  }

  .header\_\_block1 {

    flex: 1 1 100%;

    text-align: center;

    order: 1;

  }

}

**ProductPage.jsx**

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

import { products } from "../../data";

import { Card } from "../../component";

// Компонент ProductPage отображает подробности о продукте, включая его дочерние элементы (товары или варианты)

export function ProductPage() {

  // Извлекаем параметр productId из URL, используя хук useParams

  const { productId } = useParams();

  // Ищем продукт в массиве products, который соответствует переданному productId

  const product = products.find((product) => {

    return product.id === productId;  // Сравниваем id продукта с переданным в URL

  });

  return (

    <div className="product\_\_items">

      {/\* Если продукт найден, отображаем его описание, иначе выводим сообщение "Не найдено" \*/}

      {product ? product.text : "Не найдено"}

      {/\* Если продукт найден, отображаем его дочерние элементы, например, товары \*/}

      <div>

        {product && product.items.map(item => {

          // Для каждого дочернего товара отображаем карточку с деталями

          return (

            <Card

              key={item.id}  // Уникальный ключ для каждого элемента

              text={item.text}  // Название товара

              imageSrc={item.imageSrc}  // Источник изображения товара

              id={item.id}  // ID товара

              showRemoveButton={false}  // Не показывать кнопку удаления для товаров на этой странице

              price={item.price}  // Цена товара

              url={"/item/" + item.id}  // URL для страницы товара

            />

          );

        })}

      </div>

    </div>

  );

}

**RootPage.jsx**

import { Card } from "../../component";

import { products } from "../../data";

import { Slider } from "../../component/Slider/Slider";

import { useContext } from "react";

import { SearchContext } from "../../context";

import "../RootPage/RootPage.style.css";

// Компонент RootPage служит главной страницей приложения и отображает все карточки продуктов с учетом фильтра по поиску

export function RootPage() {

  // Получаем контекст поиска с помощью хука `useContext` для `SearchContext`

  const search = useContext(SearchContext);

  return (

    <div>

      {/\* Рендерим слайдер \*/}

      <Slider />

      <div className="card\_\_list">

        {/\* Фильтруем массив продуктов в зависимости от текущего значения поиска \*/}

        {products

          .filter((product) => {

            if (search.search === "") {

              // Если строка поиска пустая, отображаем все продукты

              return true;

            }

            // Иначе, фильтруем продукты, проверяя, включает ли их текст (в нижнем регистре) строку поиска

            return product.text.toLowerCase().includes(search.search.toLowerCase());

          })

          .map((product) => {

            // Для каждого отфильтрованного продукта создаем карточку

            return (

              <Card

                key={product.id}  // Уникальный ключ для каждого продукта

                id={product.id}  // ID продукта

                text={product.text}  // Название продукта

                imageSrc={product.imageSrc}  // Источник изображения продукта

                isGroup={true}  // Флаг, указывающий, что это группа продуктов (карточка для группы)

                url={"/product/" + product.id}  // Ссылка на страницу продукта

              />

            );

          })}

      </div>

    </div>

  );

}

**RootPage.style.css**

.card\_\_list{

    display: flex;

    flex-wrap: wrap;

    align-items: flex-end;

    justify-content: space-evenly;

    text-align: center;

}

**App.jsx**

import { BrowserRouter, Routes, Route } from "react-router-dom";

import { Layout } from "./page/Layout";

import { RootPage } from "./page/RootPage";

import { ProductPage } from "./page/ProductPage";

import { Basket } from "./page/Basket/Basket";

import { ItemPage } from "./page/ItemPage";

import { CurrencyContext } from "./context";

// Основной компонент приложения

export function App({ currency }) {

  return (

    // Оборачиваем все приложение в CurrencyContext.Provider,

    // чтобы предоставлять информацию о валюте в дочерних компонентах

    <CurrencyContext.Provider value={currency}>

      {/\* Используем BrowserRouter для маршрутизации по страницам \*/}

      <BrowserRouter>

        {/\* Layout - общий контейнер для всего приложения, включает Header, Footer и основной контент \*/}

        <Layout>

          {/\* Настройка маршрутов для различных страниц \*/}

          <Routes>

            {/\* Главная страница приложения \*/}

            <Route path="/" element={<RootPage />} />

            {/\* Страница продукта, идентифицируемая по параметру productId \*/}

            <Route path="product/:productId" element={<ProductPage />} />

            {/\* Страница корзины \*/}

            <Route path="basket" element={<Basket />} />

            {/\* Страница товара, идентифицируемая по параметру itemId \*/}

            <Route path="item/:itemId" element={<ItemPage />} />

            {/\* Страница для несуществующих маршрутов (ошибка 404) \*/}

            <Route path="\*" element={<h2>404</h2>} />

          </Routes>

        </Layout>

      </BrowserRouter>

    </CurrencyContext.Provider>

  );

}

**context.js**

import { createContext } from 'react';

// Создаем контекст для корзины. По умолчанию корзина пустая (пустой массив продуктов).

// Этот контекст будет использоваться для управления и доступа к данным корзины (например, список товаров).

export const BasketContext = createContext({ products: [] });

// Создаем контекст для поиска. По умолчанию строка поиска пустая.

// Этот контекст будет использоваться для хранения и обновления значения поискового запроса.

export const SearchContext = createContext({

  search: "", // Значение по умолчанию для поисковой строки

});

// Создаем контекст для пользователя. По умолчанию пользователь отсутствует (null).

// Этот контекст будет использоваться для управления состоянием пользователя (например, информация о том, вошел ли пользователь).

export const UserContext = createContext({

  user: null, // Значение по умолчанию - отсутствующий пользователь

});

// Создаем контекст для валюты. По умолчанию значение равно null.

// Этот контекст будет использоваться для передачи данных о текущей валюте (например, обменных курсах).

export const CurrencyContext = createContext(null);

**data.js**

export const products = [

  {

    id: "1",

    text: "Ноутбуки",

    imageSrc: "/img/notebook.jpg",

    items: [

      {

        id: "1",

        imageSrc: "/img/lenova.jpg",

        text: "Lenovo",

        price: 55000,

        description:

          "Ноутбук Lenovo Ideapad 1 - мощный и надежный помощник в работе и развлечениях. Оснащен процессором AMD Athlon Silver-7120U с частотой 2.4 ГГц и 2 ядрами, что обеспечивает высокую производительность и быстродействие.",

      },

      {

        id: "2",

        imageSrc: "/img/hp.jpg",

        text: "HP",

        price: 35000,

        description:

          "Ноутбук HP Laptop 15 - это современное и функциональное устройство, которое станет незаменимым помощником в работе и развлечениях.Ноутбук оснащен процессором Intel Celeron N4120 с частотой 1.1 ГГц, что обеспечивает быструю и стабильную работу. В ноутбуке установлено 8 ГБ оперативной памяти DDR4, что позволяет одновременно запускать несколько приложений и работать с большими объемами данных.",

      },

      {

        id: "3",

        imageSrc: "/img/asus.jpg",

        text: "ASUS",

        price: 75000,

        description:

          "Ноутбук ASUS VivoBook 17X M3704YA-AU052 – оптимальное решение для выполнения большого объема офисных и домашних задач. Энергоемкий литий-ионный аккумулятор позволяет использовать его на протяжении долгого времени без подзарядки.ASUS VivoBook 17X оснащен 8-ядерным процессором AMD Ryzen 7 7730U частотой 2 ГГц, что позволяет выполнять самые сложные задачи. Большой объем оперативной памяти дает возможность запускать одновременно несколько приложений и быстро переключаться между ними.",

      },

    ],

  },

  {

    id: "2",

    text: "Роутеры",

    imageSrc: "/img/router.jpg",

    items: [

      {

        id: "4",

        imageSrc: "/img/microtic.jpg",

        text: "Microtic",

        price: 15000,

        description:

          "Общие характеристики Тип устройства маршрутизатор Возможность установки в стойку есть Коммутатор/концентратор Общее количество портов 10 Базовая скорость передачи данных 1 Гбит/с Количество uplink/стек/SFP-портов и модулей 1 Максимальная скорость uplink/SFP-портов 1 Гбит/с Маршрутизатор DHCP-сервер есть Межсетевой экран (Firewall) есть Память Объем оперативной памяти 1024 МБ Объем флэш-памяти 128 МБ Поддержка стандартов Сетевые стандарты автоопределение MDI/MDIX Мониторинг и конфигурирование Консольный порт есть Web-интерфейс есть Дополнительные характеристики USB-порт есть Ширина 443 мм Высота 44 мм Глубина 92 мм Вес 0.95 кг Дополнительная информация cенсорный touchscreen LCD экран",

      },

      {

        id: "5",

        imageSrc: "/img/tp.jpg",

        text: "Tp-Link",

        price: 10000,

        description:

          "802.11ac Wave 2 Wi-Fi — до 1300 Мбит/с на диапазоне 5 ГГц и до 600 Мбит/с на 2,4 ГГц. Технология MIMO 3×3 — отправка и получение данных в три потока для идеальной работы с устройствами 3×3. Улучшенное покрытие Wi-Fi – благодаря технологии Beamforming. MU-MIMO — оптимальная производительность устройств за счёт оптимизации подключений. Повышенная безопасность – благодаря родительскому контролю, гостевой сети и управлению доступом. Умное подключение — функция Smart Connect переводит клиентов на менее загруженный канал, а Airtime Fairness оптимизирует использование эфирного времени. WPA3 — поддержка последнего стандарта шифрования, улучшающего безопасность и обеспечивающего защиту от брутфорс атак. Поддержка OneMesh — используйте усилители OneMesh, чтобы создать покрытие во всём доме и не терять сигнал при перемещении. Беспроводное вещание Стандарты Wi-Fi 5 IEEE 802.11ac/n/a 5 ГГц IEEE 802.11n/b/g 2,4 ГГц. Скорость Wi-FiAC1900 5 ГГц: 1300 Мбит/с (802.11ac) 2,4 ГГц: 600 Мбит/с (802.11n)Покрытие Wi-Fi Для трёхкомнатных квартир4 фиксированные высокопроизводительные антенны Из нескольких антенн формируется мощный сигнал, увеличивающий радиус действия Beamforming Концентрирует беспроводной сигнал в направлении клиентов для расширения радиуса действия Wi-FiПропускная способность Wi-Fi Средняя Два диапазона Присвойте каждому устройству свой диапазон для оптимальной производительности 3×3 MU-MIMO Одновременная коммуникация с множеством клиентов MU-MIMO Airtime Fairness Улучшение эффективности сети за счёт одинакового времени передачи для всех клиентов Smart Connect Автоматическое назначение каждого устройства на лучший из доступных диапазонов Wi-FiРежимы работы.",

      },

      {

        id: "6",

        imageSrc: "/img/zyxel.png",

        text: "Zyxel",

        price: 21000,

        description:

          "Cетевое оборудование Маршрутизаторы Беспроводный маршрутизатор Zyxel NBG6604 с скоростным Wi-Fi стандарта 802.11ac. Сеть WiFi работает быстрее и у нее мощнее сигнал. Четыре сетевых порта Ethernet для подключения проводных устройств.Преимущества: Новый дизайн, более эффективное охлаждение. Опция установки на стене для большей гибкости. Cветодиодный индикатор WiFi WPS. Моб. приложение ONE Connect. Доступно несколько режимов работы: маршрутизатор и точка доступа. Возможность крепления на стене.Комплектация: Полную комплектацию товара уточняйте у производителяОсобенности: Нет поддержки Dual Mode для работы IPTV (одновременно с PPPoE). Нет поддержки L2TP и PPTP.",

      },

    ],

  },

  {

    id: "3",

    text: "Модемы",

    imageSrc: "/img/modem.jpg",

    items: [

      {

        id: "7",

        imageSrc: "/img/huawei.jpg",

        text: "Huawei",

        price: 4500,

        description:

          "Подключить компьютер или ноутбук к интернету поможет модем Huawei E3372-325 3G/4G. Его компактный и легкий корпус поместится даже в кармане, поэтому его комфортно везде брать с собой. Подключение к рабочему устройству происходит через интерфейс USB. Таким образом, процесс установки не вызовет сложностей у пользователей любого уровня.",

      },

      {

        id: "8",

        imageSrc: "/img/zte.jpg",

        text: "ZTE",

        price: 5600,

        description:

          "Работает со всеми операторами мобильной связи (Мегафон, МТС, Билайн, Теле 2, Yota) с любыми тарифами. В модеме есть слот для карт памяти MicroSD 32 Гб, позволяющих превратить его в портативный накопитель информации. Устройство выполняет еще и функции Wi-Fi роутера. Может работать как точка доступа для 10 устройств через Wi-Fi на частоте 2.4 ГГц. Радиус раздачи Wi-Fi до 10 метров. Модем ZTE MF79RU передает данные в стандарте 3G одновременно по двум каналам связи, благодаря чему достигается высокая скорость - до 43.2 Мбит/с (технология DC-HSPA+). В стандарте 4G устройство достигает максимальной скорости передачи данных до 150 Мбит/с на прием данных. Два разъема TS9 для подключения внешней антенны - FDD B1/3/7/20 - TDD B38 - UMTS B1/B8 Комплектация: 1. Модем MF79U - 1 шт. 2. Антенна TS9 - 2 шт.",

      },

      {

        id: "9",

        imageSrc: "/img/yota.jpg",

        text: "Yota",

        price: 4900,

        description:

          "Работает в частотах 4G 4G+  LTE со скоростью до 150Мбсек! Описание модема : - работает со всеми операторами МТС Теле2 Мегафон Билайн Ростелеком WiFire и др. - интерфейс подключения: USB - удобный Web интерфейс - стандарты: 4G 4G+LTE - размеры: 28x12x88 мм - вес 55 г -скорость приёма данных до 150 Мбит/с скорость передачи данных - до 50 Мбит/с.",

      },

    ],

  },

  {

    id: "4",

    text: "Смартфоны",

    imageSrc: "/img/smart.jpg",

    items: [

      {

        id: "10",

        imageSrc: "/img/xiaomi.jpeg",

        text: "Xiaomi",

        price: 30000,

        description:

          "6.67 ″ процессор Mediatek G99 Ultra , камера 200MP, 8 MP + 2 MP + 16 MP, 4G, Wi-Fi, Bluetooth 5.2, RAM 12GB, 512GB встроенной памяти, USB Type-C, microSD, поддержка 2-х SIM-карт, Android 14, 1161.1 x 75 x 8 мм, 188 г.",

      },

      {

        id: "11",

        imageSrc: "/img/honor.jpg",

        text: "Honor",

        price: 35000,

        description:

          "Указана актуальная на момент презентации устройства версия операционной системы: Android 13 Материал корпуса: Пластик Количество SIM-карт: 2 Режим работы нескольких SIM-карт: попеременный Цвет: Чёрный Вес (грамм): 183 Размеры: 16,19 x 7,41 x 0,78 см Объем встроенной памяти: 256 ГБ Стандарт сотовой связи: GSM / CDMA / HSPA / CDMA2000 / LTE / 5G Поддержка диапазонов 5G: 1, 5, 8, 28, 41, 77, 78 SA/NSA Поддержка диапазонов LTE: 1, 3, 4, 5, 8, 19, 28, 34, 38, 39, 40, 41 Интерфейсы: Bluetooth, Wi-Fi, NFC Поддержка Wi-Fi: Wi-Fi 802.11 a/b/g/n/ac/6 Возможности Wi-Fi: Wi-Fi Direct Поддержка Bluetooth: Bluetooth 5.2 Возможности Bluetooth: A2DP, LE, aptX HD ",

      },

      {

        id: "12",

        imageSrc: "/img/samsung.jpg",

        text: "Samsung",

        price: 39000,

        description:

          "SAMSUNG Galaxy A55 — смартфон с большим четким дисплеем, качественной системой камер и фирменным дизайном. Устройство получило емкую батарею и прочный корпус с защитой от влаги по стандарту IP67. По вопросам гарантийного обслуживания необходимо обращаться в магазины МТС (АО РТК). Наличие сервисной поддержки Samsung в РФ.",

      },

    ],

  },

  {

    id: "5",

    text: "Планшеты",

    imageSrc: "/img/tablet.jpg",

    items: [

      {

        id: "13",

        imageSrc: "/img/tabsamsung.jpg",

        text: "Samsung",

        price: 31000,

        description:

          "Планшет Samsung Galaxy Tab A9+ 5G Graphite (SM-X216B) обладает экраном диагональю 11 дюймов разрешением 1200x1920 пикселей. Отображается 16 млн цветов. Частота обновления составляет 60 Гц, что обеспечивает плавность прокрутки страниц.  основе модели лежит восьмиядерный процессор Qualcomm Snapdragon 695 5G, частота которого достигает 2,2 ГГц. В сочетании с 8 Гб оперативной памяти он гарантирует стабильную работу. Объем хранилища данных — 128 Гб, при необходимости его можно увеличить за счет карты microSD (до 1024 Гб).Встроены модули Wi-Fi и Bluetooth для выхода в интернет и обмена файлами с совместимыми устройствами. Поддерживается технология Wi-Fi Direct. Четыре динамика воспроизводят громкий звук, микрофоны отвечают за точную передачу речи при общении, в том числе по видеосвязи — для этого предусмотрена фронтальная камера разрешением 5 Мп. Можно подключить проводные наушники через 3,5-миллиметровый разъем. Тыловая камера на 8 Мп способна снимать качественные фотографии и видеоролики.",

      },

      {

        id: "14",

        imageSrc: "/img/tabhuawei.jpg",

        text: "Huawei",

        price: 22000,

        description:

          "Планшет Huawei MatePad космический серый (год презентации — 2023) с диагональю экрана 11,5 дюйма представлен в цельнометаллическом корпусе габаритами весом 499 г и оснащен съемной клавиатурой для быстрого безошибочного ввода данных. Девайс функционирует под управлением восьмиядерного процессора Snapdragon 7 Gen 1 частотой 2,4 ГГц и видеопроцессора Adreno 644. Размер оперативной памяти — 8 Гб, встроенной — 128 Гб. Беспроводные адаптеры Wi-Fi 4, 5, 6 (двухдиапазонный) и Bluetooth 5.2 расширяют коммуникационные возможности устройства. Операционная система Harmony OS 3.1 отличается повышенным быстродействием благодаря интеллектуальной опции важности выполнения задач.",

      },

      {

        id: "15",

        imageSrc: "/img/tablenovo.jpg",

        text: "Lenovo",

        price: 18000,

        description:

          "Работает планшет на базе восьмиядерного центрального процессора Qualcomm Snapdragon 685 с тактовой частотой 2,8 ГГц, который обеспечивает оптимальный уровень быстродействия при различных сценариях использования. Интегрированный графический микроконтроллер Qualcomm Adreno 610 и 8 Гб оперативной памяти также повышают скорость обработки системных команд и запросов пользователя при загрузке операционной системы Android 13 с системным интерфейсом ZUI 15.",

      },

    ],

  },

  {

    id: "6",

    text: "SSD",

    imageSrc: "/img/ssd.jpg",

    items: [

      {

        id: "16",

        imageSrc: "/img/ssdsamsung.jpg",

        text: "Samsung",

        price: 5500,

        description:

          "Жесткий диск SSD Samsung 870 Evo выполнен в форм-факторе 2,5”. Он подходит для стационарных компьютеров и ноутбуков с интерфейсом SATA III, использующихся для игр, создания контента и обработки больших объемов информации. Корпус твердотельного накопителя изготовлен из ABS-пластика. В нем размещается флеш-память типа 3D TLC NAND. Модель совместима с устройствами с Windows 7, 8 и 10. В режиме чтения максимальная скорость передачи данных составляет 560 МБ/с, при записи — 530 МБ/с.",

      },

      {

        id: "17",

        imageSrc: "/img/ssdadata.jpg",

        text: "Adata",

        price: 5300,

        description:

          "Современные модели SSD с типом памяти 3D обеспечивает плотность хранения, открывая возможности для работы с большими объемами данных и повышая производительность. С вместимостью 1 ТБ вы сможете хранить все важные файлы, мультимедию и игры, наслаждаясь высокой производительностью и мгновенной загрузкой даже тяжелых приложений.Интерфейс SATA III обеспечивает отличную скорость передачи данных и совместимость с большинством современных систем.",

      },

      {

        id: "18",

        imageSrc: "/img/ssdcrucial.jpg",

        text: "Crucial",

        price: 4700,

        description:

          "SATA накопитель Crucial BX500 - быстрое и надежное устройство хранения, которое обеспечивает высокий уровень производительности обработки ресурсов разной сложности. Он представлен в востребованном форм-факторе 2.5 дюйма и оснащен интерфейсом подключения SATA.Устройство Crucial BX500 отличается совместимостью с ноутбуками, стационарными ПК и другими вычислительными платформами.Объем 500 ГБ гарантирует размещение большого количества документов, мультимедийных файлов, игр.",

      },

    ],

  },

  {

    id: "7",

    text: "USB",

    imageSrc: "/img/usb.jpg",

    items: [

      {

        id: "19",

        imageSrc: "/img/usbsandisk.jpg",

        text: "SanDisk",

        price: 1500,

        description:

          "Высокая Скорость и Стиль: USB-флеш-накопитель SanDisk Ultra Flair USB 3.0 гарантирует быструю передачу файлов. Экономьте время — отдайте предпочтение высокой скорости USB 3.0, которая достигает 150 МБ/с. Надежный и изящный металлический корпус накопителя выдерживает сильные удары. Защита паролем обеспечит конфиденциальность ваших личных файлов. Храните файлы на стильном USB-флеш-накопителе SanDisk Ultra Flair USB 3.0.",

      },

      {

        id: "20",

        imageSrc: "/img/usbkingston.jpg",

        text: "Kingston",

        price: 1170,

        description:

          "Флешка USB Kingston DataTraveler Kyson - стильный и надежный накопитель.Устройство DataTraveler® Kyson™ корпорации Kingston — высокопроизводительный флеш-накопитель USB типа А. Металлическая конструкция без колпачка избавит Вас от неприятностей из-за потери колпачка, а функциональное кольцо облегчает взятие устройства с собой куда угодно.Флешка USB Kingston DataTraveler Kyson предлагает надежное и быстрое решение для хранения и переноса больших объемов данных. С USB-накопителем емкостью 64 Гб, Вы можете хранить фотографии, видео, музыку, другой контент и обмениваться ими прямо на ходу.",

      },

      {

        id: "21",

        imageSrc: "/img/usbtranscend.jpg",

        text: "Transcend",

        price: 1600,

        description:

          "Флешка USB 3.0 Transcend JetFlash 790 (TS128GJF790K) - это компактное, портативное и удобное устройство для хранения и передачи цифровой информации. С его помощью можно легко переносить и хранить документы, фотографии, видео, музыку и многое другое.Флешка оснащена легко подключаемым интерфейсом USB 3.0, который обеспечивает быструю и простую передачу данных. Благодаря этому, вы сможете быстро и легко подключить ее к компьютеру или другому устройству с поддержкой USB.Флешка имеет объем памяти 128 ГБ, что позволяет хранить большое количество информации.Transcend JetFlash 790 имеет выдвижной разъем, который обеспечивает удобство использования и позволяет легко подключать флешку к различным устройствам.",

      },

    ],

  },

];

export const items = {};

products.forEach((product) => {

  if (product.items) {

    product.items.forEach((item) => {

      items[item.id] = item;

    });

  }

});

**index.css**

:root {

  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;

  line-height: 1.5;

  font-weight: 400;

  color-scheme: light dark;

  color: rgba(255, 255, 255, 0.87);

  background-color: #242424;

  font-synthesis: none;

  text-rendering: optimizeLegibility;

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

}

a {

  font-weight: 500;

  color: #646cff;

  text-decoration: inherit;

}

a:hover {

  color: #535bf2;

}

body {

  margin: 0;

  display: flex;

  place-items: center;

  min-width: 320px;

  min-height: 100vh;

}

h1 {

  font-size: 3.2em;

  line-height: 1.1;

}

button {

  border-radius: 8px;

  border: 1px solid transparent;

  padding: 0.6em 1.2em;

  font-size: 1em;

  font-weight: 500;

  font-family: inherit;

  background-color: #1a1a1a;

  cursor: pointer;

  transition: border-color 0.25s;

}

button:hover {

  border-color: #646cff;

}

button:focus,

button:focus-visible {

  outline: 4px auto -webkit-focus-ring-color;

}

@media (prefers-color-scheme: light) {

  :root {

    color: #213547;

    background-color: #ffffff;

  }

  a:hover {

    color: #747bff;

  }

  button {

    background-color: #f9f9f9;

  }

}

**main.jsx**

import { createRoot } from 'react-dom/client'

import { App } from './App'

import {getCurrency} from './component/Exchange/getCurrency'

// Получаем данные о валютных курсах с помощью асинхронной функции getCurrency

getCurrency().then((currency) => {

    // После получения данных, рендерим компонент App и передаем данные о валюте в пропс currency

    // Это позволяет компонентам, использующим CurrencyContext, получить доступ к этим данным

    createRoot(document.getElementById('root')).render(<App currency={currency} />);

});