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»

**Исполнитель**

студент 2 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Коренчук

подпись, дата

**Руководитель**

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.С. Кантарович

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

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

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

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. С. Кантарович

подпись дата инициалы и фамилия

Минск 2022

**Содержание**

[Введение 5](#_Toc122376412)

[1 Постановка задач и обзор аналогичных решений 6](#_Toc122376413)

[1.1 Актуальность темы 6](#_Toc122376414)

[1.2 Обзор аналогичных решений 6](#_Toc122376415)

[1.3 Постановка задачи 10](#_Toc122376416)

[1.4 Выводы по разделу 10](#_Toc122376417)

[2 Проектирование веб-сайта 11](#_Toc122376418)

[2.1 Выбор средств реализации 11](#_Toc122376419)

[2.2 Проектирование uml-диаграммы 12](#_Toc122376420)

[2.3 Проектирование базы данных 13](#_Toc122376421)

[2.4 Проектирование страниц 14](#_Toc122376422)

[2.5 Выбор стилевого оформления 15](#_Toc122376423)

[2.6 Разработка пользовательских элементов 16](#_Toc122376424)

[2.7 Выводы по разделу 18](#_Toc122376425)

[3 Реализация структуры веб-сайты 19](#_Toc122376426)

[3.1 Структура JSX-документа 19](#_Toc122376427)

[3.2 Структура Redux-Toolkit 24](#_Toc122376428)

[3.3 Реализация серверной части 26](#_Toc122376429)

[3.4 Структура базы данных 28](#_Toc122376430)

[3.5 Добавление таблиц стилей Sass 29](#_Toc122376431)

[3.6 Выводы по разделу 30](#_Toc122376432)

[4 Тестирование веб-сайта 31](#_Toc122376433)

[4.1 Адаптивный дизайн веб-сайта 31](#_Toc122376434)

[4.2 Кроссбраузерность веб-сайта 32](#_Toc122376435)

[4.3 Ручное тестирование 34](#_Toc122376436)

[4.4 Выводы по разделу 36](#_Toc122376437)

[5 Руководство пользователю 37](#_Toc122376438)

[Заключение 38](#_Toc122376439)

[Список использованных источников 39](#_Toc122376440)

[Приложение А Макет структуры веб-сайта 40](#_Toc122376441)

[Приложение Б Листинг JSX-документа 44](#_Toc122376442)

[Приложение В Листинг (SASS) 46](#_Toc122376443)

[Приложение Г Листинг Redux 52](#_Toc122376444)

# Введение

В настоящее время количество новых сайтов увеличивается с неимоверной скоростью. Именно поэтому, основной задачей, ставящейся при разработке сайта – сделать его максимально привлекательным для потенциальных пользователей и придать ему индивидуальность. На заре развития интернета можно было привлечь достаточно большое количество посетителей на сайт, не прикладывая к его разработке больших усилий и без особых материальных затрат. Сейчас же, в условиях возросшей конкуренции, функциональность сайта, его стиль и запоминаемость играют очень большую роль. Ведь независимо от того создается сайт для представления какой-либо фирмы или просто для заработка на рекламе, основной его функцией является привлечение как можно большей аудитории. Поэтому для достижения успеха важен правильный подход к выбору стратегии и исполнителей.

Профессиональные студии веб – дизайна предлагают создание сайтов различного предназначения, это может быть корпоративныый сайт, сайт – визитка, интернет – магазин, сайты для ведения бизнеса как онлайн, так и в реальном мире.

Наше время называют «информационным веком». Это название возникло потому, что самым важным, ценным и необходимым ресурсом является информация. Обладание информацией и умение своевременно, быстро и четко донести необходимую информацию до клиентов – ключ к успеху в реалиях современного бизнеса. В настоящий момент именно Интернет, способен оперативно и массово передавать текст, звук, изображения и даже видео-ролики, т.е. практически любую информацию. C точки зрения пользователя, Интернет – это огромный информационный ресурс, в котором можно найти все, что угодно: от прогноза погоды до личных предпочтений голливудских кинозвезд. Аудитория сети Интернет растет с каждым годом. Интернет – самое быстроразвивающееся средство передачи информации за все историю человечества. Пользователи сети интернет являются целью особого интереса для рекламодателей, потому что факт остается фактом – среди пользователей сети куча потенциальных клиентов. Актуальность темы интернет-магазина в современном мире обусловлена тем, что сегодня все больше людей предпочитают покупать товары через интернет, а не в традиционных магазинах. Это связано с тем, что в интернет-магазинах можно купить товары по более низким ценам, чем в магазинах, а также с тем, что в интернет-магазинах можно купить товары, которые в магазинах не продается.

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 купить различные товары. 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 Ozon [1].

Ozon является одним из крупнейших интернет-магазинов в России. Основным преимуществом данного сайта является его удобство и простота использования. На главной странице сайта можно увидеть разделы, которые помогут пользователю найти нужный товар. На рисунке 1.1 представлено оформление сайта. Также все товары разделены по категориям, что позволяет пользователю найти нужный товар быстрее. Веб-сайт очень простой и удобный для пользователя, а также имеет красивый дизайн. Шрифт читабельный. Сайт динамический, интерактивный. Также есть анимация при наведении на картинки. Сайт использует два основных цвета (синий и белый), что положительно сказывается на общем восприятии сайта. На этом сайте есть уникальная функция, которая позволяет пользователю сравнить товары, которые он хочет купить. Еще можно отметить, что на сайте есть режим для слабовидящих, который позволяет пользователю увеличить шрифт и контрастность. Также можно добавить товары в избранное, чтобы не потерять их, причем без регистрации.

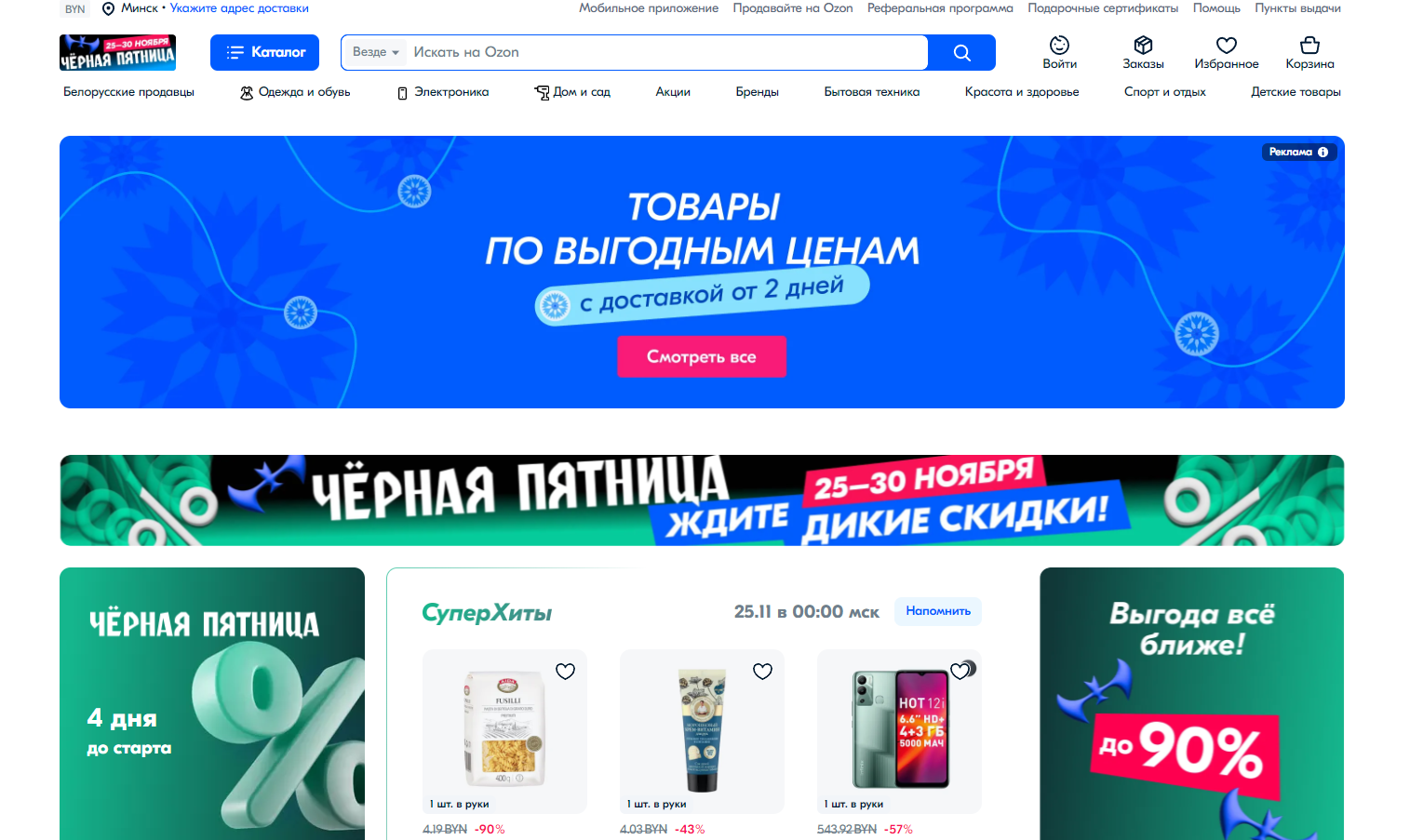


Рисунок 1.1 – Главная страница сайта Ozon

Веб-сайт имеет адаптивный дизайн, который позволяет использовать сайт на мобильных устройствах, горизонтальное меню становится выпадающим, текст остаётся читабельным. Дизайн представлен на рисунке 1.2.

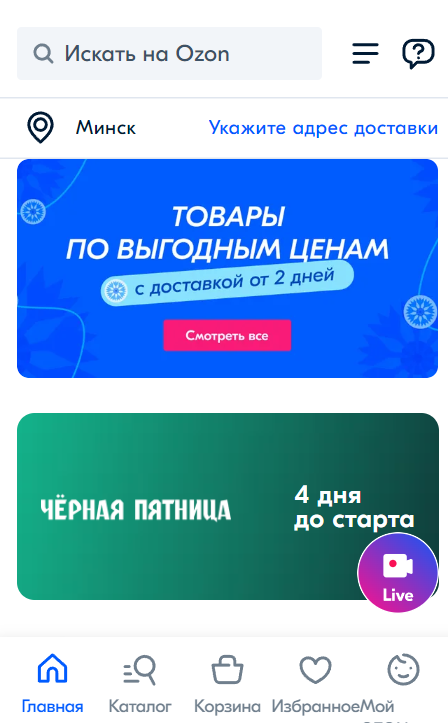


Рисунок 1.2 – Мобильная версия сайта Ozon

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

Аналог №2 – сайт Wildberries [2].

Следующим аналогом будет сайт Wildberries, который является одним из крупнейших интернет-магазинов в России. На сайте можно купить разные вещи начиная от одежды, заканчивая техникой. На сайте есть разделы, в которых можно выбрать нужный товар. Веб-сайт очень удобен для пользователя, так как он имеет простой и понятный интерфейс. Также пользователь может быстро найти нужный товар, так как на сайте есть поиск по названию товара и по категориям. Все товары имеют подробное описание, также есть отзывы о товаре, которые позволяют пользователю сделать правильный выбор. Дизайн сайта очень привлекательный, так как он имеет яркие цвета и красивые фотографии товаров. Дизайн представлен на рисунке 1.3. Одной из особенностей сайта является то, что сайт будет показывать товары, которые пользователь просматривал ранее, таким образом пользователь сможет быстро найти нужный товар. Еще из особенностей данного сайта является то, что на сайте есть быстрый поиск по названию товара. Отличие данного поиска от обычного поиска в том, что он показывает товары вместе с ценой.

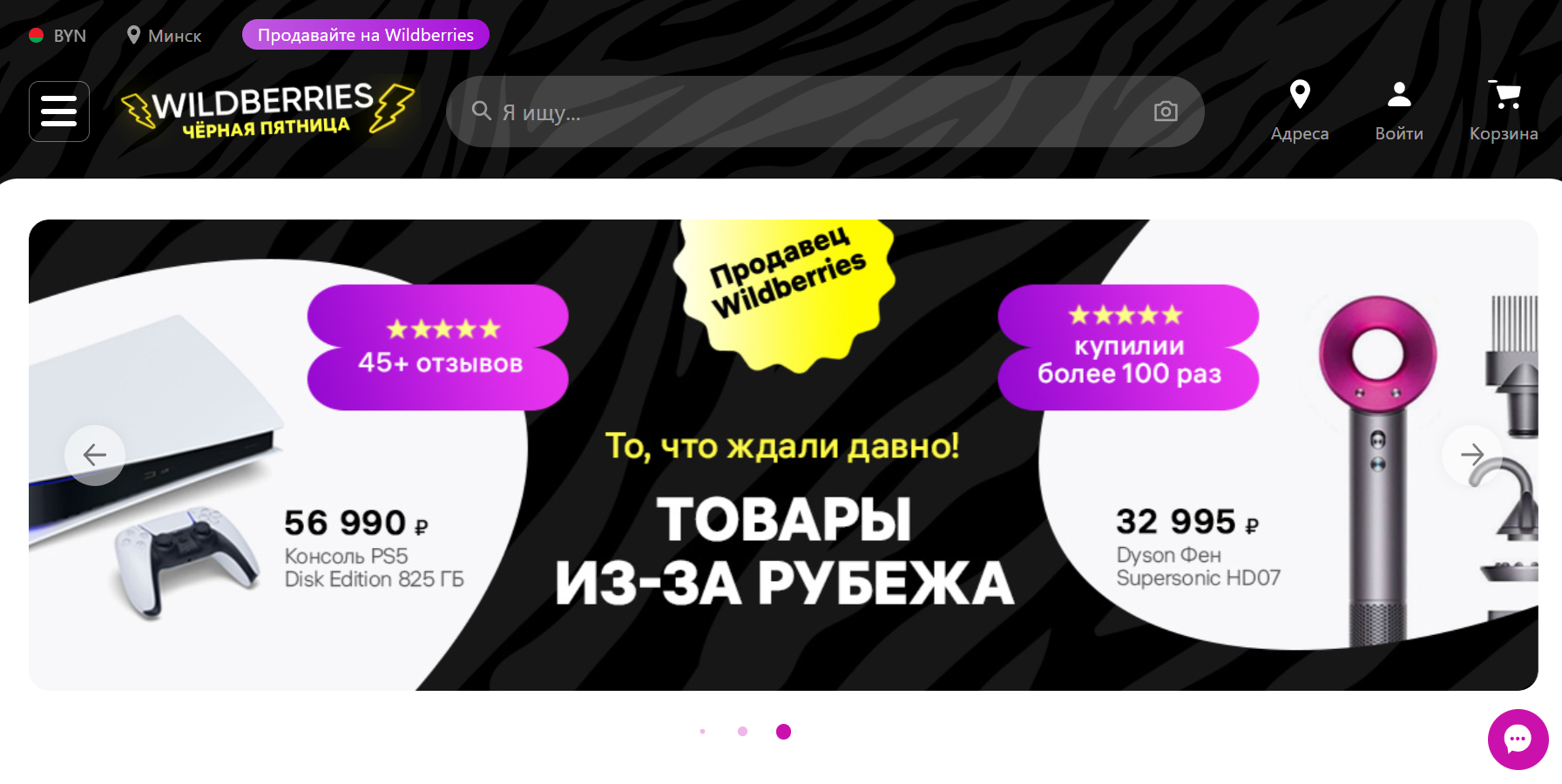


Рисунок 1.3 – Главная страница сайта Wildberries

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.

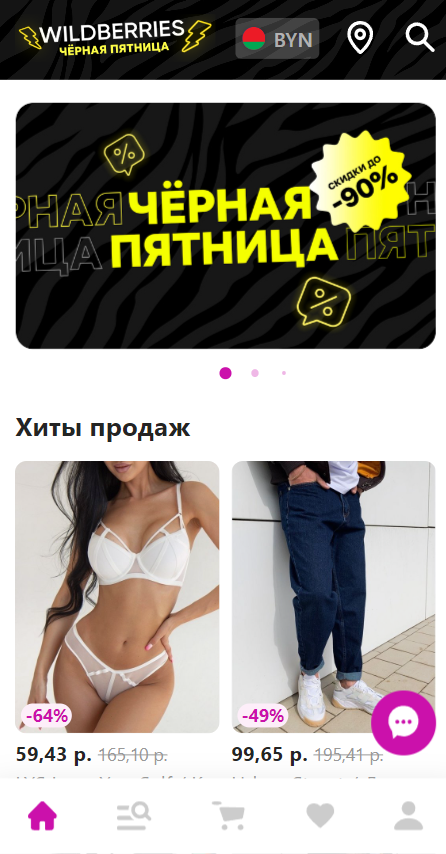


Рисунок 1.4 – Мобильная версия сайта Wildberries

Из недостатков можно выделить то, что сайт имеет много рекламы, которая может отвлекать пользователя от основного контента. Данный сайт был разработан на таких технологи как HTML, CSS, JavaScript, PHP, MySQL.

Аналог №3 – сайт Aliexpress [3].

Aliexpress – сайт, который является крупнейшим интернет-магазином в мире. На сайте можно купить товары из Китая, а также из других стран. На сайте можно найти товары по различным категориям, а также по различным ценам. Данный сайт имеет большое количество посетителей. На главной странице сайта можно найти различные товары, которые можно купить Дизайн сайта простой и удобный для пользователя. Цвета сайта яркие и не раздражают глаза. Все кнопки сайта имеют одинаковый размер, что упрощает навигацию по сайту. Из недостатков сайта можно выделить то, что сайт очень загружен и не всегда удобен для пользователя, а также не всегда можно найти нужный товар. Одним из уникальных особенностей сайта является то, что на сайте можно быструю регистрацию через социальные сети. Aliexpress был создан в 2009 году и был реализован следующими технологиями HTML, CSS, JavaScript, PHP, MySQL, Bootstrap, jQuery, Ajax. На рисунке 1.5 представлено оформление сайта Aliexpress.

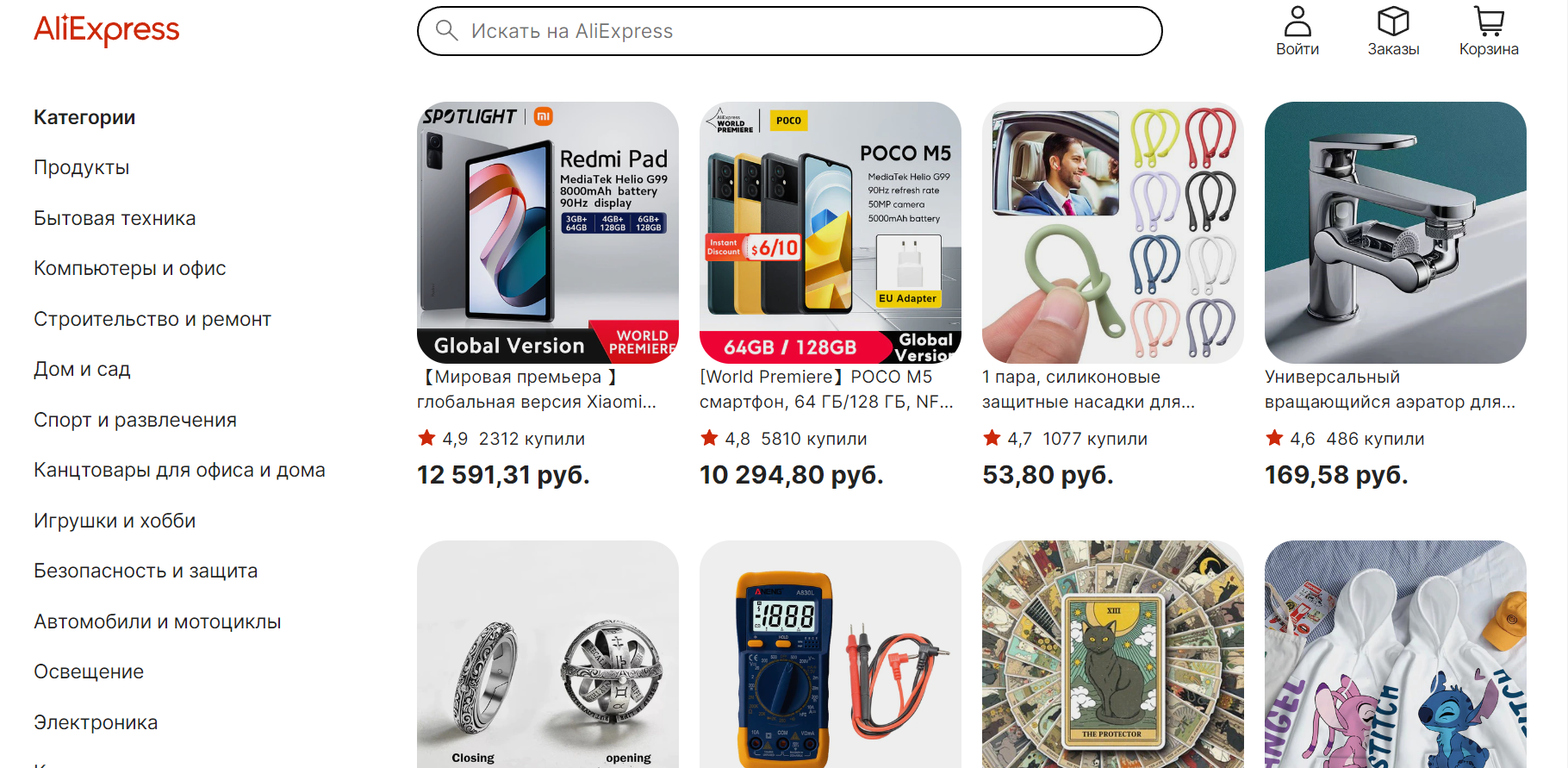


Рисунок 1.5 – Главная страница сайта Aliexpress

Также данный сайт кроссплатформенный, а также адаптирован под мобильные устройства. На рисунке 1.6 представлено оформление на мобильных устройствах.

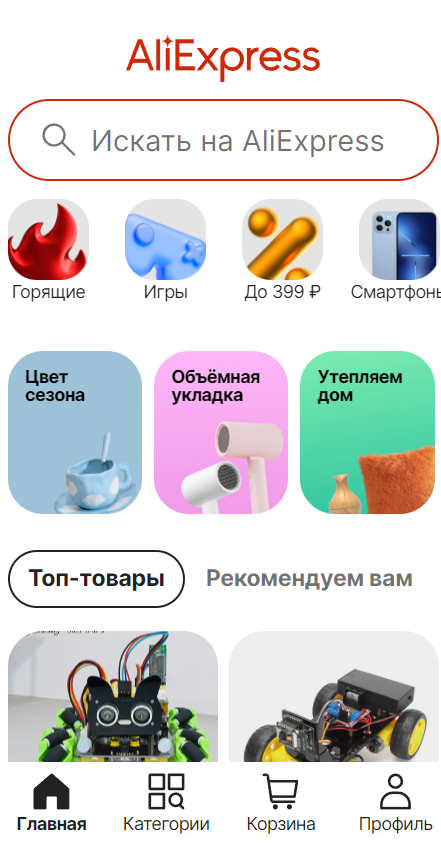


Рисунок 1.6 – Мобильная версия сайта Aliexpress

И таким образом, было проанализировано 3 сайта, которые являются аналогами нашего будущего сайта. 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 структуру веб-сайта;
* разработать возможность оставить отзыв о товаре;
* разработать возможность заказа товара;
* разработать возможность просмотра истории заказов.

## 1.4 Выводы по разделу

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

# 2 Проектирование веб-сайта

## 2.1 Выбор средств реализации

Главной задачей сайта является предоставление информации о товарах, которые можно купить. Данный интернет-магазин будет предназначен для людей, которые хотят купить различные товары. Сайт будет адаптирован под мобильные устройства, а также будет кроссплатформенным.

Интернет-магазин будет разработан в текстовом редакторе Visual Studio Code (1.73.1). Страницы сайта могут просматриваться в таких браузерах как Internet Explorer, Microsoft Edge, Safari, Opera, Mozilla Firefox, 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:

* Front-end.
* Back-end.
* База данных.

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), TypeScript (3.7.5), Redux-Toolkit (1.3.5), Axios (0.19.2), React Router (5.1.2).

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 — это библиотека JavaScript, предназначенная для разработки пользовательских интерфейсов. В отличие от других библиотек JavaScript React JS использует компонентный подход, который позволяет разрабатывать сложные интерфейсы из простых компонентов. С помощью React разработчики создают веб-приложения, которые изменяют отображение без перезагрузки страницы. Благодаря этому приложения быстро реагируют на действия пользователя, например, заполнение форм, применение фильтров, добавление товаров в корзину и так далее. React применяют для отрисовки компонентов пользовательского интерфейса. Также библиотека может полностью управлять фронтендом. В этом случае React используют с библиотеками для управления состоянием и роутинга, например, Redux и React Router.

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.

TypeScript — это язык программирования, который является надмножеством JavaScript. TypeScript добавляет в JavaScript поддержку статической типизации и классов, что позволяет разработчикам писать более качественный код. TypeScript компилируется в JavaScript, поэтому его можно использовать в любом проекте, который использует JavaScript.

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. Благодаря React Router можно создавать SPA (Single Page Application), которые не перезагружают страницу при переходе по ссылкам.

Axios — это библиотека для работы с HTTP-запросами. Axios позволяет создавать запросы к API, которые будут возвращать промисы, которые можно обрабатывать с помощью async/await.

Back-end - это часть приложения, которая отвечает за обработку данных, которые приходят от клиента и возвращают данные клиенту. Также Back-end отвечает за работу с базой данных, которая хранит данные приложения. Для создания Back-end будет использованы следующие технологии: Node.js (12.16.1), Express (4.17.1).

Node.js — это JavaScript-окружение, которое позволяет запускать JavaScript-код на сервере. Node.js позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

Express — это фреймворк для Node.js, который позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

База данных - это совокупность взаимосвязанных данных. Для создания базы данных будет использована PostgreSQL (12.2).

PostgreSQL — это объектно-реляционная система управления базами данных (СУБД), которая позволяет хранить данные в виде таблиц. PostgreSQL позволяет создавать базы данных, которые будут хранить данные 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.

Подводя итог, можно сказать, что в данном проекте будет использована PERN-стек технологий, что позволит создать приложение, которое будет работать на всех платформах, которые поддерживают JavaScript. Данный стек технологий позволяет создавать приложения, которые будут работать быстро и эффективно. Поэтому интернет-магазин будет работать быстро и эффективно, что позволит увеличить количество покупателей.

## 2.2 Проектирование uml-диаграммы

Роли пользователя — это набор прав, которые пользователь может получить в системе. В зависимости от роли пользователя, он может иметь доступ к различным функциям 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.

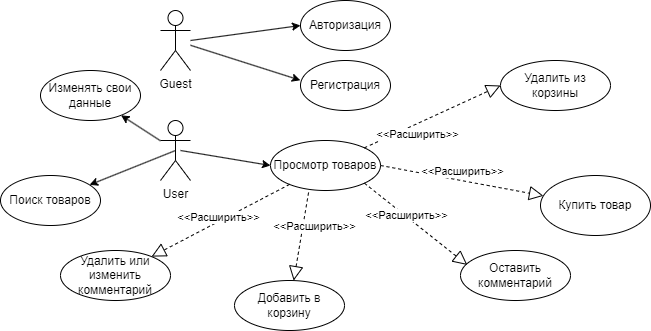


Рисунок 2.1 – UML диаграмма вариантов использования

В начале работы с сайтом пользователь является гостем (Guest). Ему будет доступна только главная страница. После регистрации пользователь становится пользователем (User).

Пользователь (User) может просматривать карточки продуктов, оформлять заказы, оставлять отзывы, просматривать историю заказов, информацию о доставке, информацию о оплате. Пользователь может изменить свои данные, изменить пароль, удалить аккаунт.

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

## 2.3 Проектирование базы данных

Проектирование базы данных — это процесс создания структуры базы данных, которая будет хранить данные, необходимые для работы сайта. В данном проекте будет создана база данных, которая будет хранить данные о пользователях, продуктах, заказах, отзывах и т.д. База данных будет состоять из следующих 5 таблиц: person, product, basket, comment, buy. В таблице person хранятся данные о пользователя. В таблице product хранятся данные о товарах. В таблице basket хранятся данные о товарах, которые добавил пользователь в корзину. В таблице comment хранятся данные о комментариях, которые оставил пользователь. В таблице buy хранятся данные о покупках, которые совершил пользователь. Связи между таблицами были реализованы с помощью внешних ключей. Таблица person имеет внешний ключ, который связывает ее с таблицей basket. Связь между таблицами basket и product была реализована с помощью внешних ключей. Таблица person имеет внешний ключ, который связывает ее с таблицей comment. Таблица person имеет внешний ключ, который связывает ее с таблицей buy. Таблица product имеет внешний ключ, который связывает ее с таблицей buy. Все таблицы приведены к нормальной форме. Нормальная форма - это набор правил, которые позволяют избежать дублирования данных в таблице. Диаграмма таблиц будет показана на рисунке 2.2.

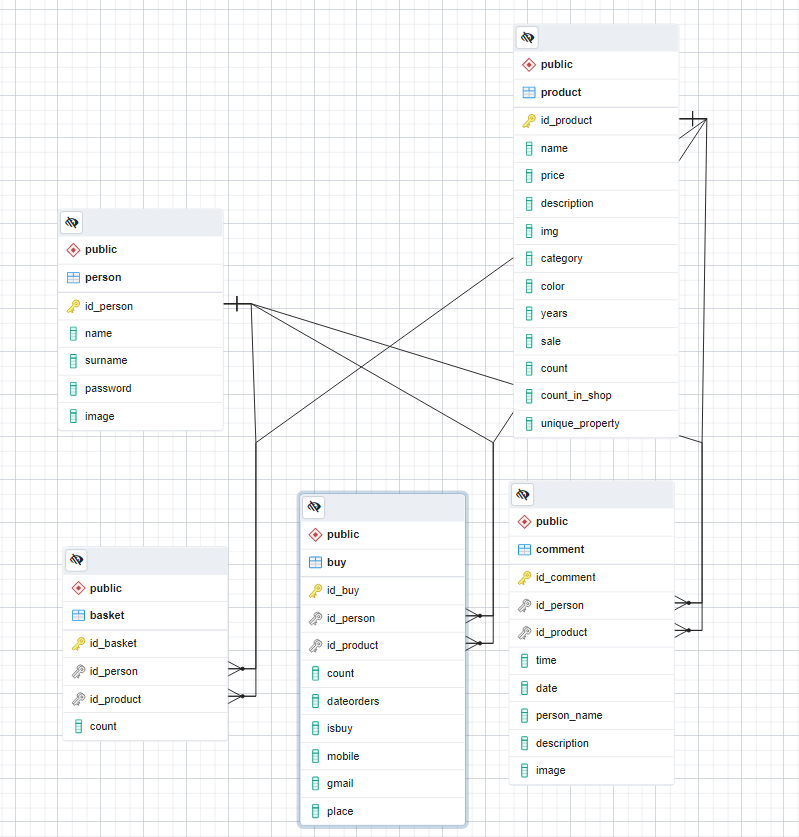


Рисунок 2.2 – Диаграмма таблиц

Таким образом, проектирование базы данных закончено. Далее будет произведена реализация проекта.

## 2.4 Проектирование страниц

В данном проекте будет использована Flexbox-верстка (Flexbox), так как она позволяет создавать адаптивные веб-страницы, которые будут отображаться корректно на всех устройствах. С помощью этой технологии можно очень просто и гибко расставить элементы в контейнере, распределить доступное пространство между ними, и выровнять их тем или иным способом даже если они не имеют конкретных размеров. CSS Flexbox поддерживается всеми используемые на сегодняшний момент современными браузерами (с использованием префиксов: IE10+, Edge12+, Firefox 2+, Chrome 4+, Safari 3.1+, Opera 12.1+, iOS Safari 3.2, Opera mini, Android 2.1+, Blackberry 7+).

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:

* + Home (Главная страница).
  + Product (Продукты).
  + Basket (Корзина).
  + Profile (Профиль).
  + Login (Вход).
  + Register (Регистрация).
  + ProductFull (Полное описание продукта).

На странице Home будет расположены слайдеры с изображениями, категории продуктов, информация о компании, контакты, форма обратной связи.

На странице Product будут расположены категории продуктов, фильтры, карточки продуктов, пагинация. При нажатии на карточку продукта, пользователь будет перенаправлен на страницу ProductFull.

На странице Basket будут расположены карточки продуктов, форма оформления заказа, информация о доставке, информация о оплате.

На странице Profile будут расположены данные пользователя, история заказов, информация о доставке, информация о оплате. На этой странице пользователь сможет изменить свои данные, изменить пароль, удалить аккаунт. Также он может посмотреть историю заказов, информацию о доставке, информацию о оплате.

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

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

На странице ProductFull будут расположены карточка продукта, форма оформления заказа, информация о доставке, информация о оплате. На этой странице пользователь сможет оформить заказ, оставить отзыв, посмотреть отзывы других пользователей.

Также будет фиксированный Navbar с ссылками на главную страницу, страницу продуктов, страницу корзины, страницу профиля, страницу входа, страницу регистрации. Navbar будет расположен вверху страницы.

## 2.5 Выбор стилевого оформления

Для реализации стилевого оформления сайта было выбрано минималистическое направление с элементами классического веб-дизайна. Такой стиль позволяет привлечь внимание новых пользователей. Сайт не загромождается лишними деталями (боковыми панелями и т.п.), большим количеством спецэффектов. Акцент на сайте ориентируется на основной контент.

Сайт будет состоять из 2-х основных цветов - белого и чёрного. Белый цвет будет использоваться для фона, а чёрный для текста и некоторых элементов дизайна. 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: Poppins. Шрифт подключался с помощью библиотеки Google Fonts.

Шрифт Poppins является одним из самых популярных шрифтов в 2019 году. Он был разработан в 2010 году, но в последние годы его популярность растет с каждым днем. Этот шрифт отличается своей читабельностью и простотой. Он подходит для любых типов дизайна, в том числе и для веб-дизайна.

## 2.6 Разработка пользовательских элементов

Разработка логотипа осуществлялась в графическом редакторе Figma. Дизайн довольно минималистичен. Было принято решение не загромождать его множеством декоративных элементов. Логотип представляет из себя название WebShop представлена на рисунке 2.3.



Рисунок 2.3 – Логотип сайта WebShop

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 WebShop

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

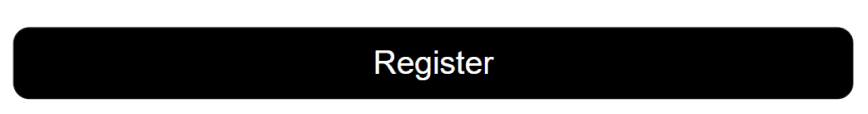


Рисунок 2.5 – Кнопка сайта WebShop

Также будет адаптивный слайдер на сайте, который будет переключаться автоматически через 2 секунды. Пример слайдера будет на рисунке 2.6.

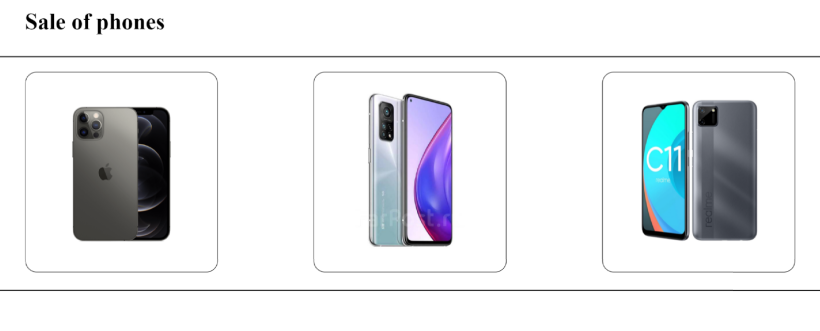


Рисунок 2.6 – Слайдер сайта WebShop

На сайте будут карточки товара, которые будут состоять из картинки и кнопок «Купить» и «В корзину». Кнопка «Купить» будет перенаправлять на страницу оформления заказа. Кнопка «В корзину» будет добавлять товар в корзину. Дизайн карточек товара будет простым и понятным. Пример карточки товара будет на рисунке 2.7.



Рисунок 2.7 – Карточка товара

Также на сайте будет блок с параметрами поиска. Пользователь сможет искать продукты по категории, по цене, по уникальному полю. Пример блока поиска товара будет рисунке 2.8.

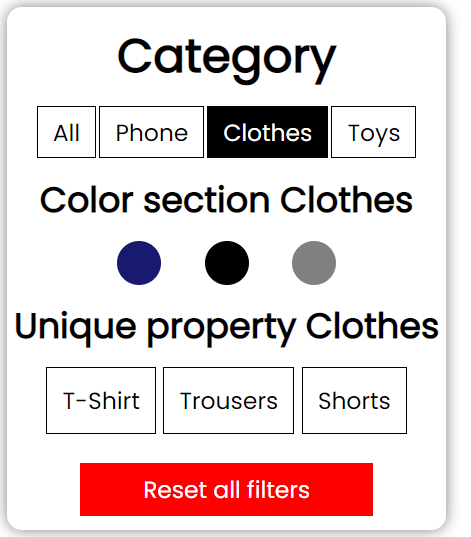


Рисунок 2.8 – Блок поиска товара

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.

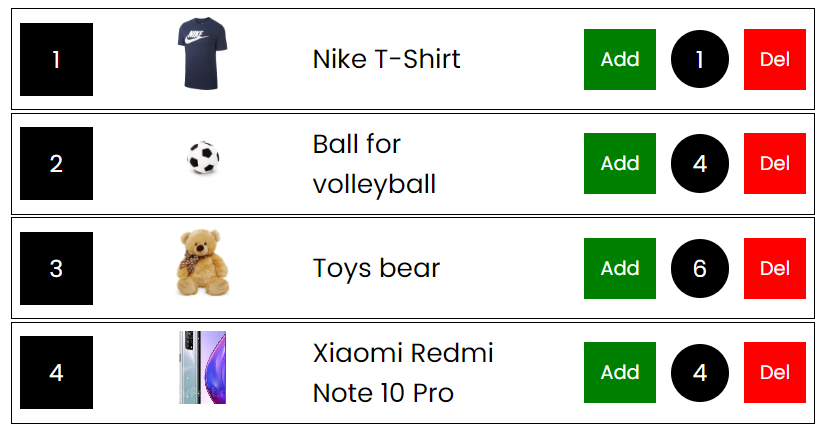


Рисунок 2.9 – Блок корзины

На сайте будут разработаны некоторые динамические эффекты и 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 Navbar будет состоять из логотипа сайта, навигационного меню и кнопки входа или выхода из аккаунта. В навигационном меню будет 4 пункта: Home, Product, Profile, Basket. Над Basket будет висеть круг с количеством товаров в корзине. На рисунке 2.10 представлено оформление Navbar.



Рисунок 2.10 – Navbar сайта WebShop

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.

## 2.7 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-сайты

## 3.1 Структура JSX-документа

Для реализации проекты было принято решение создавать разные компоненты. Компонент — это функция, которая возращает 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.

|  |
| --- |
| const Footer: FC = () => {     return (        <div className='footer'>           <div className="wrapper">              <div className="logo">                 WebShop              </div>              <ul className="nav">              <div className="contacts">                 <ul>                    {contacts.map((item: contactsTS) => {                       const { img, name, Link } = item;                       return (                          <li key={name}>                             <a href={Link}>                                <img src={img} alt={name} />                                <p>{name}</p>                             </a>                          </li>                       )                    })}                 </ul>              </div>           </div>        </div >     )  }  export default Footer |

Листинг 3.1 – Структура JSX-формата

Самым главным компонентом будет с названием файлом и функции App.tsx. Он будет содержать основные маршруты по сайту. Для этого нужно использовать пакет React-router-dom. Почему лучше использовать его, чем обычный тег <a>? Потому что React это SPA. SPA (single page application) – это веб-приложение, работающее на одной странице. Оно подгружает все необходмые javascript и css файлы при первой загрузке страницы, а затем все общение между клиентом и сервером сводится к минимуму. Т.е. при таком подходе большая часть работы сайта производится на стороне клиента, а если нужно получить данные с сервера, то это обычно делается с помощью JSON. Код примера маршрутов будет показан в листинге 3.2.

|  |
| --- |
| <BrowserRouter>          <Navbar />          <Suspense fallback={<Loading />}>            <Routes>              <Route path="/" element={<Home />} />              <Route path="/basket" element={<Basket />} />              <Route path="/product" element={<Product />} />              <Route path="/productFull/:id" element={<ProductFull />} />              <Route path="\*" element={<Error />} />            </Routes>          </Suspense>   </BrowserRouter> |

Листинг 3.2 – Структура маршрутов

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

* Фото товара.
* Названия товара.
* Описание товара.
* Цена товара.
* Кнопки добавления товара в корзину.

Кнопка добавления товара в корзину будет висеть поверх фото товара. При наведении на кнопку добавления товара в корзину, она будет менять цвет. При нажатии на кнопку добавления товара в корзину, она будет менять цвет и появится надпись, что товар добавлен в корзину. Код карточки товара будет показан в листинге 3.3.

|  |
| --- |
| const { id\_product, img } = item;  return (  <div className="product" key={index}>  {item.sale !== 0 ? (  <div className="sale">  <p className='sales'>{item.sale}%</p>  ) :  <div className="sale">  {item.price}$  </div>  }  <div className="choise\_product">  <div className="blocks">  <div className="block">  <p onClick={() => AddToBasket(item)}> 🛒</p>  </div>  <div className="block">  <Link to={`/product/:${id\_product}`}>  <p>🔍</p>  </Link>  </div>  </div>  </div>  <img src={img} alt={item.category}  />  </div> |

Листинг 3.3 – Структура карточки товара

Следующим компонентом будет Basket компонент. В нем будет находится товары, которые были добавлены в корзину. Также будет находится блок с общей суммой товаров в корзине и кнопкой оформления заказа. На товаре будут кнопки добавить в корзину и удалить из корзины. Код корзины будет показан в листинге 1.4.

|  |
| --- |
| <div className="product" key={img}>  <div className="id">  <p>{index + 1}</p>  </div>  <div className="img">  <Link to={`/product/:${id\_product}`}>  <img src={img} alt="" />  </Link>  </div>  <div className="name">  <Link to={`/product/:${id\_product}`}>  <p>{name}</p>  </Link>  </div>  <div className="buttons\_add\_del">  <button className='add' onClick={() => AddToBasket(item)}>Add</button>  <p className='count'><p>{count || 0}</p></p>  <button className='del'  onClick={() => DeleteToBasket(item)}  >Del</button>  </div>  </div>  </div>  <div className="title\_list">  <p>Total: {sum} $</p>  <button onClick={() => handleBuyOrder()}>Buy</button>  </div> |

Листинг 3.4 – Структура кода компонента Basket

После нажатия на кнопку оформления заказа, будет появляться модальное окно с формой для ввода данных для оформления заказа. После заполнения формы и нажатия на кнопку отправить, будет появляться модальное окно с сообщением о том, что заказ оформлен. Код формы будет показан в листинге 3.5.

|  |
| --- |
| {state === 1 && (<div className="form\_pages two">  <h1>Country</h1>  <select value={country} onChange={(e: ChangeEvent<HTMLSelectElement>) => setCountry(e.target.value)}>  <option value="Russia">Russia</option>  <option value="Belarus">Belarus</option>  <option value="Ukraine">Ukraine</option>  </select>  <h1>City</h1>  {  country === "Russia" && (  <select value={addres} onChange={(e: ChangeEvent<HTMLSelectElement>) => setAdress(e.target.value)}>  <option value="Moscow">Moscow</option>  <option value="Saint-Petersburg">Saint-Petersburg</option>  <option value="Novosibirsk">Novosibirsk</option>  </select>  country === "Belarus" && (  <select value={addres} onChange={(e: ChangeEvent<HTMLSelectElement>) => setAdress(e.target.value)}>  <option value="Minsk">Minsk</option>  <option value="Gomel">Gomel</option>  <option value="Brest">Brest</option>  <option value="Vitebsk">Vitebsk</option>  </select>  )  } |

Листинг 3.5 – Структура код формы

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

|  |
| --- |
| <div className="profile">  <div className="photo">  <img src={`http://localhost:3001/User/${user?.image}`  } alt="prfile"  onClick={() => PhotoModal(`http://localhost:3001/User/${user?.image}`)}  /> Update Photo  </button>  </div>  <div className="information">  <div className="info\_about\_user">  <div className="name\_info">  <div className="value\_info">  <p>  {user?.name || "No name"}  </p>  <p>  {user?.surname || "No surname"}  </p>  </div>  </div>  <div className="changeProfile">  <button  onClick={() => ChangeModel(true)}  >  Change Profile  </button>  </div>  </div> |

Листинг 3.6 – Структура компоненты Profile

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 товарах, о пользователях, о заказах, о корзине. Благодаря CreateAsyncThunk был реализован асинхронный запрос к серверу для получения данных о товарах, о пользователях, о заказах, о корзине. Асинхронный запрос к серверу реализован с помощью библиотеки Axios. Асинхронный запрос к серверу будет показан в листинге 3.7.

|  |
| --- |
| export const DeleteComment = createAsyncThunk(  "product/deleteComment",  async ({ id\_comment, id\_product, id\_person }: IDeleteComment) => {  try {  const { data } = await axios.delete("/product/deletecomment", {  data: {  id\_comment,  id\_product,  id\_person,  },  });  return data;  } catch (error) {  console.log(error);  }  }  ); |

Листинг 3.7 – Код асинхронной функции

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, асинхронные запросы к серверу. Для того, чтобы получить данные нужно использовать ExtraReducers, в котором будет находиться асинхронный запрос к серверу. Код CreateSlice будет показан в листинге 3.8.

|  |
| --- |
| export const productSlice = createSlice({  name: "product",  initialState,  reducers: {  Reset: (state) => {  state.product = [];  state.product = [];  state.basket = [];  state.basket\_length = 0;  state.orders = [];  state.top\_persons = [];  },  },  extraReducers: (builder) => {  builder.addCase(GetProducts.pending, (state) => {  state.loading = true;  state.message = "";  });  builder.addCase(GetProducts.fulfilled, (state, action) => {  state.loading = false;  state.product = Array.from(action.payload);  }); |

Листинг 3.8 – Структура кода CreatSlice

Затем все редьюсеры нужно объединить в один редьюсер и поместить его в Store. 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.

|  |
| --- |
| const store = configureStore({  reducer: {  auth: authSlice,  product: productSlice  },  });  export default store;  export type RootState = ReturnType<typeof store.getState>;  export type AppDispatch = typeof store.dispatch;  export const useAppDispatch = () => useDispatch<AppDispatch>();  export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector; |

Листинг 3.9 – Структура кода Store

В этом разделе были рассмотрены осноыные положения по созданию глобального хранилища данных с помощью Redux-Toolkit. Были рассмотрены основные понятия по созданию редьюсеров, асинхронных запросов к серверу, получения данных из Store.

## 3.3 Реализация серверной части

Серверная часть была реализована с помощью Node JS и Express JS. Благодаря Express JS был реализован сервер, который будет обрабатывать запросы от клиента. Главным файлом сервера является index.js, потому что он будет запускать сервер и соединяться с другими файлами. Код файла index.js будет показан в листинге 3.10.

|  |
| --- |
| import express from "express";  import cors from "cors";  import UserRoute from "./route/user.route.js";  import ProductRoute from "./route/product.route.js";  import fileUpload from "express-fileupload";  import dotenv from "dotenv";  dotenv.config();  const app = express();  const PORT = process.env.PORT || 3001;  //#region MIDDLEWARES  app.use(cors()); // для того, чтобы разрешить запросы с других доменов  app.use(express.json());  app.use(fileUpload());  //#endregion  //#region ROUTES  app.use("/api", UserRoute);  app.use("/api", ProductRoute);  app.use("/uploads", express.static("uploads"));  app.use(express.static("uplouds"));  app.get("/", (req, res) => {  res.send("Hello World!");  });  // по запросу hhttp://localhost:3001 будем отдавать статические файлы из папки uploads  //#endregion  app.listen(PORT, () => {  console.log(`Server started on port ${PORT}`);  }); |

Листинг 3.10 – Структура кода index.js

Также на сервере были созданы Controller и Router. Для того, чтобы получить данные с сервера нужно использовать Router, а для того, чтобы обработать запросы нужно использовать Controller. В Controller будет находиться логика обработки запросов, а в Router будет находиться логика получения данных с сервера. Код Controller и Router будет показан в листинге 3.11.

|  |
| --- |
| сonst router = Router();  router.get("/product/getproducts", ProductController.GetProducts);  router.get("/product/getproduct/:id", ProductController.GetProduct);  router.post("/product/addcomment", ProductController.AddComment);  router.put("/product/updatecomment", ProductController.UpdateComment);  router.delete("/product/deletecomment", ProductController.DeleteComment  async DeleteComment(req, res) {  try {  const { id\_comment, id\_product, id\_person } = req.body;  await pool.query(  "DELETE FROM comment WHERE id\_comment = $1 AND id\_product = $2 AND id\_person = $3",  [id\_comment, id\_product, id\_person]  );  const product = await pool.query(  "SELECT \* FROM product WHERE id\_product = $1",  [id\_product]  );  const Path = "http://localhost:3001";  const NewComment = await pool.query(  "SELECT \* FROM comment WHERE id\_product = $1 ORDER BY time",  [id\_product]  );  const productForSend = {  id\_product: product.rows[0].id\_product,  name: product.rows[0].name,  }; |

Листинг 3.11 – Код Controller и Router

Для подключения к базе данных был использован PostgreSQL. Код подключения к БД будет показан в листинге 3.12.

|  |
| --- |
| const pool = new Pool({  user: "postgres",  password: "Ghtyuioplm",  host: "localhost",  port: 5432,  database: "shop",  }); |

Листинг 3.12 – Структура подключения к БД

Подводя итог, в этом разделе были рассмотрены основные положения по созданию серверной части приложения. Было рассмотрено, как работает сервер, какие файлы нужно создать для того, чтобы сервер работал корректно.

## 3.4 Структура базы данных

База данных была реализована с помощью PostgreSQL. Для создания таблиц в базе данных был использован следующий код, который будет показан в листинге 3.13.

|  |
| --- |
| CREATE DATABASE shop;  create table person(  id\_person serial PRIMARY KEY,  name varchar(50) not null,  surname varchar(50) not null,  password text not null,  image varchar (50) DEFAULT 'default.jpg'  );  CREATE TABLE product(  id\_product serial PRIMARY KEY,  name varchar(50) not null,  price INTEGER not null,  description varchar(100) not null,  img VARCHAR(50) ,  category varchar(50) not null,  color varchar(50) not null,  years INTEGER not null,  sale INTEGER,  count INTEGER not null,  count\_in\_shop INTEGER not NULL,  unique\_property varchar(50)  );  CREATE table basket (  id\_basket serial PRIMARY KEY,  id\_person INTEGER not null,  id\_product INTEGER not null,  count INTEGER not null,  FOREIGN KEY (id\_person) REFERENCES person(id\_person),  FOREIGN KEY (id\_product) REFERENCES product(id\_product)  );  CREATE TABLE buy(  id\_buy serial PRIMARY KEY,  id\_person INTEGER not null,  id\_product INTEGER not null,  count INTEGER not null,  dateorders timestamp not null,  isbuy BOOLEAN not null,  mobile varchar(50) ,  gmail varchar(50) ,  place varchar(50) ,  FOREIGN KEY (id\_person) REFERENCES person(id\_person),  FOREIGN KEY (id\_product) REFERENCES product(id\_product)  ); |

Листинг 3.13 – Структура создания таблиц в базе данных

Таким образом, в этом разделе была рассмотрена структура базы данных. Были рассмотрены таблицы, которые были созданы для реализации базы данных, а также были рассмотрены связи между таблицами.

## 3.5 Добавление таблиц стилей Sass

Для стилизации был использован SASS, который позволяет использовать переменные, вложенность, миксины, наследование, импорт и экспорт. Код использования SASS показан в листинге 3.14.

|  |
| --- |
| .header {  position: fixed;  top: 0;  left: 0;  z-index: 100;  background-color: white;  width: 100%;  box-shadow: 0px 0px 10px 0px gray;  .wrapper {  height: 60px;  display: flex;  .burger {  display: none;  }  align-items: center;  justify-content: space-between;  .nav {  ul {  display: flex;  align-items: center;  .li\_end {  background-color: white;  //default cursor  padding: 0 10px;  cursor: default;  &:hover {  background-color: white;  }  }  li {  height: 60px;  display: flex;  align-items: center;  justify-content: center;  cursor: pointer;  position: relative;  span {  display: flex;  justify-content: center;  align-items: center;  background-color: black;  border-radius: 50%;  width: 25px;  height: 25px;  color: white;  position: absolute;  top: 40%;  left: 85%;  transform: translate(-50%, -50%);  transition: 0.4s;  } |

Листинг 3.14 – Структура кода SASS

В этом разделе была рассмотрена структура SASS, что такое SASS и как он используется в проекте.

## 3.6 Выводы по разделу

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

# 4 Тестирование веб-сайта

## 4.1 Адаптивный дизайн веб-сайта

Сегодня количество пользователей, которые заходят в Интернет с мобильных устройств очень велико, и с каждым днем их становится все больше. А потому адаптация веб-сайта под маленькие экраны – актуальная задача и для курсового проекта.

Передо мной стояла задача: создать сайт, который будет сам подстраиваться под разные типы экранов. Главными условиями адаптивности является не фиксированный размер элементов страницы.

Для создания адаптивности веб-сайта были использованы следующие инструменты: медиа-запросы (@media), единицы измерения, которые связаны непосредственно с параметрами области просмотра (представляют собой процентные величины от текущей области просмотра браузера).

После написания основной структуры страницы на JSX и внешнего стилевого оформления появился вопрос о тестировании. Веб-сайт был открыт при помощи различных браузеров. После этого было выявлено, что во всех браузерах сайт выглядит одинаково.

При отображении сайта на мобильных устройствах навигационное меню в верхнем колонтитуле меняется на выпадающее меню, открывающееся при нажатии на кнопку меню. 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 продуктами на странице для отображения располагаются друг под другом. На рисунке 4.2 можно увидеть пример.

****

Рисунок 4.2 – Блоки с продуктами на мобильных устройствах

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

## 4.2 Кроссбраузерность веб-сайта

Очень важно, чтобы наш сайт был кроссбраузерным. Кроссбраузерность – это способность веб-ресурса отображаться одинаково и работать во всех популярных браузерах, без перебоев в функционировании и ошибок в верстке, а также с одинаково корректной читабельностью контента.

У каждого браузера есть свои встроенные, экспериментальные или нестандартные свойства и для того, чтобы они корректно работали было принято решения о внедрении вендорных префиксов. Они были автоматически прописаны после компиляции файла с расширением sass.

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 Opera сайт показал схожий результат: никаких отклонений как для мобильной версии сайта, так и для компьютерной не было выявлено, текст не съехал, не произошло его наложения на картинки. На рисунке 4.4 можно увидеть пример веб-страницы в Opera.

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

Рис 4.4 – Вид веб-страницы в Opera 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 с помощью таких браузеров как Google Chrome, Яндекс и Opera общий вид веб-страницы не изменился.

Главной целью такой тщательной проверки сайта является грамотная настройка всех показателей, одной или всех страниц сайта, оперативное обнаружение и устранение всех дефектных мест на сайте, а также его адаптация под различные устройства.

Динамика сайта на разных устройствах – наиболее важная часть в написании сайта, поэтому, на данном этапе было проанализировано поведение веб-сайта при использовании с разных устройств, а также веб-ресурс был протестирован в большом количестве браузеров. По итогу выполнения данного этапа для веб-сайта был разработан адаптивный дизайн и кроссбраузерность. Работа, проделанная с сайтом, позволяет пользователям ориентироваться по сайту. Лишних элементов нет, что упрощает пользование сайтом. При пользовании сайтом при помощи мобильного устройства некоторые элементы меняют свое положение, для более удобного пользования. В ходе тестирования были проверены все ссылки (включая модальные окна), проверены формы регистрации пользователя. Была выполнена проверка контента. Также в ходе тестирования была проверена совместимость с браузерами. (Google Chrome, Opera, Яндекс).

## 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 запросов, их обработки frontend сервером, а также правильности отображение представлений.

Рассмотрим сценарий регистрации. При заполнении формы неправильными данными должно быть получено оповещение об ошибки. Заполнение формы неправильными данными и уведомление об ошибке изображено на рисунке 4.6.

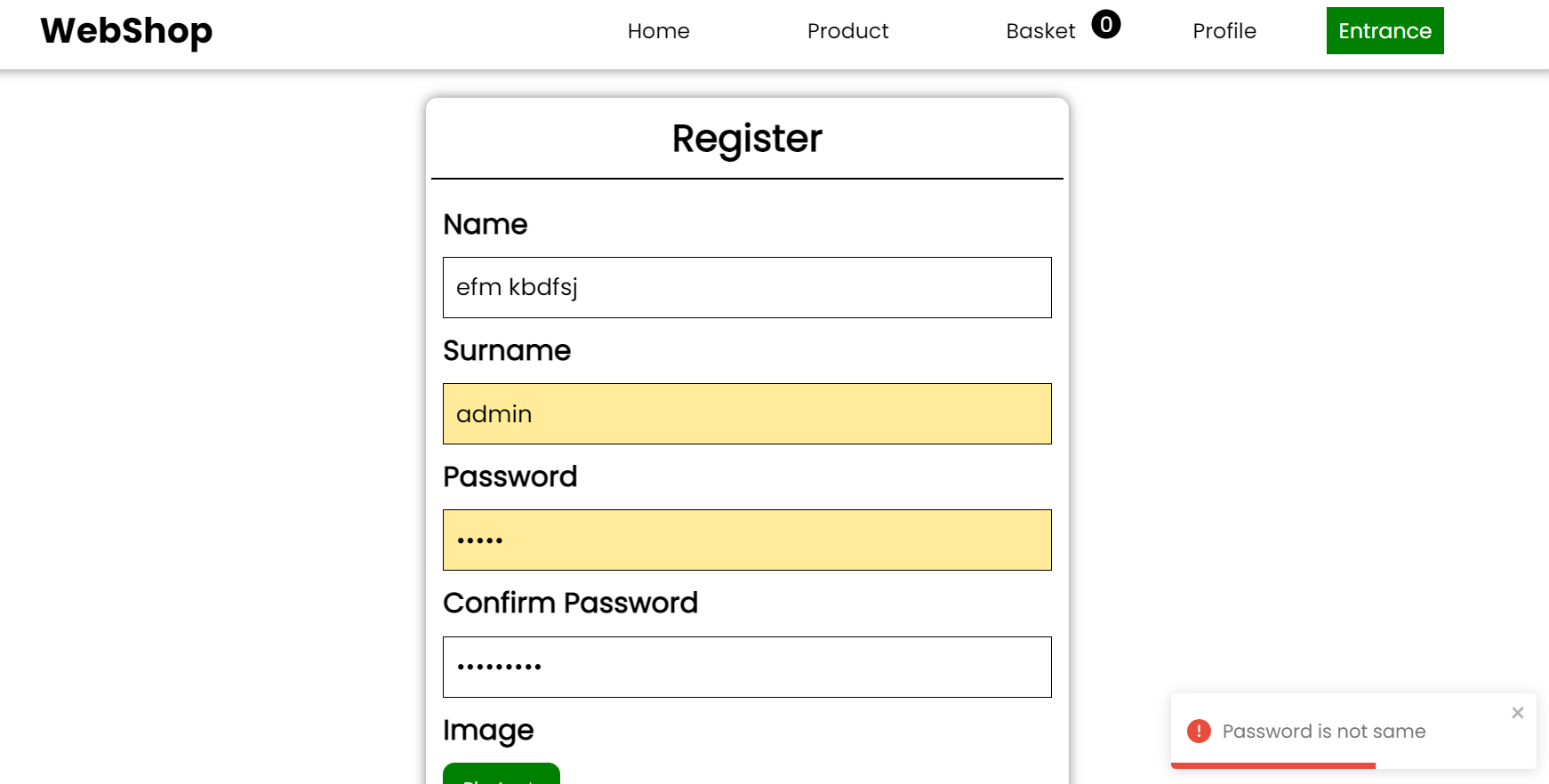


Рисунок 4.6 – Форма заполнения неверными данными

Попробуем зайти под несуществующим пользователем. При неверном введение данным, получим оповещение об ошибке. Оповещение об ошибке изображено на рисунке 4.7.

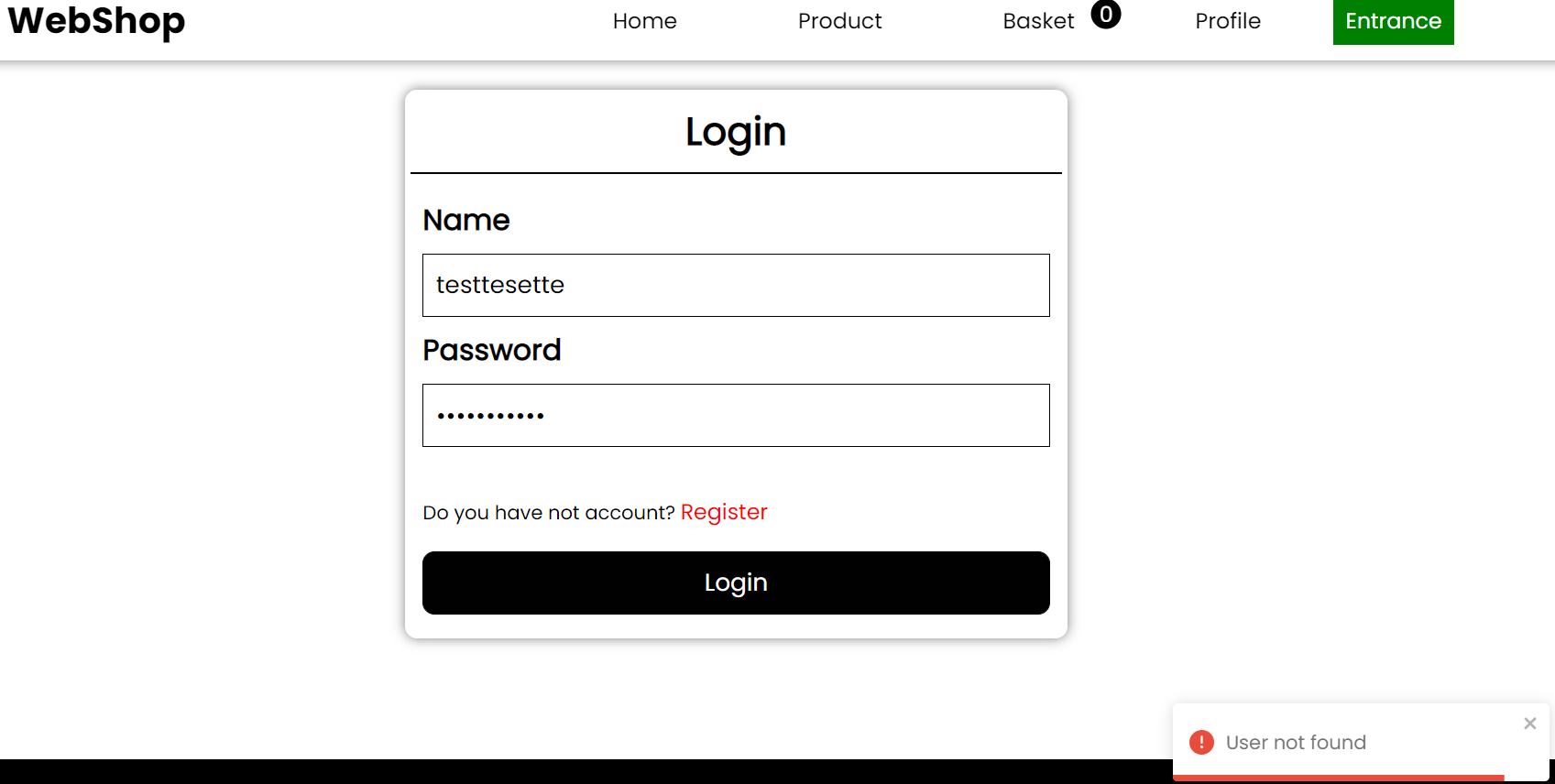


Рисунок 4.7 – Неправильный ввод данных для входа и оповещение об ошибке

Ввод правильных данных изображен на рисунке 4.8

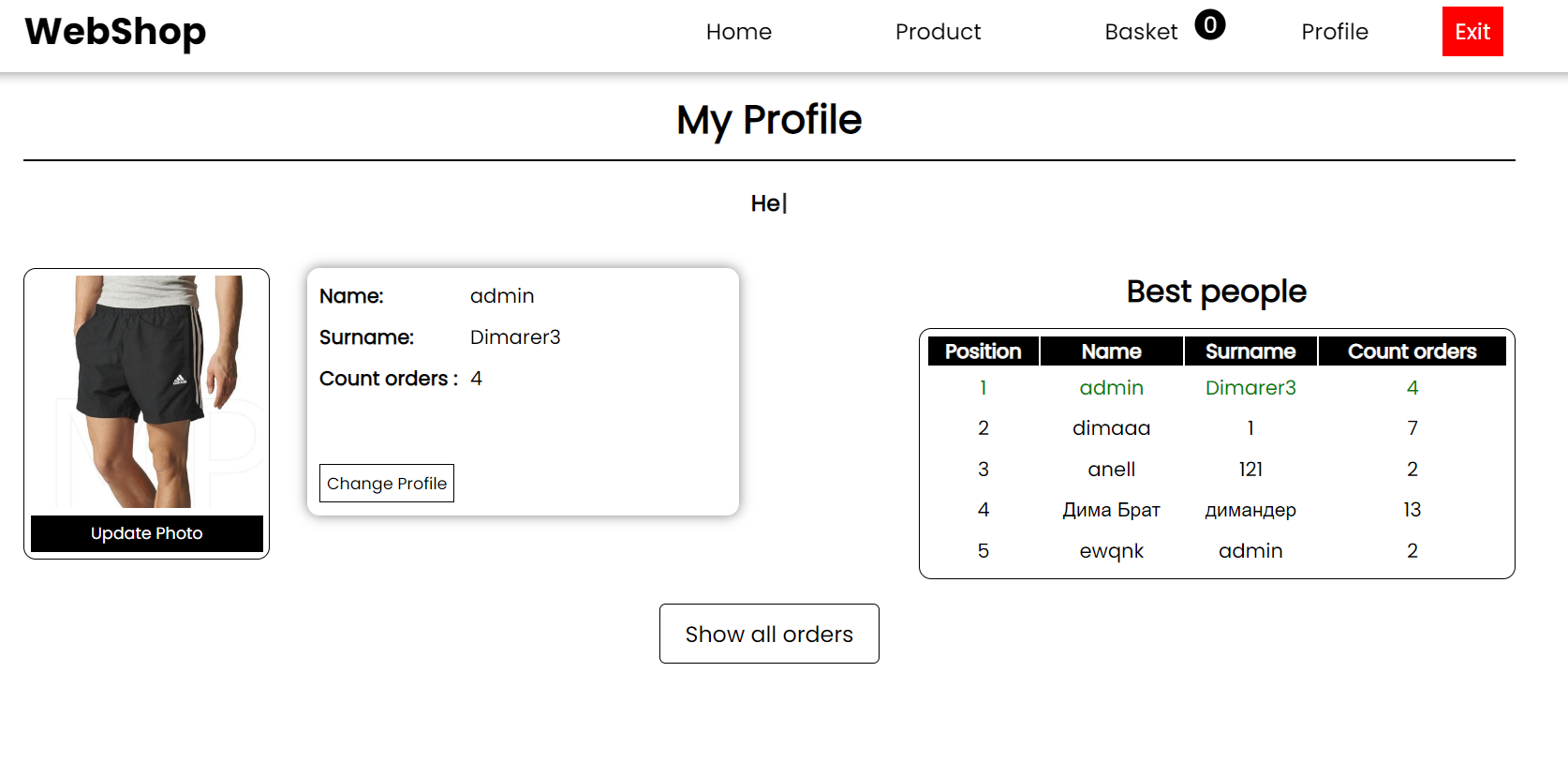


Рисунок 4.8 – Правильный ввод данных при авторизации

В ходе ручного тестирования были проведена валидация всех страниц сайта, проверены все ссылки, проверены все формы, проверен контент, проверена совместимость с браузерами. Также была проведена проверка работы корзины, где товары добавляются и удаляются из корзины. Все комментарии отображаются без ошибок. Во время тестирования не было обнаружено ошибок. Во всех браузерах работает корректно. Тестирование функционала сайта было выполнено вручную с использованием следующие браузеров Google Chrome (42.0.2311.90), Opera (29.0.1795.47) 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 регистрации 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 покупки товаров. Для того, чтобы совершить покупку, необходимо зарегистрироваться на сайте. После регистрации необходимо авторизоваться на сайте. После авторизации пользователь может просматривать товары, добавлять их в корзину, а также удалять из корзины. После того, как пользователь добавил все товары в корзину, он может оформить заказ. После оформления заказа пользователь получает уведомление о том, что заказ оформлен. После этого пользователь может просматривать свои заказы, а также отменять их.

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

На странице Product пользователь может просматривать товары, которые находятся в наличии, а также делать фильрацию товаров по категориям.

На странице ProductFull пользователь может просматривать полностью описание товара, а также добавить товар в корзину.

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

На Странице Profile пользователь может просматривать свои заказы, а также отменять их. Может также изменить свои данные.

# Заключение

Целью на курсовое проектирование было создание интернет-магазина для

продажи различных товаров для пользователей, которые хотят купить товары

в интернете.

В ходе курсовой работы были выполнены следующие задачи:

* Проанализировать и описать основные принципы работы React и Redux.
* Разработать приложение, используя React и Redux.
* Провести тестирование приложения.
* Провести анализ и оценку качества приложения.

А также были выполнены задачи по созданию интернет-магазина:

* Разработать регистрацию и авторизацию пользователей.
* Разработать возможность добавления и удаления товаров из корзины.
* Разработать структуру веб-сайта.
* Разработать возможность оставить отзыв о товаре.
* Разработать возможность заказа товара.
* Разработать возможность просмотра истории заказов.

Подводя итоги курсовой работы, можно сделать вывод, что в ходе работы были выполнены все поставленные задачи. Были изучены основные принципы работы React и Redux, что позволило разработать приложение и закрепить полученные знания.

# Список использованных источников

1. Ozon [Электронный ресурс]. – Режим доступа: https://ozon.by/. – Дата обращения 21.11.2022.
2. Wildberries [Электронный ресурс]. – Режим доступа: https://www.wildberries.by/. – Дата обращения 21.11.2022.
3. Aliexpress [Электронный ресурс]. – Режим доступа: https://aliexpress.ru/. – Дата обращения 21.11.2022.
4. Документация по React JS [Электронный ресурс]. - Режим доступа: https://reactjs.org/. – Дата обращения 21.11.2022.
5. Документация по Figma [Электронный ресурс]. - Режим доступа: <https://help.figma.com>. – Дата обращения 21.11.2022.
6. Документация по Redux/Redux-tolkit [Электронный ресурс]. - Режим доступа: https://redux-toolkit.js.org/ . – Дата обращения 21.11.2022.
7. Документация по Sass/Scss [Электронный ресурс]. - Режим доступа : <https://sass-scss.ru/guide/> – Дата обращения 21.11.2022.

# Приложение А Макет структуры веб-сайта

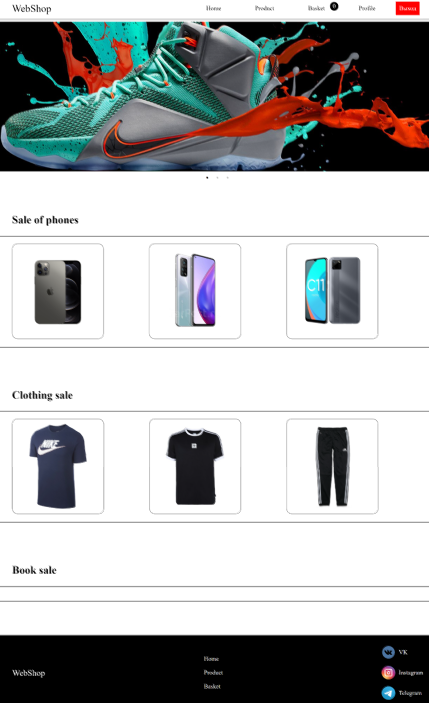
 

Рисунок 1-макет страницы “Home”

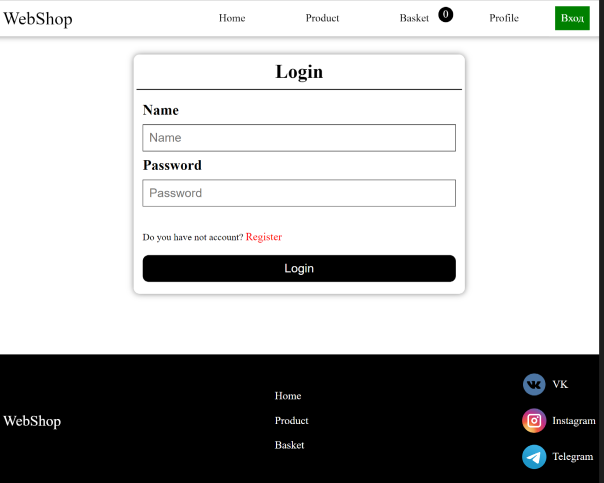
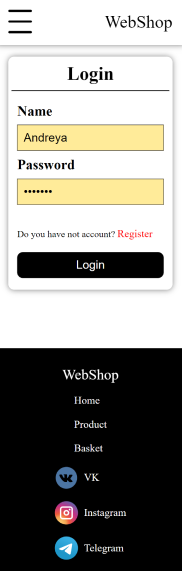
 

Рисунок 2-макет страницы “Login”

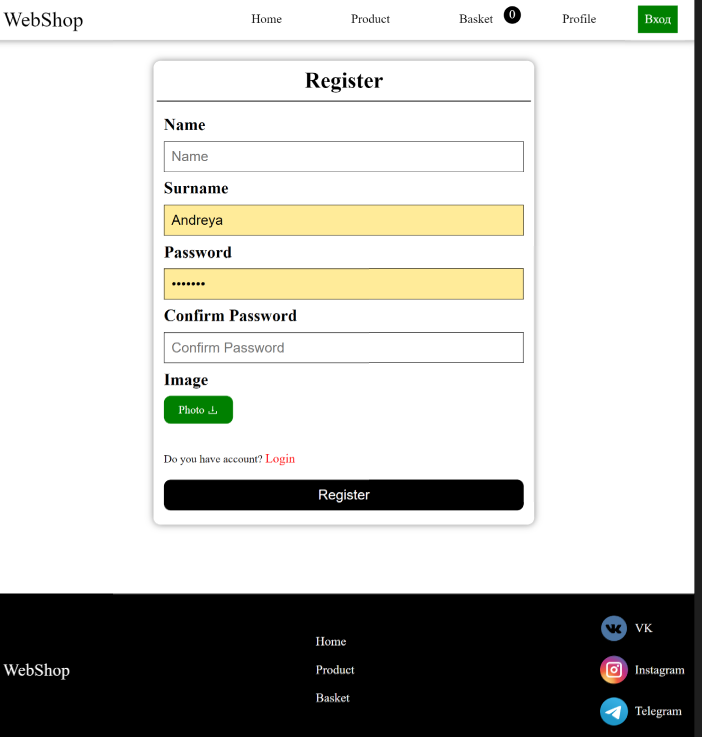
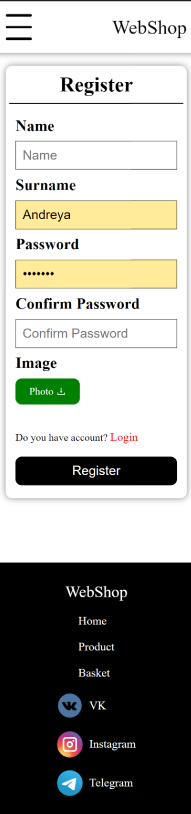
 

Рисунок 3-макет страницы “Register”

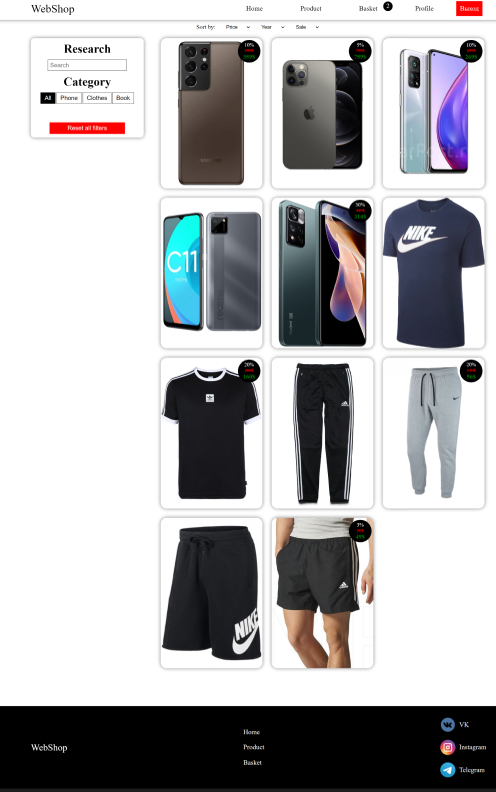
 

Рисунок 4-макет страницы “Product”

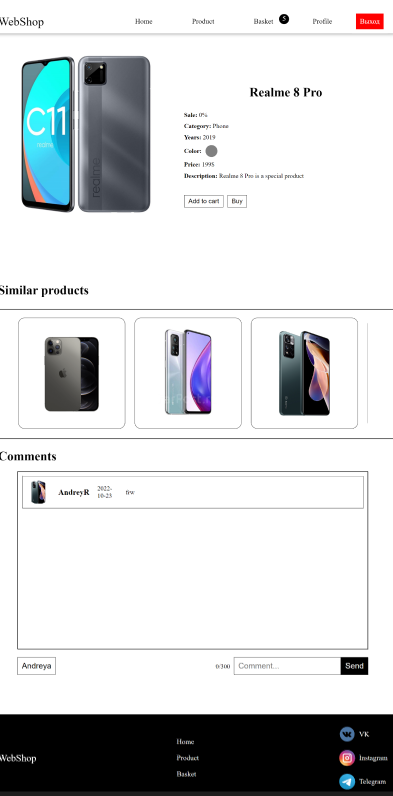
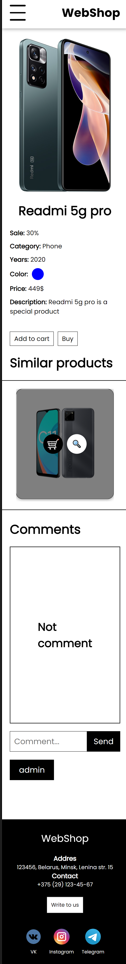
 

Рисунок 5-макет страницы “ProductFull”

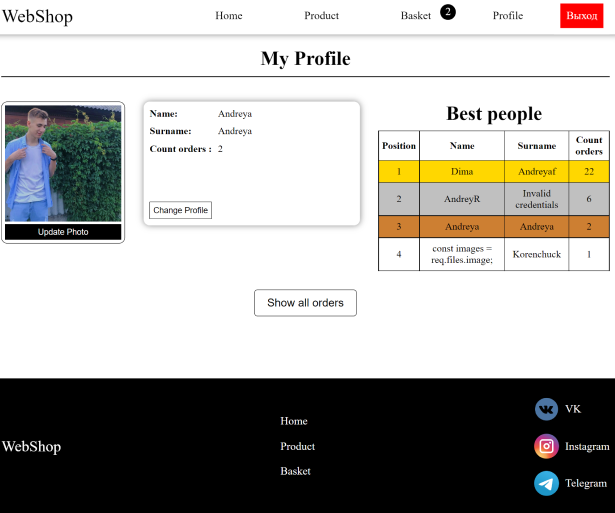
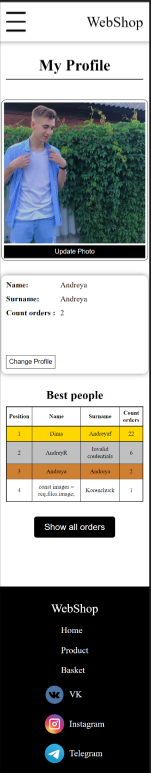
 

Рисунок 6-макет страницы “Profile”

# Приложение Б Листинг JSX-документа

|  |
| --- |
| <div  className='animate'>  <div className="wrapper">  <div className="research">  <div className="sort">  <p>Sort by:</p>  <div className="selects">  <select  onChange={(e: ChangeEvent<HTMLSelectElement>) => handleChangePrice(e.target.value)}  value={sortPrice}>  <option value="">Price</option>  <option value="ascending">Ascending</option>  <option value="descending">Descending</option>  </select>  <select  onChange={(e: ChangeEvent<HTMLSelectElement>) => handleChangeYears(e.target.value)}  value={sortYear}>  <option>Year</option>  <option value="ascending">Ascending</option>  <option value="descending">Descending</option>  </select>  <select  onChange={(e: ChangeEvent<HTMLSelectElement>) => handleChangeSale(e.target.value)}  value={sortSale}>  <option>Sale</option>  <option value="ascending">Ascending</option>  <option value="descending">Descending</option>  </select>  </div>  </div>  </div>  <div className="products\_\_input">  <div className="input">  <h1>Category</h1>  <div className="categors">  <button style={filter !== "Phone" && filter !== "Clothes" && filter !== "Toys"  ? { background: "black", color: "white" } : {}}  onClick={() => Filter(products)} >All</button>  <button style={filter === "Phone" ? { background: "black", color: "white" } : {}}  onClick={() => Filter("Phone")} >Phone</button>  <button  style={filter === "Clothes" ? { background: "black", color: "white" } : {}}  onClick={() => Filter("Clothes")} >Clothes</button>  <button  style={filter === "Toys" ? { background: "black", color: "white" } : {}}  onClick={() => Filter("Toys")} >Toys</button>  </div>  {  (testColor === undefined || testColor.length === 0) ?  <></> :  <h2>Color section {filter}</h2>  }  <div className="colors">  {testColor?.map((index, i) => {  return (  <p className='check'>  <input type="checkbox"  id={index.color}  key={i}  checked={testColor[i].checked}  onChange={() => handlerChangeColor(i)} />  <label htmlFor={index.color}>  <p style={index.color == "White" ? {  background: index.color, outline: "1px solid black"  } : { background: index.color }}  className={testColor[i].checked ? 'circle active' : 'circle'}  >  </p></label>  </p>  );  })}  </div> |

Структура JSX—документа

# Приложение В Листинг (SASS)

|  |
| --- |
| font-face {  font-family: "Poppins\_Bold";  //Regular.tt  src: url("fonts/Poppins-Bold.ttf");  }  .header {  position: fixed;  top: 0;  left: 0;  z-index: 100;  background-color: white;  width: 100%;  box-shadow: 0px 0px 10px 0px gray;  .wrapper {  height: 60px;  display: flex;  .burger {  display: none;  }  .logo {  a {  font-size: 30px;  font-family: "Poppins\_Bold", sans-serif;  }  }  align-items: center;  justify-content: space-between;  .nav {  ul {  display: flex;  align-items: center;  .li\_end {  background-color: white;  //default cursor  padding: 0 10px;  cursor: default;  &:hover {  background-color: white;  }  }  li {  height: 60px;  display: flex;  align-items: center;  justify-content: center;  cursor: pointer;  position: relative;  span {  display: flex;  justify-content: center;  align-items: center;  background-color: black;  border-radius: 50%;  width: 25px;  height: 25px;  color: white;  position: absolute;  top: 40%;  left: 85%;  transform: translate(-50%, -50%);  transition: 0.4s;  }  button {  background-color: transparent;  border: none;  cursor: pointer;  outline: none;  background-color: red;  color: white;  }  .button\_b {  background-color: red;  color: white;  height: 40px;  padding: 10px 10px;  cursor: pointer;  width: 100%;  display: flex;  align-items: center;  justify-content: center;  max-width: 100px;  }  .button\_a {  background-color: green;  color: white;  height: 40px;  padding: 10px 10px;  max-width: 100px;  width: 100%;  display: flex;  align-items: center;  justify-content: center;  }  a {  height: 60px;  width: 100%;  display: flex;  align-items: center;  justify-content: center;  padding: 0 50px;  }  transition: 0.4s;  &:hover {  background-color: black;  span {  background-color: white;  color: black;  }  a {  color: white;  }  }  }  }  }  }  }  @media (max-width: 820px) {  .header {  background-color: white;  width: 100%;  box-shadow: 0px 0px 10px 0px gray;  .wrapper {  height: 80px;  display: flex;  .burger {  display: block;  #nav-icon1 {  width: 40px;  height: 40px;  position: relative;  margin: 30px auto;  transform: rotate(0deg);  transition: 0.5s ease-in-out;  cursor: pointer;  }  #nav-icon1 span {  display: block;  position: absolute;  height: 4px;  width: 100%;  background: black;  border-radius: 9px;  opacity: 1;  left: 0;  transform: rotate(0deg);  transition: 0.25s ease-in-out;  }  #nav-icon1 span:nth-child(1) {  top: 0px;  }  #nav-icon1 span:nth-child(2) {  top: 18px;  }  #nav-icon1 span:nth-child(3) {  top: 36px;  }  #nav-icon1.open span:nth-child(1) {  top: 18px;  transform: rotate(135deg);  }  #nav-icon1.open span:nth-child(2) {  opacity: 0;  left: -60px;  }  #nav-icon1.open span:nth-child(3) {  top: 18px;  transform: rotate(-135deg);  }  }  .logo {  a {  font-size: 30px;  }  }  align-items: center;  justify-content: space-between;  .nav {  &.active {  border-top: 1px solid black;  transform: translateX(0%);  }  transition: 0.4s;  transform: translateX(-100%);  position: fixed;  top: 80px;  left: 0;  height: 100vh;  width: 100%;  background-color: white;  z-index: 100;  ul {  display: flex;  flex-direction: column;  align-items: center;  li {  width: 100%;  height: 80px;  display: flex;  align-items: center;  justify-content: center;  cursor: pointer;  position: relative;  span {  display: flex;  justify-content: center;  align-items: center;  background-color: black;  border-radius: 50%;  width: 25px;  height: 25px;  color: white;  position: absolute;  top: 40%;  transform: translate(-50%, -50%);  transition: 0.4s;  }  a {  height: 80px;  width: 100%;  display: flex;  align-items: center;  justify-content: center;  padding: 0 50px;  }  transition: 0.4s;  &:hover {  background-color: black;  span {  background-color: white;  color: black;  }  a {  color: white;  }  }  }  }  } |

Структура SASS—документа

# Приложение Г Листинг Redux

|  |
| --- |
| export const DeleteBasket = createAsyncThunk(  "product/deleteBasket",  async (basket: IDeleteBasket) => {  try {  const { id\_product, id\_person, id\_basket } = basket;  const { data } = await axios.delete("/product/deletebasket", {  data: {  id\_product,  id\_person,  id\_basket,  },  });  return data;  } catch (error) {  console.log(error);  }  }  );  export const AddOrder = createAsyncThunk(  "product/addOrder",  async (order: IOrder) => {  try {  const { id\_person, id\_product, count, dateorders, mobile, gmail, place } =  order;  const { data } = await axios.post("/product/addorder", {  id\_person,  id\_product,  count,  dateorders,  mobile,  gmail,  place,  });  return data;  } catch (error) {  console.log(error);  }  }  );  export const GetOrders = createAsyncThunk(  "product/getOrders",  async (id: string | undefined) => {  try {  const { data } = await axios.get(`/product/getorders/${id}`);  return data;  } catch (error) {  console.log(error);  }  }  );  export const DeleteAllBasket = createAsyncThunk(  "product/deleteAllBasket",  async (id\_person: number) => {  try {  const { data } = await axios.delete("/product/deleteallbasket", {  data: {  id\_person,  },  });  return data;  } catch (error) {  console.log(error);  }  }  );  export const GetTopPersons = createAsyncThunk(  "product/getTopPersons",  async () => {  try {  const { data } = await axios.get("/product/gettoppersons");  return data;  } catch (error) {  console.log(error);  }  }  );  export const UpdateOrder = createAsyncThunk(  "product/UpdateOrder",  async (order: IOrderUpdate) => {  try {  const { id\_buy, isbuy, id\_person } = order;  const { data } = await axios.put("/product/updateorder", {  id\_buy,  isbuy,  id\_person,  });  return data;  } catch (error) {  console.log(error);  }  }  );  export const productSlice = createSlice({  name: "product",  initialState,  reducers: {  Reset: (state) => {  state.product = [];  state.product = [];  state.basket = [];  state.basket\_length = 0;  state.orders = [];  state.top\_persons = [];  },  },  extraReducers: (builder) => {  builder.addCase(GetProducts.pending, (state) => {  state.loading = true;  state.message = "";  });  builder.addCase(GetProducts.fulfilled, (state, action) => {  state.loading = false;  state.product = Array.from(action.payload);  });  builder.addCase(GetProducts.rejected, (state) => {  state.loading = false;  });  builder.addCase(getProduct.pending, (state) => {  state.loading = true;  });  builder.addCase(getProduct.fulfilled, (state, action) => {  state.loading = false;  state.productID = action.payload;  });  builder.addCase(getProduct.rejected, (state) => {  state.loading = false;  });  builder.addCase(AddComment.fulfilled, (state, action) => {  state.productID = action.payload;  });  builder.addCase(UpdateComment.fulfilled, (state, action) => {  state.productID = action.payload;  });  builder.addCase(DeleteComment.fulfilled, (state, action) => {  state.productID = action.payload;  });  builder.addCase(AddBasket.fulfilled, (state, action) => {  state.message = action.payload;  state.basket\_length = action.payload.length || 0;  });  builder.addCase(GetBasket.fulfilled, (state, action) => {  state.basket = action.payload;  const Arrays = action.payload;  state.basket\_length = action.payload?.length || 0;  let sum = 0;  for (let i = 0; i < Arrays.length; i++) {  sum += Arrays[i].count \* Arrays[i].price;  }  state.sum = sum || 0;  });  builder.addCase(DeleteBasket.fulfilled, (state, action) => {  state.message = action.payload;  state.basket\_length = action.payload.length || 0;  });  builder.addCase(AddOrder.fulfilled, (state, action) => {  state.orders = action.payload;  });  builder.addCase(GetOrders.fulfilled, (state, action) => {  state.orders = action.payload;  });  builder.addCase(DeleteAllBasket.fulfilled, (state, action) => {  state.basket = [];  state.basket\_length = 0;  });  builder.addCase(GetTopPersons.fulfilled, (state, action) => {  state.top\_persons = action.payload;  });  },  });  export const { Reset } = productSlice.actions;  export default productSlice.reducer; |

|  |
| --- |
| import { configureStore } from "@reduxjs/toolkit";  import { TypedUseSelectorHook, useDispatch, useSelector } from "react-redux";  import authSlice from "./CreateAuth";  import productSlice from "./CreateProduct";  const store = configureStore({  reducer: {  auth: authSlice,  product: productSlice  },  });  export default store;  export type RootState = ReturnType<typeof store.getState>;  export type AppDispatch = typeof store.dispatch;  export const useAppDispatch = () => useDispatch<AppDispatch>();  export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector; |