МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка интернет-магазина одежды

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-562.ВКР

Научный руководитель,
профессор кафедры СП, д.ф.-м.н.,
доцент
_____ Р.Ж. Алеев

Автор работы,
студент группы КЭ-403
_____ В.В. Кириенко

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-403
Кириенко Владиславу Викторовичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка интернет-магазина одежды.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**
3.1. CSS справочник. [Электронный ресурс] URL: http://htmlbook.ru/css (дата обращения: 04.03.2024 г.).
3.2. Основы HTML. [Электронный ресурс] URL: https://htmlacad-emy.ru/courses/297 (дата обращения: 04.04.2024 г.).
4. **Перечень подлежащих разработке вопросов**
4.1. Выполнить анализ предметной области.
4.2. Спроектировать веб-сайт.
4.3. Реализовать веб-сайт.
4.4. Провести тестирование.
5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.ф.-м.н., доцент Р.Ж. Алеев

Задание принял к исполнению В.В. Кириенко

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

React – это библиотека JavaScript для создания масштабируемых пользовательских интерфейсов. Она набирает популярность в среде разработки веб-приложений.

Разработка интернет-магазина с использованием React обеспечивает высокую производительность, гибкость и масштабируемость. Это важно для обработки больших объемов данных и ускорения взаимодействия с пользователями. Это также позволяет разработчикам быстро и легко вносить изменения, не нарушая структуру и производительность приложения.

Кроме того, React имеет большое и активное сообщество разработчиков, которое обеспечивает поддержку, помощь и обновления в случае возникновения ошибок или сбоев. Это позволяет разработчикам быстро и легко находить решения различных проблем.

Поэтому использование React в разработке интернет-магазинов на сегодняшний день является перспективной темой для исследовательского сообщества. Поиск новых технологий и идей позволяет создавать более легкие и быстрые веб-приложения и улучшать их функциональность.

Постановка задач

Целью данной работы является разработка интернет-магазина одежды.

Для реализации выбранной цели необходимо решить следующие задачи:

1) провести анализ предметной области;
2) провести проектирование веб-сайта;
3) реализовать веб-сайт;
4) протестировать веб-сайт.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 40 страниц, объем списка литературы – 15 источников.

В первой главе проводится анализ предметной области, обзор на существующие средства разработки, которые будут определять основные задачи для реализации.

Во второй главе определяются функциональные и нефункциональные требования к системе, производится разработка архитектуры приложения, демонстрируется диаграмма вариантов использования и диаграмма активности, а также производится разработка модели базы данных.

Третья глава описывает реализацию системы в соответствии с определенными требованиями и поставленными задачами. Создаются модели базы данных, демонстрируется создание возможности загрузки данных пользователей и данных корзины, разрабатывается управление состоянием, а также производится создание маршрутизации контекста.

В четвертой главе описывается тестирование приложения, демонстрируются результаты функционального тестирования и тестирования верстки. Основной целью такой работы будет проработка внешнего вида сайта. Данные результаты тестирования помогут нам убедится в корректной работе получившегося при разработке веб-сайта.

В приложении предоставлен код работы JWT аутентификации. Это означает, что в данном приложении используется JWT для аутентификации и авторизации пользователей. JWT является открытым стандартом для создания токенов доступа, которые используются для безопасной передачи информации между сторонами в виде JSON-объекта. Этот токен содержит информацию о пользователе, такую как его идентификатор, роли, права доступа и другие данные.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Актуальность разработки веб-сайтов

Разработка веб-сайтов в настоящее время имеет огромную актуальность и значимость, поскольку веб-страницы стали неотъемлемой частью бизнеса, коммуникации и информационного обмена.

В современном мире большинство людей и компаний ожидает найти информацию о продуктах или услугах в сети Интернет. Поэтому веб-сайт является основным маркетинговым инструментом, который позволяет представить свою компанию, товары и услуги потенциальным клиентам. Без надлежащего онлайн присутствия компания может потерять значительное количество потенциальных клиентов и упустить возможность продаж и развития.

Иметь качественный веб-сайт важно для поддержания конкурентной позиции на рынке. В настоящее время ожидается, что компании будут иметь веб-сайт, и отсутствие такой возможности может создать впечатление, что ваша компания устарела или не серьезно относится к своему бизнесу. Кроме того, веб-сайт дает вам возможность представить свою уникальность и отличие от конкурентов. Через дизайн, контент и функциональность вы можете передать свою уникальную ценность и привлечь больше клиентов.

Разработка веб-сайтов остается актуальной и востребованной деятельностью, поскольку онлайн присутствие, доступность, конкурентоспособность и возможность коммуникации являются важными факторами для успешного бизнеса. Веб-сайт помогает представить компанию и ее продукты, привлечь новых клиентов и улучшить отношения с существующими.

Развитие интернет-магазинов одежды является актуальным направлением в свете постоянно растущего спроса на онлайн-шопинг. С учетом глобализации и цифровизации экономики, а также изменения потребительских привычек, особенно после пандемии, люди все чаще предпочитают покупать одежду в интернете. Это обусловлено удобством, возможностью срав-

нивать цены и ассортимент без необходимости посещения физических магазинов. Кроме того, разработка интернет-магазинов позволяет предпринимателям сократить расходы на аренду и обслуживание торговых площадей, а также оптимизировать логистику и управление запасами. Важным аспектом является и возможность использования данных о покупательских предпочтениях для персонализации предложений и улучшения пользовательского опыта. Таким образом, инвестиции в создание и развитие интернет-магазинов одежды могут стать ключевым фактором успеха в современной розничной торговле.

1.2. Анализ аналогичных веб-сайтов

В этом подразделе будет проведен анализ имеющихся решений в данной области и представлены результаты в виде таблицы. Будут рассмотрены различные аспекты и особенности существующих решений, а также их преимущества и недостатки. Таблица будет содержать сведения о каждом решении, его основных характеристиках и результатах сравнительного анализа. Это позволит получить полное представление о текущем состоянии рынка и выбрать наиболее подходящее решение для задачи, руководствуясь конкретными потребностями и требованиями. В конце анализа будут предоставлены возможности реализации сайта на различных платформах.

«Louis Vuitton»

Один из самых популярных магазинов одежды является «Louis Vuitton». На главной странице сайта можно увидеть меню, предназначенное для навигации по сайту, корзину с уже выбранными позициями, возможность связаться напрямую с менеджерами, а также возможность зайти в свою учетную запись. Главная страница сайта приведена на рисунке 1.

Сайт занимается продажей премиального качества одежды имеет современный дизайн сайта, который в свою очередь показывает статус бренда.

Верхняя панель меню находится в шапке веб-страницы и содержит ссылки на разделы меню, категории товаров, возможность связаться с тех. поддержкой, возможность просмотра магазинов в любой стране.

Оформление заказа происходит в несколько стандартных шагов. Пользователь выбирает желаем товары, указывает способ оплаты товара, утверждает способ получения товара. «Louis Vuitton» предоставляет большой список товаров, с удобными опциями оплаты товаров в пару кликов, и удобным способом получения заказа.

Премиальный бренд «Louis Vuitton» полностью адаптирован для мобильных устройств и создан с учетом современных требований к работе с планшетами и смартфонами. На сайте отсутствуют всплывающие окна с навязанными акциями, которые могли бы раздражать покупателей. Навигация по веб-сайту простая и не требует усилий в поиске подходящего товара, а элементы контента и меню оптимизированы для различных экранов и разрешений.

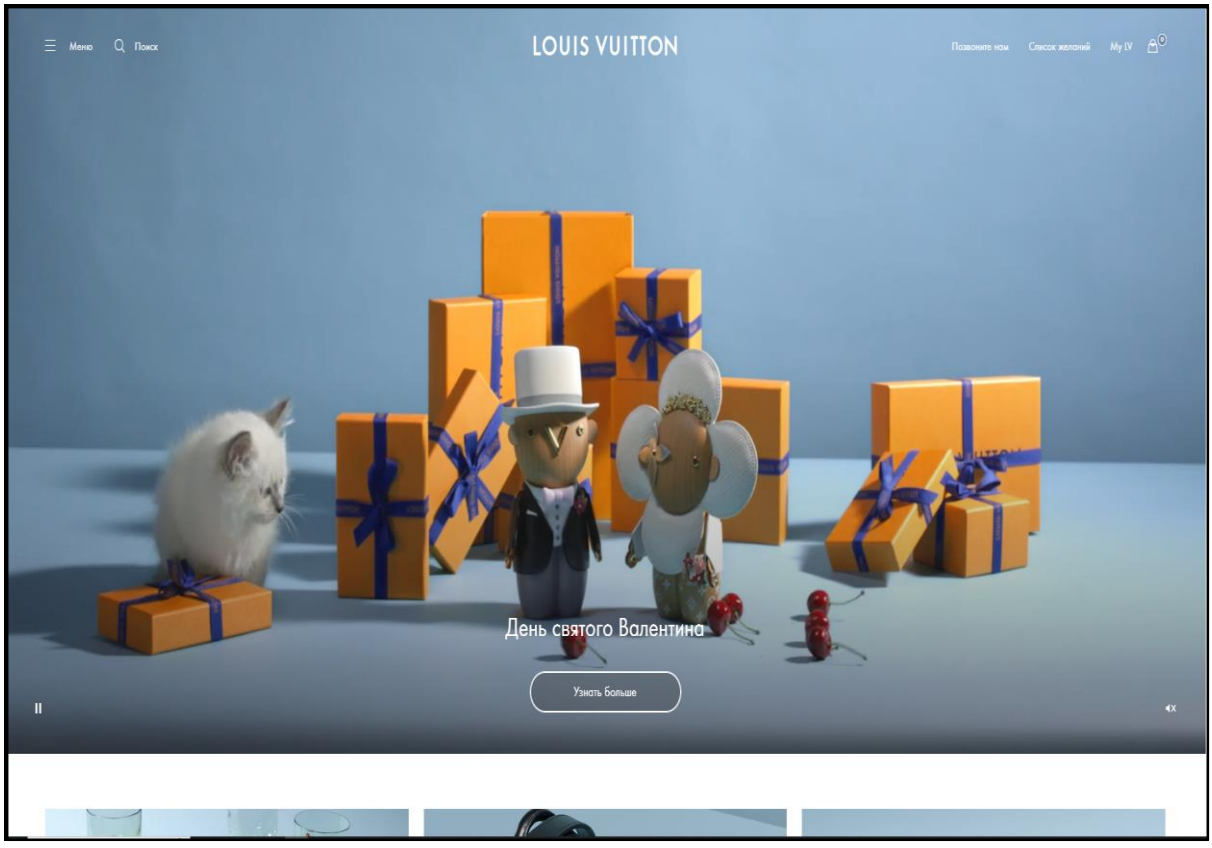

Рисунок 1 – Главная страница сайта «Louis Vuitton»

«La Redoute»

«La Redoute» – это известный французский магазин стильной одежды и даже мебели, который известен на весь мир. Бренд имеет свою популярность за счет многочисленных коллекций. На веб-странице можно увидеть категории товаров, которые упрощают поиск интересующих пользователя продукцию. Возможность связаться с технической поддержкой, помогает покупателям решать проблемы, которые появились в процессе покупок или при нахождении на сайте. Желание помочь клиенту вызывает у пользователя чувство доверия, которое провоцирует покупателя порекомендовать этот бренд знакомым. Главная страница сайта представлена на рисунке 2.

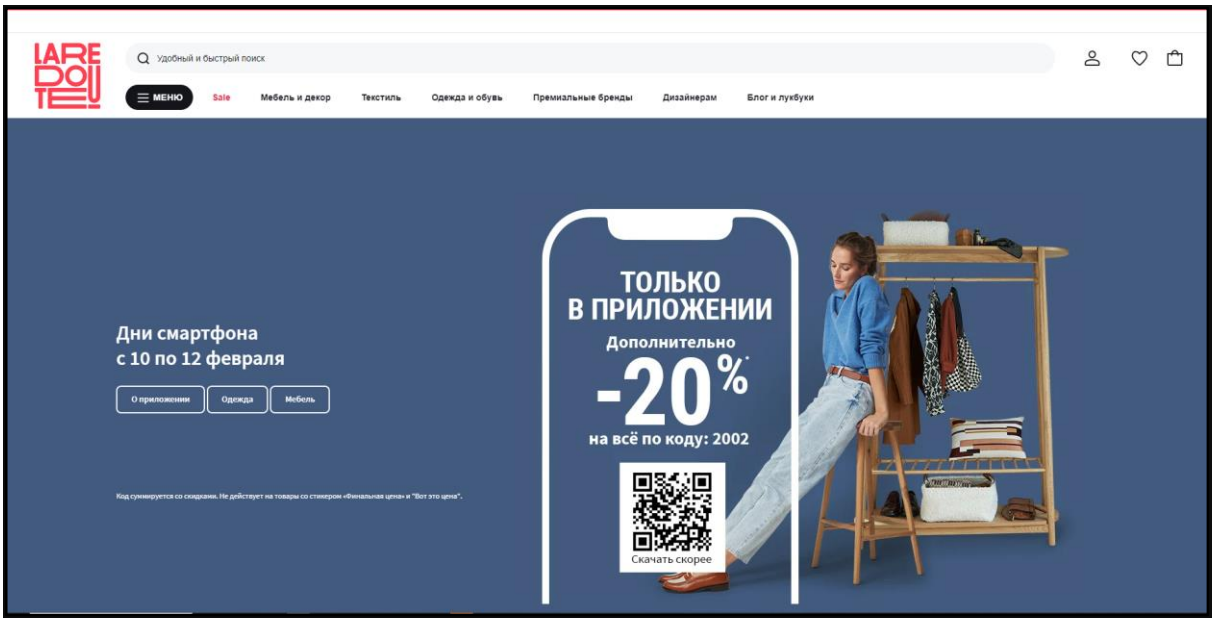

Рисунок 2 – Главная страница сайта «La Redoute»

Данный веб-сайт имеет простой и спокойный визуальный дизайн: по центру расположена реклама мобильного приложения с возможностью получения скидки, в верхней части экрана предоставлен список разделов поиска товаров «Sale», «Мебель и декор», «Текстиль», «Одежда и обувь», «Премиальные бренды», «Дизайнерам», «Блог и лукбуки».

Достаточно простой сайт для потребителя, но очень много встроенной рекламы акций на товары, что может отрицательно повлиять на желание пользователя искать необходимый ему товар.

«La Redoute» адаптирован для мобильных устройств любого вида. Навигация по веб-страницам удобна: меню расположено в центральной части экрана, что сокращает время пользователя на поиск необходимого ему товара.

«Industry Standard»

«Industry Standard» – известный интернет-магазин, который предлагает широкий ассортимент товаров. Визуальный дизайн сайта привлекателен и современен. Он представляет пользователю четкую навигацию по разделам и категориям товаров, используя привлекательные фотографии и информационные блоки. Главная страница сайта представлена на рисунке 3.

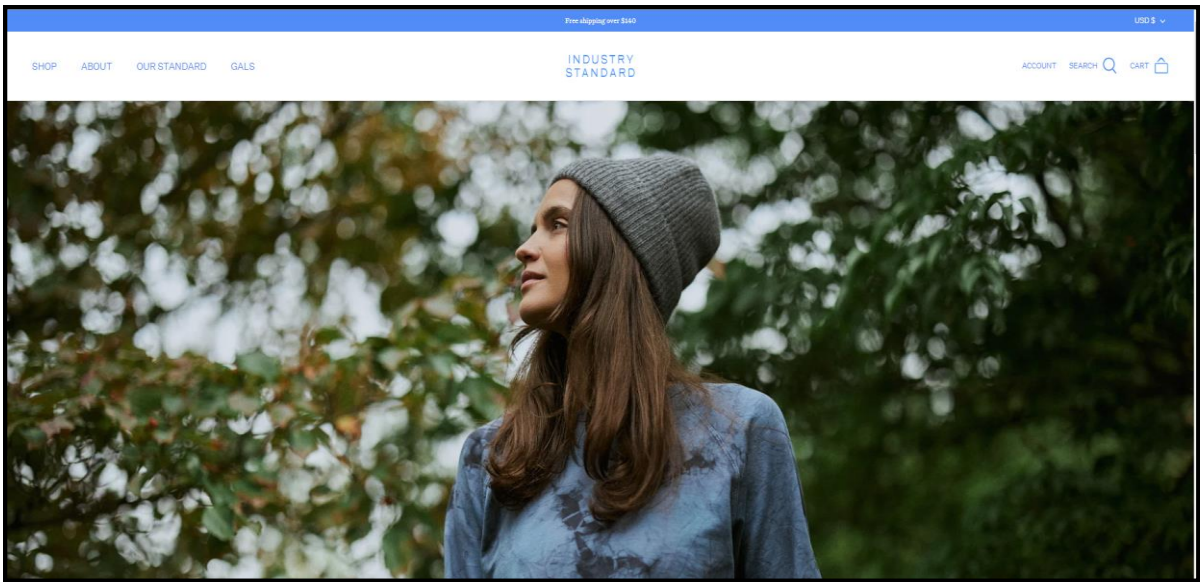

Рисунок 3 – Главная страница сайта «Industry Standard»

Верхняя панель содержит логотип, поиск по сайту, иконки социальных сетей, позволяющие пользователям связаться с магазином через популярные платформы. Наличие основных разделов («Магазин», «О нас», «Наши стандарты», «Товары», «Корзина») позволяют пользователям быстро и удобно найти нужную информацию и перейти к нужным разделам.

При клике на каждую категорию отображается страница с фотографиями и цене товара. Это позволяет пользователям быстро ознакомиться с ассортиментом и выбрать интересующий их товар. Пользователи могут добавлять товары в корзину и оформлять заказы в несколько кликов. Доступность

подробного описания и характеристик товаров, а также фотографий и отзывов позволяет пользователям принять информированное решение о покупке.

Выбор средств реализации интернет-магазина одежды

Выбранные сайты «Louis Vuitton», «La Redoute», «Industry Standard» были созданы с помощью различных языков программирования, таких как CSS, JavaScript и многих других технологий. Они имеют более сложную структуру и функциональность, поскольку они используют несколько языков программирования и технологий для интеграции платежных систем, управления заказами, создания профилей пользователей, аналитики и маркетинговых инструментов. В свою очередь, ваш сайт на React имеет более простую функциональность и инфраструктуру, не требующую использования других языков программирования и технологий.

Анализ аналогичных сайтов показал, что React позволяет создать функциональный и простой сайт для пользователя. Ниже приведены основные особенности библиотеки React.

1. Компонентный подход. React предлагает создавать веб-сайты из множества небольших и переиспользуемых компонентов, что делает код более организованным и легко поддерживаемым.

2. Виртуальный DOM. React использует виртуальный DOM для эффективного обновления только измененных частей веб-сайта вместо перерисовки всего содержимого. Это повышает производительность быстро обновлять пользовательский интерфейс.

3. Однонаправленный поток данных. React используется однонаправленный поток данных, что означает, что данные всегда передаются от родительского компонента к дочернему. Это делает код более предсказуемым и облегчает отладку и обнаружение ошибок.

4. JXS синтаксис. React использует JSX синтаксис для определения компонентов и их внешнего вида. JSX позволяет объединять HTML и JavaScript в одном файле, что упрощает разработку и обеспечивает более

декларативный подход к созданию пользовательского интерфейса.

5. Широкое сообщество и экосистема. React имеет огромное сообщество разработчиков и обширную экосистему с различными инструментами и библиотеками, которые помогают улучшить и расширить функциональность React. Это делает React одним их самых популярных выборов для создания веб-сайтов.

1.3. Обзор средств разработки

Разрабатываемый веб-сайт с использованием React будет обладать большими преимуществами благодаря высокой производительности и гибкости. Сообщество разработчиков в любой момент сможет обеспечить поддержку твоего проекта.

Visual Studio Code

Интегрированная среда разработки (IDE) от компании Microsoft для создания приложений, включая приложения для Windows, веб-сайты, веб-приложения, мобильные приложения и игры. Это одно из самых популярных и мощных средств разработки, предлагающее обширный набор инструментов и функциональности для упрощения и ускорения процесса создания программного обеспечения.

Visual Studio Code также предоставляет широкий выбор расширений, позволяющих дополнительно настраивать и расширять функциональность среды. Это позволяет адаптировать Visual Studio Code под конкретные потребности разработчиков и проектов.

React

React – это библиотека JavaScript, которая используется для создания пользовательских интерфейсов веб-приложений. Он позволяет разработчикам создавать масштабируемые и эффективные по производительности компоненты пользовательского интерфейса.

React использует подход «однонаправленного потока данных» и виртуальное дерево DOM для эффективного обновления только измененных

компонентов. Это позволяет создавать динамические, интерактивные и отзывчивые веб-приложения.

Таким образом, React очень полезен для создания профессиональных сайтов и веб-приложений. Он имеет динамический и интерактивный пользовательский интерфейс, способный конкурировать с большим количеством аналогов и оставаться каждый раз в списках лучших языков программирования в данной области.

Инструменты и технологии

При разработке интернет-магазина одежды предоставлен огромный список веб-технологий: JavaScript, React, CSS, HTML. React обладает рядом преимуществ:

1) высокая производительность;
2) односторонняя данная привязка;
3) поддерживает функциональное программирование;
4) имеет открытый исходный код.

Вывод по первой главе

После анализа наиболее эффективных решений для создания веб-сайта, необходимо было выбрать оптимальный инструмент, учитывая множество доступных программных средств. Для определения лучшего решения был проведен анализ различных вариантов и сравнений, в результате чего была выбрана библиотека React для создания сайта магазина. Конечным результатом данного выбора стало создание оптимальной платформы для разработки веб-сайта, обладающего высокой производительностью.

2. ПРОЕКТИРОВАНИЕ

В данном разделе продемонстрировано планирование разрабатываемого веб-сайта, выявлены главные функциональные и нефункциональные требования, а также сделана диаграмма вариантов использования.

2.1. Требования к разработке системы

Функциональные требования

Функциональные требования определяют – спецификации того, что должна делать система или программное обеспечение. Они определяют функции, операции, возможности и поведение системы, а также ее интерфейсы с пользователем и другими системами. Функциональные требования описывают то, что система должна предоставлять или достигать в пределах своей функциональности. Для выполнения приложения выявлены следующие функциональные требования:

- добавление товара в корзину;
- просмотр каталога товаров;
- поиск по каталогу товаров;
- оплата товара;
- очистить корзину.

Нефункциональные требования

Нефункциональные требования – это требования, которые описывают атрибуты системы, не связанные непосредственно с ее функциональностью, но влияющие на ее характеристики и качество. При разработке веб-сайта такие требования определяют параметры, ограничения и критерии, касающиеся его производительности, безопасности, надежности, масштабируемости и других аспектов. Вот несколько нефункциональных требований к разрабатываемому веб-сайту.

- сайт должен иметь адаптивный дизайн;

– сайт должен иметь быструю скорость загрузки страниц, быстро загружаться, а также практически мгновенно открываться на мобильных устройствах.

2.2. Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой графический инструмент, применяемый в процессе объектно-ориентированного анализа и разработки программных систем. Она служит для визуального представления взаимосвязей между различными участниками системы (актерами) и сценариями их взаимодействия с системой.

Варианты использования описывают конкретные сценарии взаимодействия пользователя с системой, позволяя лучше понять требования пользователей и обеспечивать основу для тестирования системы. Опираясь на основания функциональных требований к этой системе, была создана диаграмма вариантов использования сайта (рисунок 4).

Краткое описание вариантов использования для актера авторизированный пользователь приведено ниже:

1) поиск товара – авторизированый пользователь может найти интересующий его товар в строке поиска;

2) просмотр каталога – авторизированый пользователь может ознакомиться с интересующей его продукцией;

3) отфильтровать товары по категории – авторизированый пользователь может выбрать необходимые категории товаров;

4) сортировать товары по цене – авторизированый пользователь может отсорировать товары по цене;

5) добавить товар в корзину – авторизированый пользователь добавляет выбранный товар в корзину;

6) просмотреть раздел корзины – авторизированый пользователь может проверить корректность всех выбранных им товаров;

7) удалить товар из корзины – авторизированый пользователь может удалить ранее добавлеенный товар из корзины;

8) увеличить/уменьшить количество товара в корзине – авторизированый пользователь может изменить количество выбранного товара в корзине;

9) выйти из аккаунта – авторизированый пользователь может выйти из учетной записи сайта.

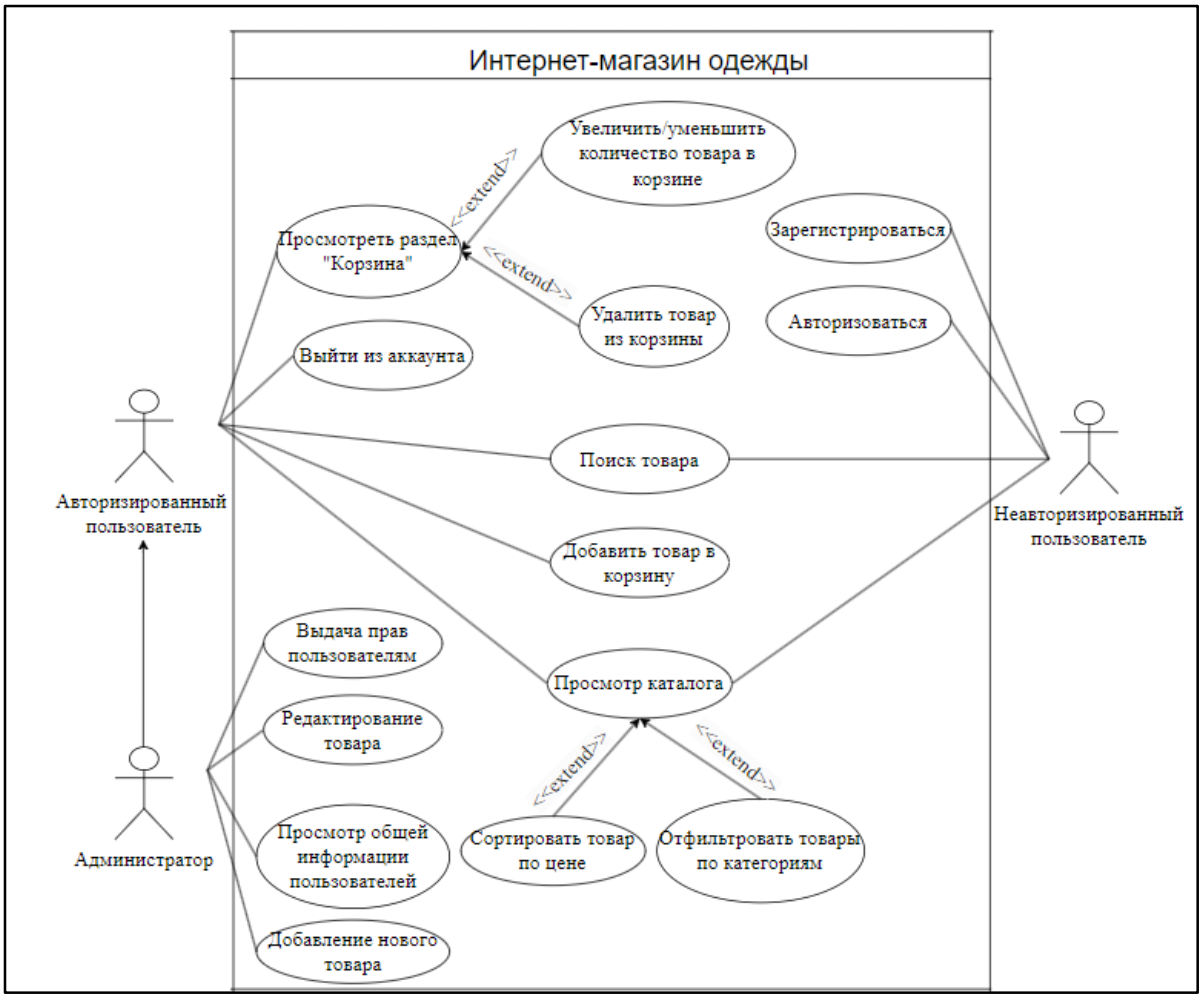

Рисунок 4 – Диаграмма вариантов использования сайта

Актер администратор наследует все классы от актера авторизированный пользователь. Данное наследование называется «общее наследование» и необходимо для доступа ко всем классам интересующего нам актера. Краткое описание вариантов использования для актера администратор приведено ниже:

1) добавление нового товара – администратор может добавлять новый товар на сайт;

2) редактирование товара – администратор может полностью изменить информацию о имеющимся товара;

3) выдача прав пользователям – администратор может изменить роль у зарегистрировавшегося пользователя;

4) просмотр общей информации пользователя – администратор может просмотреть.

Краткое описание вариантов использования для актера неавторизированный пользователь приведено ниже:

1) зарегистрироваться – неавторизированный пользователь может пройти регистрацию на сайте;

2) авторизоваться – неавторизированный пользователь имеет возможность после регистрации войти в свой аккаунт;

3) поиск товара – неавторизированный пользователь имеет возможность искать интересующий его товар на сайте;

4) добавить товар в корзину – неавторизированный пользователь может добавить в корзину понравившийся ему товар;

5) просмотр каталога – неавторизированный пользователь может просматривать каталог товаров на сайте;

6) сортировать товары по цене – неавторизированный пользователь при необходимости может отсортировать товары по цене;

7) отфильтровать товары по категориям – неавторизированный пользователь может выбрать в каталоге интересующие его категории.

2.3. Архитектура интернет-магазина

В работе будут использоваться такие компоненты как React, Redux, Express.js и MongoDB. Приложение будет представлять собой интернет-магазин, позволяющий пользователям просматривать товары, добавлять их в корзину и оформлять заказы. Основные составляющие архитектуры будут

включать фронтенд на React, управление состоянием через Redux, серверную часть на Express.js и базу данных MongoDB. Особое внимание уделяется аутентификации пользователей с использованием JWT (JSON Web Token). Архитектура разрабатываемого интернет магазина представлена на рисунке 5.

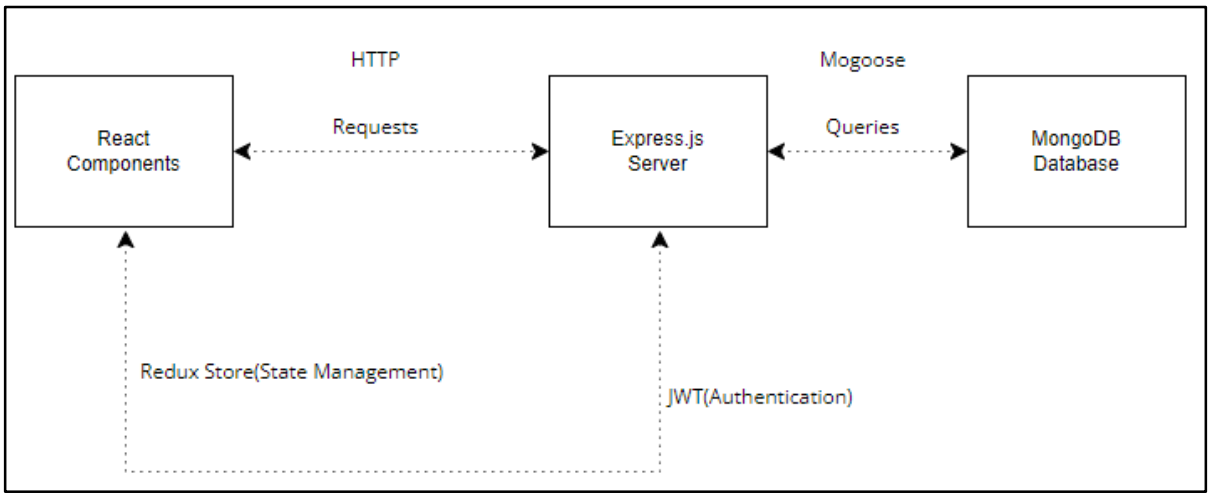

Рисунок 5 – Архитектура интернет-магазина

React components

Фронтенд приложения реализован с использованием библиотеки React, которая обеспечивает создание динамичного и интерактивного пользовательского интерфейса. Основные компоненты, такие как заголовок, подвал, список продуктов и корзина, обрабатывают взаимодействие пользователя с приложением, отправляют HTTP-запросы на сервер и отображают полученные данные. React позволяет быстро и эффективно обновлять интерфейс в ответ на действия пользователя, обеспечивая плавное и интуитивно понятное взаимодействие.

Redux Store

Redux используется для управления состоянием приложения, предоставляя централизованное хранилище для данных пользователя и состояния корзины. Это упрощает и оптимизирует обмен данными между компонентами, позволяя избежать проблем с синхронизацией состояния и обеспечивая предсказуемость поведения приложения. С помощью Redux все данные

хранятся в едином месте, что упрощает отладку и тестирование.

JWT (JSON Web Token)

Аутентификация в приложении будет реализована с использованием JWT. При входе в систему пользователю будет выдаваться токен, который будет хранится в куки и отправляться с каждым последующим запросом для валидации пользователя. Это обеспечит безопасность данных пользователя и позволит серверу удостоверяться в том, что запросы поступают от авторизованных пользователей. JWT обеспечивает удобство и безопасность, поскольку токены можно легко проверять на стороне сервера без необходимости хранения сессий. В разрабатываемой программе будет используется JSON Web Token аутентификация, данный токен необходим для проверки подлинности пользователей при входе в систему. Реализация аутентификации будет представлена в приложении в листинге 1 и 2.

MongoDB

База данных MongoDB используется для хранения информации о пользователях и продуктах. MongoDB обеспечивает гибкость в управлении данными и их структуре, что особенно полезно для приложений, которые могут потребовать быстрых изменений в модели данных. Для взаимодействия с базой данных используется библиотека Mongoose, которая предоставляет удобный интерфейс для работы с MongoDB, позволяя легко создавать, читать, обновлять и удалять данные.

2.4. Диаграмма активности

Диаграмма активности представляет собой один из видов диаграмм в унифицированном языке моделирования (UML), который используется для визуального представления и описания потока управления в системе или бизнес-процессе. Данная диаграмма позволяет наглядно отобразить последовательность действий, решений и потоков в рамках некоторого процесса. Так же диаграмма активности может использоваться для описания внутренней логики работы программного компонента или всего приложения.

Одной из ключевых задач сайта предоставить возможность добавления товара в корзину. Диаграмма активности отлично демонстрирует данный алгоритм на рисунке 6.

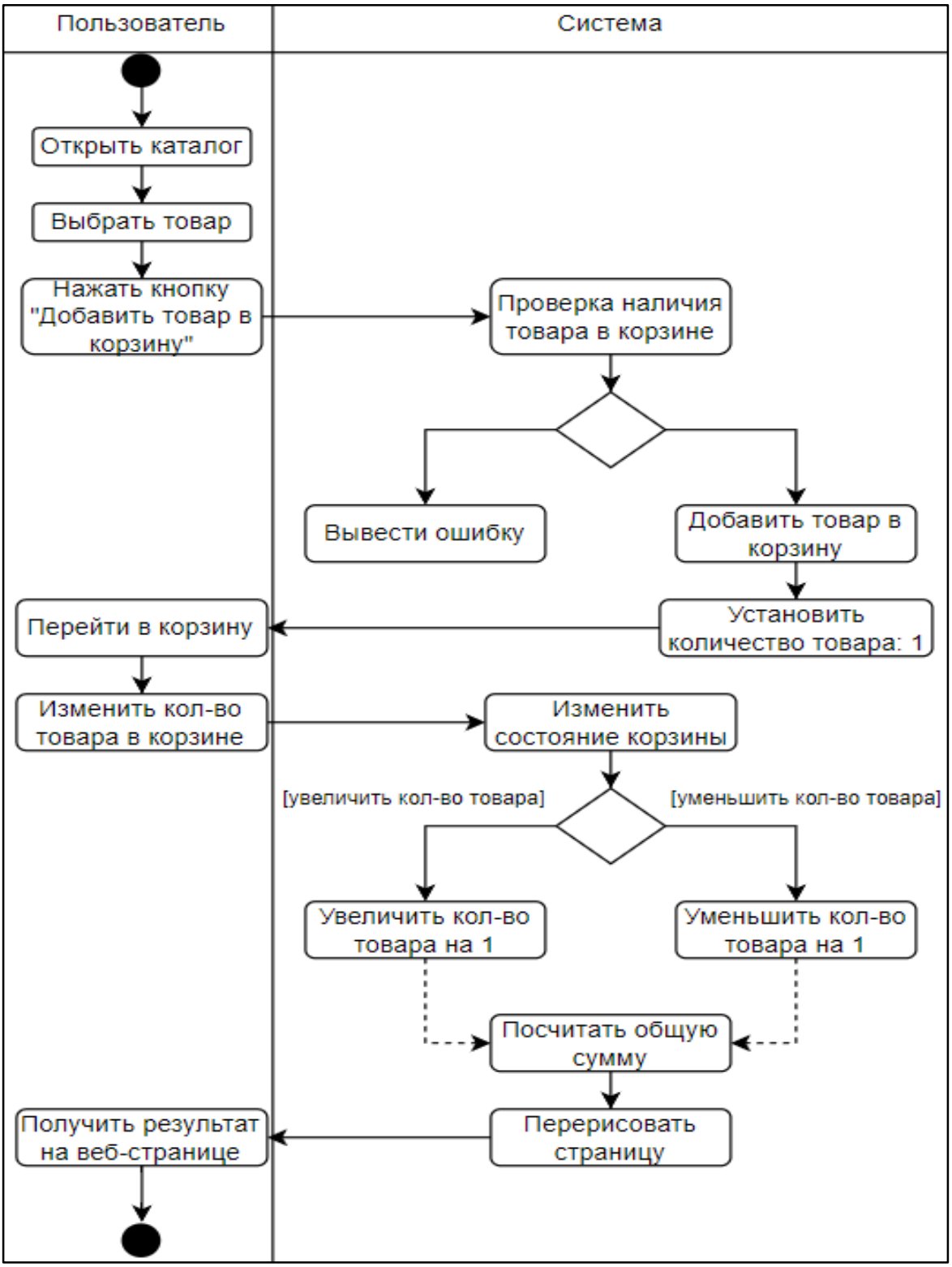

Рисунок 6 – Диаграмма активности

В момент, когда пользователь нашел интересующий его товар, он нажимает на кнопку добавления товара в корзину. В этот момент, система проверяет находится ли объект с соответствующим идентификаторов в массиве, который хранит все добавленные в корзину товары. Если выбранный объект присутствует, то поле количества товара увеличивается на 1. В ином случае, при выборе нового объекта, его количество устанавливается равным единице.

На следующем этапе подсчитывается общая сумма товаров в корзине. После обновления страницы пользователь может увидеть окончательную сумма за все товары в корзине.

При желании пользователь может редактировать содержимое корзины, изменять количество товаров или удалять их. Вся эта информация динамически обновляется на веб-странице, обеспечивая максимальное удобство и простоту использования интерфейса. Кроме того, пользователь может продолжить покупки, добавляя другие товары в корзину. Система будет также изменять состояние корзины, учитывая новые решения пользователя по приобретению новых товаров, а также изменения текущего выбора товаров.

2.5. Проектирование базы данных

Для этого проекта была выбрана MongoDB, которая является документоориентированной системой управления базами данных. Она предоставляет гибкость в хранении данных без строгой схемы, что позволяет быстро разрабатывать и изменять требования.

Была спроектирована базы данных для поддержки функциональности интернет-магазина одежды, которая позволяет просматривать, добавлять и управлять товарами в корзине покупок, а также осуществлять регистрацию пользователей на сайте (рисунок 7).

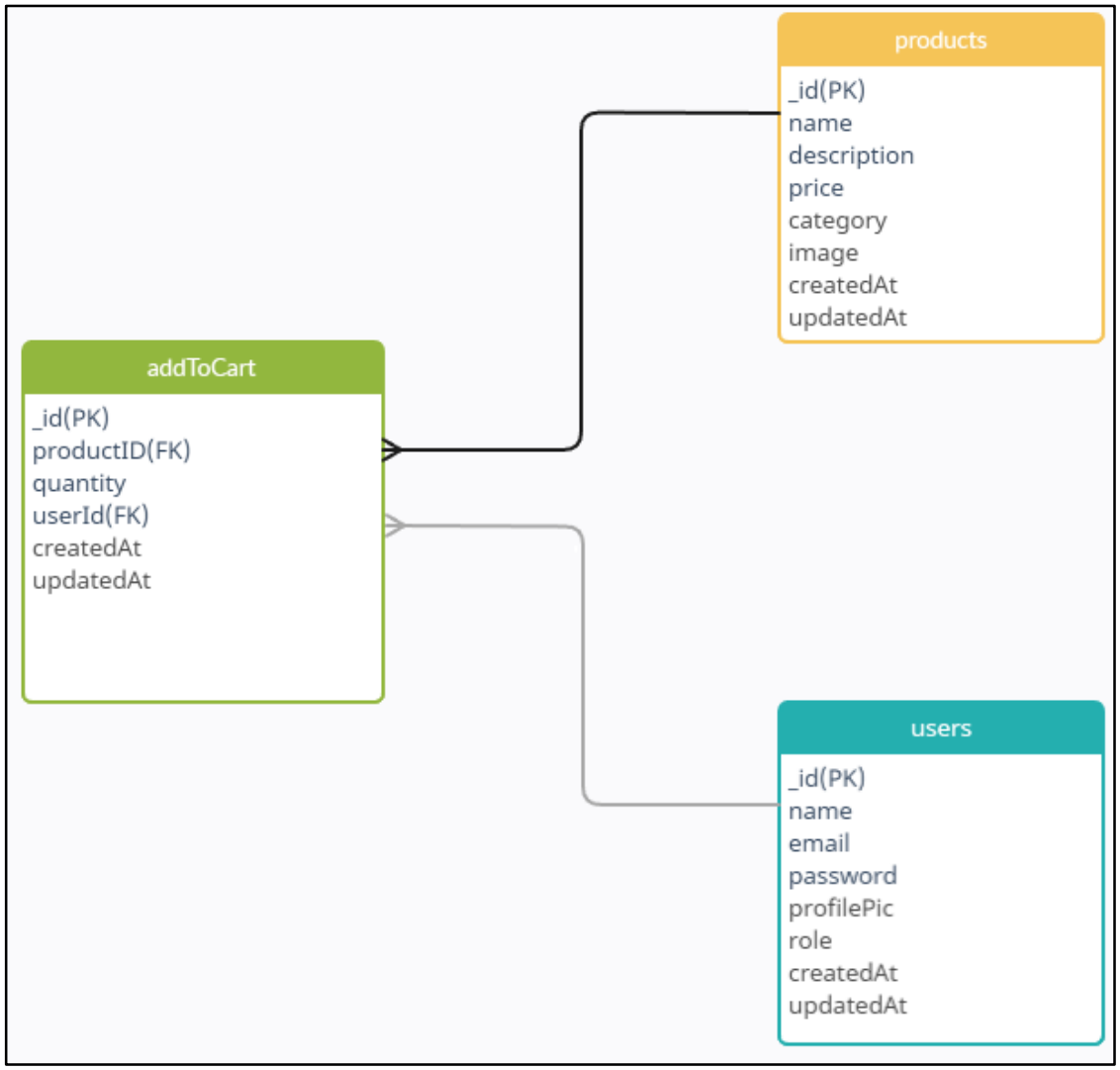

Рисунок 7 – Диаграмма активности

Структура базы данных

Таблица «users» содержит такие поля как:

1) id – уникальный идентификатор пользователя (primary key);
2) name – имя пользователя;
3) emal – адрес электронной почты пользователя (уникальный);
4) password – хэш пароля пользователя;
5) profilePic – путь к изображению профиля пользователя;
6) role – роль пользователя в системе (например, «ADMIN», «GENERAL»);
7) createdAt – дата и время создания записи;
8) updatedAt –дата и время последнего обновления записи.

Таблица «products» содержит такие поля как:

1) id – уникальный идентификатор товара (primary key);
2) name – название товара;
3) description – описание товара;
4) price – цена товара;
5) category – категория товара;
6) image – путь к изображению товара;
7) createdAt – дата и время создания записи;
8) updateAt – дата и время последнего обновления записи.

Таблица «addToCart» содержит такие поля как:

1) id – уникальный идентификатор записи корзины (primary key);
2) userId – идентификатор пользователя, который добавил товар в корзину (foreign key);
3) productId – идентификатор товара, добавленного в корзину (foreign key);
4) quantity – количество товара в корзине;
5) createdAt – дата и время создания записи;
6) updatedAt – дата и время последнего обновления записи.

Связи между таблицами

Таблица «users» и «addToCart» связаны отношениями один ко многим, так как каждый пользователь может иметь несколько записей в корзине.

Таблица «products» и «addToCart» также связаны отношениями один ко многим, так как каждый продукт может быть добавлен в корзину несколькими пользователями.

2.6. Проектирование дизайна веб-сайта

Сайт должен иметь хороший дизайн. Цвета сайта должны быть приятными и не вызывать раздражение. Все компоненты сайта должны расположены так, чтобы пользователь мог их с легкостью найти. Первое впечат-

ление потенциального покупателя очень важно, поэтому важно иметь стандартную верхнюю панель в дизайне проектируемого сайта.

Основные цвета сайта имеют теплую цветовую гамму, чтобы создать легкость в глазах пользователя. Правильное сочетание оттенков позволяет увидеть всю необходимую информацию с первого взгляда на макете сайта (рисунок 8).

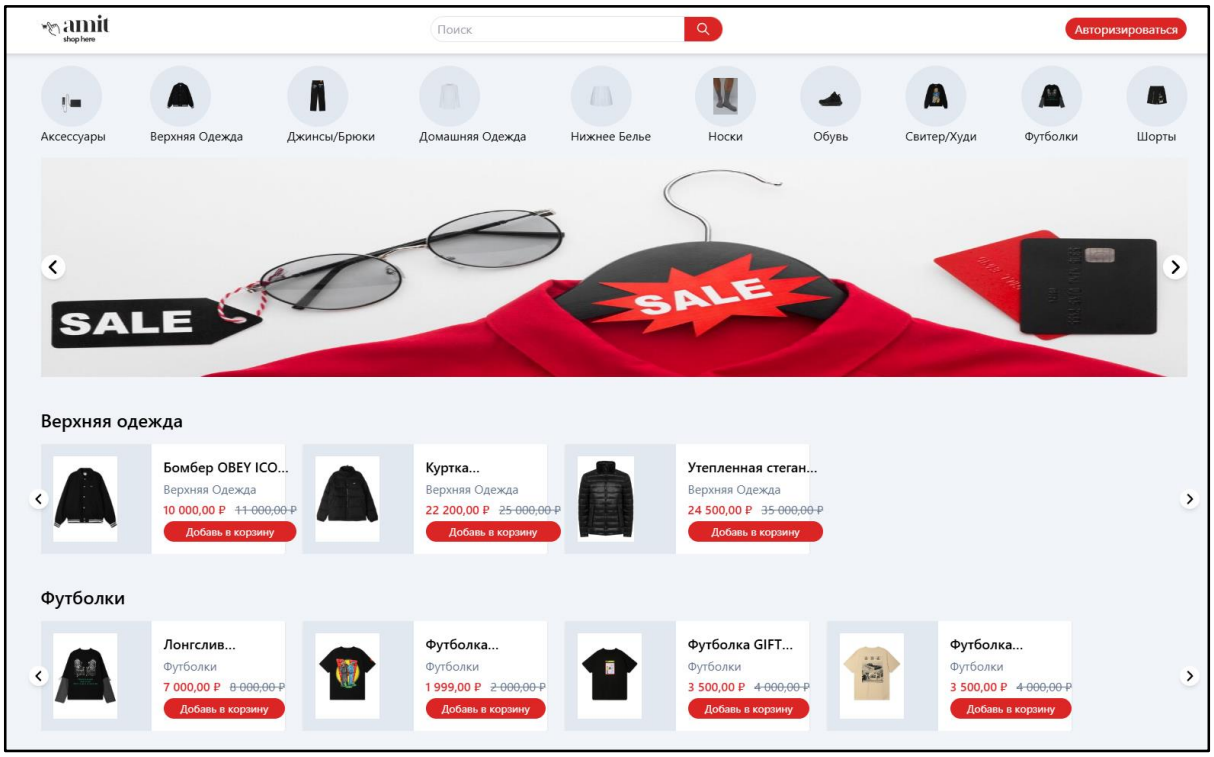

Рисунок 8 – Макет главной страницы сайта

Проектирование страницы товара также имеет важную роль для первого впечатления пользователя. Необходимо создать приятный дизайн, который обеспечит удобство для потенциального покупателя.

Важной составляющей является качественные изображения продукции с разных ракурсов, это нужно для того, чтобы покупатель получил максимально полное представление о товаре. Создание дизайна страницы товара, поможет увеличить конверсию и удовлетворенность потребностей пользователей. Также важно, чтобы изображения были четкими и выделяли особенности товара (рисунок 9).

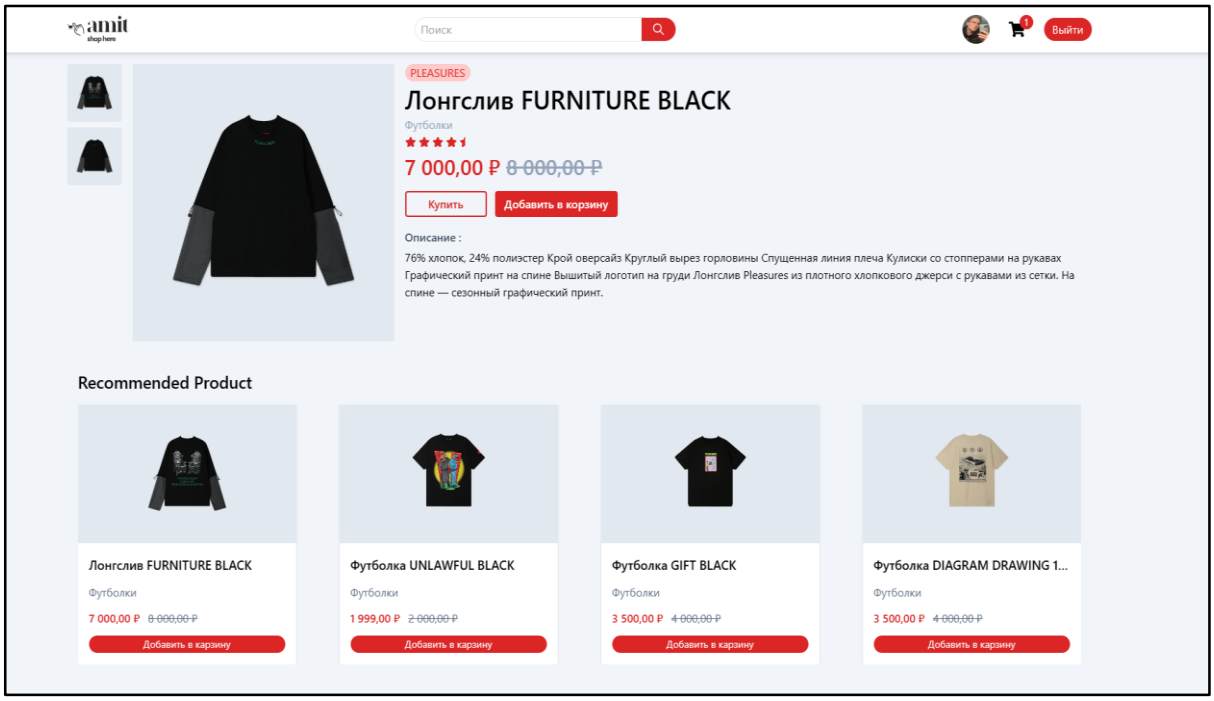

Рисунок 9 – Страница товара

Вывод по второй главе

В данной главе было проведено тщательное планирование и проектирование разрабатываемого веб-сайта интернет-магазина одежды. Была продемонстрирована архитектура разрабатываемого приложения.

Была создана диаграмма вариантов использования, которая наглядно демонстрирует взаимодействие различных актеров (авторизированные пользователи, администраторы и неавторизированные пользователи) с системой.

Общая структура веб-сайта была представлена в виде схемы, на которой показаны все страницы и разделы интернет-магазина. Диаграмма активности была разработана для ключевого процесса добавления товара в корзину, что позволяет наглядно продемонстрировать алгоритм работы этого процесса.

3. РЕАЛИЗАЦИЯ

Создание проекта

Для создания React-приложения необходимо прописать команду `create-react-app`. Данная команда создает проект со всеми встроенными базовыми модулями. Необходимым шагом при старте разработки веб-сайта будут являться пакеты, которые несут ответственность за данные в приложении, такие как React Redux и Redux Toolkit. Для их установки требуется прописать в консоли соответствующие команды, загрузка протекает через пакетный менеджер, благодаря команде `npm install`.

Создание моделей базы данных

Модель товара

В процессе разработки была определена модель товаров «productModel.js», в которой была создана схема для товара с использованием библиотеки Mongoose. Схема включает поля, такие как название товара, бренд, категория, изображение товара, описание, цена и цена продажи, дата и время создания записи, дата и время обновления записи. Код для модели товара представлен в листинге 1.

Листинг 1 – Модель товара

```
const mongoose = require('mongoose')
const productSchema = mongoose.Schema({
    productName : String,
    brandName : String,
    category : String,
    productImage : [],
    description : String,
    price : Number,
    sellingPrice : Number
},{
    timestamps : true
})
const productModel = mongoose.model("product",productSchema)
module.exports = productModel
```

Модель пользователя

Была реализована модель пользователя «userModel.js», в которой создана схема всех необходимых при регистрации пунктов, таких как: имя пользователя, почта, пароль, изображение профиля, роль на сайте, дата и

время создания записи, дата и время обновления записи. Код для модели пользователя представлен в листинге 2.

Листинг 2 – Модель пользователя

```
const mongoose = require('mongoose')
const userSchema = new mongoose.Schema({
    name : String,
    email : {
        type : String,
        unique : true,
        required : true
    },
    password : String,
    profilePic : String,
    role : String,
},{
    timestamps : true
})
const userModel =  mongoose.model("user",userSchema)
module.exports = userModel
```

Модель корзины

Реализация модели корзины «cartProduct.js» необходима для создания возможности пользователям добавлять существующие на сайте товары в корзину. Схема корзины содержит пункты: идентификатор пользователя, идентификатор товара, количество товара в корзине, дата и время создания записи, дата и время обновления записи. Код для модели корзины представлен в листинге 3.

Листинг 3 – Модель корзины

```
const mongoose = require('mongoose');
const addToCart = mongoose.Schema({
    productId: {
        ref: 'product',
        type: String,
    },
    quantity: Number,
    userId: String, // Добавляем поле для хранения userId
}, {
    timestamps: true
});
const addToCartModel = mongoose.model("addToCart", addToCart);
module.exports = addToCartModel;
```

Загрузка данных пользователя и данных корзины

Одной из основных задач сайта является загрузка и отображение дан-

ных пользователя и содержимого его корзины. Для этого в главном компоненте «App.js» реализованы функции `fetchUserDetails` и `fetchUserAdd–ToCart`, которые выполняют запросы к серверу для получения необходимой информации. Код компонента «App.js» представлен в листинге 4.

Листинг 4 – Загрузка данных

```
function App() {
  const dispatch = useDispatch()
  const [cartProductCount,setCartProductCount] = useState(0)
  const fetchUserDetails = async()=>{
      const dataResponse = await fetch(SummaryApi.current_user.url,{
        method : SummaryApi.current_user.method,
        credentials : 'include'
      })
      const dataApi = await dataResponse.json()
      if(dataApi.success){
        dispatch(setUserDetails(dataApi.data))
      }
  }
  const fetchUserAddToCart = async()=>{
    const dataResponse = await fetch(SummaryApi.addToCartProductCount.url,{
      method : SummaryApi.addToCartProductCount.method,
      credentials : 'include'
    })
    const dataApi = await dataResponse.json()
    setCartProductCount(dataApi?.data?.count)
  }
  useEffect(()=>{
    fetchUserDetails()
    fetchUserAddToCart()
  },[fetchUserDetails])
  return (
    <>
      <Context.Provider value={{
          fetchUserDetails,
          cartProductCount,
          fetchUserAddToCart
      }}>
        <ToastContainer
          position='top-center'
        />

        <Header/>
        <main className='min-h-[calc(100vh-120px)] pt-16'>
          <Outlet/>
        </main>
        <Footer/>
      </Context.Provider>
    </>);
}
export default App;
```

В данном коде можно увидеть, как происходит загрузка данных пользователя, так `fetchUserDetails` функция делает запрос на API для получе-

ния текущих данных пользователя. Если запрос успешен, данные пользователя сохраняются в Redux-хранилище с помощью действия `setUser–Details`.

Загрузка же данных корзины производится за счет `fetchUserAdd–ToCart`. Данная функция делает запрос к API для получаения количества товаров в корзине. Полученное значение сохраняется в локальном состоянии компонента с помощью `useState`.

Управление состоянием с помощью Redux

Для управления состоянием пользователя используется Redux. Он позволяет централизованно хранить и управлять состоянием приложения, что упрощает разработку и поддержку кода. Код компонента «userSlice.js» представлен в листинге 5.

Листинг 5 – Управление состоянием

```
import { createSlice } from '@reduxjs/toolkit'
const initialState = {
    user : null}
  export const userSlice = createSlice({
    name: 'user',
    initialState,
    reducers: {
      setUserDetails : (state,action)=>{
        state.user = action.payload
      }},})
  export const { setUserDetails } = userSlice.actions
  export default userSlice.reducer
```

Функция `createSlice` из Redux Toolkit упрощает создание Redux-срезов. Также в коде имеется объект `reducers`, в котором содержаться функции-редукторы, которые изменяют состояние. В данном случае, функция `setUserDetails` обновляет состояние пользователя. Благодаря этому, происходит управлением состоянием нашего приложения.

Маршрутизация

Маршрутизация в приложении реализована с использованием библиотеки react-router-dom. Этот пакет позволяет легко управлять переходами между различными страницами приложения. Ваша конфигурация маршрутизации включена в компонент «App.js».

Структура маршрутизации

В компоненте «App.js» используется компонент Outlet для отображения вложенных маршрутов, а также компоненты Header и Footer для постоянных частей интерфейса. Описание работы маршрутизации.

1. Импорт компонентов – в начале файла импортируются необходимые компоненты, включая Outlet из react-router-dom, который используется для рендеринга дочерних маршрутов.

2. Компоненты Header и Footer – эти компоненты отображаются на всех страницах и добавляются непосредственно в компонент App.

3. Компонент Outlet – этот компонент используется для отображения дочерних маршрутов, определенных в конфигурации маршрутов. Он автоматически создает соответствующий компонент в зависимости от текущего маршрута.

4. ToastContainer – компонент из библиотеки react-toastify, который отвечает за отображение уведомлений.

Создание контекста

Контекст в React используется для передачи данных через дерево компонентов без необходимости явно передавать пропсы на каждом уровне. В данном приложении контекст используется для передачи функций и данных о пользователе и корзине. Код создания контекста представлен в листинге 6.

Листинг 6 – Создание контекста

```
import { createContext } from "react";
const Context = createContext(null)
export default Context
```

Пример использования контекста

В компоненте «App.js» используется `Context.Provider` для предоставления данных и функций контекста всем дочерним компонентам. Пример создания контекста представлен в листинге 7.

Листинг 7 – Пример использования контекста

```
<Context.Provider value={{
          fetchUserDetails,
          cartProductCount,
```

Окончание контекста

```
      fetchUserAddToCart}}>
    <ToastContainer
      position='top-center'/>
    <Header/>
    <main className='min-h-[calc(100vh-120px)] pt-16'>
      <Outlet/>
    </main>
    <Footer/>
  </Context.Provider>
```

Описание работы контекста представлено ниже.

1. Создание значений контекста. Объект `value` передается в `Context.Provider`, включающий функции `fetchUserDetails` и `fetchUserAddToCart`, а также значение `cartProductCount`.

2. Передача контекста. `Context.Provider` оборачивает компоненты Header, Outlet и Footer, предоставляя им доступ к значениям контекста.

3. Использование контекста в дочерних компонентах. Любой компонент внутри `Context.Provider` может получить доступ к значениям контекста с помощью `useContext(Context)`.

В данном приложении маршрутизация и контекст реализованы для эффективного управления переходами между страницами и передачи данных по дереву компонентов.

Вывод по третьей главе

В третьей главе была рассмотрена реализация интернет-магазина одежды с использованием технологии React. Основное внимание уделено маршрутизации и контексту, которые обеспечивают динамичное взаимодействие пользователя с веб-приложением и его компонентами.

4. ТЕСТИРОВАНИЕ

4.1. Функциональное тестирование

Функциональное тестирование необходимо для проверки соответствия разрабатываемого продукты на функциональные требования. Результаты проделанного тестирования продемонстрированы в таблице 1.

Таблица 1 – Функциональное тестирование

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
1	Проверка возможности регистрации	Переход на форму «Зарегистрироваться», ввод необходимой информации	Пользователю становится доступна возможность авторизации	Да
2	Проверка возможности авторизации	Переход на форму «Авторизация», пользователь вводит свои данные от учетной записи	Пользователь вошел в свою учетную запись	Да
3	Проверка поиска товаров	Ввести в поисковую строку название интересующей одежды	Отображаются все товары, в названии которых найдено введенное слово	Да
4	Проверка фильтрации товаров	Выбрать и нажать на одну из доступны категорий товаров	Отображаются все товары, которые относятся к выбранной категории	Да
5	Проверка сортировки товаров по цене	Нажать, находясь в каталоге, на доступный тип сортировки товара по цене.	Все товары, интересующие пользователя категорий, отсортированы по стоимости	Да
6	Проверка пагинации	На главной странице нажать кнопку перелистывания товаров в категории	Отобразятся новые товара находящиеся в имеющейся категории	Да
7	Проверка перехода на страницу корзины	Нажать на пиктограмму корзины в шапке сайта	Отображаются интерфейс корзины товаров. В корзине присутствуют все добавленные ранее товары и их общая стоимость	Да

Окончание таблицы 1

№	Название теста	Действие	Ожидаемый результата	Тест пройден?
8	Проверка уменьшения/увеличения товара в корзине	Добавить хотя бы 1 товар в корзину. Находясь в корзине нажать на кнопку «минус»/«плюс» возле одного из добавленных товаров	Счетчик товара изменяется в меньшую/большую сторону на 1. Если количество товара единица, кнопка уменьшения недоступна.	Да
9	Проверка добавления товара в корзину	На карточке товара нажать на копку «Добавить в корзину»	Кнопка «Добавить в корзину» при повторном нажатии выводит сообщение «Товар уже добавлен в корзину». Выбранный товар отображается в корзине со значением 1. Сумма товара добавлена к общей сумме корзины.	Да
10	Проверка удаления товара из корзины	Добавить хотя бы 1 товар в корзину. Находясь в корзине нажать на кнопку «удалить» возле одного из добавленных товаров	Товар полностью удаляется из корзины.	Да
11	Проверка возможности администратора переходить в «Панель Администратора»	Авторизация проходит успешно, и Администратор переходит по нажатию на кнопку в «Панель Администратора».	Администратор находится в «Панель Администратора»	Да
12	Проверка возможности администратора менять роль пользователям	Администратор в «Панель Администратора» нажимает на кнопку «Все пользователи», дальше кликает на пиктограмму в столбце «Информация» и графе роль меняет состояние пользователя	Роль интересующего администратора пользователя изменена	Да

4.2. Тестирование верстки

Тестирование верстки сайта необходимо для обеспечения корректного отображения и функционирования веб-страниц на различных устройствах, браузерах и операционных системах. Это позволяет гарантировать, что сайт выглядит и работает одинаково хорошо на разных платформах, обеспечивая последовательность и единообразие пользовательского опыта. Тестирование верстки помогает выявить и устранить проблемы с совместимостью, несоответствия в размерах, расположении элементов, цветовой гамме и других визуальных аспектах, а также обеспечить соответствие сайта стандартам и рекомендациям веб-разработки. Это критически важно для создания высококачественных, отзывчивых и доступных веб-ресурсов, которые будут хорошо восприниматься пользователями на любых устройствах.

Тестирование интерфейса было проведено в различных браузерах и на разных размерах экрана. Тест проводился в следующих браузерах:

- Opera;
- Google Chrome;
- Microsoft Edge;
- Safari.

Интерфейс пользователя во всех перечисленных браузерах остается идентичным, весь контент остается на своих местах. Также было проведено тестирование верстки на адаптивность в разных браузерах. При увеличении и уменьшении размера экрана контент адаптируется под устройство. Тесты успешно пройдены.

Вывод по четвертой главе

Проведенное функциональное тестирование приложения с JWT аутентификацией показало, что все основные функциональные требования были успешно реализованы и протестированы. Пользователи могут беспрепятственно регистрироваться, авторизоваться, искать, фильтровать и сортировать товары, управлять корзиной покупок, а администраторы - переходить в панель управления и изменять роли пользователей. Тестирование

верстки также подтвердило, что интерфейс корректно отображается и функционирует на различных устройствах, браузерах и операционных системах, обеспечивая последовательность и единообразие пользовательского опыта. В целом, результаты тестирования демонстрируют, что разрабатываемый продукт соответствует всем заявленным функциональным требованиям и обеспечивает высокое качество и доступность для пользователей на любых платформах.

Помимо функционального тестирования, для обеспечения качества разрабатываемого продукта необходимо также провести нефункциональное тестирование. Нефункциональное тестирование играет ключевую роль в обеспечении качества программного продукта, поскольку оно позволяет выявить проблемы, которые могут негативно повлиять на пользовательский опыт, производительность или безопасность системы. Такое тестирование направлено на проверку таких аспектов, как производительность системы, ее надежность, безопасность, удобство использования и соответствие стандартам. Это позволит выявить и устранить возможные проблемы, связанные с быстродействием, стабильностью, защитой данных и юзабилити, гарантируя, что продукт будет соответствовать ожиданиям пользователей и требованиям бизнеса.

ЗАКЛЮЧЕНИЕ

В рамках данного проекта был создан интернет-магазина одежды на платформе React. В ходе работы были решены следующие задачи.

1. Проведен обзор аналогичных проектов, что позволило изучить существующие подходы в области разработки веб-сайтов для интернет-магазина одежды. Выявлены основные требования, которые должны быть реализованы в данном проекте.
2. Подобраны инструменты разработки, учитывая требования и масштаб проекта.
3. Определены функциональные и нефункциональные требования для обеспечения удобства использования и удовлетворения потребностей пользователей.
4. Выбрана подходящая архитектура приложения.
5. Разработан сайт с учетом цветовой гаммы, шрифтов и других дизайнерских аспектов, внешнего вида и удобного использования.
6. Реализован веб-сайт с учетом всех поставленных требований.
7. Проведено функциональное тестирование и тестирование верстки, что позволило убедится в соответствии реализованного проекта поставленным требованиям.

ЛИТЕРАТУРА

1. Welcome to UML Web Site. [Электронный ресурс] URL: https://www.uml.org/ (дата обращения: 11.05.2024 г.).
2. CSS справочник. [Электронный ресурс] URL: http://htmlbook.ru/css (дата обращения: 10.04.2024 г.).
3. React documentation. [Электронный ресурс] URL: https://reactjs.org/docs/getting-started.html (дата обращения: 11.04.2024 г.).
4. Основы HTML. [Электронный ресурс] URL: https://htmlacademy.ru/courses/297 (дата обращения: 02.04.2024 г.).
5. Основы Sass. [Электронный ресурс] URL: https://sass-scss.ru/guide (дата обращения: 09.04.2024 г.).
6. The top programming languages. [Электронный ресурс] URL: https://octoverse.github.com/2022/top-programming-languages (дата обращения 30.03.2024 г.).
7. Куценко Д.А., Богданова О.Б. Разработка фронтенд части сайта // Аллея науки, 2021. – Т. 1. – №. 6. – С. 1152–1157.
8. Back4app. [Электронный ресурс] URL: https://blog.back4app.com/backend-programming-languages-list/ (дата обращения 29.03.2024 г.).
9. Интернет-магазин «La Redoute»[Электронный ресурс] URL: https://www.laredoute.ru (дата обращения: 08.04.2024 г.).
10. Потапенко Н.И., Олеферович А.В., Кудлацкая М.Ф. Основы веб-дизайна: учебно-методическое пособие. // БГТУ, 2020. – 161 с.
11. Redux documentation. [Электронный ресурс] URL: https://redux-toolkit.js.org (дата обращения: 30.03.2024 г.).
12. Мардан А. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. – СПб: Питер, 2019. – 560 с.
13. Хекслет. [Электронный ресурс] URL: https://redux-toolkit.js.org (дата обращения: 30.03.2024 г.).

14. MongoDB. [Электронный ресурс] URL: https://www.mongodb.com (дата обращения: 30.03.2024 г.).

15. Петрова О.Б. Разработка и анализ требований проектирования программного обеспечения: практикум: учебное пособие. // Санкт-Петербург: СПбГУТ им. М.А. Бонч-Бруевича, 2022. – 37 с.

ПРИЛОЖЕНИЕ. Разработка JWT для аутентификации

Листинг 1 – JWT аутентификация

```
const bcrypt = require('bcryptjs')
const userModel = require('../../models/userModel')
const jwt = require('jsonwebtoken');
async function userSignInController(req,res){
    try{
        const { email , password} = req.body
        if(!email){
            throw new Error("Пожалуйста укажите почту")
        }
        if(!password){
             throw new Error("Пожалуйста укажите пароль")
        }
        const user = await userModel.findOne({email})
       if(!user){
            throw new Error("Пользователь не найден")
       }
       const checkPassword = await bcrypt.compare(password,user.password)
       console.log("checkPassoword",checkPassword
       if(checkPassword){
        const tokenData = {
            _id : user._id,
            email : user.email,
        }
        const token = await jwt.sign(tokenData, process.env.TOKEN_SE-
CRET_KEY, { expiresIn: 60 * 60 * 8 });
        const tokenOption = {
            httpOnly : true,
            secure : true
        }
        res.cookie("token",token,tokenOption).status(200).json({
            message : "Успешный вход",
            data : token,
            success : true,
            error : false
        })
       }else{
         throw new Error("Пожалуйста, проверьте пароль")
}
    }catch(err){
        res.json({
            message : err.message || err  ,
            error : true,
            success : false,
        })
    }
}
module.exports = userSignInController
```

Листинг 2 – Код проверки токена

```
const jwt = require('jsonwebtoken')
async function authToken(req,res,next){
    try{
        const token = req.cookies?.token
        console.log("token",token)
```

```
        if(!token){
            return res.status(401).json({
                message : "Пожалуйста авторизируйтесь!",
                error : true,
                success : false
            })
        }
        jwt.verify(token, process.env.TOKEN_SECRET_KEY, func-
tion(err, decoded) {
            console.log(err)
            console.log("decoded",decoded)

            if(err){
                console.log("ошибка авторизации", err)
            }
            req.userId = decoded?._id
            next()
       });
    }catch(err){
        res.status(400).json({
            message : err.message || err,
            data : [],
            error : true,
            success : false
        })
    }
}

module.exports = authToken
```