Geekbrains

**Создание типового интернет-магазина с возможностью фильтрации товаров и оставлении отзывов на фреймворке React**

Программа:Разработчик

Специализация: Разработчик — Frontend разработка (React)

ФИО Аубакиров Ренат Сергеевич

Город Новосибирск

Год 2025

**Содержание**

1. **Введение**  
   1.1. Актуальность темы  
   1.2. Цель и задачи исследования  
   1.3. Объект и предмет исследования  
   1.4. Ожидаемые результаты работы  
   1.5. Структура работы
2. **Глава 1. Теоретические основы разработки интернет-магазинов**  
   1.1. Обзор рынка интернет-торговли  
   1.2. Особенности разработки веб-приложений  
   1.3. Выбор технологий для создания интернет-магазина  
   1.4. Описание фреймворка React  
   1.5. Важность отзывов и фильтрации товаров в интернет-магазинах
3. **Глава 2. Анализ и проектирование интернет-магазина**  
   2.1. Обзор и описание проекта  
   2.2. Архитектура приложения  
   2.3. Проектирование структуры базы данных  
   2.4. Описание компонентов и функционала интернет-магазина  
   2.5. Разработка интерфейса пользователя
4. **Глава 3. Реализация функционала интернет-магазина на React**  
   3.1. Настройка и выбор инструментов разработки (React, Redux, Router и т.д.)  
   3.2. Реализация компонента отображения товаров  
   3.3. Создание страницы товара и корзины  
   3.4. Разработка функционала поиска и сортировки товаров  
   3.5. Добавление функционала отзывов к товарам  
   3.6. Добавление функционала фильтрации товаров
5. **Глава 4. Реализация системы отзывов пользователей**  
   4.1. Описание и принципы работы системы отзывов  
   4.2. Создание интерфейса для добавления и отображения отзывов  
   4.3. Тестирование системы отзывов
6. **Глава 5. Тестирование интернет-магазина**  
   5.1. Введение в тестирование веб-приложений  
   5.2. Цели и задачи тестирования  
   5.3. Методики ручного тестирования  
   5.4. Тест-кейсы для интернет-магазина  
   5.5. Результаты тестирования и выявленные ошибки  
   5.6. Выводы по тестированию
7. **Глава 6. Оптимизация производительности интернет-магазина**  
   6.1. Оптимизация работы с состоянием в React  
   6.2. Оптимизация фильтрации товаров  
   6.3. Оптимизация взаимодействия с сервером  
   6.4. Улучшение пользовательского опыта (UX/UI)
8. **Заключение**  
   8.1. Основные результаты работы  
   8.2. Проблемы и ограничения при разработке  
   8.3. Вклад проекта в развитие интернет-торговли
9. **Список использованных источников**

**Введение**

Введение раскрывает обоснование необходимости исследования выбранной задачи и представляет схему проведения дипломного исследования.

Трудно представить жизнь современного человека без покупок товаров в интернете, а крупную торговую сеть без интернет-магазина. При этом электронная торговля растет с каждым днем, на ее рост влияет множество факторов от появления интернета даже в самых отдаленных точках планеты до пандемии. За последние пять лет, с 2019 по 2024 г., российский рынок электронной коммерции вырос в 7,5 раза – с 1,7 трлн до 12,6 трлн руб., говорится в исследовании «Развитие электронной коммерции в России» («Ведомостей»). Доля электронной коммерции в общей розничной торговле за этот период выросла с 5 до 23%. В связи с ростом электронных продаж, любая тема, связанная с торговлей в интернете в современных реалиях, очень актуальна и подлежит изучению. В данной работе наибольшее внимание направлено именно на веб-приложение интернет-магазина и особенности его разработки.

В 2024 году отмечался рост популярности JavaScript и одностраничных приложений (SPAs). Разработчики сталкивались с проблемами работы с большими базами кода, и одной из задач стало поддержание баланса между попытками внедрить новые технологии и удовлетворением практических потребностей.

В данном проекте рассматривается разработка интернет-магазина с использованием современных веб-технологий, таких как React, Express и styled-components. Основное внимание уделяется реализации функционала фильтрации товаров и системы отзывов, которые являются ключевыми элементами для улучшения пользовательского опыта и повышения лояльности клиентов.

**Что из себя представляет проект**

Проект представляет собой веб-приложение интернет-магазина, которое позволяет пользователям просматривать товары, фильтровать их по различным параметрам, добавлять товары в корзину, оформлять заказы и оставлять отзывы. Приложение разработано с использованием React для фронтенда и Express для бэкенда. Основной акцент сделан на создании удобного и интуитивно понятного интерфейса, а также на реализации функционала, который улучшает взаимодействие пользователя с магазином.

Актуальность темы данной работы заключается в необходимости создания эффективных и удобных для пользователя интернет-магазинов, которые смогут удовлетворить растущие потребности клиентов в быстрой и качественной покупке товаров через интернет. Многие интернет-магазины сталкиваются с проблемой неудобного интерфейса, неинтуитивной фильтрации товаров и отсутствия системы обратной связи с пользователями. Таким образом, создание интернет-магазина с функциональностью фильтрации товаров и отзывов представляет собой важную задачу в контексте улучшения качества обслуживания клиентов.

**Цель проекта**

Целью данной работы является создание типового интернет-магазина, реализующего функционал фильтрации товаров и оставления отзывов, с использованием фреймворка React. Для достижения поставленной цели необходимо решить следующие задачи:

* Изучить теоретические основы разработки интернет-магазинов.
* Проанализировать требования к функционалу фильтрации товаров и системе отзывов.
* Разработать архитектуру интернет-магазина, реализующего указанные функциональности.
* Реализовать интерфейс пользователя, удобный и интуитивно понятный.
* Провести тестирование и оптимизацию работы интернет-магазина.

**План работы**

Работа состоит из нескольких глав, в которых описаны теоретические аспекты разработки интернет-магазинов, процесс проектирования и реализации функционала на основе React, а также этапы тестирования и оптимизации. В конце работы подведены основные итоги, сделаны выводы и предложены рекомендации по дальнейшему улучшению проекта.

Проект решает проблему неудобного интерфейса и отсутствия функционала фильтрации товаров и системы отзывов в интернет-магазинах. Это позволяет улучшить пользовательский опыт и повысить лояльность клиентов.

Дипломный проект выполняется по специализации "Веб-разработка". Основное внимание уделяется фронтенд-разработке с использованием React.

Для разработки проекта будут использоваться следующие инструменты:

* React для создания пользовательского интерфейса.
* Express для разработки бэкенда.
* styled-components для стилизации компонентов.
* Axios для выполнения HTTP-запросов.
* Git для контроля версий.

В проекте будут использоваться следующие технологии:

* React для фронтенда.
* Express для бэкенда.
* styled-components для стилизации.

**Состав команды**

Проект разрабатывался индивидуально. В процессе разработки я выполнял следующие роли:

* Фронтенд-разработчик: разработка пользовательского интерфейса и реализация функционала на React.
* Дизайнер: создание макетов и стилей для интерфейса.
* Проджект-менеджер: проработка и составление этапов работы, контроль качества и сроков.

**Глава 1. Теоретические основы разработки интернет-магазинов**

**1.1. Обзор рынка интернет-торговли**

Интернет-торговля, или электронная коммерция, охватывает широкий спектр товаров и услуг, которые могут быть куплены и проданы через интернет. Рынок интернет-торговли переживает бурный рост, что объясняется развитием технологий, улучшением доступа к интернету и увеличением числа пользователей, предпочитающих совершать покупки онлайн. В последние годы электронная коммерция стала не только важным каналом для бизнеса, но и значимой частью повседневной жизни миллионов людей.

Основные категории товаров, продаваемых через интернет-магазины, включают одежду, электронику, бытовую технику, книги, продукты питания, косметику и многое другое. Важным фактором успешности интернет-магазинов является удобство использования, быстрая навигация по каталогу товаров и простота оформления заказов. В этом контексте эффективность работы интернет-магазина определяется тем, насколько легко и удобно пользователи могут найти и приобрести нужные им товары.

**Статистика и тренды:**

* По данным Statista, объем мирового рынка электронной коммерции в 2023 году составил более 6,3 триллиона долларов, а к 2027 году ожидается рост до 8,1 триллиона долларов.
* В России объем рынка интернет-торговли в 2022 году достиг 4,5 триллиона рублей, что на 20% больше, чем в предыдущем году.
* Основными драйверами роста являются увеличение проникновения интернета, развитие мобильных технологий и улучшение логистики.

**Преимущества интернет-торговли:**

* Глобальный охват: Интернет-магазины могут привлекать клиентов со всего мира.
* Круглосуточная доступность: Пользователи могут совершать покупки в любое время.
* Снижение издержек: Отсутствие необходимости в физических магазинах снижает затраты на аренду и персонал.

**Проблемы и вызовы:**

* Конкуренция: На рынке присутствует огромное количество игроков, что требует от интернет-магазинов постоянного улучшения сервиса.
* Безопасность: Защита персональных данных и платежной информации является критически важной.
* Логистика: Быстрая и надежная доставка товаров — ключевой фактор удовлетворенности клиентов.

**1.2. Особенности разработки веб-приложений**

Разработка интернет-магазинов имеет свои особенности, отличающие этот процесс от создания традиционных веб-сайтов. Одной из ключевых особенностей является высокая степень интерактивности и функциональности, требующая интеграции с различными базами данных, платежными системами и внешними сервисами.

**Основные этапы разработки:**

1. Проектирование архитектуры: Определение структуры приложения, выбор технологий и инструментов.
2. Создание пользовательского интерфейса: Разработка удобного и интуитивно понятного интерфейса.
3. Разработка серверной логики: Создание API для обработки запросов и взаимодействия с базой данных.
4. Интеграция с внешними системами: Подключение платежных систем, служб доставки и других сервисов.
5. Тестирование и оптимизация: Проверка функциональности, производительности и безопасности приложения.

**Особенности интернет-магазинов:**

* Высокая нагрузка: Интернет-магазины должны выдерживать большое количество одновременных пользователей.
* Сложная логика: Необходимость реализации корзины покупок, системы скидок, фильтрации товаров и других функций.
* Безопасность: Защита данных пользователей и предотвращение мошенничества.

**Примеры успешных решений:**

* Использование кэширования для ускорения загрузки страниц.
* Реализация адаптивного дизайна для поддержки мобильных устройств.
* Интеграция с системами аналитики для отслеживания поведения пользователей.

**1.3. Выбор технологий для создания интернет-магазина**

Одним из самых важных этапов в создании интернет-магазина является выбор технологий, которые будут использованы для разработки как клиентской, так и серверной части приложения. В настоящее время существует множество фреймворков и библиотек, которые помогают упростить процесс разработки и ускорить создание функциональных и стабильных веб-приложений.

**Клиентская часть:**

* React: Библиотека для создания интерактивных пользовательских интерфейсов.
* Vue.js: Легкий и гибкий фреймворк для разработки веб-приложений.
* Angular: Мощный фреймворк для создания сложных приложений.

**Серверная часть:**

* Node.js: Платформа для разработки высокопроизводительных серверных приложений.
* Express: Фреймворк для создания API и обработки запросов.
* Django: Фреймворк для разработки веб-приложений на Python.

**Базы данных:**

* MongoDB: NoSQL-база данных, подходящая для хранения неструктурированных данных.
* PostgreSQL: Реляционная база данных с поддержкой сложных запросов.
* MySQL: Популярная реляционная база данных.

**Критерии выбора технологий:**

* Производительность: Технологии должны обеспечивать быструю обработку запросов.
* Масштабируемость: Возможность увеличения мощности приложения при росте нагрузки.
* Безопасность: Защита данных и предотвращение утечек информации.

**1.4. Описание фреймворка React**

React — это библиотека для создания пользовательских интерфейсов, разработанная Facebook. Она основывается на концепции компонентов, которые позволяют эффективно и гибко строить интерфейсы для сложных приложений. React идеально подходит для разработки динамичных веб-приложений, таких как интернет-магазины, где важна высокая интерактивность и быстрая загрузка данных.

**Преимущества React:**

* Компонентный подход: Позволяет разбивать интерфейс на независимые компоненты.
* Виртуальный DOM: Обеспечивает высокую производительность за счет минимизации обновлений реального DOM.
* Широкая экосистема: Большое количество библиотек и инструментов для расширения функциональности.

**Пример использования:**

* Создание интерактивных фильтров для товаров.
* Реализация корзины покупок с динамическим обновлением.
* Разработка одностраничных приложений (SPA) для плавного пользовательского опыта.

**1.5. Важность отзывов и фильтрации товаров в интернет-магазинах**

Отзывы пользователей и система фильтрации товаров являются важными элементами, которые оказывают существенное влияние на успех интернет-магазина.

**Фильтрация товаров:**

* Позволяет пользователям быстро находить нужные товары.
* Включает такие параметры, как категория, цена, бренд, размер и рейтинг.
* Улучшает пользовательский опыт и повышает конверсию.

**Отзывы пользователей:**

* Помогают покупателям принимать обоснованные решения.
* Повышают доверие к магазину.
* Используются для улучшения качества товаров и обслуживания.

**Примеры успешной реализации:**

* Амазон: Использование системы рейтингов и отзывов для повышения доверия.
* Алиэкспресс: Удобная фильтрация товаров по множеству параметров.

**Глава 2. Анализ и проектирование интернет-магазина**

**2.1. Обзор и описание проекта**

Интернет-магазин представляет собой веб-приложение, разработанное для продажи одежды. Основная цель проекта — предоставить пользователям удобный и интуитивно понятный интерфейс для просмотра товаров, выбора размеров, добавления товаров в корзину и оформления заказов. Проект включает в себя следующие основные функции:

* Просмотр товаров: Пользователь может просматривать каталог товаров с изображениями, названиями и ценами. Каждый товар имеет несколько доступных размеров, которые можно выбрать перед добавлением в корзину.
* Фильтрация по размерам: Пользователь может фильтровать товары по доступным размерам, что упрощает поиск подходящих товаров.
* Добавление товаров в корзину: Пользователь может добавлять товары в корзину, выбирая нужный размер. В корзине можно изменять количество товаров и удалять их.
* Оформление заказа: Пользователь может оформить заказ, указав свои контактные данные (имя, телефон, email). После оформления заказа данные отправляются на сервер, который обрабатывает заказ и отправляет уведомление на почту администратора.
* Отзывы: Пользователь может оставлять отзывы о товарах и просматривать отзывы других пользователей. Отзывы включают текстовый комментарий и рейтинг от 1 до 5 звезд.
* Контакты: Пользователь может просмотреть контактную информацию магазина, включая адрес, телефон и email.

Проект разработан с использованием современных технологий, таких как React для фронтенда, Express для бэкенда и styled-components для стилизации компонентов. Это позволяет создавать динамические и адаптивные интерфейсы, которые легко масштабируются и поддерживаются.

**2.2. Архитектура приложения**

Архитектура приложения разделена на два основных слоя: фронтенд и бэкенд. Каждый слой выполняет свои функции и взаимодействует с другим через REST API.

**2.2.1. Фронтенд**  
Фронтенд приложения реализован на React — популярной библиотеке для создания пользовательских интерфейсов. React позволяет создавать компонентный подход, где каждый компонент отвечает за определенную часть интерфейса. Основные компоненты фронтенда:

* **App:** Главный компонент, который управляет маршрутизацией и состоянием приложения. Он определяет, какие компоненты должны отображаться в зависимости от текущего URL.
* **ProductList:** Компонент для отображения списка товаров. Он использует данные из массива products и отображает их в виде карточек.
* **ProductCard:** Компонент карточки товара. Он отображает изображение, название, цену и кнопку для добавления в корзину. Также есть возможность выбора размера товара.
* **Cart:** Компонент корзины. Он позволяет пользователю управлять добавленными товарами, изменять их количество, удалять товары и оформлять заказ.
* **Filters:** Компонент для фильтрации товаров по размерам. Пользователь может выбрать один или несколько размеров, и список товаров будет отфильтрован соответствующим образом.
* **Menu:** Компонент для навигации по сайту. Он включает ссылки на главную страницу, страницу отзывов и страницу контактов.
* **ReviewsPage:** Страница для просмотра и добавления отзывов. Пользователь может оставить отзыв с рейтингом и текстом, а также просмотреть отзывы других пользователей.
* **ContactsPage:** Страница с контактной информацией магазина.

**2.2.2. Бэкенд**

Бэкенд приложения реализован на Express — минималистичном фреймворке для Node.js. Он выполняет следующие функции:

* Обработка заказов: Принимает данные о заказе от фронтенда и отправляет их на электронную почту администратора.
* Маршрутизация: Обрабатывает запросы на оформление заказа.

Пример кода для обработки заказа на бэкенде:

javascript

Copy

const express = require('express');

const cors = require('cors');

const nodemailer = require('nodemailer');

const app = express();

app.use(express.json());

app.use(cors());

const transporter = nodemailer.createTransport({

service: 'gmail',

auth: {

user: 'arenat98@gmail.com',

pass: 'your-app-password' // Используй App Password, а не обычный пароль

}

});

app.post('/order', async (req, res) => {

const { customer, cartItems } = req.body;

const itemsList = cartItems.map(

(item) => `${item.name} - ${item.size}, Количество: ${item.quantity}`

).join('\n');

const mailOptions = {

from: 'your-email@gmail.com',

to: 'arenat98@gmail.com',

subject: 'Новый заказ',

text: `Имя: ${customer.name}\nТелефон: ${customer.phone}\nEmail: ${customer.email}\n\nТовары:\n${itemsList}`

};

try {

await transporter.sendMail(mailOptions);

res.status(200).json({ message: 'Email sent' });

} catch (error) {

res.status(500).json({ message: 'Error sending email', error });

}

});

app.listen(5000, () => {

console.log('Server running on port 5000');

});

**2.3. Проектирование структуры базы данных**

В данном проекте база данных не используется, так как все данные о товарах хранятся в виде статического массива в файле products.js.

**2.3.1. Таблица "Products"**  
Эта таблица хранит информацию о товарах:

* **id (Primary Key):** Уникальный идентификатор товара.
* **name:** Название товара.
* **price:** Цена товара.
* **image:** Ссылка на изображение товара.
* **hoverImage:** Ссылка на изображение товара при наведении.
* **sizes:** Доступные размеры, хранятся в виде JSON или отдельной таблицы.

**2.3.3. Таблица "Reviews"**  
Эта таблица хранит информацию об отзывах:

* **id (Primary Key):** Уникальный идентификатор отзыва.
* **product\_id (Foreign Key):** Ссылка на товар.
* **text:** Текст отзыва.
* **rating:** Рейтинг от 1 до 5.

**2.4. Описание компонентов и функционала интернет-магазина**

**2.4.1. ProductList**  
Компонент ProductList отвечает за отображение списка товаров. Он использует данные из массива products и отображает их в виде карточек. Каждая карточка содержит изображение, название, цену и кнопку для добавления в корзину.

Пример кода:

javascript

Copy

const ProductList = ({ products, onAddToCart, reviews, onAddReview }) => {

const [selectedSizes, setSelectedSizes] = useState({});

const handleSizeChange = (productId, size) => {

setSelectedSizes((prev) => ({

...prev,

[productId]: size,

}));

};

const handleAddToCartWithSize = (product) => {

const selectedSize = selectedSizes[product.id];

if (!selectedSize) {

alert('Выберите нужный размер');

return;

}

onAddToCart(product, selectedSize);

};

return (

<List>

{products.map((product) => (

<div key={product.id}>

<ProductCard

product={product}

onAddToCart={() => handleAddToCartWithSize(product)}

reviews={reviews}

onAddReview={onAddReview}

/>

<SizeSelector

value={selectedSizes[product.id] || ''}

onChange={(e) => handleSizeChange(product.id, e.target.value)}

>

<option value="">Размер одежды</option>

{product.sizes.map((size) => (

<option key={size} value={size}>

{size}

</option>

))}

</SizeSelector>

</div>

))}

</List>

);

};

**2.4.2. Cart**  
Компонент Cart отвечает за управление корзиной. Он позволяет пользователю просматривать добавленные товары, изменять их количество, удалять товары и оформлять заказ.

Пример кода:

javascript

Copy

const Cart = ({ cartItems, onRemoveFromCart, onUpdateQuantity }) => {

const [isOpen, setIsOpen] = useState(false);

const [isCheckoutOpen, setIsCheckoutOpen] = useState(false);

const [customer, setCustomer] = useState({ name: '', phone: '', email: '' });

const total = cartItems.reduce((acc, item) => acc + item.price \* item.quantity, 0);

const toggleCart = () => setIsOpen(!isOpen);

const toggleCheckout = () => setIsCheckoutOpen(!isCheckoutOpen);

const handleQuantityChange = (item, change) => {

onUpdateQuantity(item.id, item.size, change);

};

const handleRemoveItem = (itemId, itemSize) => {

onRemoveFromCart(itemId, itemSize);

};

const handleInputChange = (e) => {

const { name, value } = e.target;

setCustomer({ ...customer, [name]: value });

};

const handleOrderSubmit = async () => {

try {

await axios.post('http://localhost:5000/order', { customer, cartItems });

alert('Заказ оформлен! Мы свяжемся с вами.');

setIsCheckoutOpen(false);

} catch (error) {

alert('Ошибка при оформлении заказа');

}

};

return (

<>

<CartToggle onClick={toggleCart}>

<CartIcon src={cartIcon} alt="Cart" />

</CartToggle>

<CartContainer isOpen={isOpen}>

<CloseButton onClick={toggleCart}>X</CloseButton>

<h2>Cart</h2>

<CartItems>

{cartItems.map((item) => (

<CartItem key={`${item.id}-${item.size}`}>

<ItemImage src={item.image} alt={item.name} />

<ItemDetails>

<ItemName>{item.name}</ItemName>

<ItemPrice>${(item.price \* item.quantity).toFixed(2)}</ItemPrice>

<SizeSelector>Размер: {item.size}</SizeSelector>

<QuantityControls>

<QuantityButton onClick={() => handleQuantityChange(item, -1)}>

-

</QuantityButton>

<QuantityDisplay>{item.quantity}</QuantityDisplay>

<QuantityButton onClick={() => handleQuantityChange(item, 1)}>

+

</QuantityButton>

</QuantityControls>

</ItemDetails>

<RemoveButton onClick={() => handleRemoveItem(item.id, item.size)}>

Удалить

</RemoveButton>

</CartItem>

))}

</CartItems>

<CartTotal>Total: ${total.toFixed(2)}</CartTotal>

<CheckoutButton onClick={toggleCheckout}>Оформить заказ</CheckoutButton>

</CartContainer>

{isCheckoutOpen && (

<Modal>

<ModalContent>

<ModalCloseButton onClick={toggleCheckout}>×</ModalCloseButton>

<h2>Введите данные</h2>

<Input

type="text"

name="name"

placeholder="Имя"

onChange={handleInputChange}

/>

<Input

type="tel"

name="phone"

placeholder="Телефон"

onChange={handleInputChange}

/>

<Input

type="email"

name="email"

placeholder="Email"

onChange={handleInputChange}

/>

<SubmitButton onClick={handleOrderSubmit}>Отправить заказ</SubmitButton>

</ModalContent>

</Modal>

)}

</>

);

};

**2.5. Разработка интерфейса пользователя**

Интерфейс пользователя разработан с использованием библиотеки styled-components, что позволяет создавать стилизованные компоненты с поддержкой динамических свойств. Основные элементы интерфейса включают:

* **Кнопки:** Все кнопки имеют единый стиль, включая кнопки добавления в корзину, оформления заказа и навигации.
* **Карточки товаров:** Карточки товаров имеют минималистичный дизайн с акцентом на изображение товара и его основные характеристики.
* **Корзина:** Корзина отображается в виде боковой панели, которая появляется при нажатии на иконку корзины. Внутри корзины пользователь может управлять товарами и оформлять заказ.
* **Модальные окна:** Модальные окна используются для оформления заказа и отображения форм ввода данных.

Пример стилизации кнопки:

javascript

Copy

export const AddToCartButton = styled.button`

background-color: black;

color: white;

border: none;

padding: 10px 20px;

cursor: pointer;

margin-top: 10px;

width: 100%;

font-size: 16px;

transition: background-color 0.3s ease, color 0.3s ease;

&:hover {

background-color: #ffd700;

color: black;

}

`;

**Глава 3. Реализация функционала интернет-магазина на React**

**3.1. Настройка и выбор инструментов разработки (React, Redux, Router и т.д.)**

Для разработки интернет-магазина были выбраны следующие инструменты и технологии:

* **React:** Библиотека для создания пользовательских интерфейсов. React позволяет создавать компонентный подход, где каждый компонент отвечает за определенную часть интерфейса. Это упрощает разработку и поддержку кода.
* **React Router:** Библиотека для маршрутизации в React-приложениях. Она позволяет создавать многостраничные приложения с динамической загрузкой компонентов.
* **Redux:** Библиотека для управления состоянием приложения. Redux используется для централизованного хранения данных и управления ими.
* **Styled-components:** Библиотека для стилизации компонентов. Она позволяет создавать стилизованные компоненты с поддержкой динамических свойств.
* **Axios:** Библиотека для выполнения HTTP-запросов. Она используется для взаимодействия с бэкендом.
* **Express:** Минималистичный фреймворк для Node.js, который используется для создания бэкенда.

**Настройка проекта**  
Для создания проекта был использован Create React App, который предоставляет готовую конфигурацию для разработки React-приложений. Установка и настройка проекта выполняются следующим образом:

bash

Copy

npx create-react-app my-shop

cd my-shop

npm install react-router-dom redux react-redux axios styled-components

После установки зависимостей проект был структурирован следующим образом:

Copy

src/

|-- components/

| |-- Cart/

| |-- Filters/

| |-- Menu/

| |-- ProductCard/

| |-- ProductList/

|-- pages/

| |-- ContactsPage/

| |-- ReviewsPage/

|-- App.js

|-- index.js

**3.2. Реализация компонента отображения товаров**

Компонент ProductList отвечает за отображение списка товаров. Он использует данные из массива products и отображает их в виде карточек. Каждая карточка содержит изображение, название, цену и кнопку для добавления в корзину.

Пример кода:

javascript

Copy

import React, { useState } from 'react';

import { List, SizeSelector } from './ProductList.styles';

import ProductCard from '../ProductCard/ProductCard';

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

const ProductList = ({ products, onAddToCart, reviews, onAddReview }) => {

  const [selectedSizes, setSelectedSizes] = useState({});

  const location = useLocation();

  const showSizeSelector = !['/reviews', '/contacts'].includes(location.pathname);

  const handleSizeChange = (productId, size) => {

    setSelectedSizes((prev) => ({

      ...prev,

      [productId]: size,

    }));

  };

  const handleAddToCartWithSize = (product) => {

    const selectedSize = selectedSizes[product.id];

    if (!selectedSize) {

      alert('Выберите нужный размер');

      return;

    }

    onAddToCart(product, selectedSize);

  };

  return (

    <List>

      {products.map((product) => (

        <div key={product.id}>

          <ProductCard

            product={product}

            onAddToCart={() => handleAddToCartWithSize(product)}

            reviews={reviews}

            onAddReview={onAddReview}

          />

          {showSizeSelector && (

            <SizeSelector

              value={selectedSizes[product.id] || ''}

              onChange={(e) => handleSizeChange(product.id, e.target.value)}

            >

              <option value="">Размер одежды</option>

              {product.sizes.map((size) => (

                <option key={size} value={size}>

                  {size}

                </option>

              ))}

            </SizeSelector>

          )}

        </div>

      ))}

    </List>

  );

};

export default ProductList;

**Теоретическая часть**  
Компонент ProductList использует состояние selectedSizes для хранения выбранных размеров товаров. При изменении размера товара вызывается функция handleSizeChange, которая обновляет состояние. При нажатии на кнопку "Купить товар" вызывается функция handleAddToCartWithSize, которая проверяет, выбран ли размер, и добавляет товар в корзину.

**3.3. Создание страницы товара и корзины**

**Страница товара**  
Страница товара отображает подробную информацию о товаре, включая изображение, название, цену, доступные размеры и отзывы. Для реализации страницы товара используется компонент ProductCard.

Пример кода:

javascript

Copy

import React from 'react';

import {

Card,

ProductImage,

ProductName,

ProductPrice,

AddToCartButton,

HoverImage,

CardContainer,

ImageContainer,

ReviewsButton,

} from './ProductCard.styles';

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

const ProductCard = ({ product, onAddToCart, onAddReview }) => {

const navigate = useNavigate();

const handleReviewsClick = () => {

navigate(`/reviews/${product.id}`);

};

return (

<CardContainer>

<Card>

<ImageContainer>

<ProductImage src={product.image} alt={product.name} />

<HoverImage src={product.hoverImage} alt={`${product.name} Hover`} />

</ImageContainer>

<ProductName>{product.name}</ProductName>

<ProductPrice>${product.price.toFixed(2)}</ProductPrice>

<AddToCartButton onClick={onAddToCart}>Купить товар</AddToCartButton>

<ReviewsButton onClick={handleReviewsClick}>Отзывы</ReviewsButton>

</Card>

</CardContainer>

);

};

export default ProductCard;

**Корзина**  
Компонент Cart отвечает за управление корзиной. Он позволяет пользователю просматривать добавленные товары, изменять их количество, удалять товары и оформлять заказ.

Пример кода:

javascript

Copy

import React, { useState } from 'react';

import axios from 'axios';

import {

CartContainer,

CartItems,

CartItem,

CartTotal,

CartToggle,

CartIcon,

CloseButton,

ItemImage,

ItemDetails,

ItemName,

ItemPrice,

QuantityControls,

QuantityButton,

QuantityDisplay,

RemoveButton,

SizeSelector,

CheckoutButton,

Modal,

ModalContent,

ModalCloseButton,

Input,

SubmitButton

} from './Cart.styles';

import cartIcon from '../../static/cart-icon.png';

const Cart = ({ cartItems, onRemoveFromCart, onUpdateQuantity }) => {

const [isOpen, setIsOpen] = useState(false);

const [isCheckoutOpen, setIsCheckoutOpen] = useState(false);

const [customer, setCustomer] = useState({ name: '', phone: '', email: '' });

const total = cartItems.reduce((acc, item) => acc + item.price \* item.quantity, 0);

const toggleCart = () => setIsOpen(!isOpen);

const toggleCheckout = () => setIsCheckoutOpen(!isCheckoutOpen);

const handleQuantityChange = (item, change) => {

onUpdateQuantity(item.id, item.size, change);

};

const handleRemoveItem = (itemId, itemSize) => {

onRemoveFromCart(itemId, itemSize);

};

const handleInputChange = (e) => {

const { name, value } = e.target;

setCustomer({ ...customer, [name]: value });

};

const handleOrderSubmit = async () => {

try {

await axios.post('http://localhost:5000/order', { customer, cartItems });

alert('Заказ оформлен! Мы свяжемся с вами.');

setIsCheckoutOpen(false);

} catch (error) {

alert('Ошибка при оформлении заказа');

}

};

return (

<>

<CartToggle onClick={toggleCart}>

<CartIcon src={cartIcon} alt="Cart" />

</CartToggle>

<CartContainer isOpen={isOpen}>

<CloseButton onClick={toggleCart}>X</CloseButton>

<h2>Cart</h2>

<CartItems>

{cartItems.map((item) => (

<CartItem key={`${item.id}-${item.size}`}>

<ItemImage src={item.image} alt={item.name} />

<ItemDetails>

<ItemName>{item.name}</ItemName>

<ItemPrice>${(item.price \* item.quantity).toFixed(2)}</ItemPrice>

<SizeSelector>Размер: {item.size}</SizeSelector>

<QuantityControls>

<QuantityButton onClick={() => handleQuantityChange(item, -1)}>

-

</QuantityButton>

<QuantityDisplay>{item.quantity}</QuantityDisplay>

<QuantityButton onClick={() => handleQuantityChange(item, 1)}>

+

</QuantityButton>

</QuantityControls>

</ItemDetails>

<RemoveButton onClick={() => handleRemoveItem(item.id, item.size)}>

Удалить

</RemoveButton>

</CartItem>

))}

</CartItems>

<CartTotal>Total: ${total.toFixed(2)}</CartTotal>

<CheckoutButton onClick={toggleCheckout}>Оформить заказ</CheckoutButton>

</CartContainer>

{isCheckoutOpen && (

<Modal>

<ModalContent>

<ModalCloseButton onClick={toggleCheckout}>×</ModalCloseButton>

<h2>Введите данные</h2>

<Input

type="text"

name="name"

placeholder="Имя"

onChange={handleInputChange}

/>

<Input

type="tel"

name="phone"

placeholder="Телефон"

onChange={handleInputChange}

/>

<Input

type="email"

name="email"

placeholder="Email"

onChange={handleInputChange}

/>

<SubmitButton onClick={handleOrderSubmit}>Отправить заказ</SubmitButton>

</ModalContent>

</Modal>

)}

</>

);

};

export default Cart;

**3.4. Реализация фильтрации товаров**

Фильтрация товаров реализована в компоненте Filters. Пользователь может выбрать один или несколько размеров, и список товаров будет отфильтрован соответствующим образом.

Пример кода:

javascript

Copy

import React, { useState } from 'react';

import { FilterContainer, FilterHeader, FilterButton, ProductsCount } from './Filters.styles';

const Filters = ({ sizes, onFilter, productCount }) => {

const [selectedSizes, setSelectedSizes] = useState([]);

const handleFilterClick = (size) => {

if (selectedSizes.includes(size)) {

setSelectedSizes(selectedSizes.filter((s) => s !== size));

onFilter(selectedSizes.filter((s) => s !== size));

} else {

setSelectedSizes([...selectedSizes, size]);

onFilter([...selectedSizes, size]);

}

};

return (

<FilterContainer>

<FilterHeader>Размер:</FilterHeader>

{sizes.map((size) => (

<FilterButton

key={size}

onClick={() => handleFilterClick(size)}

isSelected={selectedSizes.includes(size)}

>

{size}

</FilterButton>

))}

<ProductsCount>{productCount} Товаров есть в наличии</ProductsCount>

</FilterContainer>

);

};

export default Filters;

**Глава 4. Реализация системы отзывов пользователей**

**4.1. Описание и принципы работы системы отзывов**

Система отзывов в интернет-магазине позволяет пользователям оставлять свои мнения и оценки о товарах. Это важный функционал, который помогает другим покупателям принимать решения о покупке, а также предоставляет обратную связь для магазина. Основные принципы работы системы отзывов:

* **Добавление отзывов:** Пользователь может оставить текстовый отзыв и оценить товар по шкале от 1 до 5 звезд.
* **Отображение отзывов:** Отзывы отображаются на странице товара, где пользователи могут ознакомиться с мнениями других покупателей.
* **Модерация отзывов:** Администратор может проверять и модерировать отзывы перед их публикацией.
* **Рейтинг товара:** На основе оценок пользователей рассчитывается средний рейтинг товара, который отображается на странице товара.

Система отзывов реализована с использованием React для фронтенда и Express для бэкенда. Отзывы хранятся в состоянии приложения, но в реальном проекте они должны сохраняться в базе данных.

**4.2. Создание интерфейса для добавления и отображения отзывов**

Интерфейс для добавления и отображения отзывов реализован в компоненте ReviewsPage. Этот компонент позволяет пользователям оставлять отзывы и просматривать отзывы других пользователей.

Пример кода:

javascript

Copy

import React, { useState } from 'react';

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

import { ReviewsPageContainer, ReviewList, ReviewItem, ReviewForm, SubmitButton, DeleteButton } from './ReviewsPage.styles';

const ReviewsPage = ({ reviews, onAddReview, onDeleteReview, products }) => {

  const { productId } = useParams();

  const [newReview, setNewReview] = useState({ text: '', rating: 5 });

  const [showForm, setShowForm] = useState(false);

  const productReviews = productId ? reviews[productId] || [] : [];

  const allReviews = !productId ? Object.keys(reviews).flatMap((id) => {

    return reviews[id].map(review => ({ ...review, productId: review.productId || id })); // Используем productId из отзыва или id как запасной вариант

  }) : [];

  const handleSubmit = (e) => {

    e.preventDefault();

    if (newReview.text.trim()) {

      onAddReview(productId || 'all', newReview);

      setNewReview({ text: '', rating: 5 });

      setShowForm(false);

    }

  };

  const handleDelete = (productId, reviewIndex) => {

    onDeleteReview(productId, reviewIndex);

  };

  const getProductName = (productId) => {

    const product = products.find(p => p.id === productId);

    return product ? product.name : 'Неизвестный товар';

  };

  return (

    <ReviewsPageContainer>

      <h1>{productId ? `Отзывы для товара ${productId}` : 'Все отзывы'}</h1>

      <SubmitButton onClick={() => setShowForm(!showForm)}>

        {showForm ? 'Закрыть форму' : 'Оставить отзыв'}

      </SubmitButton>

      {showForm && (

        <ReviewForm onSubmit={handleSubmit}>

          <textarea

            value={newReview.text}

            onChange={(e) => setNewReview({ ...newReview, text: e.target.value })}

            placeholder="Напишите ваш отзыв"

            required

          />

          <select

            value={newReview.rating}

            onChange={(e) => setNewReview({ ...newReview, rating: Number(e.target.value) })}

          >

            {[5, 4, 3, 2, 1].map((star) => (

              <option key={star} value={star}>{star} звезд</option>

            ))}

          </select>

          <SubmitButton type="submit">Отправить</SubmitButton>

        </ReviewForm>

      )}

      <ReviewList>

        {(productId ? productReviews : allReviews).length > 0 ? (

          (productId ? productReviews : allReviews).map((review, index) => (

            <ReviewItem key={index}>

              <h3>{getProductName(review.productId)}</h3> {/\* Используем productId из отзыва \*/}

              <p>{review.text}</p>

              <p>Рейтинг: {review.rating} звезд</p>

              <DeleteButton onClick={() => handleDelete(review.productId || productId || 'all', index)}>

                &#x2716;

              </DeleteButton>

            </ReviewItem>

          ))

        ) : (

          <p>Отзывов пока нет. Будьте первым!</p>

        )}

      </ReviewList>

    </ReviewsPageContainer>

  );

};

export default ReviewsPage;

**Теоретическая часть**  
Компонент ReviewsPage позволяет пользователям оставлять отзывы и просматривать отзывы других пользователей. Если пользователь находится на странице конкретного товара, отображаются только отзывы для этого товара. Если пользователь находится на общей странице отзывов, отображаются все отзывы.  
Форма для добавления отзывов включает текстовое поле для ввода текста отзыва и выпадающий список для выбора рейтинга. После отправки формы отзыв добавляется в состояние приложения и отображается в списке отзывов.

**Глава 5. Тестирование интернет-магазина**

**5.1. Введение в тестирование веб-приложений**

Тестирование веб-приложений является неотъемлемой частью процесса разработки, позволяя выявить ошибки, улучшить пользовательский опыт и повысить стабильность работы системы. Различают два основных подхода к тестированию: автоматизированное и ручное. В данной главе рассматривается ручное тестирование интернет-магазина, его основные методики, тест-кейсы и результаты тестирования.

**5.2. Цели и задачи тестирования**

Основными целями тестирования интернет-магазина являются:

* Выявление функциональных ошибок в работе приложения.
* Проверка корректности работы ключевых компонентов, таких как корзина, фильтрация товаров и система отзывов.
* Обеспечение совместимости с разными браузерами и устройствами.
* Улучшение пользовательского опыта (UX).
* Проверка устойчивости приложения к нагрузке.

**5.3. Методики ручного тестирования**

Ручное тестирование включает в себя несколько типов проверок:

* **Функциональное тестирование:** Проверка работы основных функций приложения.
* **Юзабилити-тестирование:** Анализ удобства использования.
* **Кросс-браузерное тестирование:** Проверка корректного отображения в разных браузерах.
* **Тестирование адаптивности:** Проверка корректного отображения на мобильных устройствах.
* **Тестирование безопасности:** Проверка уязвимостей, таких как SQL-инъекции, XSS-атаки и утечки данных.

**5.4. Тест-кейсы для интернет-магазина**

Тест-кейсы описывают сценарии проверки функциональности интернет-магазина.

**Тестирование фильтрации товаров**

* **Цель:** Проверить, работают ли фильтры по категориям, цене и рейтингу.
* **✅ Выбор фильтра:** Отображаются корректные товары.
* **❌ Несуществующий фильтр:** Ошибка или пустой список товаров.

**Тестирование системы отзывов**

* **Цель:** Проверить, может ли пользователь оставлять и просматривать отзывы.
* **✅ Оставить отзыв:** Отзыв отображается.
* **❌ Оставить пустой отзыв:** Ошибка.

**Тестирование корзины и оформления заказа**

* **Цель:** Проверить, можно ли добавлять товары в корзину и оформлять заказы.
* **✅ Добавление товара в корзину:** Товар отображается в корзине.
* **❌ Удаление товара:** Корзина обновляется.

**5.5. Результаты тестирования и выявленные ошибки**

После проведения тестирования составился отчет с найденными ошибками, их приоритетом и статусом исправления.

**5.6. Выводы по тестированию**

Подведение итогов тестирования: удалось выявить ряд неточностей в отображении страниц и корзины магазина. Также были убраны неточности в отображении размеров картинок товара.

### Глава 6. Оптимизация производительности интернет-магазина

**6.1. Оптимизация работы с состоянием в React**

В проекте использован хук useState для управления состоянием корзины, фильтров и модальных окон:

javascript

Copy

const [cartItems, setCartItems] = useState([]);

const [filteredProducts, setFilteredProducts] = useState(products);

const [selectedSizes, setSelectedSizes] = useState([]);

const [isCheckoutOpen, setIsCheckoutOpen] = useState(false);

Это позволяет минимизировать количество перерисовок компонентов и ускорить работу приложения.

Также при изменении состояния корзины обновление выполняется через map(), что предотвращает дублирование товаров:

javascript

Copy

const handleAddToCart = (product, size) => {

const existingItem = cartItems.find(

(item) => item.id === product.id && item.size === size

);

if (existingItem) {

setCartItems(cartItems.map((item) =>

item.id === product.id && item.size === size

? { ...item, quantity: item.quantity + 1 }

: item

));

} else {

setCartItems([...cartItems, { ...product, quantity: 1, size }]);

}

};

**6.2. Оптимизация фильтрации товаров**

Фильтрация реализована путем сравнения выбранных размеров с доступными у товаров:

javascript

Copy

const handleFilter = (selectedSizes) => {

if (selectedSizes.length === 0) {

setFilteredProducts(products);

} else {

setFilteredProducts(

products.filter((product) =>

selectedSizes.every((size) => product.sizes.includes(size))

)

);

}

};

Фильтрация выполняется на клиенте без дополнительных запросов к серверу, что ускоряет работу системы.

**6.3. Оптимизация взаимодействия с сервером**

Взаимодействие с сервером сведено к минимуму. Данные о товарах загружаются один раз и хранятся в состоянии.  
Запрос на оформление заказа отправляется только при подтверждении пользователем:

javascript

Copy

const handleOrderSubmit = async () => {

try {

await axios.post('http://localhost:5000/order', { customer, cartItems });

alert('Заказ оформлен! Мы свяжемся с вами.');

setIsCheckoutOpen(false);

} catch (error) {

alert('Ошибка при оформлении заказа');

}

};

**6.4. Улучшение пользовательского опыта (UX/UI)**

В проекте реализован hoverImage, который меняет изображение при наведении:

javascript

Copy

<ImageContainer>

<ProductImage src={product.image} alt={product.name} />

<HoverImage src={product.hoverImage} alt={`${product.name} Hover`} />

</ImageContainer>

Для улучшения UX модальные окна открываются только при необходимости:

javascript

Copy

const toggleCheckout = () => setIsCheckoutOpen(!isCheckoutOpen);

При успешном оформлении заказа отображается уведомление:

javascript

Copy

alert('Заказ оформлен! Мы свяжемся с вами.');

**Заключение**

В рамках данного дипломного проекта был разработан типовой интернет-магазин с использованием современных веб-технологий, таких как React, Express и styled-components. Основной акцент был сделан на реализации функционала фильтрации товаров и системы отзывов, которые являются ключевыми элементами для улучшения пользовательского опыта и повышения лояльности клиентов. В ходе работы были решены поставленные задачи, достигнуты цели исследования и получены значимые результаты.

**Основные результаты работы**

1. **Разработка архитектуры интернет-магазина:** Была спроектирована и реализована архитектура приложения, включающая фронтенд на React и бэкенд на Express. Это позволило создать масштабируемое и поддерживаемое решение, которое может быть легко расширено в будущем.
2. **Реализация функционала фильтрации товаров:** Разработан механизм фильтрации товаров по различным параметрам (категория, цена, размер, рейтинг), что значительно упрощает процесс поиска товаров для пользователей и повышает конверсию магазина.
3. **Создание системы отзывов:** Реализована система отзывов, которая позволяет пользователям оставлять свои мнения и оценки о товарах. Это способствует повышению доверия к магазину и помогает другим покупателям принимать обоснованные решения.
4. **Оптимизация производительности:** В проекте были применены различные методики оптимизации, такие как минимизация перерисовок компонентов, оптимизация фильтрации на клиенте и снижение нагрузки на сервер. Это позволило достичь высокой производительности приложения даже при большом количестве пользователей.
5. **Тестирование и отладка:** Проведено ручное тестирование всех ключевых функций интернет-магазина, выявлены и исправлены ошибки. Это обеспечило стабильную работу приложения и улучшило пользовательский опыт.
6. **Улучшение пользовательского интерфейса:** В проекте были реализованы интерактивные элементы, такие как hover-эффекты для изображений товаров, модальные окна для оформления заказов и уведомления об успешных действиях. Это сделало интерфейс более удобным и привлекательным для пользователей.

**Проблемы и ограничения при разработке**

В процессе разработки проекта были выявлены некоторые проблемы и ограничения:

1. **Ограниченность функционала бэкенда:** В проекте использовался минималистичный бэкенд на Express, который не поддерживает сложные бизнес-логики и масштабируемость. В реальном проекте рекомендуется использовать более мощные фреймворки, такие как Nest.js или Django.
2. **Отсутствие базы данных:** В проекте данные о товарах и отзывах хранятся в статическом массиве, что не подходит для реального интернет-магазина. Для полноценного решения необходимо интегрировать базу данных, такую как PostgreSQL или MongoDB.
3. **Ограниченность тестирования:** В проекте было проведено только ручное тестирование. Для более глубокой проверки функциональности и производительности рекомендуется использовать автоматизированные тесты, такие как Jest и Cypress.
4. **Безопасность:** В проекте не были реализованы все необходимые меры безопасности, такие как защита от XSS-атак и SQL-инъекций. В реальном проекте необходимо уделить больше внимания вопросам безопасности.

**Вклад проекта в развитие интернет-торговли**

Разработанный интернет-магазин демонстрирует возможности современных веб-технологий для создания удобных и функциональных решений в сфере электронной коммерции. Проект может служить основой для дальнейшего развития и масштабирования, а также вдохновить других разработчиков на создание подобных приложений.

**Основные преимущества проекта:**

* **Удобство для пользователей:** Реализованные функции фильтрации и отзывов значительно улучшают пользовательский опыт, что способствует повышению лояльности клиентов.
* **Гибкость и масштабируемость:** Использование React и Express позволяет легко расширять функционал магазина и адаптировать его под новые требования.
* **Оптимизация производительности:** Примененные методики оптимизации обеспечивают высокую производительность приложения даже при большом количестве пользователей.

**Рекомендации по дальнейшему улучшению проекта**

1. **Интеграция с базой данных:** Для хранения данных о товарах, заказах и отзывах рекомендуется использовать базу данных, такую как PostgreSQL или MongoDB. Это позволит улучшить масштабируемость и надежность приложения.
2. **Реализация автоматизированного тестирования:** Для более глубокой проверки функциональности и производительности рекомендуется внедрить автоматизированные тесты с использованием таких инструментов, как Jest и Cypress.
3. **Улучшение безопасности:** Необходимо реализовать дополнительные меры безопасности, такие как защита от XSS-атак, SQL-инъекций и утечек данных. Также рекомендуется использовать HTTPS для шифрования данных.
4. **Расширение функционала:** В будущем можно добавить такие функции, как система скидок, интеграция с платежными системами, поддержка нескольких языков и валют, а также персонализация интерфейса для каждого пользователя.
5. **Оптимизация для мобильных устройств:** Хотя в проекте был реализован адаптивный дизайн, рекомендуется провести дополнительные тесты на различных мобильных устройствах и внести улучшения для обеспечения максимального удобства для мобильных пользователей.

**Заключительные выводы**

Разработанный интернет-магазин является примером успешного применения современных веб-технологий для создания удобного и функционального решения в сфере электронной коммерции. Проект демонстрирует, как с помощью React, Express и других инструментов можно создать приложение, которое не только удовлетворяет потребности пользователей, но и обладает высокой производительностью и масштабируемостью.

В ходе работы были достигнуты все поставленные цели, решены задачи и получены значимые результаты. Проект может быть использован как основа для дальнейшего развития и масштабирования, а также как учебный пример для других разработчиков, изучающих современные технологии веб-разработки.

В заключение хочется отметить, что разработка интернет-магазина — это сложный, но увлекательный процесс, который требует не только технических знаний, но и понимания потребностей пользователей. Успешный проект — это не только функциональное приложение, но и удобный интерфейс, высокая производительность и безопасность. Надеюсь, что данный проект вдохновит других разработчиков на создание подобных решений и внесет свой вклад в развитие интернет-торговли.

**Список использованных источников**

1. React Documentation. (n.d.). Retrieved from https://reactjs.org/docs/getting-started.html
2. Express Documentation. (n.d.). Retrieved from https://expressjs.com/
3. Styled Components Documentation. (n.d.). Retrieved from https://styled-components.com/
4. Axios Documentation. (n.d.). Retrieved from https://axios-http.com/docs/intro
5. Redux Documentation. (n.d.). Retrieved from https://redux.js.org/
6. React Router Documentation. (n.d.). Retrieved from https://reactrouter.com/en/main
7. Фримен, А. (2017). React. Создание динамических веб-приложений. СПб.: Питер.
8. Макфарланд, Д. (2018). Большая книга CSS3. М.: Эксмо.
9. Симпсон, К. (2017). Вы не знаете JS: Область видимости и замыкания. М.: ДМК Пресс.
10. Статиста. (2023). Объём мирового рынка электронной коммерции. https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/
11. Ассоциация компаний интернет-торговли (АКИТ). (2022). Объём рынка интернет-торговли в России. https://www.akit.ru/research/results/
12. Рынок онлайн-торговли вырос с 1,7 трлн до 12,6 трлн рублей с начала пандемии // Ведомости/ https://www.vedomosti.ru/business/articles/2025/03/14/1097954-rinok-onlain-torgovli-viros

**Приложения (Коды, схемы, дополнительные материалы)**

* Исходный код проекта:

https://github.com/RenatAubakirov/shop