МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема: Web-сайт «Интернет-магазин велосипедов»

**Исполнитель**

студент 2 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.О. Лемешевский

подпись, дата

**Руководитель**

Доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Жиляк

(должность, уч. звание) (подпись, дата)

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Жиляк

подпись дата инициалы и фамилия

Минск 2023

Содержание

[Введение 4](#_Toc153230326)

[1 Постановка задач и обзор аналогичных решений 5](#_Toc153230327)

[1.1 Актуальность темы 5](#_Toc153230328)

[1.2 Обзор аналогичных решений 5](#_Toc153230329)

[1.3 Постановка задачи 9](#_Toc153230330)

[1.4 Выводы по разделу 9](#_Toc153230331)

[2 Проектирование веб-сайта 10](#_Toc153230332)

[2.1 Выбор средств реализации 10](#_Toc153230333)

[2.2 Проектирование uml-диаграммы 11](#_Toc153230334)

[2.3 Проектирование страниц 12](#_Toc153230335)

[2.4 Выбор стилевого оформления 13](#_Toc153230336)

[2.5 Разработка пользовательских элементов 14](#_Toc153230337)

[2.6 Выводы по разделу 17](#_Toc153230338)

[3 Реализация структуры веб-сайта 18](#_Toc153230339)

[3.1 Структура JSX-документа 18](#_Toc153230340)

[3.2 Структура Redux-Toolkit 25](#_Toc153230341)

[3.3 Добавление таблиц стилей Sass/CSS3 29](#_Toc153230342)

[3.4 Выводы по разделу 32](#_Toc153230343)

[4 Тестирование веб-сайта 33](#_Toc153230344)

[4.1 Адаптивный дизайн веб-сайта 33](#_Toc153230345)

[4.2 Кроссбраузерность веб-сайта 34](#_Toc153230346)

[4.3 Ручное тестирование 35](#_Toc153230347)

[4.4 Выводы по разделу 37](#_Toc153230348)

[5 Руководство пользователю 38](#_Toc153230349)

[Заключение 39](#_Toc153230350)

[Список использованных источников 40](#_Toc153230351)

[Приложение А Макет структуры веб-сайта 41](#_Toc153230352)

[Приложение Б Листинг JSX-документа 43](#_Toc153230353)

[Приложение В Листинг (SASS) 48](#_Toc153230354)

[Приложение Г Листинг Redux 50](#_Toc153230355)

# Введение

В наше время наблюдается взрывной рост числа новых веб-сайтов, что делает разработку сайта особенно важной задачей. Основной целью при создании сайта становится формирование привлекательного и индивидуального пространства для потенциальных пользователей. В ранние годы интернета было возможно привлечь много посетителей без значительных усилий и материальных затрат. Однако в условиях увеличивающейся конкуренции, решающую роль начинают играть функциональность, стиль и запоминаемость сайта.

В современных условиях, независимо от того, представляет ли сайт компанию или служит для рекламы, его главной целью остается привлечение максимальной аудитории. В этом контексте правильный выбор стратегии и исполнителей становится ключевым фактором для достижения успеха. Профессиональные студии веб-дизайна предлагают разработку сайтов разного назначения: корпоративные, визитки, интернет-магазины, а также сайты для ведения бизнеса как в онлайне, так и в реальном мире.

Современную эпоху называют "информационным веком", где информация является наиболее ценным ресурсом. Умение оперативно и четко предоставлять необходимую информацию клиентам через Интернет становится ключом к успеху в современном бизнесе. С точки зрения пользователя, Интернет представляет собой огромный информационный ресурс, способный передавать текст, звук, изображения и видео-ролики. Аудитория сети Интернет постоянно расширяется, что делает ее наиболее быстроразвивающимся средством передачи информации в истории человечества.

Пользователи Интернета представляют особый интерес для рекламодателей, так как среди них сосредоточено множество потенциальных клиентов. Актуальность интернет-магазина в современном мире обусловлена тем, что все больше людей предпочитают онлайн-покупки из-за более выгодных цен и доступности товаров, которые могут быть недоступны в традиционных магазинах.

Цель курсовой работы: разработать интернет-магазин с использованием React JS, а также с применением Sass/CSS3.

Задачи курсовой работы:

* проанализировать и описать основные принципы работы React и Redux;
* разработать приложение, используя React и Redux;
* провести тестирование приложения;
* провести анализ и оценку качества приложения.

# 1 Постановка задач и обзор аналогичных решений

## Актуальность темы

Актуальность темы заключается в том, что интернет-магазин является одним из самых популярных видов сайтов, которые посещают пользователи. Как для покупателей важно, чтобы было удобно и быстро совершать покупки, так и для продавцов важно, чтобы было удобно и быстро добавлять товары в каталог, а также управлять ими.

Сегодня на рынке существует множество интернет-магазинов, которые используют различные технологии и платформы для создания своих сайтов. Но, как правило, они используют один из самых популярных фреймворков для создания интернет-магазинов – React JS.

## 1.2 Обзор аналогичных решений

Веб-сайт – это набор взаимосвязанных веб-страниц, которые представляют собой информацию, предназначенную для просмотра в браузере. Веб-сайт представляет собой совокупность веб-страниц, которые могут быть связаны между собой посредством гипертекстовых ссылок.

Для создания быстрого и динамического веб-сайта необходимо использовать современные технологии, которые позволяют создавать веб-сайт с минимальными затратами времени и средств. Веб-сайт должен быть удобным для пользователя, а также должен быть адаптирован под мобильные устройства. Тема веб-сайта должна быть востребованной, а также должна быть актуальной. Поэтому было принято решение разработать интернет-магазин, который будет представлять собой веб-сайт, на котором можно будет заказать велосипеды BMX и их комплектующие. Прежде чем приступить к разработке веб-сайта, необходимо проанализировать аналоги, чтобы понять, какие технологии используются для создания веб-сайтов, а также какие есть недостатки и преимущества веб-сайтов аналогов. И самое главное найти в каждом аналоге свои уникальные особенности, которые будут отличать этот сайт от других. Для этого будет рассмотрены 3 аналога.

Аналог №1 – сайт Stressshop [1].

Stressshop.ru — это интернет-магазин BMX, который предлагает широкий выбор велосипедов, запчастей и аксессуаров от разных брендов, включая отечественный бренд STRESS.

Основное преимущество сайта — это его целевая аудитория, которая состоит из любителей и профессионалов BMX. Сайт также предлагает бесплатную доставку по Москве, России и СНГ при заказе от 15 000 рублей.

На главной странице сайта можно увидеть разделы, которые помогут пользователю найти нужный товар по категориям, брендам, цене, размеру и другим параметрам1. На рисунке 1.1 представлено оформление сайта.

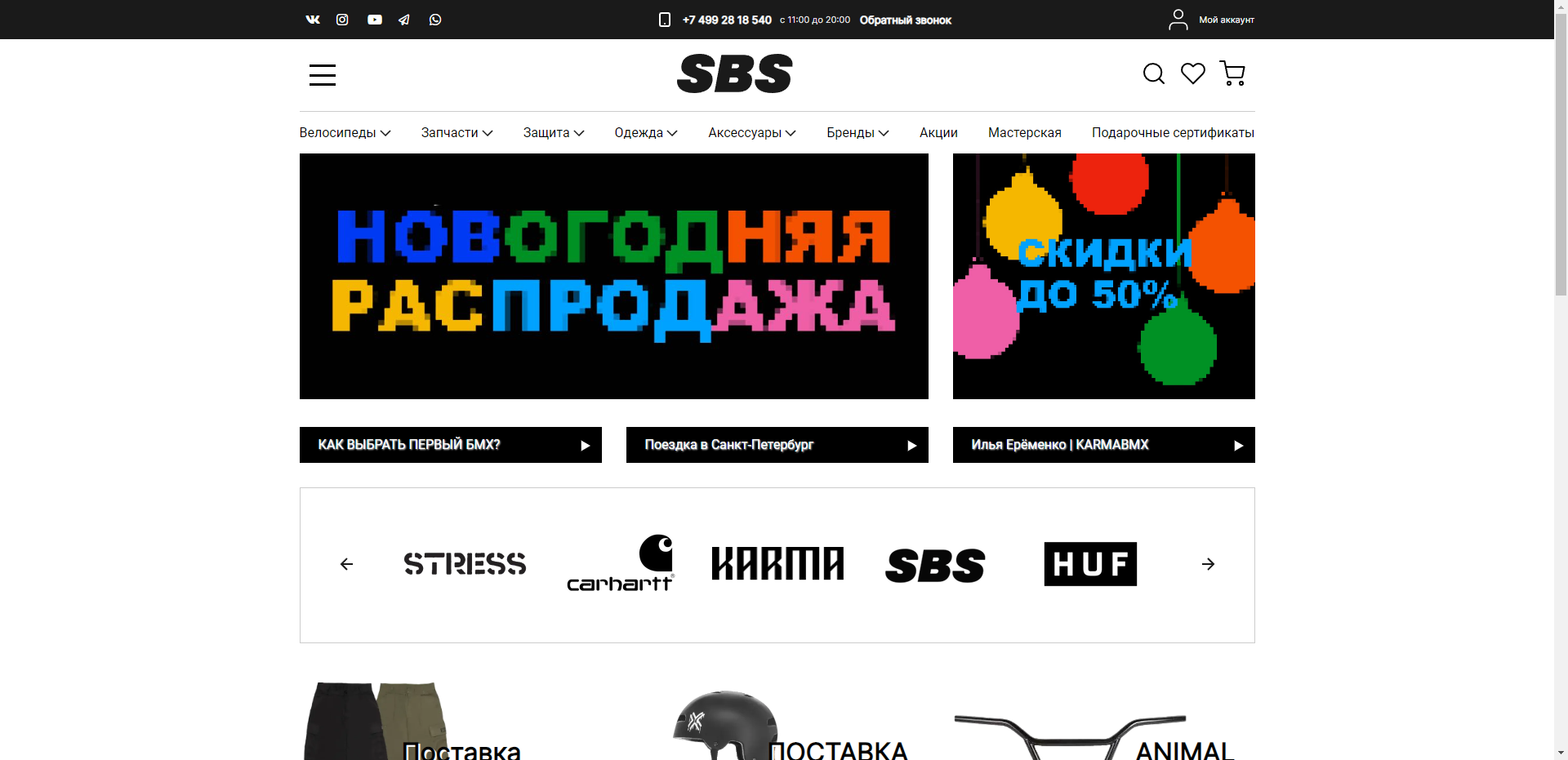


Рисунок 1.1 – Главная страница сайта Stressshop

Веб-сайт достаточно простой и удобный для пользователя, но имеет некоторые недостатки в дизайне. Шрифт маленький и не очень читабельный, цветовая гамма слишком яркая и контрастная, анимация при наведении на картинки выглядит дешево и неестественно. Сайт использует три основных цвета (черный, белый и красный), что не очень гармонично сочетается с общим стилем сайта. Веб-сайт имеет адаптивный дизайн, который позволяет использовать сайт на мобильных устройствах. Горизонтальное меню становится выпадающим, текст не перекрывается картинками, дизайн выглядит аккуратно и удобно. Дизайн представлен на рисунке 1.2

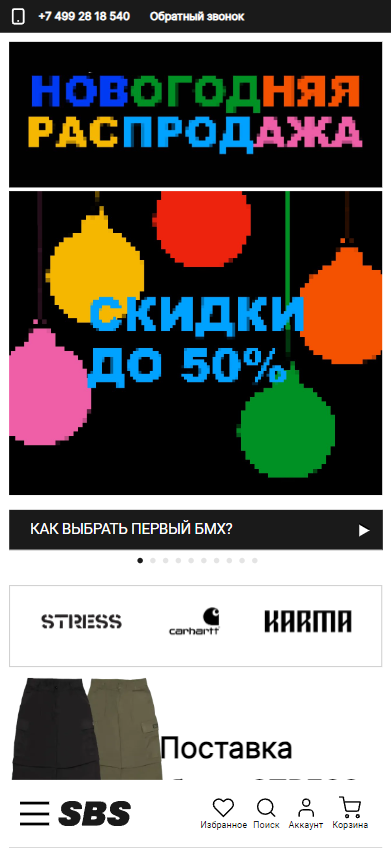


Рисунок 1.2 – Мобильная версия сайта Stressshop

Из минусов данного веб-сайта можно выделить то, что он долго загружается из-за большого количества товаров, которые находятся на сайте.

Из плюсов данного веб-сайта можно выделить то, что он быстро загружается из-за небольшого количества товаров, которые находятся на сайте. Данный веб-сайт был разработан с помощью HTML, CSS, JavaScript, PHP, MySQL.

Аналог №2 – сайт iBike [2].

ibike.by - это крупнейший экстрим магазин в Беларуси, который предлагает широкий выбор трюковых велосипедов, самокатов, запчастей, защиты и аксессуаров1.

На главной странице сайта можно увидеть разделы, которые помогут пользователю найти нужный товар по категориям, брендам, цене, размеру и другим параметрам. Главная страница представлена на рисунке 1.3

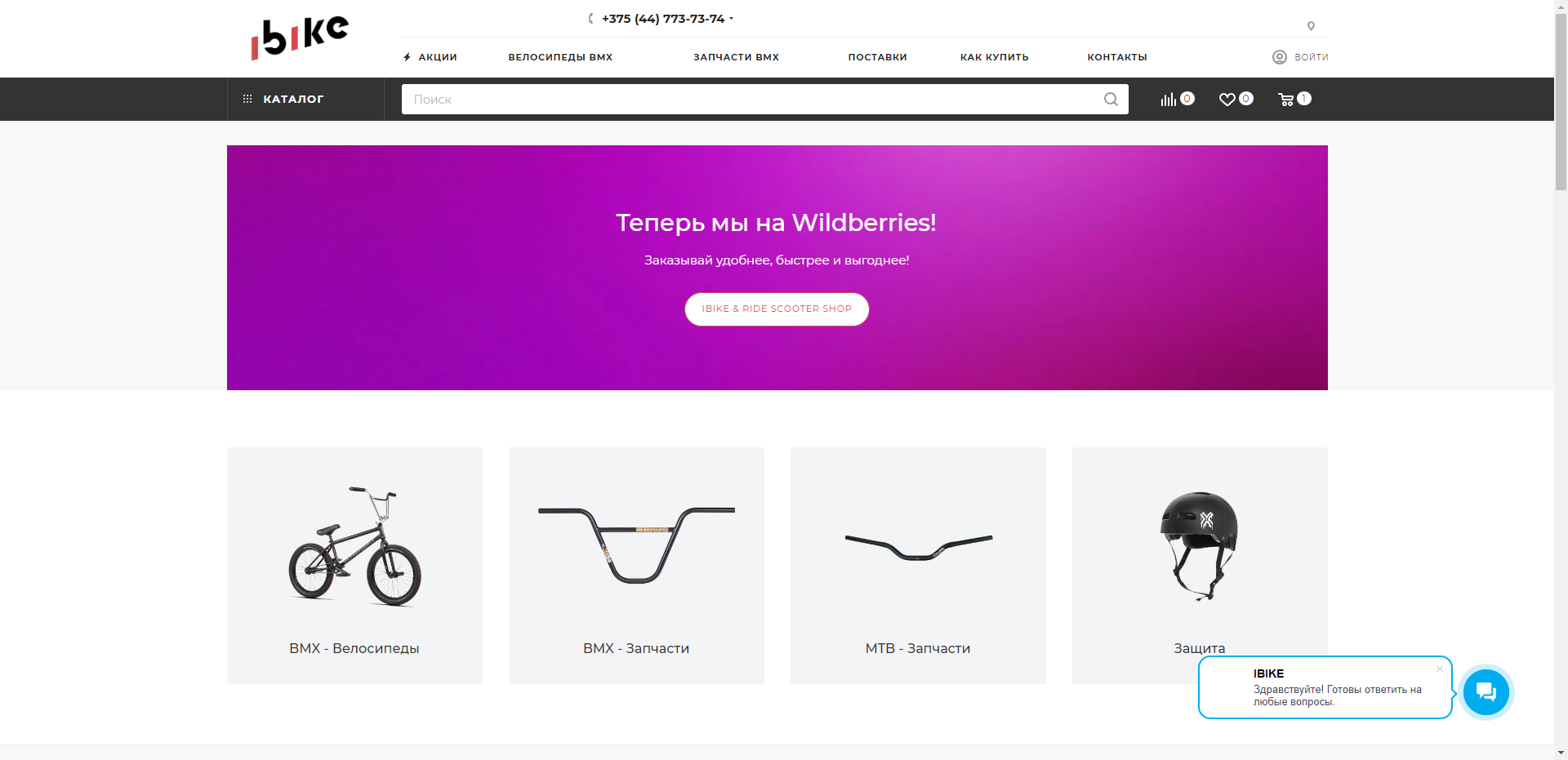


Рисунок 1.3 – Главная страница сайта iBike

Веб-сайт достаточно простой и удобный для пользователя, но имеет некоторые недостатки в дизайне. Шрифт маленький и не очень читабельный, цветовая гамма слишком яркая и контрастная, анимация при наведении на картинки выглядит дешево и неестественно. Сайт использует три основных цвета (черный, белый и красный), что не очень гармонично сочетается с общим стилем сайта. Также сайт имеет адаптивный дизайн, который позволяет просматривать сайт на мобильных устройствах. У данного сайта нет никаких проблем с адаптивностью, на разных устройствах сайт выглядит привлекательно. На рисунке 1.4 представлено оформление на мобильных устройствах.

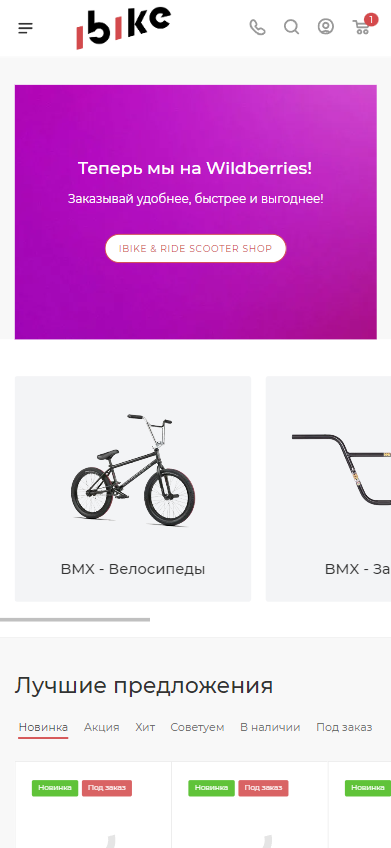


Рисунок 1.4 – Мобильная версия сайта iBike

Из плюсов данного веб-сайта можно выделить то, что он быстро загружается из-за небольшого количества товаров, которые находятся на сайте. Данный веб-сайт был разработан с помощью HTML, CSS, JavaScript, PHP, MySQL.

Аналог №3 – сайт Alienbike [3].

AlienBike - это крупнейший экстрим магазин в Беларуси, который предлагает широкий выбор трюковых велосипедов, самокатов, запчастей, защиты и аксессуаров. На главной странице сайта можно увидеть разделы, которые помогут пользователю найти нужный товар по категориям, брендам, цене, размеру и другим параметрам. На рисунке 1.5 представлено оформление сайта.

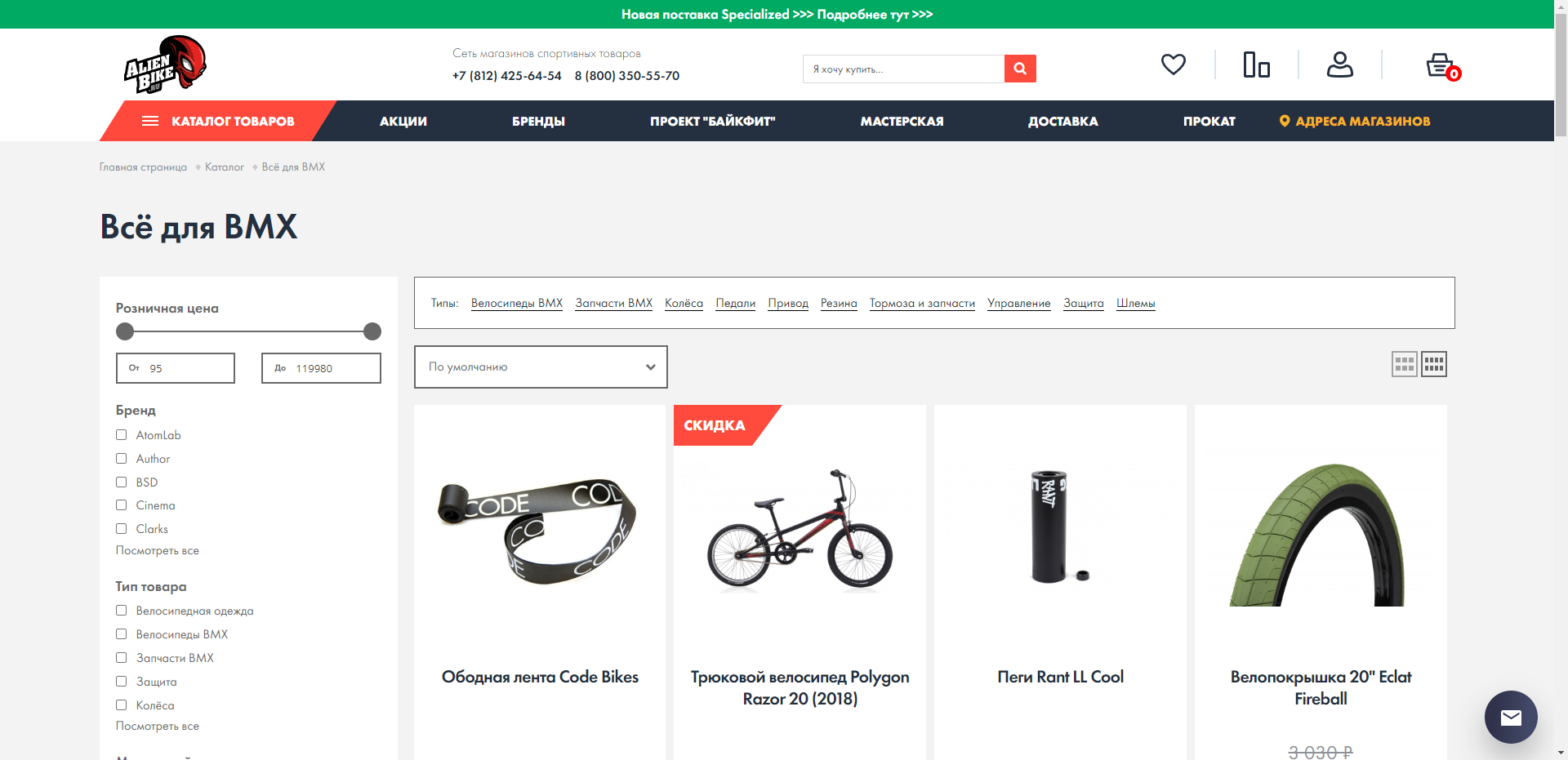


Рисунок 1.5 – Главная страница сайта Alienbike

Веб-сайт достаточно простой и удобный для пользователя, но имеет некоторые недостатки в дизайне. Шрифт маленький и не очень читабельный, цветовая гамма слишком яркая и контрастная, анимация при наведении на картинки выглядит дешево и неестественно. Сайт использует три основных цвета (черный, белый и красный), что не очень гармонично сочетается с общим стилем сайта. Веб-сайт имеет адаптивный дизайна, который позволяет использовать сайт на мобильных устройствах. Дизайн представлен на рисунке 1.6.

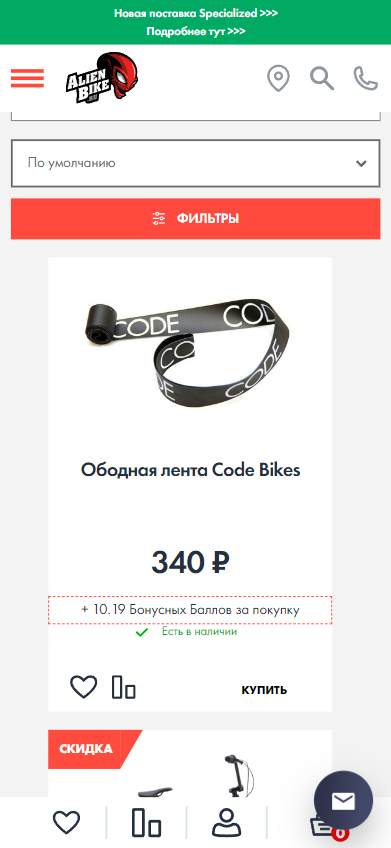


Рисунок 1.6 – Мобильная версия сайта Alienbike

И таким образом, было проанализировано 3 сайта, которые являются аналогами нашего будущего интернет-магазина. Основные преимущества и недостатки этих сайтов были выделены, а также были выделены технологии, которые используются для создания сайтов. Можно сделать вывод, что сайт, который будет создан в рамках данной работы, будет иметь следующие преимущества:

* быстрая загрузка сайта;
* простой и удобный дизайн;
* яркие и не раздражающие глаза цвета;
* кроссплатформенность;
* адаптивность под мобильные устройства;
* удобная навигация по сайту.

## Постановка задачи

Основные задачи для интернет-магазина:

* разработать регистрацию и авторизацию пользователей;
* разработать возможность добавления и удаления товаров из корзины;
* разработать возможность добавления и удаления товаров из избранного;
* разработать структуру веб-сайта;
* разработать возможность посмотреть информацию о товаре;
* разработать возможность оставить отзыв о товаре;
* разработать возможность заказа товара;
* разработать возможность просмотра истории покупок;
* разработать возможность редактирования профиля.

## 1.4 Выводы по разделу

В данном разделе была подчеркнута необходимость создания интернет-магазина, а также были выделены основные преимущества и недостатки существующих сайтов по продаже велосипедов BMX. Были выделены технологии, которые используются для создания сайтов. Были выделены основные задачи для интернет-магазина.

# 2 Проектирование веб-сайта

## 2.1 Выбор средств реализации

Основная цель данного веб-ресурса заключается в предоставлении подробной информации о продуктах, доступных для приобретения. Этот интернет-магазин рассчитан на тех, кто заинтересован в приобретении разнообразных товаров. Веб-сайт будет оптимизирован для использования на мобильных устройствах и обеспечит кроссплатформенное взаимодействие.

Интернет-магазин будет разработан в текстовом редакторе Visual Studio Code. Страницы сайта могут просматриваться в таких браузерах как Microsoft Edge, Safari, Google Chrome.

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса и средства для рефакторинга.

Для реализации будущего сайта будет использована следующая архитектура и технологии:

Front-end — это часть сайта, которая видна пользователю. В данном случае это будет внешний вид сайта, его дизайн, а также навигация по сайту. В данном случае будет использованы следующие технологии: React JS (16.13.1), SASS (4.13.0),CSS ( Redux-Toolkit (1.3.5), React Router (5.1.2), Redux Persist (6.0.0).

JavaScript – это язык программирования, который используется для создания динамических веб-страниц. JavaScript позволяет создавать интерактивные веб-страницы, которые могут реагировать на действия пользователя.

React JS представляет собой библиотеку JavaScript, предназначенную для создания пользовательских интерфейсов. Отличительной особенностью React JS является использование компонентного подхода, который позволяет разрабатывать сложные интерфейсы, состоящие из простых компонентов. При использовании React разработчики могут создавать веб-приложения, способные изменять отображение без перезагрузки страницы. Это обеспечивает быстрый отклик приложений на действия пользователя, такие как заполнение форм, применение фильтров, добавление товаров в корзину и другие. React применяется для отрисовки компонентов пользовательского интерфейса, а также может полностью управлять фронтендом. В этом случае React используют с библиотеками для управления состоянием и роутинга, например, Redux и React Router.

CSS — язык стилей, используемый для описания внешнего вида документа, написанного с использованием языков разметки, таких как HTML или XML. Он определяет, как элементы веб-страницы должны быть отображены на экране, печати или других медиа.

SASS — это расширение CSS, которое добавляет в CSS мощные возможности, такие как переменные, вложенные правила, импорт и многое другое. Но самое главное, что SASS позволяет использовать препроцессоры, которые позволяют писать более краткий и лаконичный код.

Redux — это библиотека для управления состоянием приложения.

Redux Toolkit — это набор инструментов для упрощения работы с Redux. Redux Toolkit позволяет создавать более простые и понятные редьюсеры, а также упрощает работу с асинхронными запросами.

React Router — это библиотека для маршрутизации в React-приложениях. React Router позволяет создавать маршруты, которые будут отображаться в зависимости от URL-адреса.

Redux Persist — это библиотека для управления состоянием приложения в связке с библиотекой Redux в контексте приложений на React. Она предоставляет механизм для сохранения и восстановления состояния Redux в постоянном хранилище, таком как локальное хранилище браузера. Это особенно полезно для сохранения данных, таких как пользовательские настройки, темы или другие важные параметры, между сеансами работы приложения.

Для передачи данных между клиентом и сервером будет использоваться формат JSON.

JSON — это текстовый формат обмена данными, который основан на JavaScript. JSON позволяет передавать данные между клиентом и сервером. JSON позволяет передавать данные в виде объектов, массивов, строк, чисел, логических значений и null.

Итак, краткий итог по использованному стеку технологий для проекта интернет-магазина BMX: текстовый редактор Visual Studio Code для удобной кроссплатформенной разработки, фронтенд будет построен с применением React JS (16.13.1), SASS (4.13.0), CSS, Redux Toolkit (1.3.5), React Router (5.1.2) и Redux Persist (6.0.0) для эффективного управления пользовательским интерфейсом и состоянием приложения. Обмен данными между клиентом и сервером будет осуществляться в формате JSON, обеспечивая эффективное взаимодействие между компонентами и передачу информации.

## 2.2 Проектирование uml-диаграммы

Роли пользователя представляют собой определенные привилегии, которые пользователь может обладать в системе. В зависимости от уровня его роли, пользователю предоставляется доступ к различным функциональным возможностям системы.В данном проекте роли пользователей будут следующими:

* Пользователь (User).
* Гость (Guest).

На основе предоставленного списка ролей необходимо построить UML диаграмму. UML диаграмма изображена на рисунке 2.1.

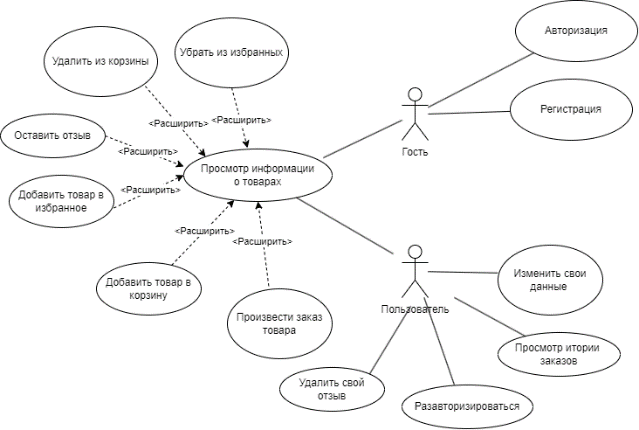


Рисунок 2.1 – UML диаграмма вариантов использования

В начале работы с сайтом пользователь является гостем (Guest). Ему будет доступно главная страница, страница для регистрации и авторизации, полная информация о товаре. После регистрации гость переходит в роль пользователя (User).

Пользователь (User) может оформлять заказы, оставлять отзывы, просматривать историю заказов, информацию о товарах. Пользователь может изменить свои данные, изменить имя, выйти с аккаунта, менять собственный аватар.

Подводя итог, можно сказать, что в данном проекте будет две роли пользователей. Пользователь (User) будет иметь доступ к основным функциям сайта, Гость (Guest) будет иметь доступ только ограниченный доступ к функционалу.

## 2.3 Проектирование страниц

В этом проекте применяется Flexbox-вёрстка, поскольку она обеспечивает создание адаптивных веб-страниц, гарантируя корректное отображение на всех типах устройств. С использованием данной технологии легко и гибко располагать элементы в контейнере, эффективно распределять доступное пространство между ними и выравнивать их в соответствии с заданными параметрами, даже если у элементов нет явно указанных размеров. Применение CSS Flexbox поддерживается всеми современными браузерами, что обеспечивает надежность и согласованность визуализации на всех платформах.

Исходя из выбора способа верстки были расположены все основные элементы сайта и разработаны макеты страниц. Макеты смотреть в [приложении Б](#_Приложение_Б_Макет).

Первым шагом было принято создать макет сайта на основе рассмотренных аналогов, сделать макет удобным и легким. По макету будет понятно, где будут располагаться основные блоки, картинки, текст. Макет будет представлен в [приложении А.](#_Приложение_A_Прототипы)

Вторым шагом было принято решение создать макеты страниц сайта. Макеты страниц сайта будут представлены в приложении Б.

На сайте будут представлены следующие страницы:

* + Главная страница.
  + Полная информация о товаре.
  + Корзина.
  + Избранное.
  + Профиль.
  + Вход.
  + Регистрация.

На главной странице будет расположены все продаваемые карточки товаров и их краткая информация, видео-ролик о продаваемых товарах, поисковой фильтр, контекстное меню для выбора категорий, а так же нижняя часть сайта с полезной информацией.

На странице с полной информацией о товаре будет расположены их фотографии, описание, цена, названия, отзывы, а так же функционал добавления товара в корзину и в избранное.

На странице корзины будут расположены карточки товаров, форма оформления заказа, информация о покупке.

На странице избранное будет располагаться товары добавленные пользователем в избранное.

На странице профиля будут расположены данные пользователя, история заказов, информация о стоимости товара. На этой странице пользователь сможет изменить свои данные, сменить аккаунт.

На странице входа будут расположены форма входа, ссылка на страницу регистрации.

На странице регистрации будут расположены форма регистрации, ссылка на страницу входа.

Также будет навигационное меню с ссылками на страницу избранных товаров, страницу корзины, страницу профиля, страницу входа, страницу регистрации. Оно будет расположено вверху страницы.

## 2.4 Выбор стилевого оформления

Дизайн сайта выбран очень простой и минималистичный. Он состоит из нескольких страниц, включая главную страницу, страницу категорий, страницу продуктов и страницу корзины.

Цветовая палитра сайта состоит из белого, черного и серого цветов, что делает его очень чистым и современным. Эти цвета были подобраны так, потому что они будут приятны для глаз и это цвета, на которое пользователь сразу обратит внимания на сайт.

В данном проекте будут использован один основной шрифт: Roboto. Шрифт подключался с помощью библиотеки Google Fonts.

Шрифт Roboto - это шрифт с двойной природой, который имеет механический скелет и в основном геометрические формы. В то же время, шрифт обладает дружелюбными и открытыми кривыми. Шрифт был разработан Google и является одним из самых популярных шрифтов в мире. Он используется во многих приложениях и на многих сайтах, включая Android, Google Fonts и другие.

## 2.5 Разработка пользовательских элементов

Разработка логотипа осуществлялась в графическом редакторе Figma. Дизайн не минималистичен. Было принято решение поставить на виду главный предмет магазина. Логотип представляет из себя название BMShop представлена на рисунке 2.3.



Рисунок 2.3 – Логотип сайта BMShop

В проекте будет использована иконка сайта, которая будет отображаться во вкладке браузера. Иконка будет иметь формат .ico и будет иметь размер 32x32 пикселя. Иконка будет создана в программе Adobe Photoshop. Иконка представлена на рисунке 2.4.

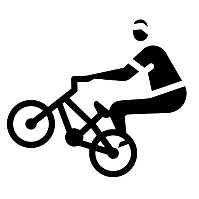


Рисунок 2.4 – Иконка сайта BMShop

В проекте будут пользовательские элементы, которые будут отвечать за взаимодействие пользователя с сайтом. Это будут следующие элементы: кнопки, модальные окна, формы и т.д. Пример формы будет на рисунке 2.5.

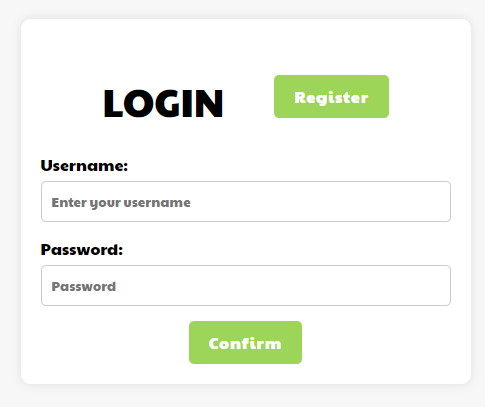


Рисунок 2.5 –Форма входа сайта BMShop

Также будет адаптивный слайдер на с фотографиями товаров, который может показать изображение в полном размере. Пример слайдера будет на рисунке 2.6.



Рисунок 2.6 – Слайдер товара сайта BMShop

На сайте будут карточки товара, которые будут состоять из картинки и кнопок «В корзину» и «В избранное». Кнопка «В корзину» будет добавлять товар в корзину. Кнопка «В избранное» будет добавлять товар в избранное. А также можно будет перейти на страницу с подробной информацией о товаре. Дизайн карточек товара будет простым и понятным. Пример карточки товара будет на рисунке 2.7.

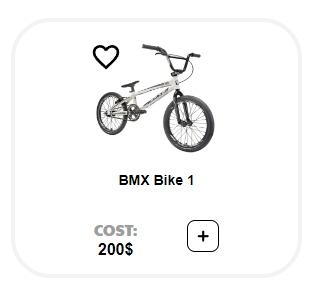


Рисунок 2.7 – Карточка товара BMShop

Также на сайте будет блок с категориями товаров и полем для поиска по имени. Пользователь сможет искать товары. Пример блока поиска товара будет рисунке 2.8.



Рисунок 2.8 – Блок поиска товара

Следующим основным элементом сайта будет блок корзины. Пользователь сможет добавлять продукты в корзину, удалять их из корзины, оформлять заказы. Пример блока корзины будет рисунке 2.9.

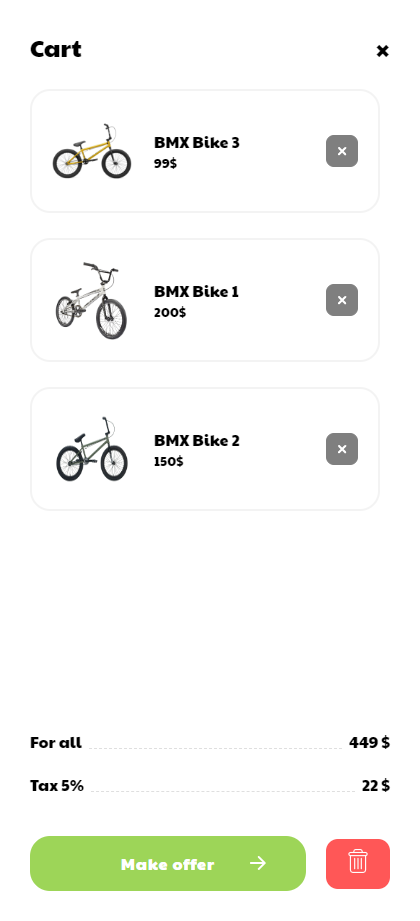


Рисунок 2.9 – Блок корзины

На сайте будут разработаны некоторые анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Анимация будет при появлении страницы на сайте. А также при наведении на кнопки.

Элемент навигации будет состоять из логотипа сайта, навигационного меню. В навигационном меню будет 3 пункта: Корзина, Избранное, Профиль. На рисунке 2.10 представлено оформление навигационного меню.

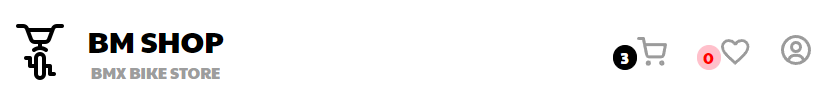


Рисунок 2.10 –навигационное меню сайта BMShop

На данном этапе были созданы макеты сайта, которые будут использоваться для дальнейшей разработки сайта. Были выбраны основные цвета и шрифты, которые будут использоваться на сайте. Были разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Самое главное, чтобы сайт был удобным для пользователя и отвечал его потребностям.

## 2.6 Выводы по разделу

Подводя итоги этого раздела можно сказать, что были разработаны макеты сайта, которые будут использоваться для дальнейшей разработки сайта. Были выбраны основные цвета и шрифты, которые будут использоваться на сайте. Были разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Самое главное, чтобы сайт был удобным для пользователя и отвечал его потребностям.

# 3 Реализация структуры веб-сайта

## 3.1 Структура JSX-документа

Для реализации проекта было принято решение создавать разные компоненты. Компонент — это функция, которая возвращает JSX формат.  JavaScript XML (JSX) — это расширение синтаксиса JavaScript, которое позволяет использовать HTML-подобный синтаксис для описания структуры интерфейса. Как правило, компоненты написаны с использованием JSX, но также есть возможность использования обычного JavaScript. Код примера JSX-формата показан в листинге 3.1.

|  |
| --- |
| export default function Logo(){  return(  <div className="logo" id="logo">  <img src="./img/logo.png" alt=""/>  <span className="summury">  <p>BM SHOP<small>BMX BIKE STORE</small></p>  </span>  </div>  )  } |

Листинг 3.1 – Структура JSX-формата

Ключевым компонентом будет файл с именем и функцией App.js, который будет включать основные маршруты по сайту. Для этого предпочтительно использовать пакет React-router-dom. Почему выбор пал на него вместо обычного тега <a>? Потому что React представляет собой SPA (Single Page Application) — веб-приложение, функционирующее на одной странице. Оно загружает все необходимые JavaScript и CSS файлы при первой загрузке страницы, а затем минимизирует взаимодействие между клиентом и сервером. Таким образом, при таком подходе основная часть работы веб-сайта выполняется на стороне клиента, и обмен данными с сервером, как правило, осуществляется посредством формата JSON. Пример кода для маршрутов представлен в листинге 3.2.

|  |
| --- |
| <Router>  <div className="app-body">  <div className="overlay" style={{ display: display }}>  <Cart {...cartProps} />  </div>  <Routes>  <Route path="/login" element={<Login {...loginProps}/>} />  <Route path="/bikePage/:id" element={ <SnackbarProvider maxSnack={3}> <BikePage {...bikeProps}/> </SnackbarProvider>} />  <Route path="/" element={<>  <Header {...headerProps} />  <SnackbarProvider maxSnack={3}>  <Main {...mainProps} />  </SnackbarProvider>  <Footer />  </>} />  <Route path="/\*" element={<ErrorTab/>}/>  </Routes>  </div>  </Router> |

Листинг 3.2 – Структура маршрутов

Следующим компонентом будет BikePage компонент. В нем будет находится карточка товара. Карточка товара будет состоять из:

* Фотографий товара.
* Названия товара.
* Описание товара.
* Цены товара.
* Кнопки добавления товара в корзину и в избранное.

Кнопка добавления товара в корзину будет внизу всей информации товара. При нажатии на кнопку добавления товара в корзину, она будет менять цвет и текст. При нажатии на кнопку добавления товара в избранное, она будет менять цвет. Код карточки товара будет показан в листинге 3.3.

|  |
| --- |
| return(  <>  <h1 className="info\_sum">Item info:</h1>  <div className="bike\_container">  <BikeSlider cardToShow={cardToShow} setOverflow={setOverflow}/>  {/\* <img className="info\_img" src={cardToShow.img} width="500" height="350" alt="" /> \*/}  <div className="bike-info-container">  <h2 className="inner\_info\_sum"> {cardToShow.name} <svg onClick={() => { addFavBtn(cardToShow);}} className="bp-fav" focusable="false"  aria-hidden="true" viewBox="0 0 24 24"  data-testid="FavoriteBorderIcon" fill={bikeFillColors[cardToShow.id - 1]}>  <path d="M16.5 3c-1.74 0-3.41.81-4.5 2.09C10.91 3.81 9.24 3 7.5 3 4.42 3 2 5.42 2 8.5c0 3.78 3.4 6.86 8.55 11.54L12 21.35l1.45-1.32C18.6 15.36 22 12.28 22 8.5 22 5.42 19.58 3 16.5 3zm-4.4 15.55-.1.1-.1-.1C7.14 14.24 4 11.39 4 8.5 4 6.5 5.5 5 7.5 5c1.54 0 3.04.99 3.57 2.36h1.87C13.46 5.99 14.96 5 16.5 5c2 0 3.5 1.5 3.5 3.5 0 2.89-3.14 5.74-7.9 10.05z"></path>  </svg></h2>  <p className="b-cost">Cost: <span>{cardToShow.cost} $</span> </p>  <h3>About:</h3>  <p className="bike-desq">{cardToShow.desq}</p>  <div className="btns-container">  <div className="add\_but" onClick={() => addToCart(cardToShow)}>  <button style={{ backgroundColor: added ? 'rgb(239, 85, 85)' : '#8cbe00' }} className="cart\_bt" onClick={handleClickVariant(added ? 'error' : 'success') }><p>{added ? `Remove` : "Add"}</p></button>  </div>    </div>  </div>  </div>  <div className="comments-container">  <h2>Reviews</h2>  <div className="type-container">  <input value={commentInput} maxLength={50} className="new-text" type='text' name="comment" placeholder="Enter your view" onChange={(e) => setCommentInput(e.target.value)} />  <button onClick={addComment} >Send</button>  </div>  <div className="total-comments">  { commentsList.map((comment) => (  <div key={comment.id} className="comment">  <div className="comment-head">  <div className="comm-from">  <img src={comment.avatar} className="comment-avatar" alt=" "/>  <h3>{comment.userName}:<br/><small>{comment.bikeName}</small></h3>  </div>  <div>  <div className="com-btn">  <button onClick={() => deleteComment(comment.id,comment.userName)}><img src='/img/trash.png' alt='clean'/></button>  </div>  <p className="time">{comment.time}</p>  </div>  </div>  <h4>{comment.text}</h4>  </div>))}  </div>  </div>  <div className="link-container">  <Link to="/" className="link\_back"><button className="go-back"><img className='info-img' src='/img/go-back.png' alt=''/><p className='back-sum' href="#">Back</p></button></Link>  </div>  </>  )  } |

Листинг 3.3 – Структура компонента BikePage

Следующим компонентом будет Cart компонент. В нем будет находится товары, которые были добавлены в корзину. Также будет находится блок с общей суммой товаров в корзине, налогом и кнопкой оформления заказа или очистки корзины. На товаре будет кнопка удалить из корзины. Код корзины будет показан в листинге 1.4.

|  |
| --- |
| return(    <div className='drawer'>  <h2>Cart <span className='cart-closer' onClick={displayNone}>&times;</span></h2>  <div className='cart-empty' style={bikes.length === 0 ? {display:'flex'} : {display:'none'}}>  <img src='./img/empty-cart.png' alt='empty-cart'/>  <p>Cart is empty</p>  <button onClick={displayNone}><img className='empty-btn-img' src='./img/go-back.png' alt=''/>Go back</button>  </div>  <div className='items' >  {bikes.map((bike) => (  <div className='cartItem' key={bike.id}>  <img src={bike.img[0]} alt='' />  <div>  <p>{bike.name}<small>{bike.cost}$</small></p>    </div>  <img className='remover' src='/img/remove.svg' onClick={()=>{delFromCart(bike.id)}} alt=''/>  </div>  ))}  </div>  <div className='cartTotalBlock' style={bikes.length === 0 ? {display:'none'} : {display:'block'}}>  <ul className='cart-foot'>  <li><span>For all </span><div></div><b>{bikes.reduce((total,element) => total + element.cost , 0)} $</b></li>  <li><span>Tax 5%</span><div></div><b>{Math.floor((bikes.reduce((total, element) => total + element.cost, 0) \* 0.05))} $</b></li>  </ul>  <div className='cart-btns'>  <button className='greenBtn' onClick={addToBuys}>  Make offer <img src='./img/arrow.png' alt=''/></button>  <button className='clear-btn' onClick={clearCart}><img src='./img/trash.png' alt='clean'/></button>  </div>  </div>    </div>  )  } |

Листинг 3.4 – Структура кода компонента Cart

После нажатия на кнопку оформления заказа, товар будет добавлен в истрию покупок пользователя или будет отмена заказа если пользователь не был зарегистрирован. Код отображения купленного товар будет показан в листинге 3.5.

|  |
| --- |
| {boughtItems.length === 0 ? <div className={styles.nth\_bought}> <img src="./img/favorite.png"/> <h2>Nothing bought.</h2> </div> :  boughtItems.map((bike) => (  <div className='cartItem' key={bike.id}>  <img src={bike.img[0]} alt='' />  <div>  <Link to={`/bikePage/${bike.id}`} className="bike\_link\_info">  <p>{bike.name}</p> </Link> <small>{bike.cost}$</small>  </div>  </div>  ))  } |

Листинг 3.5 – Структура кода купленных товаров

Следующим основным компонентом является Login, в котором можно посмотреть информацию о пользователе. Пользователь сможет редактировать свои данные, просматривать историю заказов. Код Profile будет показан в листинге 3.6.

|  |
| --- |
| if(isAuthorized) {  return (  <>  <div className="header">  <div className="logo">  <img src="./img/logo.png" alt=""/>  <span className="summury">  <p>BM SHOP<small>BMX BIKE STORE</small></p>  </span>  </div>  </div>  <div className={styles.user\_profile}>  <UploadAvatar/>  <div className={styles.user\_profile\_data}>  <h2>Username:<br/>  <UserEditor handleEditName={handleEditName}  usernameRef={usernameRef}  displayInput={displayInput}  />  </h2>  <div className='user-name-block' style={{display:displayNameBlock}}>  <span>{userName} </span>  <img src='./img/nameEdit.svg'  onClick={() => handleShowEditor()}  className='name-edit' alt=' '/>  </div>  <p>Email: <br/> {userEmail}</p>  <div className={styles.user\_profile\_data\_buttons}>  <Link to="/"><button >Back to main</button></Link>  <button onClick={handleLogout} className='logout-btn'>Logout</button>  </div>  </div>  </div>  {boughtItems.length !== 0 ? <h2 className={styles.if\_bought}>Purshases:</h2> : <></>}  <div className={styles.purss\_items}>  {boughtItems.length === 0 ? <div className={styles.nth\_bought}> <img src="./img/favorite.png"/> <h2>Nothing bought.</h2> </div> :  boughtItems.map((bike) => (  <div className='cartItem' key={bike.id}>  <img src={bike.img[0]} alt='' />  <div>  <Link to={`/bikePage/${bike.id}`} className="bike\_link\_info">  <p>{bike.name}</p> </Link> <small>{bike.cost}$</small>  </div>  </div>  ))  }  </div>  </>  );  } |

Листинг 3.6 – Структура компонента Login

Таким образом, в этом разделе были рассмотрены основные компоненты приложения, которые будут использоваться в дальнейшем. Хотелось бы отметить, что JSX-формат очень удобный для написания кода, так как он позволяет писать код в HTML-формате, что упрощает его чтение и понимание.

## 3.2 Структура Redux-Toolkit

Для того, чтобы удобно работать с данными, был использован Redux-Toolkit. В нем будет хранится информация о корзине, о пользователях, об избранных товарах, о комментариях. Здесь также был использован Redux Persist совместно для сохранения и восстановления состояния Redux в постоянном хранилище.

Благодаря CreateSlice был реализован редьюсер для получения данных о товарах, о пользователях, о заказах, о корзине. В CreateSlice будет находиться имя редьюсера, начальное состояние, редьюсеры для изменения состояния. Код CreateSlice будет показан в листинге 3.7.

|  |
| --- |
| const initialState = {  cart: [],  favorite: [],  userName:'',  userEmail:'',  userPassword:'',  userAvatar:'/img/avatarDefault.jpg',  preview:null,  isAuthorized: false,  // comments:[],  commentsList:[],  };  const commentsListSlice = createSlice({  name:'comment',  initialState,  reducers:{  setCommentsList:(state,action) => {  state.commentsList = action.payload  },  }  })  // user slice  const userSlice = createSlice({  name: 'user',  initialState,  reducers: {  authorize: (state, action) => {  state.isAuthorized = true;  state.userName = action.payload.userName;  state.userEmail = action.payload.userEmail;  state.userPassword = action.payload.userPassword;  state.userAvatar = '/img/avatarDefault.jpg';  },  unauthorize: (state) => {  state.isAuthorized = false;  },  setName: (state, action) => {  if (!state.isAuthorized) return;  state.userName = action.payload;  },  setAvatar: (state, action) => {  if (!state.isAuthorized) return;  state.userAvatar = action.payload;  },  setPreview:(state,action) => {  if(!state.isAuthorized) return;  state.preview = action.payload;  },  setEmail: (state, action) => {  if (!state.isAuthorized) return;  state.userEmail = action.payload;  },  setPassword: (state, action) => {  if (!state.isAuthorized) return;  state.userPassword = action.payload;  },  }  });  // c and f slice  const appSlice = createSlice({  name: "app",  initialState,  reducers: {  setCart: (state, action) => {  state.cart = action.payload;  },  setFavorite:(state,action) => {  state.favorite = action.payload;  }    },  }); |

Листинг 3.7 – Структура кода CreatSlice

Для сохранения состаяния в Redux Persist были использваны специльные методы combineReducers для обьединения всех редьюсеров и дальнейшей передачи их в store. А для конкретного создания одного редьюсера был использован метод persistReducer для отслеживаемых редьюсеров. Код структуры persistReducer представлен в листинге 3.8

|  |
| --- |
| const rootReducer = combineReducers({  reducer: appSlice.reducer,  user: userSlice.reducer,  // comment:commentSlice.reducer,  comment:commentsListSlice.reducer,  });  const persistConfig = {  key: 'root',  storage,  }  const persistedReducer = persistReducer(persistConfig,rootReducer) |

Листинг 3.8 – Структура кода persistReducer

Затем редьюсер нужно поместить его в Store вместе с его действиями Actions. В Store будет находиться все состояние приложения. Для того, чтобы получить данные из Store нужно использовать useSelector. Для того, чтобы изменить данные в Store нужно использовать useDispatch. Код структуры Store будет показан в листинге 3.9.

|  |
| --- |
| export const {  setCart,setFavorite  } = appSlice.actions;  // actions for users  export const {  authorize,  unauthorize,  setName,  setAvatar,  setPreview,  setEmail,  setPassword  } = userSlice.actions  export const {  setCommentsList,  } = commentsListSlice.actions  // store  const store = configureStore({  reducer: persistedReducer,  middleware: (getDefaultMiddleware) =>  getDefaultMiddleware({  serializableCheck: {  ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],  },  }),  });  export const persistor = persistStore(store);  export default store; |

Листинг 3.9 – Структура кода Store

В этом разделе были рассмотрены основные положения по созданию глобального хранилища данных с помощью Redux-Toolkit и Redux Persist. Были рассмотрены основные понятия по созданию редьюсеров, получения данных из Store.

## 3.3 Добавление таблиц стилей Sass/CSS3

Для оформления стилей были применены SASS и CSS3, что обеспечивает возможность использования переменных, вложенности, наследования, импорта и экспорта. Пример кода, демонстрирующий применение SASS, представлен в листинге 3.14.

|  |
| --- |
| @import url('https://fonts.googleapis.com/css2?family=Libre+Baskerville&family=Lilita+One&display=swap');  @import url('https://fonts.googleapis.com/css2?family=Patua+One&family=Paytone+One&family=Roboto&family=Sarabun:wght@700&display=swap');  \*{  transition: 0.15s;  box-sizing: border-box;  color: #000;  // font-family: 'Lilita One', cursive;  font-family: 'Paytone One', sans-serif;  }  $currentColor: white;  body,  html {  margin: 0;  padding: 0;  padding-top: 28px;  padding-bottom: 15px;  overflow-x: hidden;  scroll-behavior: smooth;  background: linear-gradient(-45deg, #7cdde0, #d5e04b, #23a6d5, #16d7de);  background-size: 400% 400%;  animation: gradient 15s ease infinite;    }  .ht-over {  overflow: hidden;  }  .ht-no-over {  overflow: auto;  }  body {  display: flex;  justify-content: center;  #root {  border-radius: 20px;  width: 1280px;  height: 100%;  background: #FFF;  box-shadow: 0px 10px 20px 0px rgba(0, 0, 0, 0.04);  }  }  @keyframes gradient {  0% {  background-position: 0% 50%;  }  50% {  background-position: 100% 50%;  }  100% {  background-position: 0% 50%;  }  }  //#notistack-snackbar svg {  // fill: #f7f7f7;  //}  //  //#notistack-snackbar {  // color: white;  // font-weight: 200;  // font-size: 1.1rem;  //}  @media screen and (max-width:600px) {  body #root {  width: 1080px;  border-radius: 0;  }  body,  html {  padding-top: 0;  padding-bottom: 0;  }  } |

Листинг 3.14 – Структура кода SASS

И для остальных компонентов в основном использовался CSS3.

В этом разделе была рассмотрена структура SASS и CSS3, что такое SASS и CSS3 и как они используется в проекте. Пример кода, демонстрирующий применение CSS3, представлен в листинге 3.15.

|  |
| --- |
| .bike-desq {  font-family: 'Patua One', serif;  }  .comment h4 {  overflow: hidden;  text-overflow: ellipsis;  white-space: nowrap;  font-weight: 100;  margin: 0;  margin-top: 20px;  word-break: break-all;  font-family: 'Patua One', serif;  }  .carousel .slide img {  width: 100%;  }  .comment-avatar {  width: 42px;  border-radius: 20px;  }  .comm-from {  display: flex;  align-items: center;  gap: 5px;  }  .go946087465{  color: white;  }  .go946087465 svg path {  fill: white;  }  @media screen and (max-width:600px) {  .bike\_container {  flex-direction: column;  }  .info\_img {  width: 300px;  height: 250px;  }  .go-back {  width: 90%;  }  .type-container {  flex-direction: column;  gap: 15px;  width: 80%;  }  .new-text {  height: 100px;  }  .bike-desq {  font-weight: 100;  width: auto;  }  .bike-info-container {  margin-top: -80px;  }  }  @media (min-height:1000px) and (max-height:1300px) {  .bike\_container {  flex-direction: column;  }  .bike-info-container {  margin-top: -80px;      }  .carousel .slide img {  width: 50%;  }  } |

## 3.4 Выводы по разделу

Таким образом, были рассмотрены основные моменты, которые будут реализованы в проекте. В проекте будет реализована авторизация, регистрация, добавление товаров в корзину и в избранное, оформление заказа, получение информации о товарах, о пользователях, о корзине.

# 4 Тестирование веб-сайта

## 4.1 Адаптивный дизайн веб-сайта

В наше время наблюдается значительное увеличение числа пользователей, предпочитающих доступ в Интернет через мобильные устройства, и этот тренд продолжает расти. В связи с этим адаптация веб-сайта под небольшие экраны становится важной задачей, которую необходимо учитывать и в рамках курсового проекта.

Моя задача заключалась в создании веб-сайта, который автоматически подстраивается под различные типы экранов. Основными критериями адаптивности является отсутствие фиксированных размеров элементов страницы. Для реализации адаптивности были использованы медиа-запросы (@media) и единицы измерения, привязанные к параметрам области просмотра, представляющие собой процентные значения от текущей области просмотра браузера.

После создания основной структуры страницы на JSX и внешнего стилевого оформления возник вопрос о тестировании. Веб-сайт был протестирован в различных браузерах, и было отмечено, что во всех случаях сайт отображается однородно, подтверждая его адаптивность и согласованность визуального восприятия.

При отображении сайта на мобильных устройствах навигационное меню в верхнем колонтитуле меняется на столбцовый формат. На рисунке 4.1 можно увидеть пример адаптивного дизайна для мобильных устройств.



Рисунок 4.1 – Адаптивный главной страницы для телефонов

Также при использовании мобильного устройства блоки с товарми на странице для отображения располагаются друг под другом. На рисунке 4.2 можно увидеть пример.

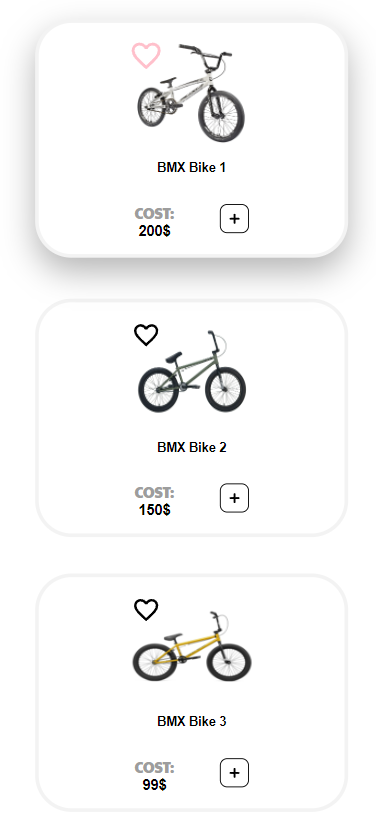


Рисунок 4.2 – Блоки с товарами на мобильных устройствах

И таким образом можно сказать, что все блоки располагаются друг за другом, потому что пользователю было все хорошо видно на его мобильном устройстве.

## 4.2 Кроссбраузерность веб-сайта

Имеет большое значение, чтобы наш веб-сайт был кроссбраузерным. Кроссбраузерность представляет собой характеристику веб-ресурса, обеспечивающую однаковое отображение и надежную работу во всех широко используемых браузерах. Это включает в себя отсутствие сбоев в функциональности, ошибок в верстке, а также обеспечение равнозначной читаемости контента на всех платформах. При проверке веб-сайта в Google Chrome никаких искажений текста, анимации, переходов не было выявлено. На рисунке 4.3 можно увидеть пример веб-страницы в Google Chrome.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рисунок 4.3 – Вид веб-страницы в Google Chrome а) мобильная версия; б) компьютерная версия

В Яндекс всё так же работает без неисправностей, весь контент обладает правильным шрифтом, не наблюдается сдвиг текста или отсутствие картинок. На рисунке 4.5 можно увидеть пример веб-страницы в Яндекс.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рис 4.5 – Вид веб-страницы в Яндекс а) мобильная версия; б) компьютерная версия

Также для тестирования сайта было использовано Linter. Linter - это инструмент, который проверяет код на наличие ошибок. В нашем случае он проверяет код на наличие ошибок в HTML и CSS.

При просмотре веб-сайта в браузерах, таких как Google Chrome и Яндекс, общий внешний вид веб-страницы остался неизменным. Основная цель тщательной проверки заключается в настройке всех параметров страниц сайта, оперативном выявлении и устранении дефектов, а также в адаптации к различным устройствам.

Динамика сайта на разных устройствах играет ключевую роль в его разработке, поэтому на этапе анализа производилось тщательное изучение поведения веб-сайта при использовании различных устройств, а также проведено тестирование в разнообразных браузерах.

Этот этап позволил создать адаптивный дизайн и обеспечить кроссбраузерность веб-сайта. Проведенная работа с сайтом обеспечивает удобство пользователям при навигации. Отсутствие лишних элементов упрощает использование сайта. На мобильных устройствах некоторые элементы меняют свое расположение, что улучшает удобство пользования.

В ходе тестирования были проверены все ссылки, включая модальные окна, а также протестированы формы регистрации пользователя и контент. Совместимость с браузерами (Google Chrome, Яндекс) также была подвергнута проверке.

## 4.3 Ручное тестирование

Ручное тестирование – это тестирование, которое выполняется вручную. В данном случае ручное тестирование было проведено с целью проверки веб-сайта на соответствие требованиям, а также на наличие ошибок.

Тестирование frontend части включает в себе проверку правильности осуществления перехода между страницами, которые совершают пользователи системы, также проверку правильности отправления отзывов пользователей, регистрации, авторизации, валидации и т.п.

Рассмотрим сценарий отправки отзыва. При не полном заполнении формы должно быть получено оповещение об этом (Валидация). Заполнение формы данными и уведомление изображено на рисунке 4.6.

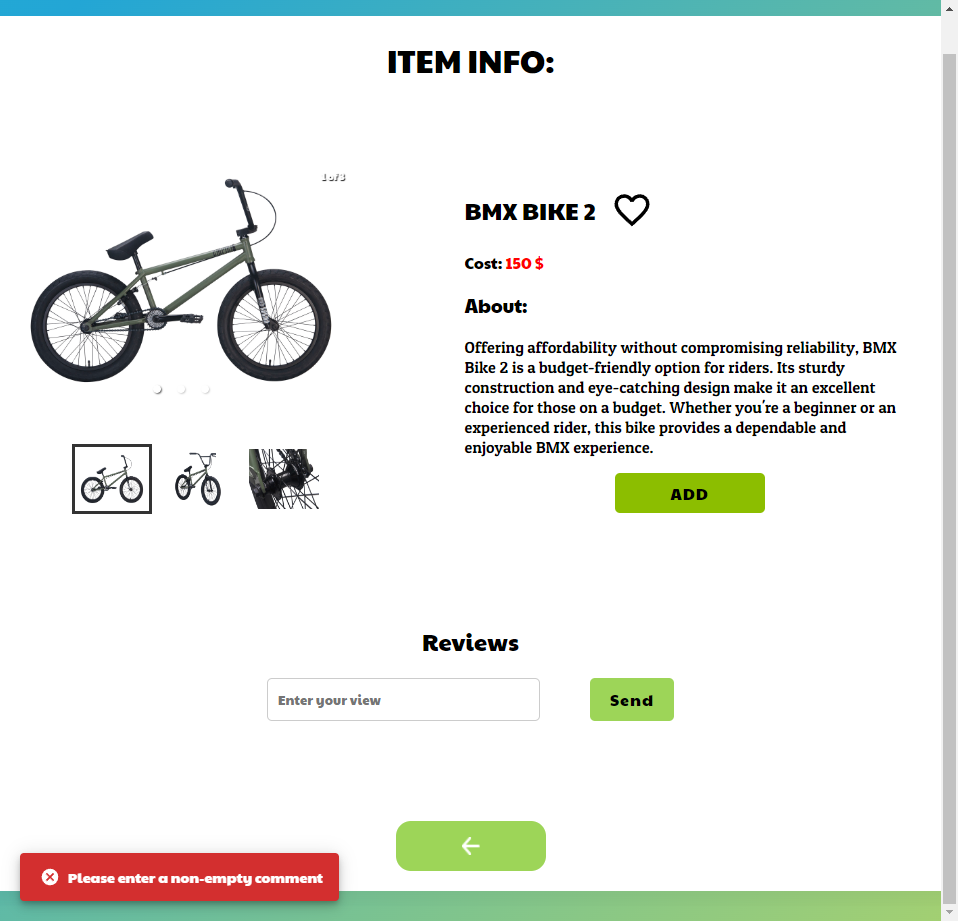


Рисунок 4.6 – Форма заполнения неверными данными

При попытке ввода еще одного отзыва от пользователя будет так же выдано уведомление об ошибке. Пример показан на рисунке 4.7

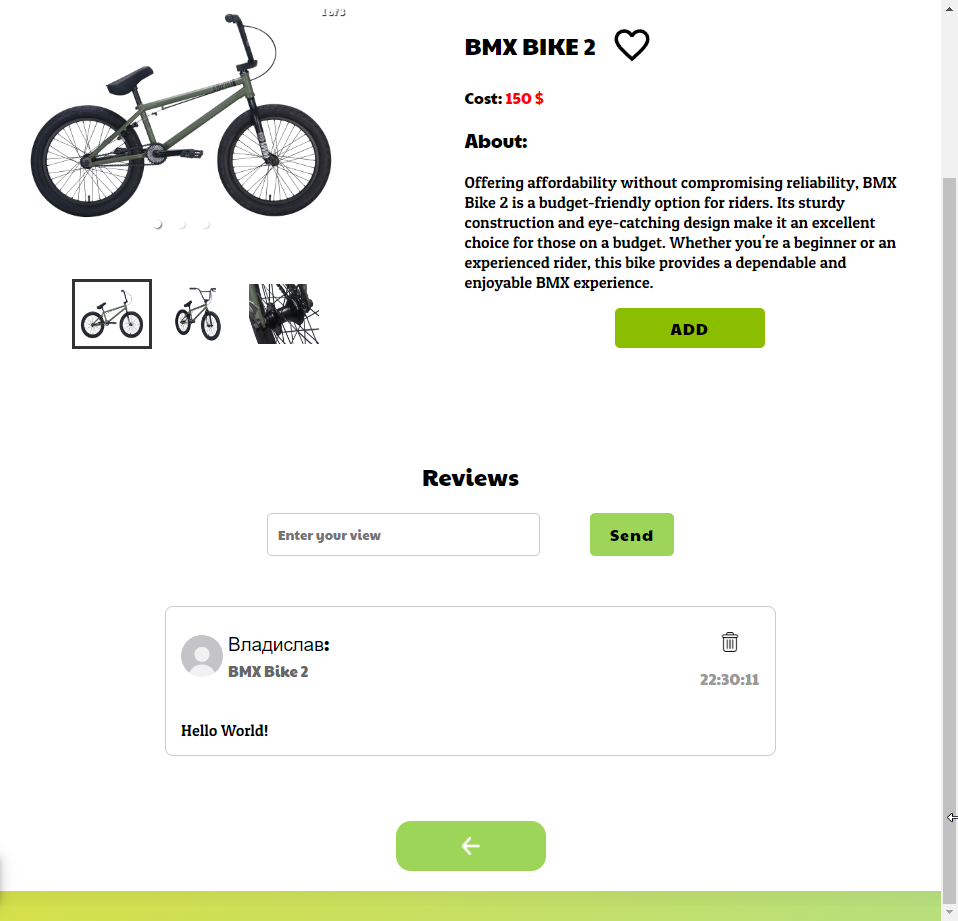


Рисунок 4.7 – Повторная отправка отзыва от одного пользователя

При попытке отправки отзыва незарегистрированным пользователем будет так же произведена валидация. Пример представлен на рисунке 4.8

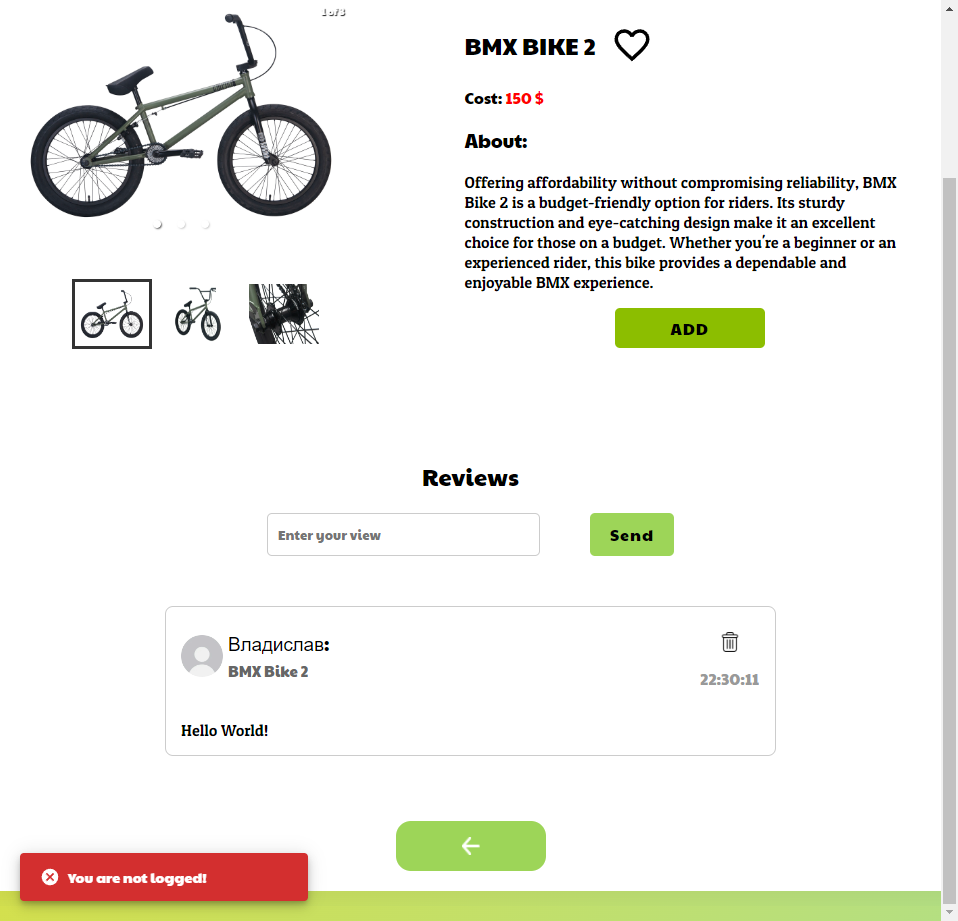


Рисунок 4.8 – Отправка отзыва незарегистрированным пользователем.

В ходе ручного тестирования были проведена валидация всех страниц сайта, проверены все ссылки, проверены все формы, проверен контент, проверена совместимость с браузерами. Тестирование функционала сайта было выполнено вручную с использованием следующие браузеров Google Chrome (42.0.2311.90) и Яндекс (22.1.0.2510)

Подводя итоги тестирования функционала сайта, можно сказать, что все функции сайта работают корректно. В ходе тестирования были проверены все ссылки (включая модальные окна), проверены формы отправки отзыва пользователя. Была выполнена проверка контента.

## 4.4 Выводы по разделу

В этом разделе было проведено тестирование функционала сайта. Были проверены все функции сайта, которые должны работать корректно, а также были проверены все элементы, которые должны быть доступны для пользователя. Не было обнаружено никаких ошибок, которые могли бы повлиять на работу сайта.

# 5 Руководство пользователю

Данный веб-сайт предназначен для заказа велосипедов BMX и комплектующих . Для того, чтобы совершить заказ, необходимо зарегистрироваться или авторизоваться на сайте. Пользователь может просматривать товары, добавлять их в корзину и в избранное, а также удалять. После того, как пользователь добавил все товары в корзину, он может оформить заказ. После оформления заказа пользователь может посмотреть историю заказов в своем профиле.

На странице Main пользователь может просматривать товары, которые находятся в наличии.

На странице BikePage пользователь может просматривать полную информацию о товаре с предоставленным функционалом.

На странице Cart пользователь может просматривать товары, которые он добавил в корзину, а также удалять их из корзины.

На странице Favorit пользователь может просматривать товары, которые он добавил в избранное, а также удалять их из избранных.

На Странице Login пользователь может авторизоваться или зарегистрироваться, просматривать свои заказы.

# Заключение

Целью курсового проекта было создание интернет-магазина велосипедов, предназначенного для пользователей, желающих приобрести товары по данной тематике в онлайн-режиме. В ходе курсовой работы были выполнены следующие задачи:

* Проанализировали и описали основные принципы работы React и Redux.
* Разработали приложение, используя React и Redux.
* Протестировали приложение.
* Провели анализ и оценку качества приложения.

А также были выполнены задачи по созданию интернет-магазина:

* Разработали регистрацию и авторизацию пользователей;
* Разработали возможность добавления и удаления товаров из корзины;
* Разработали возможность добавления и удаления товаров из избранного;
* Разработали структуру веб-сайта;
* Разработали возможность посмотреть информацию о товаре;
* Разработали возможность оставить отзыв о товаре;
* Разработали возможность заказа товара;
* Разработали возможность просмотра истории покупок;
* Разработали возможность редактирования профиля.

В результате выполнения курсового проекта по созданию интернет-магазина велосипедов были успешно достигнуты поставленные цели и выполнены задачи. Проанализированы и описаны основные принципы работы React и Redux, что позволило эффективно разработать приложение, используя современные технологии.

Созданное приложение прошло тщательное тестирование, что подтвердило его функциональность и надежность. Проведен анализ и оценка качества приложения, при этом особое внимание уделено аспектам регистрации и авторизации пользователей, управлению корзиной и избранным, а также другим функциональным возможностям интернет-магазина.

В рамках разработки магазина реализованы ключевые элементы, такие как добавление и удаление товаров из корзины и избранного, структура веб-сайта, просмотр информации о товаре, оставление отзывов, заказ товаров, просмотр истории покупок и редактирование профиля пользователя.

Общий результат работы позволяет утверждать, что цели проекта достигнуты, а интернет-магазин велосипедов предоставляет полный функционал для удобного и надежного онлайн-приобретения товаров по данной тематике.

# Список использованных источников

1. Stresshop [Электронный ресурс]. – Режим доступа: https://stressshop.ru/– Дата обращения 03.12.2023.
2. iBike [Электронный ресурс]. – Режим доступа: https://ibike.by/– Дата обращения 03.12.2023.
3. Alienbike [Электронный ресурс]. – Режим доступа: https://www.alienbike.ru/– Дата обращения 03.12.2023.
4. Документация по React JS [Электронный ресурс]. - Режим доступа: https://reactjs.org/. – Дата обращения 03.12.2023.
5. Документация по Redux Persist [Электронный ресурс]. – Режим доступа:

www.npmjs.com/package/redux-persist/ – Дата обращения 03.12.2023.

1. Документация по Figma [Электронный ресурс]. - Режим доступа: <https://help.figma.com>. – Дата обращения 03.12.2023.
2. Документация по Redux/Redux-tolkit [Электронный ресурс]. - Режим доступа: https://redux-toolkit.js.org/ . – Дата обращения 03.12.2023.
3. Документация по Sass/Scss [Электронный ресурс]. - Режим доступа : <https://sass-scss.ru/guide/> – Дата обращения 03.12.2023.

# Приложение А Макет структуры веб-сайта

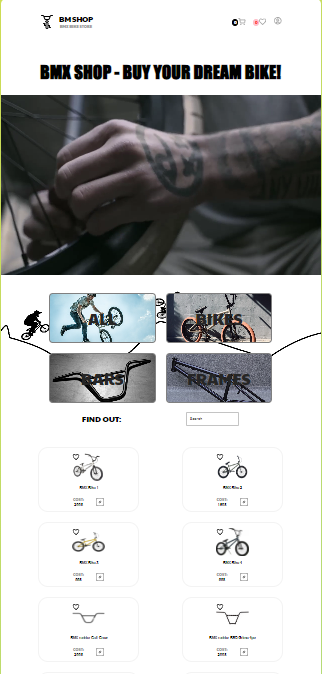
 

Рисунок 1 – Макет главной страницы

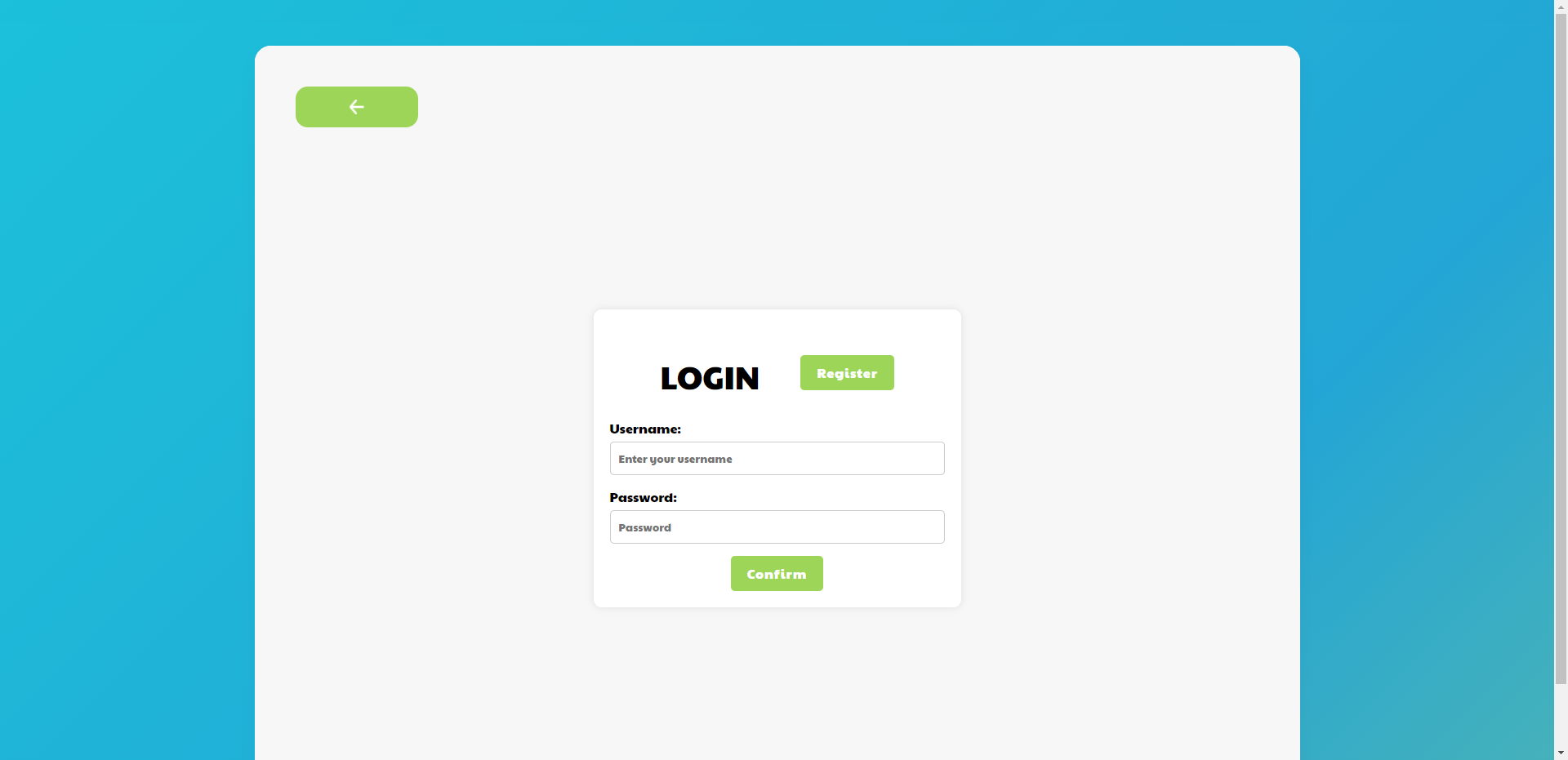
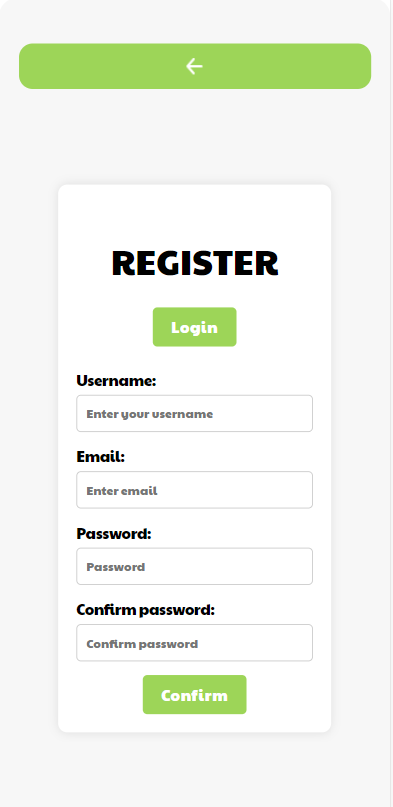
 

Рисунок 2 – Макет страницы регистрации

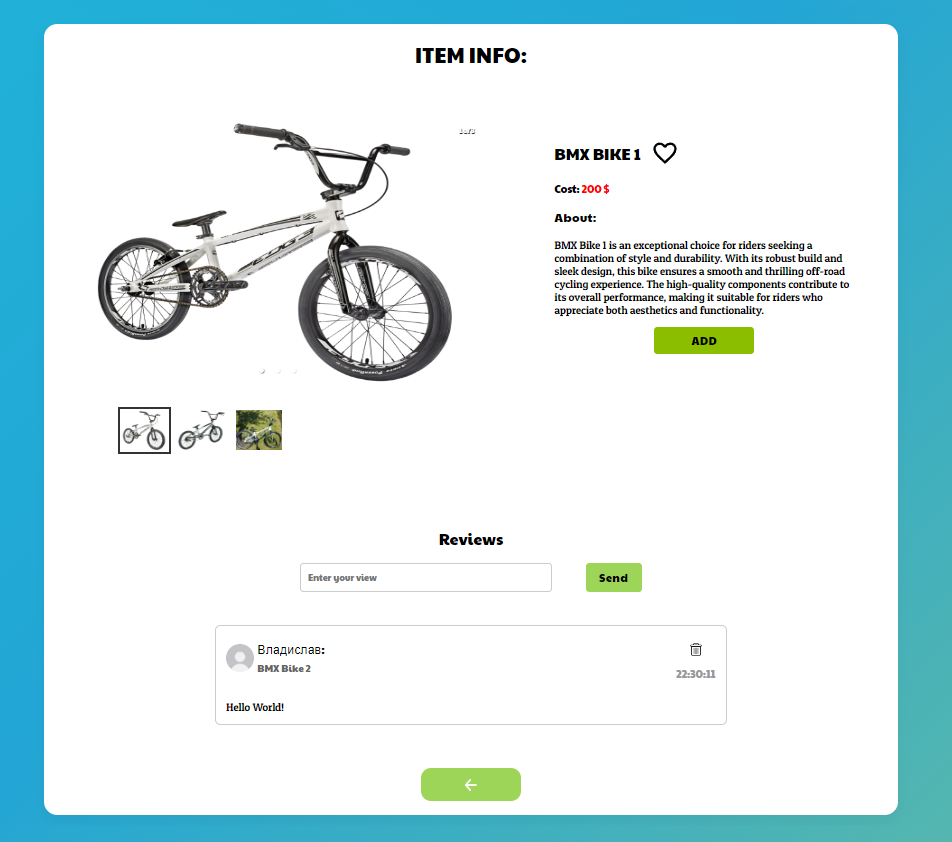
 

Рисунок 3 – Макет страницы товара

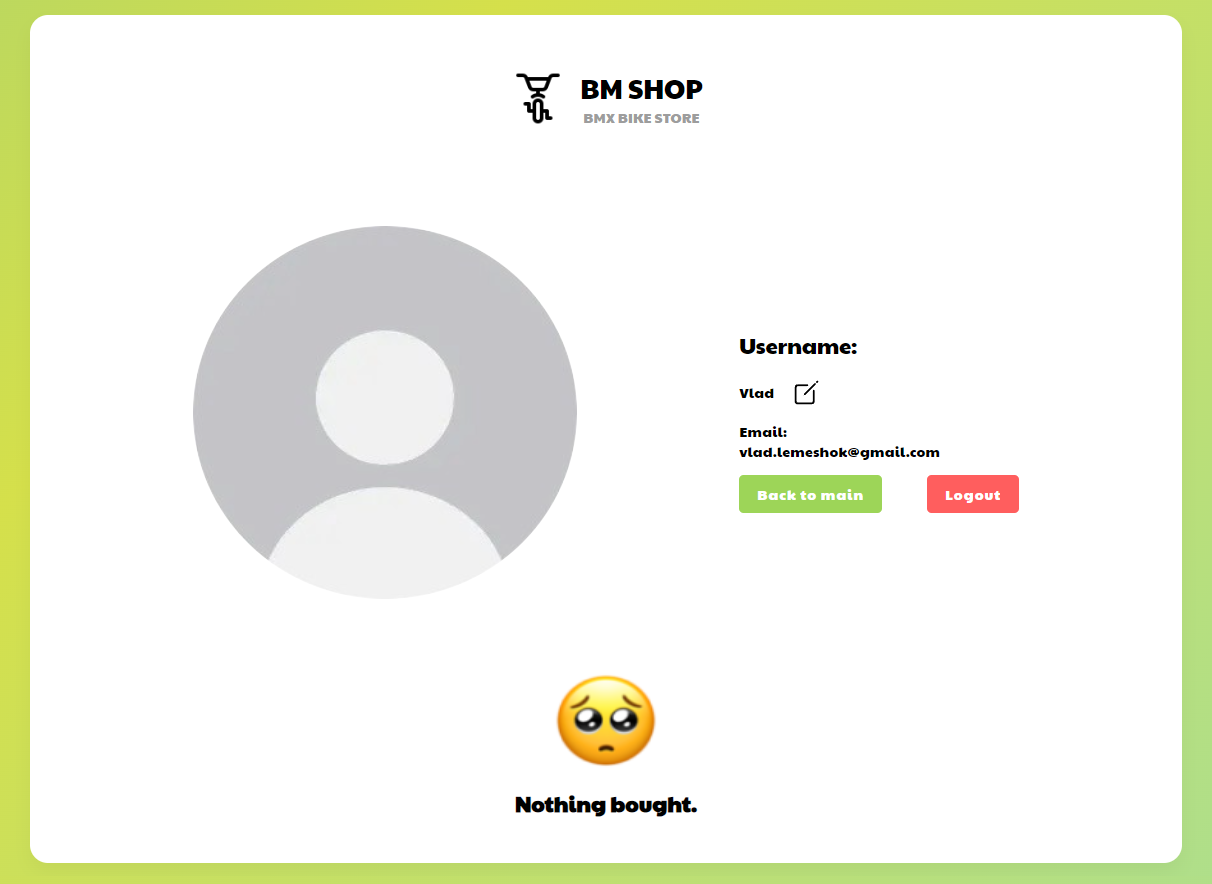
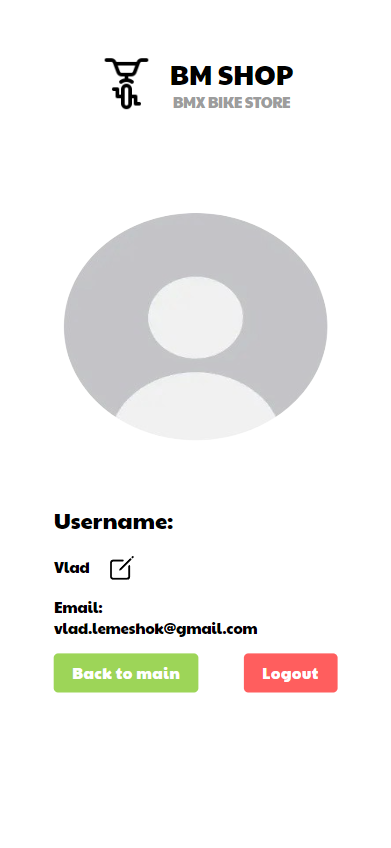
 

Рисунок 4 – Макет страницы профиля

# Приложение Б Листинг JSX-документа

|  |
| --- |
| import React, { useState } from "react";  import Header from "./Header/Header";  import Main from "./Main/Main";  import Footer from "./Footer/Footer";  import Cart from "./Cart/Cart";  import BikesData from "./AllData.json";  import { SnackbarProvider } from "notistack";  import Login from "./Accout/Login";  import {BrowserRouter as Router, Route, Routes} from "react-router-dom";  import { useDispatch, useSelector } from "react-redux";  import { setCart, setFavorite } from "./Redux/Store";  import BikePage from "./BikePage/BikePage";  import ErrorTab from "./404/404";  function App (){  const [display,setDisplay] = useState('none');  // const [cart,setCart] = useState([]);  // const [favorite,setFavorite] = useState([])  const [bikes,setBikes] = useState(BikesData);  const [added , setAdded] = useState(false);  const [showMainContent, setShowMainContent] = useState(true);  const [displayFavContainer, setDisplayFavContainer] = useState(true);  const [boughtItems, setBoughtItems] = useState([]);  const [overFlow , setOverflow] = useState('auto')  const dispatch = useDispatch();  const cart = useSelector((state) => state.reducer.cart);  const favorite = useSelector((state) => state.reducer.favorite)  const isAuthorized = useSelector((state) => state.user.isAuthorized);  // FILTER CATEGORIES ARRAYS :    const handleCategoryChange = (category) => {  let jsonData = require("./AllData.json");  switch(category) {  case "Bikes":  jsonData = jsonData.filter(bike => bike.name.toLowerCase().includes('bike')).sort((a , b) => a.id - b.id);  setBikes(jsonData)  break;  case "Rudders":  jsonData = jsonData.filter(bike => bike.name.toLowerCase().includes('rudder')).sort((a,b) => a.id - b.id);  setBikes(jsonData)  console.log(jsonData)  break;  case "Frames":  jsonData = jsonData.filter(bike => bike.name.toLowerCase().includes('frame')).sort((a , b) => a.id - b.id);  setBikes(jsonData)  break;  default:  jsonData = require("./AllData.json");  setBikes(jsonData);  break;  }  }  // -----------------------    const toggleFavContainer = () => {  setDisplayFavContainer(!displayFavContainer);  };  const toggleMainContent = () => {  setShowMainContent(!showMainContent);  };  // const addToCart = (bike) => {  // setCart((prev) => ([...prev,bike]))  // }  const clearCart = () => {  dispatch(setCart([]))  }  const clearFav = () => {  dispatch(setFavorite([]))  }  const addToBuys = () => {  if(isAuthorized){  cart.forEach((bike) => {  setBoughtItems((prevItems) => [...prevItems,bike])  });  dispatch(setCart([]))  }  else{  console.log("ERR")  }  }  const addFavBtn = (bike) => {  const isAlreadyFav = favorite.some((favBike) => favBike.id === bike.id);  if(isAlreadyFav){  delFavItem(bike.id);  }  else {  // setFavorite((prev) => ([...prev,bike]) )  dispatch(setFavorite([...favorite,bike]))  }  }  const addToCart = (bike) => {  const isAlreadyInCart = cart.some((cartBike) => cartBike.id === bike.id)  if(!cart.filter(bike => bike.name.toLowerCase().includes(bike.name))){  setAdded(false)  }  else { setAdded(true) }  if(isAlreadyInCart){  delFromCart(bike.id);  }  else {  dispatch(setCart([...cart, bike]));  }  }  // const delFromCart = (id) => {  // setCart((prevCart) => {  // const idx = prevCart.findIndex((bike) => bike.id === id);  // if(idx !== -1){  // const newArray = [...prevCart.slice(0,idx),...prevCart.slice(idx+1)];  // return newArray;  // }  // return prevCart;  // })  // }  const delFromCart = (id) => {  const idx = cart.findIndex((bike) => bike.id === id);  if (idx !== -1) {  const newArray = [...cart.slice(0, idx), ...cart.slice(idx + 1)];  dispatch(setCart(newArray));  setAdded(false);  }  }  // --------------------------    const delFavItem = (id) => {  // setFavorite((prevFav) => {  const idx = favorite.findIndex((bike) => bike.id === id);  if(idx !== -1){  const newFavArray = [...favorite.slice(0,idx),...favorite.slice(idx+1)];  // return newFavArray;  dispatch(setFavorite(newFavArray))  }    // })  }  const displayBlock = () => {  setDisplay('block');  setOverflow('hidden');  }  const displayNone = () => {  setDisplay('none');  setOverflow('auto');    }  document.documentElement.style.overflow = overFlow  // -----------------------------------------|  const headerProps = {  toggleFavContainer:toggleFavContainer,  displayChange:displayBlock ,  displayMain:toggleMainContent,  bikes:cart,  added:favorite,  }  const cartProps = {  bikes:cart,  displayNone:displayNone,  delFromCart:delFromCart,  clearCart:clearCart,  addToBuys :addToBuys,    }  const mainProps = {  displayMain: toggleMainContent,  toggleFavContainer: toggleFavContainer,  showMainContent: showMainContent,  displayFavContainer: displayFavContainer,  bikes: bikes,  addToCart: addToCart,  addFavBtn: addFavBtn,  favorite: favorite,  delFavItem: delFavItem,  handleCategoryChange: handleCategoryChange,  cart:cart,  added:added,  clearFav:clearFav,  };  const loginProps = {  bikes: bikes,  boughtItems: boughtItems,  clearFav:clearFav,  clearCart:clearCart,  setBoughtItems:setBoughtItems,  }  const bikeProps = {  bikes:bikes,  addToCart:addToCart,  added:added,  addFavBtn: addFavBtn,  favorite: favorite,  setOverflow:setOverflow,    }  return(  <Router>  <div className="app-body">  <div className="overlay" style={{ display: display }}>  <Cart {...cartProps} />  </div>  <Routes>  <Route path="/login" element={<Login {...loginProps}/>} />  <Route path="/bikePage/:id" element={ <SnackbarProvider maxSnack={3}> <BikePage {...bikeProps}/> </SnackbarProvider>} />  <Route path="/" element={<>  <Header {...headerProps} />  <SnackbarProvider maxSnack={3}>  <Main {...mainProps} />  </SnackbarProvider>  <Footer />  </>} />  <Route path="/\*" element={<ErrorTab/>}/>  </Routes>  </div>  </Router>  )  }  export default App; |

Листинг 1 - структура JSX—документа

# Приложение В Листинг (SASS)

|  |
| --- |
| @import url('https://fonts.googleapis.com/css2?family=Libre+Baskerville&family=Lilita+One&display=swap');  @import url('https://fonts.googleapis.com/css2?family=Patua+One&family=Paytone+One&family=Roboto&family=Sarabun:wght@700&display=swap');  \*{  transition: 0.15s;  box-sizing: border-box;  color: #000;  // font-family: 'Lilita One', cursive;  font-family: 'Paytone One', sans-serif;  }  $currentColor: white;  body,  html {  margin: 0;  padding: 0;  padding-top: 28px;  padding-bottom: 15px;  overflow-x: hidden;  scroll-behavior: smooth;  background: linear-gradient(-45deg, #7cdde0, #d5e04b, #23a6d5, #16d7de);  background-size: 400% 400%;  animation: gradient 15s ease infinite;    }  .ht-over {  overflow: hidden;  }  .ht-no-over {  overflow: auto;  }  body {  display: flex;  justify-content: center;  #root {  border-radius: 20px;  width: 1280px;  height: 100%;  background: #FFF;  box-shadow: 0px 10px 20px 0px rgba(0, 0, 0, 0.04);  }  }  @keyframes gradient {  0% {  background-position: 0% 50%;  }  50% {  background-position: 100% 50%;  }  100% {  background-position: 0% 50%;  }  }  @media screen and (max-width:600px) {  body #root {  width: 1080px;  border-radius: 0;  }  body,  html {  padding-top: 0;  padding-bottom: 0;  }  } |

Листинг 2 - структура SASS

# Приложение Г Листинг Redux

|  |
| --- |
| import { configureStore, createSlice, combineReducers } from "@reduxjs/toolkit";  import {  persistStore,  persistReducer,  FLUSH,  REHYDRATE,  PAUSE,  PERSIST,  PURGE,  REGISTER  } from "redux-persist";  import storage from "redux-persist/lib/storage";  // упрощающая настройку хранилища с настройками по умолчанию.  const initialState = {  cart: [],  favorite: [],  userName:'',  userEmail:'',  userPassword:'',  userAvatar:'/img/avatarDefault.jpg',  preview:null,  isAuthorized: false,  // comments:[],  commentsList:[],  };  const commentsListSlice = createSlice({  name:'comment',  initialState,  reducers:{  setCommentsList:(state,action) => {  state.commentsList = action.payload  },  }  })  // user slice  const userSlice = createSlice({  name: 'user',  initialState,  reducers: {  authorize: (state, action) => {  state.isAuthorized = true;  state.userName = action.payload.userName;  state.userEmail = action.payload.userEmail;  state.userPassword = action.payload.userPassword;  state.userAvatar = '/img/avatarDefault.jpg';  },  unauthorize: (state) => {  state.isAuthorized = false;  },  setName: (state, action) => {  if (!state.isAuthorized) return;  state.userName = action.payload;  },  setAvatar: (state, action) => {  if (!state.isAuthorized) return;  state.userAvatar = action.payload;  },  setPreview:(state,action) => {  if(!state.isAuthorized) return;  state.preview = action.payload;  },  setEmail: (state, action) => {  if (!state.isAuthorized) return;  state.userEmail = action.payload;  },  setPassword: (state, action) => {  if (!state.isAuthorized) return;  state.userPassword = action.payload;  },  }  });  // c and f slice  const appSlice = createSlice({  name: "app",  initialState,  reducers: {  setCart: (state, action) => {  state.cart = action.payload;  },  setFavorite:(state,action) => {  state.favorite = action.payload;  }    },  });  const rootReducer = combineReducers({  reducer: appSlice.reducer,  user: userSlice.reducer,  // comment:commentSlice.reducer,  comment:commentsListSlice.reducer,  });  const persistConfig = {  key: 'root',  storage,  }  const persistedReducer = persistReducer(persistConfig,rootReducer)  // actions for cart and favs  export const {  setCart,setFavorite  } = appSlice.actions;  // actions for users  export const {  authorize,  unauthorize,  setName,  setAvatar,  setPreview,  setEmail,  setPassword  } = userSlice.actions  export const {  setCommentsList,  } = commentsListSlice.actions  // store  const store = configureStore({  reducer: persistedReducer,  middleware: (getDefaultMiddleware) =>  getDefaultMiddleware({  serializableCheck: {  ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],  },  }),  });  export const persistor = persistStore(store);  export default store; |

Листинг 3 - структура Redux