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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Круглик Д. В.

подпись, дата

**Руководитель**

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

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

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

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

дата, подпись

Курсовой проект защищен с оценкой

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

подпись дата инициалы и фамилия

Минск 2023

**Содержание**

[Введение 5](#_Toc153244831)

[1 Постановка задач и обзор аналогичных решений 6](#_Toc153244832)

[1.1 Актуальность темы 6](#_Toc153244833)

[1.2 Обзор аналогичных решений 6](#_Toc153244834)

[1.3 Постановка задачи 11](#_Toc153244835)

[1.4 Выводы по разделу 12](#_Toc153244836)

[2 Проектирование веб-сайта 12](#_Toc153244837)

[2.1 Выбор средств реализации 12](#_Toc153244838)

[2.4 Проектирование страниц 13](#_Toc153244839)

[2.5 Выбор стилевого оформления 14](#_Toc153244840)

[2.6 Разработка пользовательских элементов 15](#_Toc153244841)

[2.7 Выводы по разделу 17](#_Toc153244842)

[3 Реализация структуры веб-сайты 18](#_Toc153244843)

[3.1 Структура JSX-документа 18](#_Toc153244844)

[3.2 Структура Redux-Toolkit 23](#_Toc153244845)

[3.3 Реализация серверной части 25](#_Toc153244846)

[3.4 Структура базы данных 27](#_Toc153244847)

[3.5 Добавление таблиц стилей Sass 28](#_Toc153244848)

[3.6 Выводы по разделу 29](#_Toc153244849)

[4 Тестирование веб-сайта 30](#_Toc153244850)

[4.1 Адаптивный дизайн веб-сайта 30](#_Toc153244851)

[4.2 Кроссбраузерность веб-сайта 31](#_Toc153244852)

[4.3 Ручное тестирование 33](#_Toc153244853)

[4.4 Выводы по разделу 35](#_Toc153244854)

[5 Руководство пользователю 36](#_Toc153244855)

[Заключение 37](#_Toc153244856)

[Список использованных источников 38](#_Toc153244857)

[Приложение А Макет структуры веб-сайта 39](#_Toc153244858)

[Приложение Б Листинг JSX-документа 42](#_Toc153244859)

[Приложение В Листинг (SASS) 44](#_Toc153244860)

[Приложение Г Листинг Redux 50](#_Toc153244861)

# Введение

В настоящее время наблюдается стремительный рост числа вновь создаваемых веб-сайтов. Именно поэтому ключевой целью при разработке сайта является его максимальная привлекательность для потенциальных пользователей и придание ему уникальности. В ранние годы развития Интернета можно было привлечь значительное количество посетителей на сайт, не прилагая больших усилий и не влагая значительных материальных ресурсов. Тем не менее, в условиях увеличивающейся конкуренции важную роль начинают играть функциональность сайта, его стиль и запоминаемость. Вне зависимости от того, создается ли сайт для представления фирмы или с целью заработка на рекламе, его основной задачей остается привлечение максимальной аудитории. Поэтому для достижения успеха критичен правильный подход к выбору стратегии и исполнителей.

Профессиональные студии веб-дизайна предлагают создание сайтов различного назначения: корпоративные сайты, сайты-визитки, интернет-магазины, а также сайты для ведения бизнеса как в онлайн, так и в реальном мире.

Наше время называют "информационным веком" из-за того, что информация стала самым важным, ценным и необходимым ресурсом. Владение информацией и умение передавать нужную информацию клиентам своевременно, быстро и ясно - ключ к успеху в условиях современного бизнеса. В настоящее время именно Интернет способен оперативно и массово передавать текст, звук, изображения и даже видеоролики, то есть практически любую информацию. С точки зрения пользователя, Интернет представляет собой огромный информационный ресурс, где можно найти все, что угодно - от прогноза погоды до личных предпочтений голливудских кинозвезд. Аудитория сети Интернет постоянно растет, и она представляет особый интерес для рекламодателей, поскольку факт остается фактом: среди пользователей сети множество потенциальных клиентов. Актуальность темы интернет-магазина в современном мире обусловлена тем, что все больше людей предпочитают покупать товары через Интернет, а не в традиционных магазинах. Это связано с тем, что в интернет-магазинах можно приобрести товары по более низким ценам, чем в магазинах, а также с тем, что в интернет-магазинах можно купить товары, которые недоступны в обычных 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 данной темы проявляется в том, что интернет-магазин становится одним из наиболее востребованных типов веб-сайтов среди пользователей. Важно как для покупателей, чтобы процесс совершения покупок был удобным и быстрым, так и для продавцов, чтобы добавление товаров в каталог и их управление также были простыми и эффективными.

На сегодняшний день на рынке существует множество интернет-магазинов, использующих разнообразные технологии и платформы для создания своих веб-сайтов. Тем не менее, чаще всего они ориентируются на один из наиболее распространенных фреймворков для разработки интернет-магазинов — React JS.

## 1.2 Обзор аналогичных решений

Веб-сайт представляет собой совокупность взаимосвязанных веб-страниц, предназначенных для просмотра в веб-браузере. Эти веб-страницы могут быть связаны между собой посредством гипертекстовых ссылок.

Для создания веб-сайта, который быстро отвечает и обладает динамичностью, необходимо применять современные технологии, позволяющие минимизировать временные и финансовые затраты. Важным аспектом является удобство использования веб-сайта для пользователя, а также его адаптация под мобильные устройства. Тематика веб-сайта должна быть актуальной и востребованной. В связи с этим было принято решение разработать интернет-магазин, который представит собой веб-сайт, предназначенный для покупки различных товаров.

Перед тем как приступить к разработке веб-сайта, необходимо провести анализ аналогов, чтобы изучить используемые технологии, выявить недостатки и преимущества аналогичных веб-сайтов. Ключевым моментом также является выявление уникальных особенностей каждого аналога, которые выделят создаваемый веб-сайт среди других. Проанализируем 3 аналога:

Аналог №1 – сайт Ozon [1].

Ozon является одним из крупнейших интернет-магазинов в России, привлекая внимание своим удобством и простотой использования. На главной странице сайта размещены разделы, способствующие быстрому поиску нужного товара. Структура сайта, представленная на рисунке 1.1, обращает на себя внимание своим дизайном, который сочетает простоту с красивым оформлением. Товары разбиты на категории, упрощая поиск для пользователей. Веб-сайт предоставляет удобство использования, а его шрифт читаем и понятен.

Особенности сайта включают динамичность и интерактивность, а также привлекательную анимацию при взаимодействии с изображениями. Сайт использует две основные цветовые гаммы – синий и белый, что благоприятно сказывается на визуальном восприятии. Уникальной функцией является возможность сравнения товаров, что облегчает выбор для пользователя. Сайт также обеспечивает удобство для слабовидящих, предлагая режим с увеличенным шрифтом и контрастностью. Покупатели могут также добавлять товары в избранное без необходимости регистрации, чтобы не утратить выбранные товары.

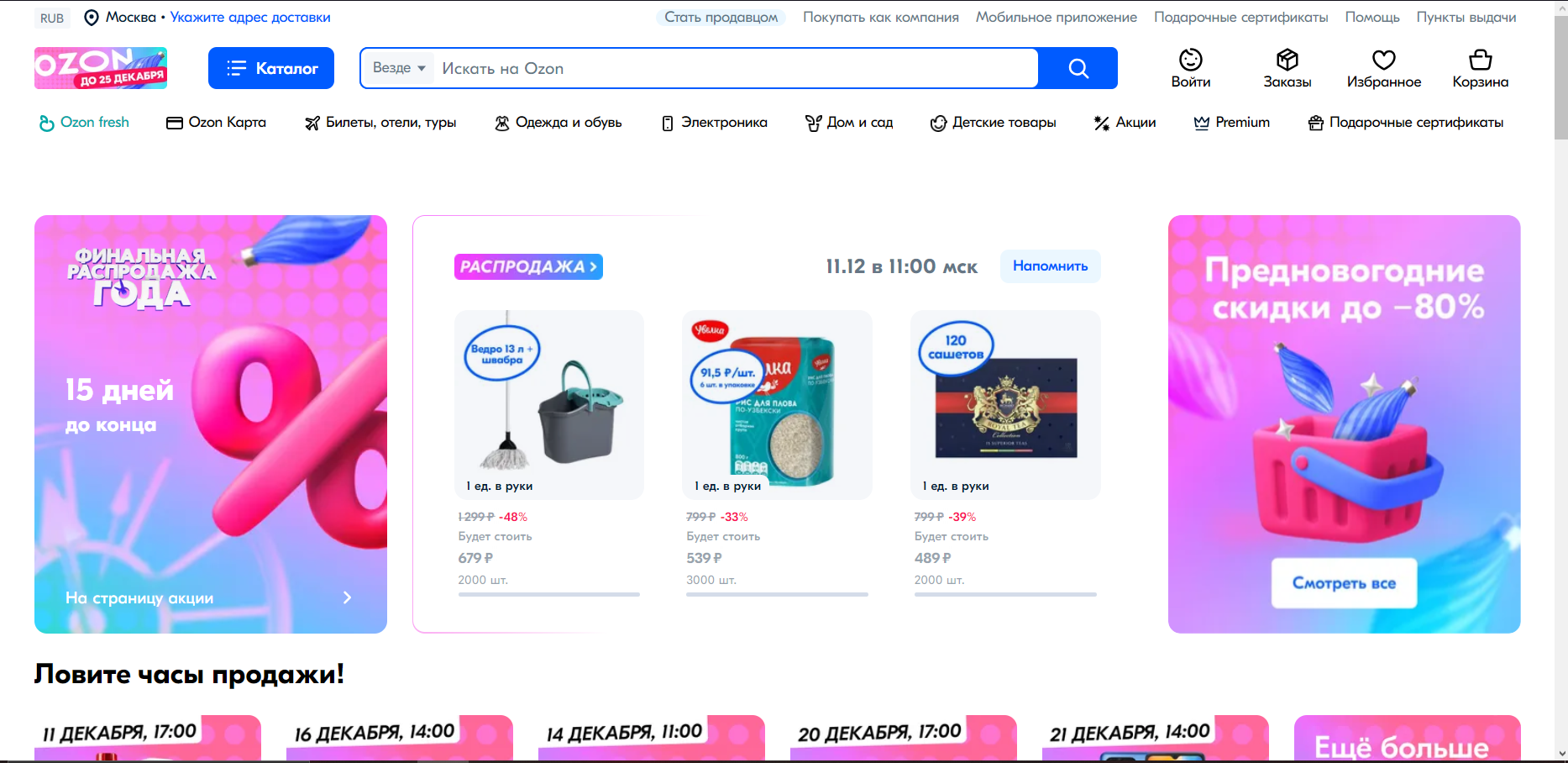


Рисунок 1.1 – Главная страница сайта Ozon

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.

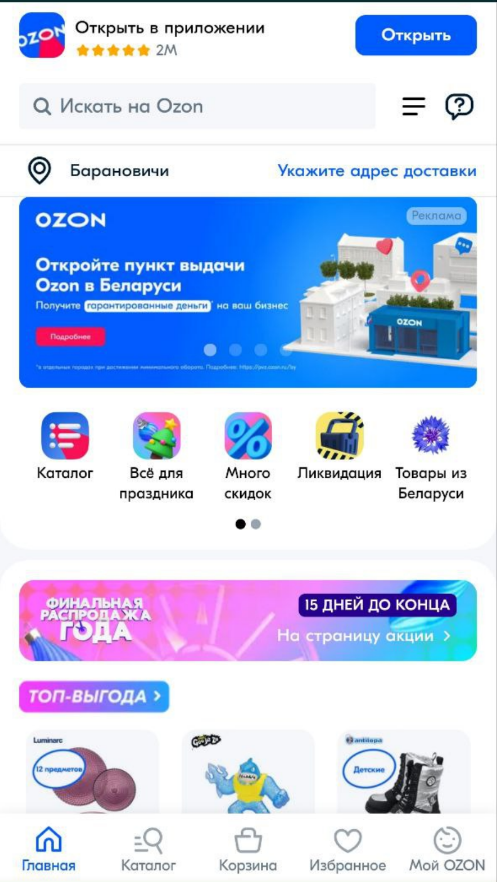


Рисунок 1.2 – Мобильная версия сайта Ozon

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 Kufar [2].

Kufar занимает лидирующую позицию среди крупных онлайн-площадок в Беларуси, привлекая внимание своей простотой и удобством использования. На главной странице портала находятся разделы, способствующие быстрому поиску необходимых товаров. Дизайн сайта, представленный на рисунке 1.3, привлекает внимание своей структурой, которая сочетает простоту с красивым оформлением. Товары систематизированы по категориям, что упрощает процесс поиска для пользователей. Портал обеспечивает удобство в использовании, а его шрифт четкий и понятный.

Особенности сайта включают в себя динамичность и интерактивность, а также привлекательные анимации при взаимодействии с изображениями. Платформа использует две основные цветовые гаммы – зеленый и белый, что благотворно сказывается на визуальном восприятии. Уникальной функцией является возможность сравнения товаров, что облегчает выбор для пользователя. Сайт также предоставляет удобства для людей с ограниченными возможностями, предлагая режим с увеличенным шрифтом и контрастностью. Покупатели могут добавлять товары в избранное без необходимости регистрации, чтобы не утратить выбранные предметы.

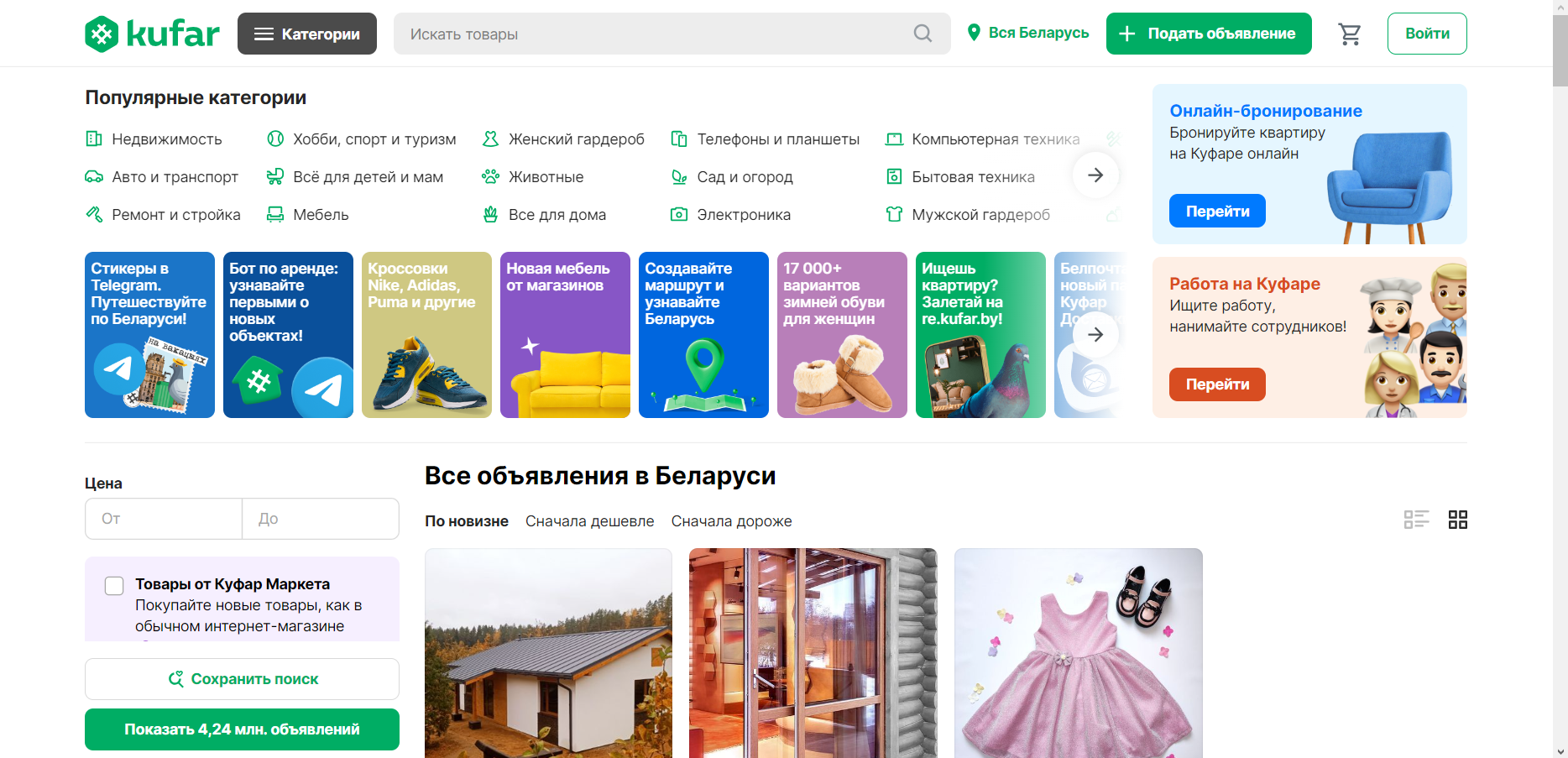


Рисунок 1.3 – Главная страница сайта Kufar

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.

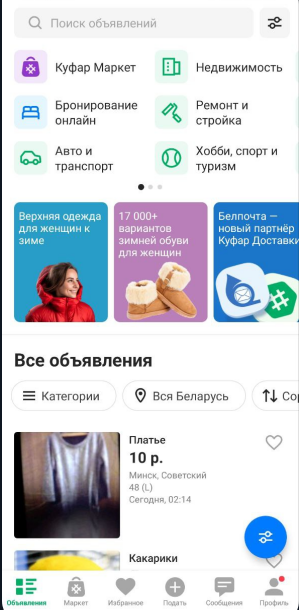


Рисунок 1.4 – Мобильная версия сайта Kufar

Среди недостатков следует отметить наличие многочисленной рекламы, которая может отвлекать пользователя от основного контента. Технологии, использованные при разработке данного портала, включают HTML, CSS, JavaScript, PHP и 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.

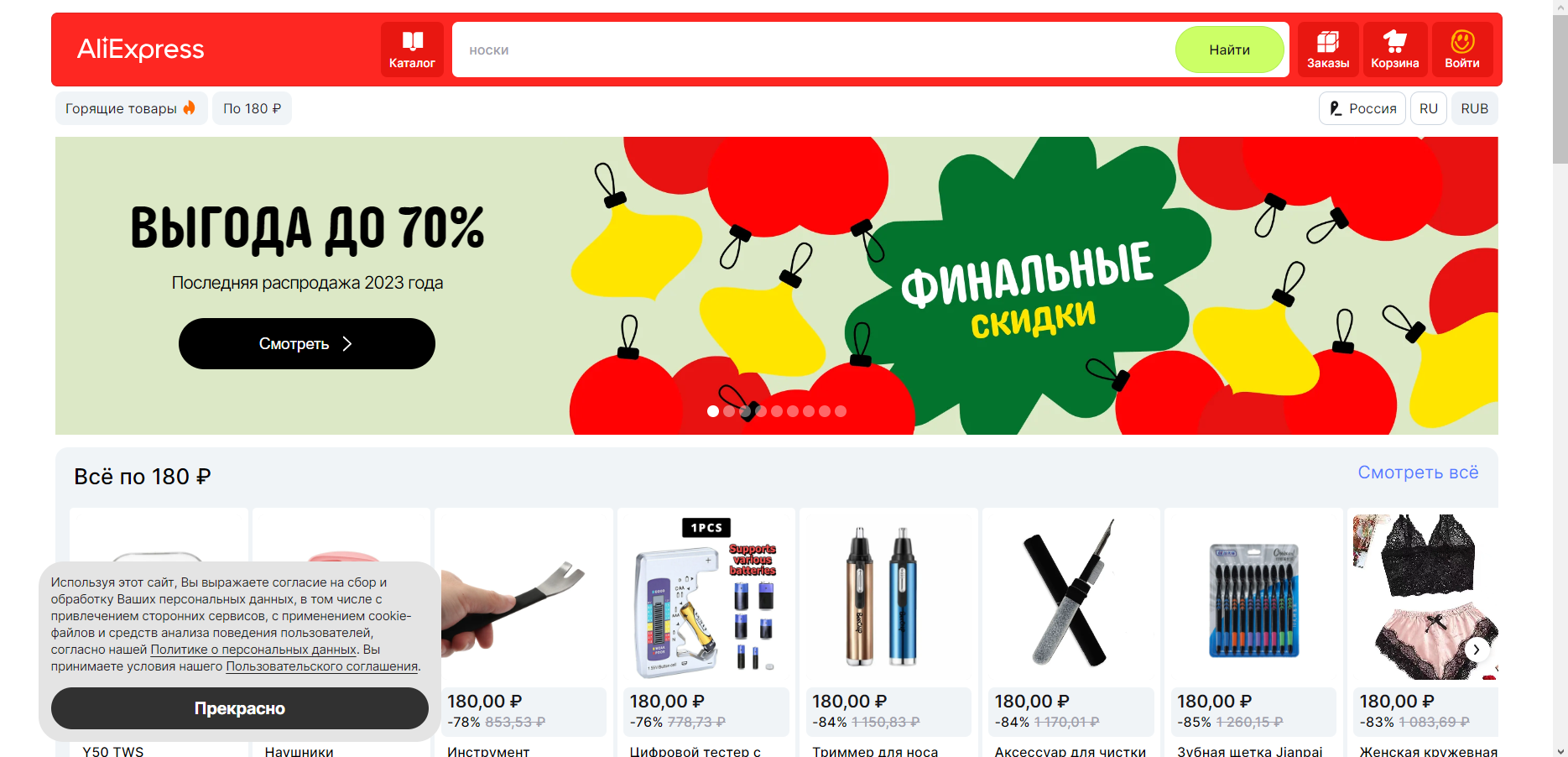


Рисунок 1.5 – Главная страница сайта Aliexpress

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

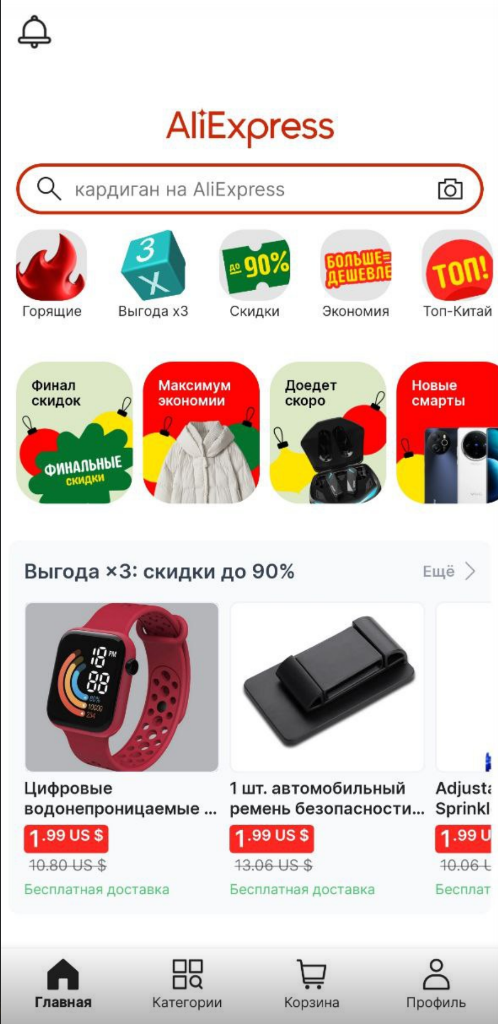


Рисунок 1.6 – Мобильная версия сайта Aliexpress

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 возможность добавления и удаления товаров из корзины;
* разработать структуру веб-сайта;
* 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.

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), Redux-Toolkit (1.3.5), 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.

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.

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).

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.

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 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:

* + MainPage (Главная страница).
  + ProductPage (Продукты).
  + Cart (Корзина).

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

На странице ProductPage будут расположены категории продуктов, фильтры, карточки продуктов, пагинация.

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

Также будет Navbar с ссылками на главную страницу, страницу каталога, страницу корзины. Navbar будет расположен вверху страницы.

## 2.3 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 3-х основных цветов – белого, красного и чёрного. Белый цвет будет использоваться для фона, а чёрный и красный 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: Arial.

Шрифт Arial 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.

## 2.4 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 FoxyExpress представлена на рисунке 2.1.



Рисунок 2.1 – Логотип сайта FoxyExpress

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 2.2.



Рисунок 2.2 – Кнопка сайта FoxyExpress

На сайте будут карточки товара, которые будут состоять из картинки и кнопок «Подробнее» и «В корзину». Кнопка «В корзину» будет перенаправлять на страницу с заказом. Кнопка «В корзину» будет добавлять товар в корзину. Дизайн карточек товара будет простым и понятным. Пример карточки товара будет на рисунке 2.3.

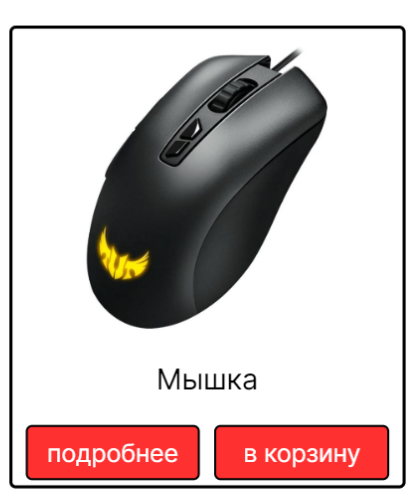


Рисунок 2.3 – Карточка товара

Также на сайте будет блок с поиском. Пользователь сможет искать продукты и разделы. Пример блока поиска товара будет рисунке 2.4.

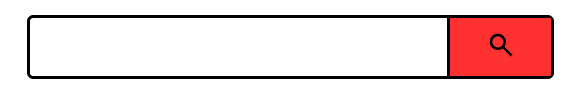


Рисунок 2.4 – Блок поиска товара

Следующим основным элементом сайта будет корзина. Пользователь сможет добавлять продукты в корзину, удалять их из корзины, оформлять заказы. Пример блока корзины будет рисунке 2.5.

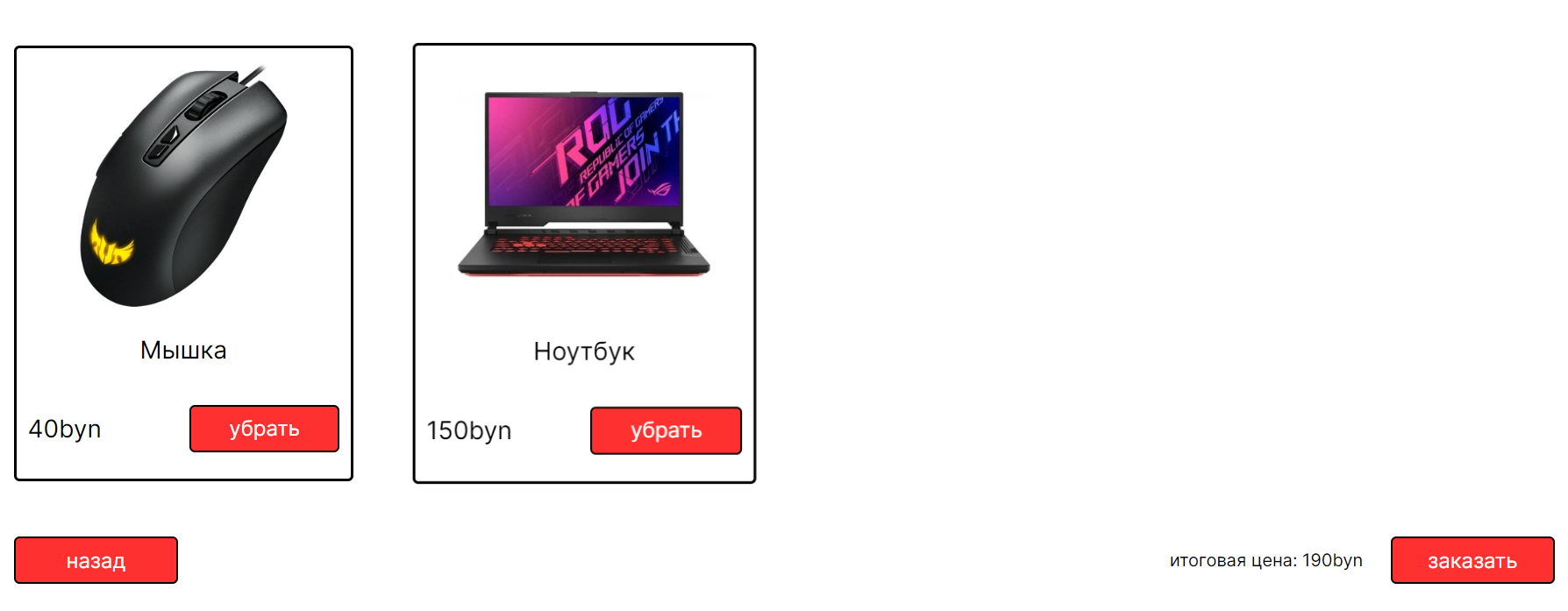


Рисунок 2.5 – Блок корзины

На сайте будут разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Анимация при наведении на кнопки.

Элемент Navbar будет состоять из логотипа сайта и навигационного меню. В навигационном меню будет 4 пункта: Каталог, Поиск, Адрес доставки, Корзина. На рисунке 2.6 представлено оформление Navbar.

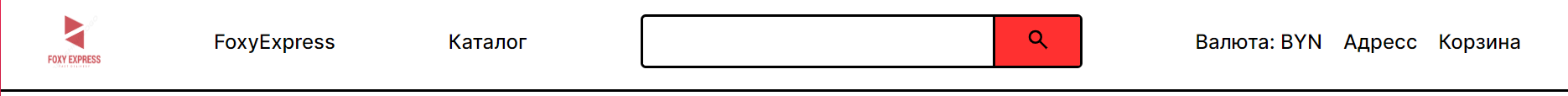


Рисунок 2.6 – Navbar сайта FoxyExpress

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.

## 2.5 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.

|  |
| --- |
| export default function TypeCard({ props }) {    const navigate = useNavigate();    const dispatch = useDispatch();    return (      <section        className={styles.type\_card}        onClick={() => {          dispatch(setType(props?.type));          navigate("/catalog");        }}      >        <img src={`http://localhost:5173/src/assets/pictures/${props?.img}`} />        <p>{props?.name}</p>      </section>    );  } |

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.

|  |
| --- |
| <div className={styles.navigation}>                <img                  onClick={() => {                    navigate("/");                  }}                  src={logo}                  className={styles.logo}                />                <p className={styles.title}>FoxyExpress</p>                <p className={styles.to\_hide}>Каталог</p>                <section className={styles.search\_field + " " + styles.to\_hide}>                  <input                    type="text"                    value={searchValue}                    onChange={(e) => {                      setSearchValue(e.target.value);                    }}                  />                  <button onClick={filterByName}>                    <svg                      xmlns="http://www.w3.org/2000/svg"                      height="24"                      viewBox="0 -960 960 960"                      width="24"                    >                      <path d="M784-120 532-372q-30 24-69 38t-83 14q-109 0-184.5-75.5T120-580q0-109 75.5-184.5T380-840q109 0 184.5 75.5T640-580q0 44-14 83t-38 69l252 252-56 56ZM380-400q75 0 127.5-52.5T560-580q0-75-52.5-127.5T380-760q-75 0-127.5 52.5T200-580q0 75 52.5 127.5T380-400Z" />                    </svg>                  </button>                </section>                <div className={styles.nav\_interface}>                  <p onClick={swapIndex}>Валюта: {vals[index]}</p>                  <Link to="/Order">Адрес</Link>                  <Link to="/cart">Корзина</Link>                </div>                <svg                  xmlns="http://www.w3.org/2000/svg"                  height="24"                  viewBox="0 -960 960 960"                  width="24"                  className={styles.adp\_btn}                  onClick={() => {                    setAdpMeny(!adpMeny);                  }}                >                  <path d="M120-240v-80h720v80H120Zm0-200v-80h720v80H120Zm0-200v-80h720v80H120Z" />                </svg>              </div>              <div                className={                  styles.adp\_meny + " " + (adpMeny ? styles.on : styles.off)                }              >                <div className={styles.nav\_interface}>                  <p>Валюта: BYN</p>                  <Link to="/Order">Адрес</Link>                  <Link to="/cart">Корзина</Link>                </div> |

Листинг 3.2 – Структура маршрутов

Следующим компонентом будет ProductPage 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 находится ниже картинки с товаром. Код карточки товара будет показан в листинге 3.3.

|  |
| --- |
| return (      <div className={styles.type\_card}>        <img className={styles.represent} src={`http://localhost:5173/src/assets/pictures/${props?.img}` } />        <p>{props?.name}</p>        <section className={styles.buttons}>          <button            onClick={() => {              dispatch(setProduct(props));              navigate("/product");            }}          >            подробнее          </button>          <button            onClick={() => {              dispatch(addToCart(props));            }}          >            в корзину          </button>        </section>      </div>    );  } |

Листинг 3.3 – Структура карточки товара

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 и кнопкой оформления заказа. Код корзины будет показан в листинге 3.4.

|  |
| --- |
| <main className={styles.container + " " + styles.main\_catalog}>          {cartFull?.length > 0 ? (            cartFull?.map((product, i) => <CartItem key={i} props={product} />)          ) : (            <h1>Пусто</h1>          )}        </main>        <div className={styles.dop\_info + " " + styles.container}>          <button            className={styles.back}            onClick={() => {              navigate("/");            }}          >            назад          </button>          <div className={styles.order}>            <p>итоговая цена: {finalPrice}byn</p>            <button              onClick={() => {                navigate("/order");              }}            >              заказать            </button>          </div>        </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. Код формы будет показан в листинге 3.5.

|  |
| --- |
| <main className={styles.container + " " + styles.form}>          <h2></h2>          <form action="">            <div>              <p>Имя</p>              <input type="text" />            </div>            <div>              <p>Фамилия</p>              <input type="text" />            </div>            <div>              <p>Отчество</p>              <input type="text" />            </div>            <div>              <p>Электронная почта</p>              <input type="text" />            </div>            <div>              <p>Номер телефона</p>              <input type="text" />            </div>            <div>              <p>Адресс доставки</p>              <input type="text" />            </div>            <div>              <p>Почтовый индекс</p>              <input type="text" />            </div>            <button className={styles.form\_btn}>Оформить заказ</button>          </form>        </main> |

Листинг 3.5 – Структура код формы

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 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.

|  |
| --- |
| const CartSLice = createSlice({    name: "cart",    initialState,    reducers: {      getCartFromLocalStorage(state, action) {        let localData = localStorage.getItem("cart");        if (!localData) return;        state.cart = JSON.parse(localData);      },      addToCart(state, action) {        let filtrationResult = state.cart.find(          (obj) => obj.id == action.payload.id        );        if (filtrationResult !== undefined) {          state.cart = state.cart.map((obj) => {            if (obj.id === action.payload.id) {              return { ...obj, count: obj.count + 1 };            }            return obj;          });        } else {          state.cart.push({ ...action.payload, count: 1 });        }        let cart = JSON.stringify(state.cart);        localStorage?.setItem("cart", cart);      },      removeFromCart(state, action) {        state.cart = state.cart.filter((item) => item.id !== action.payload.id);        localStorage?.setItem("cart", JSON.stringify(state.cart));      },    },  }); |

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.

|  |
| --- |
| export const store = configureStore({    reducer: {      products: ProdutsReducers,      cart: CartReducers,    },  }); |

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, получения данных из Store.

## 3.3 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 3.10.

|  |
| --- |
| .basic\_header {    width: 100vw;    & nav {      transition: 0.4s;      width: 100%;      height: 12vh;      &.on {        height: 20vh;      }      border-bottom: 3px solid black;    }  }  .navigation {    display: flex;    justify-content: space-between;    align-items: center;    padding: 10px 0;    & p,    a {      font-size: 1.2em;      font-weight: 500;      font-family: "Inter";    }    & img {      cursor: pointer;    }  }  .nav\_interface {    display: flex;    justify-content: space-between;    align-items: center;    gap: 20px;    & p,    a {      text-decoration: none;      color: black;      cursor: pointer;    }  } |

Листинг 3.10 – 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 добавление товаров в корзину, получение информации о товарах, 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.

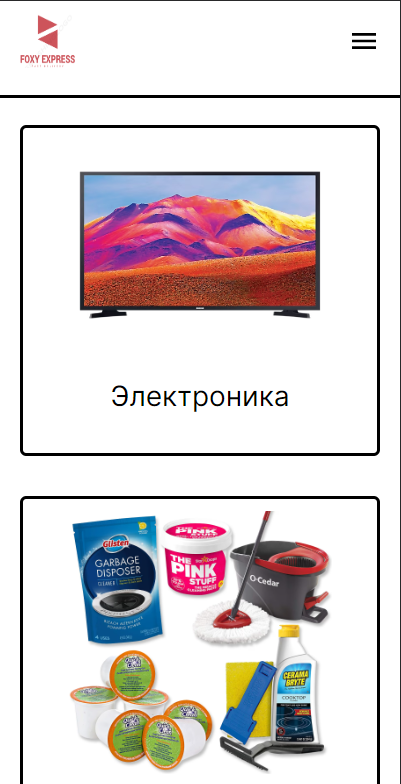


Рисунок 4.1 – Адаптивный главной страницы для телефонов

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

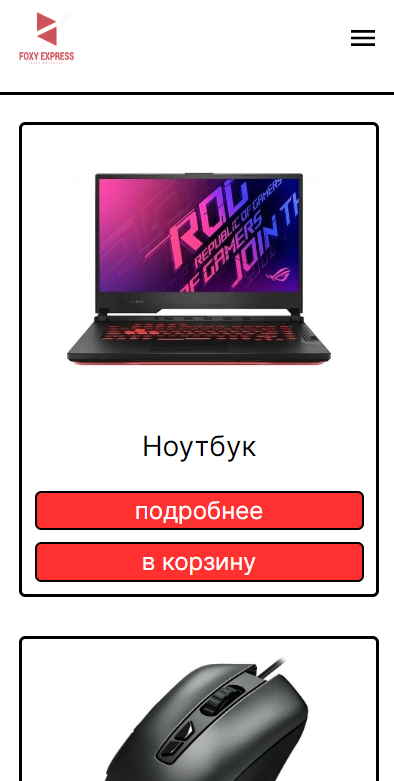
****

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.

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. 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). 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. Не было обнаружено никаких ошибок, которые могли бы повлиять на работу сайта.

# 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 возможность добавления и удаления товаров из корзины.
* Разработать структуру веб-сайта.

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 21.11.2023.
3. Aliexpress [Электронный ресурс]. – Режим доступа: https://aliexpress.ru/. – Дата обращения 21.11.2023.
4. Документация по React JS [Электронный ресурс]. - Режим доступа: https://reactjs.org/. – Дата обращения 21.11.2023.
5. Документация по Figma [Электронный ресурс]. - Режим доступа: <https://help.figma.com>. – Дата обращения 21.11.2023.
6. Документация по Redux/Redux-tolkit [Электронный ресурс]. - Режим доступа: https://redux-toolkit.js.org/ . – Дата обращения 21.11.2023.
7. Документация по Sass/Scss [Электронный ресурс]. - Режим доступа : <https://sass-scss.ru/guide/> – Дата обращения 21.11.2023.

# Приложение А Макет структуры веб-сайта



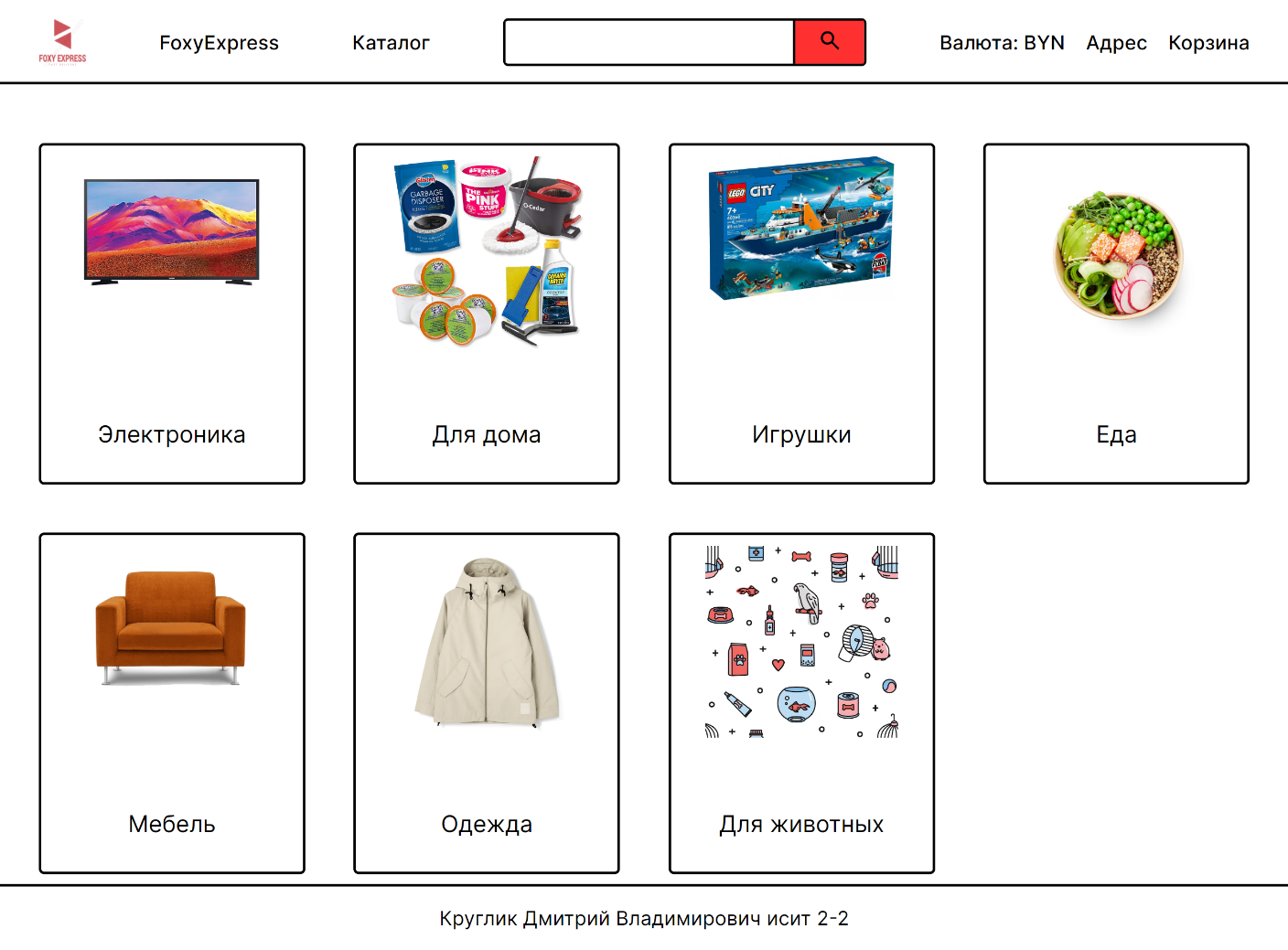


Рисунок 1 - макет страницы “MainPage”

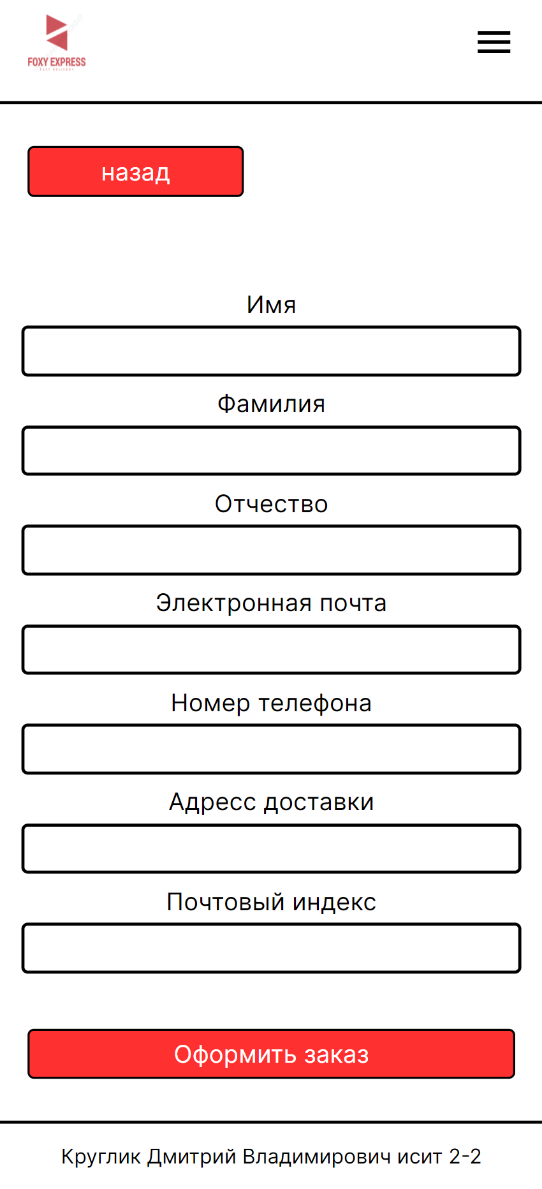
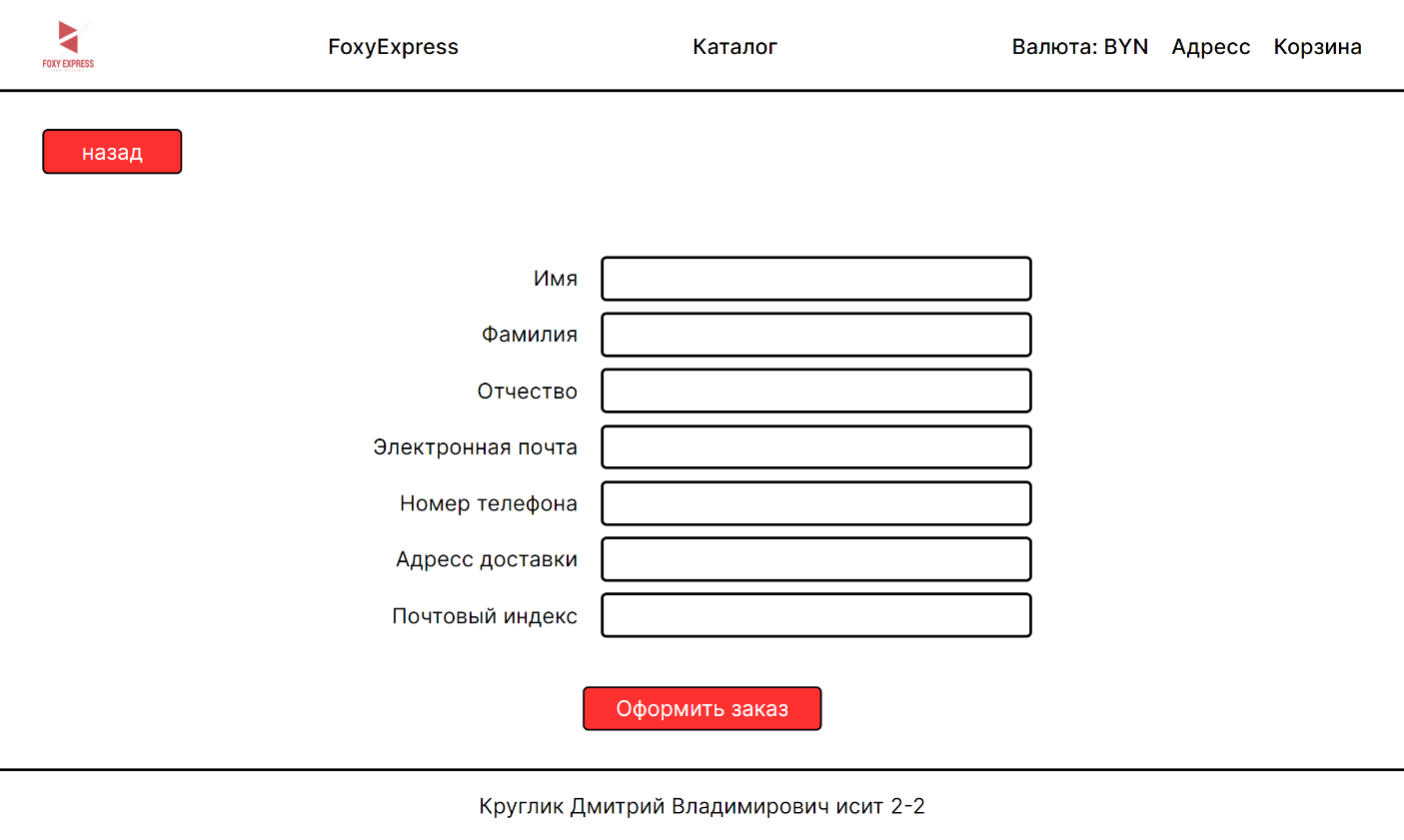


Рисунок 2 - макет страницы “Order”

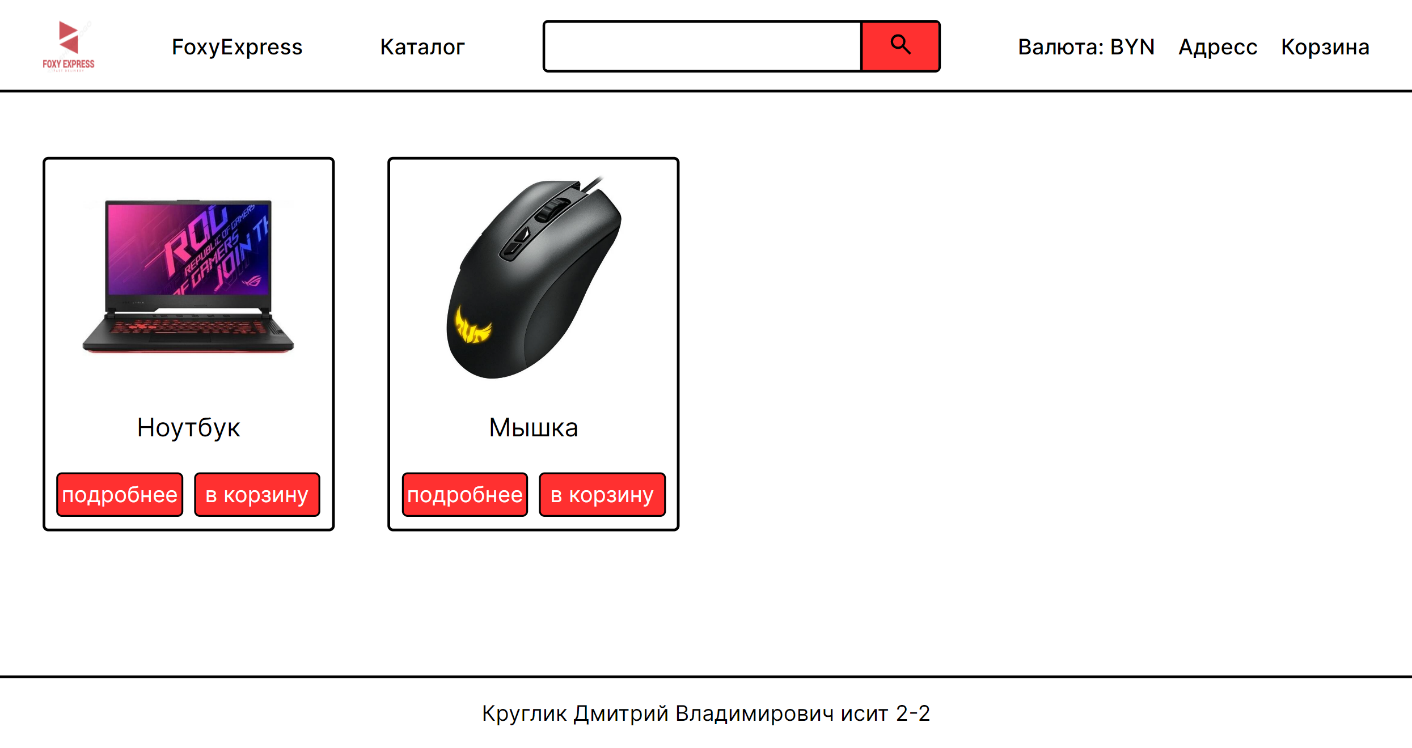
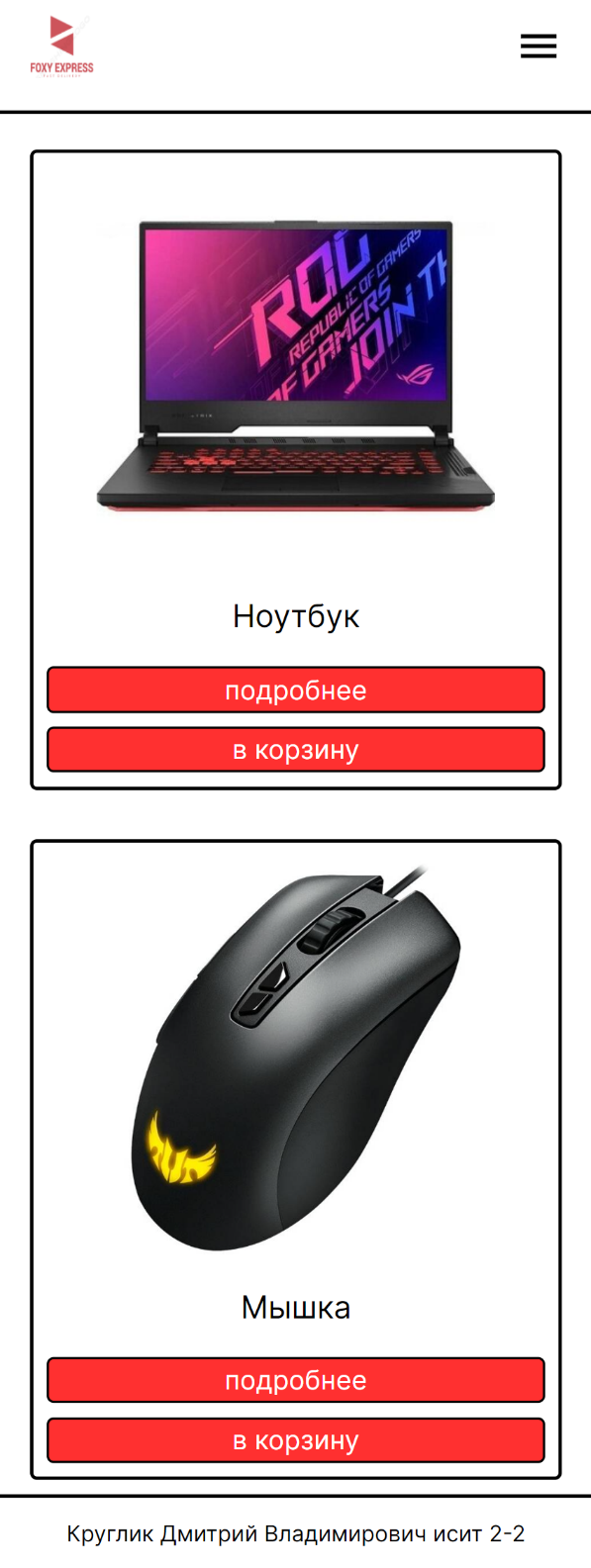


Рисунок 3-макет страницы “Catalog”

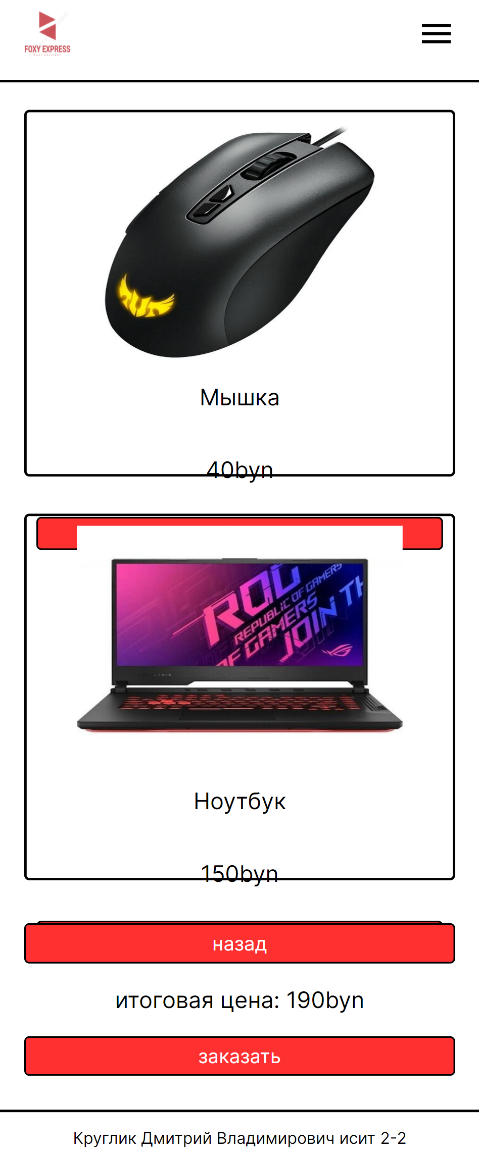
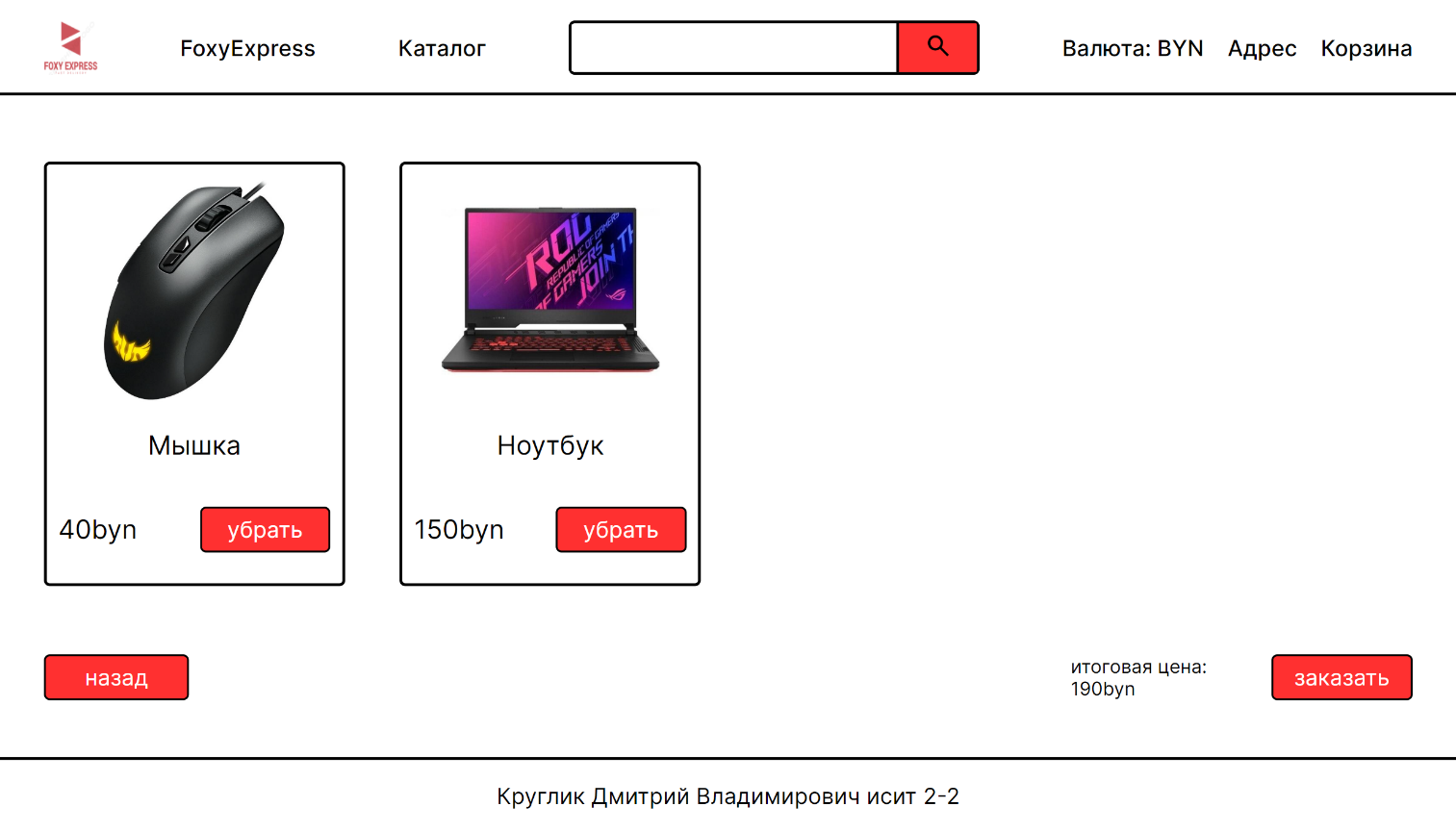


Рисунок 4-макет страницы “Cart”

# Приложение Б Листинг JSX-документа

|  |
| --- |
| import { useState, useEffect } from "react";  import { useSelector } from "react-redux";  import { Link, useNavigate } from "react-router-dom";  import logo from "../../assets/pictures/logo.png";  import TypeCard from "./TypeCard";  import styles from "../Style.module.scss";  const vals = ["BYN", "USD"];  export default function MainPage() {    const navigate = useNavigate();    const [adpMeny, setAdpMeny] = useState(false);    const [searchValue, setSearchValue] = useState("");    const { typesRepresents } = useSelector((state) => state.products);    const [prWithFilter, setPrWithFilter] = useState(typesRepresents);    const [index, setIndex] = useState(0);    const filterByName = () => {      if (searchValue.trim().length === 0) {        setPrWithFilter(typesRepresents);        return;      }      setPrWithFilter(        typesRepresents.filter((i) => i.name.includes(searchValue))      );    };    const swapIndex = () => {      setIndex(index + 1 >= vals.length ? 0 : index + 1);    };    return (      <>        <header className={styles.basic\_header}>          <nav className={adpMeny ? styles.on : styles.off}>            <div className={styles.container}>              <div className={styles.navigation}>                <img                  onClick={() => {                    navigate("/");                  }}                  src={logo}                  className={styles.logo}                />                <p className={styles.title}>FoxyExpress</p>                <p className={styles.to\_hide}>Каталог</p>                <section className={styles.search\_field + " " + styles.to\_hide}>                  <input                    type="text"                    value={searchValue}                    onChange={(e) => {                      setSearchValue(e.target.value);                    }}                  />                  <button onClick={filterByName}>                    <svg                      xmlns="http://www.w3.org/2000/svg"                      height="24"                      viewBox="0 -960 960 960"                      width="24"                    >                      <path d="M784-120 532-372q-30 24-69 38t-83 14q-109 0-184.5-75.5T120-580q0-109 75.5-184.5T380-840q109 0 184.5 75.5T640-580q0 44-14 83t-38 69l252 252-56 56ZM380-400q75 0 127.5-52.5T560-580q0-75-52.5-127.5T380-760q-75 0-127.5 52.5T200-580q0 75 52.5 127.5T380-400Z" />                    </svg>                  </button>                </section>                <div className={styles.nav\_interface}>                  <p onClick={swapIndex}>Валюта: {vals[index]}</p>                  <Link to="/Order">Адрес</Link>                  <Link to="/cart">Корзина</Link>                </div>                <svg                  xmlns="http://www.w3.org/2000/svg"                  height="24"                  viewBox="0 -960 960 960"                  width="24"                  className={styles.adp\_btn}                  onClick={() => {                    setAdpMeny(!adpMeny);                    console.log("es");                  }}                >                  <path d="M120-240v-80h720v80H120Zm0-200v-80h720v80H120Zm0-200v-80h720v80H120Z" />                </svg>              </div>              <div                className={                  styles.adp\_meny + " " + (adpMeny ? styles.on : styles.off)                }              >                <div className={styles.nav\_interface}>                  <p>Валюто: BYN</p>                  <Link to="/Order">Адресс</Link>                  <Link to="/cart">Корзина</Link>                </div>                <section                  className={styles.search\_field + " " + styles.search\_field}                >                  <input                    type="text"                    value={searchValue}                    onChange={(e) => {                      setSearchValue(e.target.value);                    }}                  />                  <button onClick={filterByName}>                    <svg                      xmlns="http://www.w3.org/2000/svg"                      height="24"                      viewBox="0 -960 960 960"                      width="24"                    >                      <path d="M784-120 532-372q-30 24-69 38t-83 14q-109 0-184.5-75.5T120-580q0-109 75.5-184.5T380-840q109 0 184.5 75.5T640-580q0 44-14 83t-38 69l252 252-56 56ZM380-400q75 0 127.5-52.5T560-580q0-75-52.5-127.5T380-760q-75 0-127.5 52.5T200-580q0 75 52.5 127.5T380-400Z" />                    </svg>                  </button>                </section>              </div>            </div>          </nav>        </header>        <main className={styles.container + " " + styles.main\_catalog}>          {prWithFilter.length === 0 ? (            <h1 style={{ textAlign: "center", width: "100%" }}>Ничего нет</h1>          ) : (            prWithFilter?.map((item, i) => <TypeCard props={item} key={i} />)          )}        </main>        <footer>          <p>Круглик Дмитрий Владимирович исит 2-2</p>        </footer>      </>    );  } |

Листинг 1 - структура JSX—документа

# Приложение В Листинг (SASS)

|  |
| --- |
| @import "card";  body,  html {    margin: 0;    padding: 0;    overflow-x: hidden;  }  \* {    font-family: "Inter";  }  .container {    width: 94vw;    margin: 0 auto;  }  .titlle {    cursor: default;  }  .basic\_header {    width: 100vw;    & nav {      transition: 0.4s;      width: 100%;      height: 12vh;      &.on {        height: 20vh;      }      border-bottom: 3px solid black;    }  }  .navigation {    display: flex;    justify-content: space-between;    align-items: center;    padding: 10px 0;    & p,    a {      font-size: 1.2em;      font-weight: 500;      font-family: "Inter";    }    & img {      cursor: pointer;    }  }  .nav\_interface {    display: flex;    justify-content: space-between;    align-items: center;    gap: 20px;    & p,    a {      text-decoration: none;      color: black;      cursor: pointer;    }  }  .logo {    width: 7vh;    height: 7vh;  }  .search\_field {    display: flex;    justify-content: space-between;    align-items: center;    width: 30%;    height: 7vh;    border: 3px solid black;    border-radius: 5px;    box-sizing: border-box;    & button {      background: #ff3030;      width: 20%;      height: 100%;      border-top-right-radius: 1px;      border-bottom-right-radius: 1px;      border: none;      border-left: 3px solid black;      box-sizing: border-box;      outline: none;      cursor: pointer;    }    & input {      height: 90%;      width: 80%;      outline: none;      border: none;      font-size: 1.2em;    }  }  .full\_buttons {    display: flex;    justify-content: space-between;    align-self: center;    width: 100%;    & button {      transition: 0.3s;      display: flex;      justify-content: center;      align-items: center;      width: 31%;      height: 6vh;      background: #ff3030;      border: 2px solid black;      border-radius: 5px;      color: white;      font-size: 1.2em;      cursor: pointer;      &:hover {        transition: 0.3s;        transform: translateY(-5px);      }    }  }  .get\_back {    margin-top: 5vh;    margin-bottom: 5vh;    & button {      transition: 0.2s;      display: flex;      justify-content: center;      align-items: center;      height: 6vh;      width: 10vw;      background: #ff3030;      border: 2px solid black;      border-radius: 5px;      color: white;      font-size: 1.2em;      cursor: pointer;      &:hover {        transition: 0.2s;        transform: translateY(-5px);      }    }  }  .dop\_info {    display: flex;    justify-content: space-between;    align-items: center;    margin-top: 5vh;    margin-bottom: 5vh;    & button {      transition: 0.2s;      display: flex;      justify-content: center;      align-items: center;      height: 6vh;      background: #ff3030;      border: 2px solid black;      border-radius: 5px;      color: white;      font-size: 1.2em;      cursor: pointer;      &:hover {        transition: 0.2s;        transform: translateY(-5px);      }    }    & .back {      width: 10vw;    }  }  .order {    display: flex;    justify-content: space-between;    align-items: center;    width: 25%;    & button {      width: 10vw;    }  }  .buttons {    display: flex;    align-items: center;    justify-content: space-between;    width: 100%;    & button {      transition: 0.2s;      display: flex;      justify-content: center;      align-items: center;      width: 48%;      height: 6vh;      background: #ff3030;      border: 2px solid black;      border-radius: 5px;      color: white;      font-size: 1.2em;      cursor: pointer;      &:hover {        transition: 0.2s;        transform: translateY(-5px);      }    }  }  .description {    width: 100%;    padding-bottom: 15px;    border-bottom: 1px solid black;  }  .product\_data {    display: flex;    min-height: 60vh;    margin: 10vh auto !important;    & .product\_intro {      display: flex;      flex-direction: column;      align-items: center;      justify-content: space-between;      width: 45%;      & img {        width: 100%;      }    }    & .product\_info {      display: flex;      flex-direction: column;      justify-content: space-between;      width: 50%;      & p,      li {        font-size: 1.3em;      }    }  }  .main\_catalog {    display: flex;    align-items: center;    flex-wrap: wrap;    gap: calc((100% - 22% \* 4) / 3);    padding: 10px 0;  }  .content {    display: flex;    flex-direction: column;    min-height: 100vh;  }  footer {    display: flex;    justify-content: center;    align-items: center;    margin-top: auto;    width: 100vw;    border-top: 3px solid black;    font-size: 1.2em;  }  .adp\_btn {    display: none;    height: 8vw;    width: 8vw;  }  .adp\_meny {    transition: 0.2s;    display: none;    justify-content: space-between;    align-items: center;    &.half {      justify-content: center;    }    &.on {      transform: translateX(0vw);    }    &.off {      transform: translateX(-100vw);    }    & div {      display: flex;      gap: 10px;    }  }  .form\_btn {    transition: 0.2s;    display: flex;    justify-content: center;    align-items: center;    margin: 5vh auto;    width: 17vw;    height: 6vh;    background: #ff3030;    border: 2px solid black;    border-radius: 5px;    color: white;    font-size: 1.2em;    cursor: pointer;    &:hover {      transition: 0.2s;      transform: translateY(-5px);    }  }  .form {    display: flex;    flex-direction: column;    justify-content: center;    align-items: center;    & form {      display: flex;      flex-direction: column;      gap: 10px;      & div {        display: flex;        justify-content: flex-end;        align-items: center;        gap: 20px;        & p {          font-size: 1.2em;          margin: 0;        }        & input {          height: 5vh;          width: 30vw;          border: 3px solid black;          border-radius: 5px;        }      }    }  }  @font-face {    font-family: "Inter";    src: url("../assets/Fonts/Inter-VariableFont\_slnt\,wght.ttf");  }  @media (max-width: 1000px) and (min-width: 700px) {    .navigation p,    .navigation a {      font-size: 1em;    }    .get\_back {      & button {        width: 13vw;      }    }    .nav\_interface {      gap: 10px;    }    .main\_catalog {      gap: calc((100% - 31% \* 3) / 2);    }    .type\_card {      width: 31%;      height: 45vh;      margin-top: calc((100% - 31% \* 3) / 2);    }    .dop\_info {      & .order {        width: 40%;      }    }  }  @media (max-width: 700px) and (min-width: 500px) {    .navigation p,    .navigation a {      font-size: 1em;    }    .get\_back {      & button {        width: 16vw;      }    }    .dop\_info {      .order {        width: 50%;      }      & button {        width: 20vw !important;      }    }    .basic\_header {      & nav {        height: 15vh;      }    }    .product\_data {      justify-content: space-between;      & .product\_intro {        width: 55%;      }      & .product\_info {        width: 40%;      }    }    .form\_btn {      width: 50vw;    }    .full\_buttons {      flex-wrap: wrap;      & button {        width: 48%;        font-size: 1em;      }      & button:last-child {        width: 100%;        margin-top: 10px;      }    }    .form {      & form {        & div {          & input {            width: 50vw;          }        }      }    }    .search\_field {      width: 40%;      height: 5vh;      & input {        width: 60%;      }      & button {        width: 40%;      }    }    .navigation {      flex-wrap: wrap;    }    .nav\_interface {      width: 100%;      justify-content: center;      gap: 10px;    }    .main\_catalog {      gap: calc((100% - 48% \* 2));    }    .type\_card {      width: 48%;      height: 48vh;      margin-top: calc((100% - 48% \* 2));    }  }  @media (max-width: 500px) and (min-width: 0px) {    .container {      width: 90vw;    }    .form\_btn {      width: 100%;    }    .form {      width: 90vw;      & form {        width: 100%;        & div {          flex-direction: column;          gap: 5px;          width: 100%;          & input {            width: 100%;          }        }      }    }    .get\_back {      & button {        width: 40vw;      }    }    .dop\_info {      .order {        flex-direction: column;        width: 100%;        & p {          font-size: 1.4em;        }      }      & button {        width: 100% !important;      }    }    .dop\_info {      flex-direction: column;    }    footer {      font-size: 1em;    }    .product\_data {      flex-direction: column;      margin: 2vh auto !important;      & .product\_intro {        width: 100%;      }      & .product\_info {        width: 100%;      }    }    .full\_buttons {      flex-wrap: wrap;      & button {        width: 48%;        font-size: 1em;      }      & button:last-child {        width: 100%;        margin-top: 10px;      }    }    .navigation p,    .navigation a {      font-size: 1em;    }    .adp\_btn {      display: block;    }    .adp\_meny {      display: flex;      gap: 30px;    }    .search\_field {      height: 5vh;      width: 50%;      & input {        width: 60%;      }      & button {        width: 40%;      }    }    .title {      display: none;    }    .nav\_interface {      display: none;    }    .main\_catalog {      flex-direction: column;      gap: 20px;    }    .type\_card {      height: auto;      width: 100%;    }    .to\_hide {      display: none;    }    .buttons {      flex-direction: column;      gap: 10px;      & button {        width: 100%;        height: 5vh;        font-size: 1.2em;      }    }  } |

Листинг 2 - структура SASS

# Приложение Г Листинг Redux

|  |
| --- |
| import { createSlice } from "@reduxjs/toolkit";  const initialState = {    cart: [],  };  const CartSLice = createSlice({    name: "cart",    initialState,    reducers: {      getCartFromLocalStorage(state, action) {        let localData = localStorage.getItem("cart");        if (!localData) return;        state.cart = JSON.parse(localData);      },      addToCart(state, action) {        let filtrationResult = state.cart.find(          (obj) => obj.id == action.payload.id        );        if (filtrationResult !== undefined) {          state.cart = state.cart.map((obj) => {            if (obj.id === action.payload.id) {              return { ...obj, count: obj.count + 1 };            }            return obj;          });        } else {          state.cart.push({ ...action.payload, count: 1 });        }        let cart = JSON.stringify(state.cart);        localStorage?.setItem("cart", cart);      },      removeFromCart(state, action) {        state.cart = state.cart.filter((item) => item.id !== action.payload.id);        localStorage?.setItem("cart", JSON.stringify(state.cart));      },    },  });  export const CartReducers = CartSLice.reducer;  export const { addToCart, removeFromCart, getCartFromLocalStorage } =    CartSLice.actions; |

Листинг 3 - структура Redux