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

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема «Разработка интернет-магазина головоломок»

**Исполнитель**

студент 2 курса 7 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Воликов Д. А.

подпись, дата

**Руководитель**

Доцент каф. ИСиТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жиляк Н. А.

должность, учен. степень, ученое звание подпись, дата

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

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

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

дата, подпись

Курсовой проект защищен с оценкой

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

подпись дата инициалы и фамилия

**Содержание**

[Введение 5](#_Toc121885877)

[1. Постановка задачи 6](#_Toc121885878)

[1.1. Обзор аналогичных решений 6](#_Toc121885879)

[1.2. Техническое задание 7](#_Toc121885880)

[1.3. Выбор средств реализации программного средства 8](#_Toc121885881)

[1.4. Вывод 10](#_Toc121885882)

[2. Проектирование веб-приложения 11](#_Toc121885883)

[2.1. Выбор способа верстки 11](#_Toc121885884)

[2.2. Выбор стилевого оформления 11](#_Toc121885885)

[2.3. Выбор шрифтового оформления 11](#_Toc121885886)

[2.4. Разработка логотипа 12](#_Toc121885887)

[2.5. Разработка пользовательских элементов 12](#_Toc121885888)

[2.7. Выводы 13](#_Toc121885889)

[3. Реализация структуры веб-приложения 15](#_Toc121885890)

[3.1. Клиентская часть приложения 15](#_Toc121885891)

[3.2. Серверная часть приложения 24](#_Toc121885892)

[3.3. Выводы 32](#_Toc121885893)

[4. Тестирование веб-приложения 33](#_Toc121885894)

[4.1. Адаптивный дизайн веб-сайта 33](#_Toc121885895)

[4.2. Крооссбраузерность веб-приложения 33](#_Toc121885896)

[4.3. Запрос данных из БД 34](#_Toc121885897)

[4.4. Вывод 35](#_Toc121885898)

[Заключение 36](#_Toc121885899)

[Список использованных источников 37](#_Toc121885900)

# Введение

В настоящее время Интернет становиться все более развитой средой для осуществления коммуникаций. В связи с глобальным развитием сети Интернет, в программировании все более резко начала выделяться отдельная его отрасль — веб-программирование.

Сейчас, чтобы привлечь внимание клиентов, покупателей или партнёров, просто необходимо заявить о себе в интернете, путём создания веб-сайта. Эффективно работающий ресурс положительно влияет на уровень продаж компании, укрепляет ее позиции среди конкурентов. Поэтому роль сайта чрезвычайно велика, и от того, насколько правильно он подобран, во многом зависит успешность бизнеса.

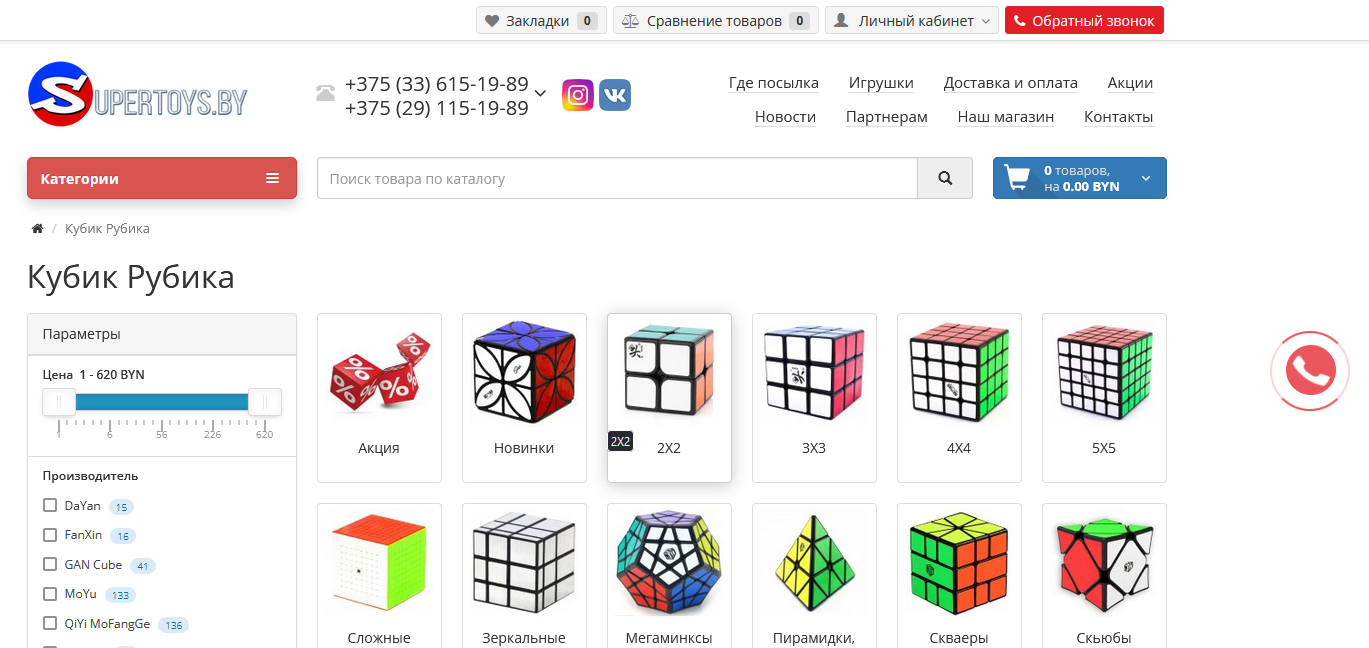
Тема моей курсовой работы «Разработка интернет-магазина головоломок». Веб-сайт предоставляет информацию о компании «Puzrain», которая занимается продажей и поставкой головоломок. Сегодня головоломки имеют большую популярность среди молодёжи, а также среди и старшего поколения.

Задачи данной курсовой работы: реализовать адаптивность и кроссбраузерность веб-сайта; подобрать уникальный дизайн веб-сайта, который будет выделять его среди конкурентов и создавать комфортные ощущения у пользователя. Для этого требуется исследовать несколько популярных веб-сайтов этой же темы и выделить основные вещи, которые будут реализованы у меня.

# 1. Постановка задачи

## 1.1. Обзор аналогичных решений

Прежде, чем начать разработку собственного интернет-магазина, следует исследовать сайты с такой же темой. Исследование сайтов даст нам понятие того, как следует распределять элементы на странице. Что стоит перенять, а что стоит точно изменить.

SuperToys [[1]](#_Список_использованных_источников). SuperToys– один из самых популярных интернет-магазинов в Беларуси (было бы довольно глупо обойти его стороной). Сам веб-сайт работает давно и как только не менялся его дизайн. Однако сейчас он выглядит достаточно простым и неброским.

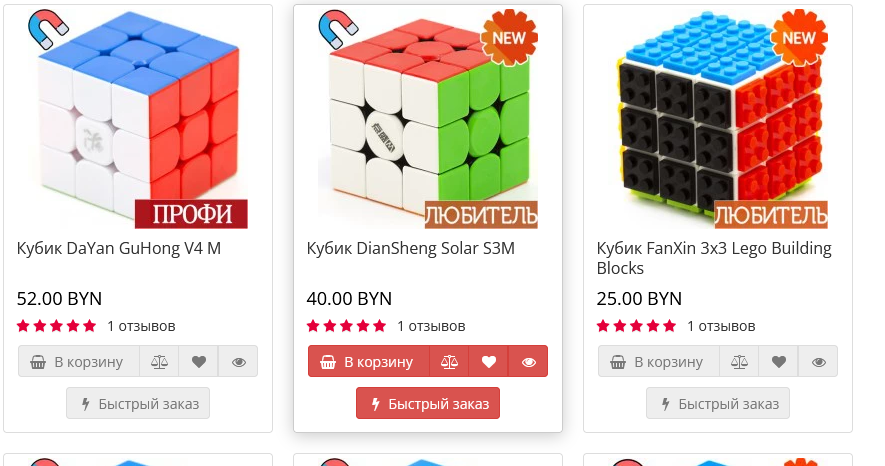
Рисунок 1.1 – Главная страница supertoys.by

Рисунок 1.2 – Главная страница supertoys.by

Список элементов, которые стоит перенять:

* Броский логотип с названием компании и неким девизом
* Страницу корзины товаров
* Страницу избранных товаров
* Реализовать меню конкретных тем, которые покупателю стоит прочитать

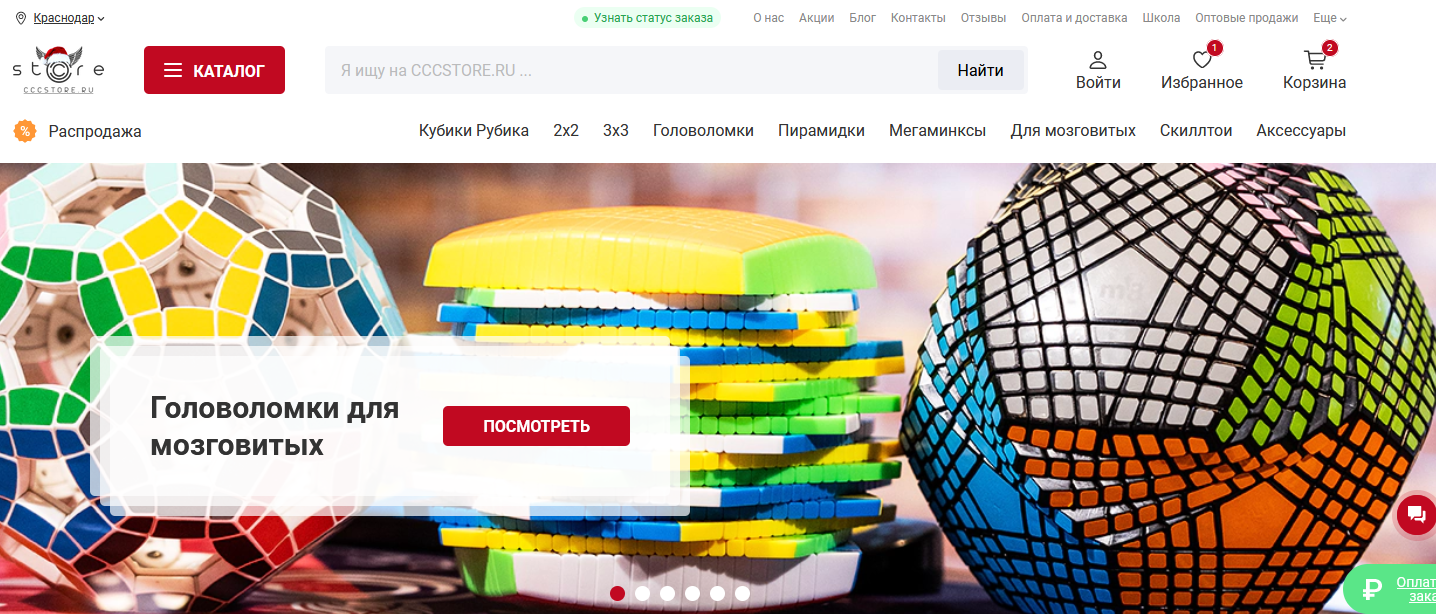
CCCSTORE [[2]](#_Список_использованных_источников). CCCSTORE – также считается одним из популярных интернет-магазинов головоломок, но в России. Стиль дизайна бросок, сразу захватывают взгляд пользователя какие-либо предложения и акции. Цветовая схема подобрана грамотно.

Рисунок 1.3 – Главная страница muz.by

Список элементов, которые стоит перенять:

* Перечислить на главной странице коллективы, которые являются частыми покупателями интернет-магазина и доверяют ему
* Список городов в виде некого выпадающего меню с названиями городов Беларуси
* Не добавлять в логотип домен сайта, ибо это выглядит отталкивающе

## 1.2. Техническое задание

Основными требованиями к программному продукту являются адаптивность, лёгкость. На создаваемом веб-приложении не может быть больших программных функций или очень сложных анимаций – всё это может сильно замедлить скорость загрузки информации и ухудшить её восприятие. Адаптивность разрабатывалась исходя из существующих актуальных браузеров и устройств.

Шапка веб-приложения, которая содержит меню навигации, логотип, ссылка на корзину, ссылка на избранное и ссылка на страницу входа должна быть на всех страницах сайта одинаковая. Меню навигации не должно быть выпадающим, а просто должно содержать переход на другие страницы. Шапка сайта должна быть фиксированной.

Данный продукт должен реализовывать те функции, которые являются основными функциями интернет-магазина: вывод каталога инструментов на страницу; список инструментов одной категории; вывод страницы товара. Страницы не должны быть сильно запичканы какой-то информацией. Только по существу, только по делу.

Программный продукт должен предоставлять возможности пользователям по оформлению заказов. Для этого есть реализация некоторой маленькой формы, где пользователь должен оставить контактную информацию.

Пользователь должен знать, каким способами он может оплачивать товар. Для этого реализована отдельная страница с информацией об этой теме. Также следует добавить информацию о доставке самого товара.

Список товаров каталога должен иметь структуру сетки. Самый простой и привлекательный способ вывода товаров на страницу.

## 1.3. Выбор средств реализации программного средства

Веб-приложение разрабатывалось в текстовом редакторе Visual Studio Code. Страницы сайта могут просматриваться в таких браузерах как Microsoft Edge, Safari, Opera, Mozilla Firefox, Google Chrome и подобных.

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.

HTML (от английского HyperText Markup Language) — это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных. HTML используется, чтобы сообщать вашему браузеру, как отображать веб-страницы, которые вы посещаете. Браузер интерпретирует HTML-документ, выстраивая его структуру (DOM) и отображая ее в соответствии с инструкциями, включенными в этот файл (таблицы стилей, скрипты). Если разметка правильная, то в окне браузера будет отображена страница, содержащая HTML-элементы — заголовки, таблицы, изображения и т.д. В 2014 году был создан HTML5. Разработчики нового языка ориентировались на устранение проблем предыдущих версий и реализацию современных функций.

JavaScript – это интерпретируемый язык, то есть исходный код скриптов не нуждается в компиляции. Интерпретаторы этого языка автоматически управляют памятью и очищают неиспользуемые блоки, следовательно, программисту не нужно следить за выделением и очисткой памяти, как, например, в C++. JavaScript отличается от большинства объектно-ориентированных языков программирования тем, что в нём отсутствует понятие класса. Все объекты являются копиями уже существующего экземпляра объекта. Синтаксис JavaScript был вдохновлён языками C и Java. Все идентификаторы (названия переменных, функций и т.д.) являются регистрозависимыми. Операторы разделяются при помощи точки с запятой «;», однако использование этого символа необязательно: оператор автоматически завершается в конце строки.

CSS (Cascading Style Sheets, каскадные таблицы стилей) – это язык описания внешнего вида документа, написанного с использованием языка разметки. CSS позволяет представлять один и тот же документ в различных стилях. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

JSON (*JavaScript Object Notation*) — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом. Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

GitHub. GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций. React может использоваться для разработки одностраничных и мобильных приложений.

программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

MongoDB — документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных. Написана на языке C++.

Mongoose — это библиотека объектно-ориентированного программирования JavaScript, которая создает соединение между MongoDB и средой выполнения JavaScript Node.js.

Express.js, или просто Express, фреймворк web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js.

Redux — библиотека для JavaScript с открытым исходным кодом, предназначенная для управления состоянием приложения. Чаще всего используется в связке с React или Angular для разработки клиентской части. Содержит ряд инструментов, позволяющих значительно упростить передачу данных хранилища через контекст.

## 1.4. Вывод

Что в итоге требуется от хорошего приложения, созданном на React, сегодня?

Основными требованиями можно назвать понимание работы с HTML5, CSS3 и JavaScript. Так, как любое React-приложение создаётся из отдельных компонентов, построенных на этих трёх языках, можно смело говорить о том, что данные технологии являются важными для любого разработчика веб-приложения, чтобы строить клиентскую часть самого приложения.

Также нужно выделить умение понимать работу серверной части приложения, настраивать её и связывать её с клиентской частью. Для всего этого можно использовать такие популярные технологии, как Node.js и Express.js. С помощбю этих двух технологий можно быстро развернуть сервер, и использовать его в своих дальнейших целях.

# 2. Проектирование веб-приложения

## 2.1. Выбор способа верстки

Веб-приложение построено на блочной системе. Каждый элемент – это отдельный блок, в то же время каждый отдельный блок обёрнут в свой родительский блок, а те же в свою очередь тоже обёрнуты в свои родительские блоки. Структуру вёрстки страницы можно представить в виде дерева: от главной части отходят дочерние элементы, и у дочерних элементов есть свои дочерние элементы.

В cвою очередь, блоки могут представлять из себя сеточную (GRID) или гибкую (FLEX). Эти два метода верстки самые популярные на сегодняшний день, ведь они позволяют на красиво расположить элементы на странице: в ряд, в колонку или в виде некоторой таблицы.

Тип вёрстки - computers first. Сайты такого типа должны быть лучше стилизованы именно на компьютерах, ноутбуках и планшетах, ибо у этих девайсов достаточно большой дисплей. В мобильной версии все довольно просто: выстроить всё в колонку, уменьшив при этом размер шрифта.

## 2.2. Выбор стилевого оформления

Была реализована попытка построить довольно простое веб-приложение, где не будет лишних элементов, а только всё по факту информации. Это можно назвать полу минимализм.

Продукт содержит 5 основных броских цветов, которые может заметить пользователь: красный, черный, зеленый и оранжевый. Цвета были подобраны для приятного оформления сайта, дабы у пользователя не было желания покинуть его.

## 2.3. Выбор шрифтового оформления

 Подобрать шрифт было довольно нетрудно. Главная задача шрифта – это передать «настроение». Шрифт должен быть без засечек, ибо тогда бы он указывал не некую строгость, а сайт музыкальных инструментов такое не нужно. Поэтому выбор пал на такие шрифты, как Montserrat и Verdana (если присмотреться на картинку ниже, то можно заметить нотки джаза).

Рисунок 2.1 – шрифт Montserrat

Рисунок 2.2 – Шрифт Verdana

Размер текста в основном 16px, заголовки страниц имеют размер побольше (и так понятно почему). Некая важная информация выделяется шрифтом в размер 18px.

При изменении дисплея устройства текст меняет свой размер. Во-первых, это упрощает реализацию адаптивного дизайна, а во-вторых, на маленьких устройствах неудобно читать информацию со слишком большим шрифтом.

## 2.4. Разработка логотипа

При дизайне логотипа была поставлена задача показать смысл веб-сайта – продажа головоломок.

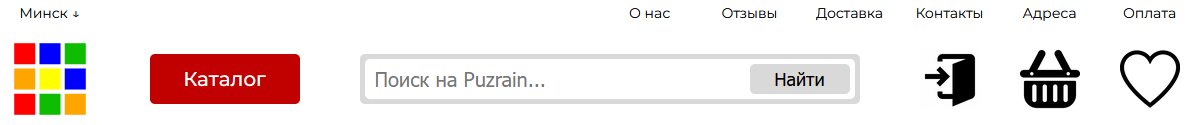
 Выбор рисунка исходился из мыслей людей о головоломках. Наверное не секрет, что одной из самых популярных головоломок в мире является кубик рубика. Поэтому я нарисовал наклейки кубика как логотип веб-сайта.

Рисунок 2.3 – логотип веб-приложения

## 2.5. Разработка пользовательских элементов

Пользовательские элементы – это компоненты React-приложения, которые взаимодействую каким-либо образом с пользователем. Веб-приложение имеет следующие основные пользовательские компоненты.

Шапка. Шапка – это верхняя часть страницы, которая содержит основные ссылки на различные страницы приложения и логотип. Реализация привлекательной глазу пользователя шапки – это из главнейших задач. В этом компоненте не должно быть напихано всего. Только самое основное, что было описано выше по тексту. Исходя их этих условий веб-приложение компании Puzrain имеет следующий вид шапки.

Рисунок 2.4 – Шапка приложения

Подвал. Подвал всегда хранит некие контактные ссылки на контактную информацию и социальные сети компании, сайт которой посещает пользователь.

 Для компании Puzrain был сделан необычный подвал, с ссылками на социальные сети.

Рисунок 2.5 – Подвал приложения

 Карточка товара. Карточка товара вышла самой простой, без лишней информации. Здесь есть изображение товара, его название, цена и кнопки управления.

Рисунок 2.6 – Пример карточки товара

## 2.7. Выводы

При проектировании веб-сайта были учтены такие пункты, как стиль текста, его размер и цвет. Все макеты были разработаны в современных приложениях для этих целей, а внешний вид элементов соответствует популярным аналогам в данный момент. Анимации добавляют «жизни» сайту, что так обязательно в наше время.

# 3. Реализация структуры веб-приложения

## 3.1. Клиентская часть приложения

Как и оговорено в техническом задании, клиентская часть построена на компонентах со своими стилями. Сами компоненты имеют подкомпоненты, что также даёт нам такую возможность, как реально разделить каждую часть приложения на отдельные «кирпичики», которые строят наш продукт.

import React from 'react';

import ReactDOM from 'react-dom/client';

import { Provider } from 'react-redux';

import store from './store';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<Provider store={store}>

<App />

</Provider>

</React.StrictMode>

);

Листинг 3.1 – код файла index.js

import { Routes, BrowserRouter, Route } from 'react-router-dom';

import Header from './components/header/Header';

import Footer from './components/footer/Footer';

import Main from './pages/main';

import About from './pages/about';

import Delivery from './pages/delivery';

import Contacts from './pages/contacts';

import Payment from './pages/payment';

import BasketPage from './pages/basket';

import PDPPage from './pages/pdp';

import PLPPage from './pages/plp';

import Reviews from './pages/reviews';

import FavPage from './pages/fav';

import Login from './pages/login';

import Register from './pages/register';

import ErrorPage from './pages/error';

import './App.css';

import URLS from './URLS';

function App() {

if (URLS.indexOf(window.location.pathname) === -1)

return window.location.pathname = "/error";

return (

<>

<Header />

<main>

<BrowserRouter>

<Routes>

<Route path="/" element={<Main />}/>

<Route path="/about" element={<About />}/>

<Route path="/delivery" element={<Delivery />}/>

<Route path="/payment" element={<Payment />}/>

<Route path="/contacts" element={<Contacts />}/>

<Route path="/basket" element={<BasketPage />}/>

<Route path="/product" element={<PDPPage />}/>

<Route path="/catalog" element={<PLPPage />}/>

<Route path="/reviews" element={<Reviews />}/>

<Route path="/favourite" element={<FavPage />}/>

<Route path="/login" element={<Login />}/>

<Route path="/register" element={<Register />}/>

<Route path="/error" element={<ErrorPage />}/>

</Routes>

</BrowserRouter>

</main>

<Footer />

</>

);

}

export default App;

Листинг 3.2 – код файла App.js

import RegionSwitcher from './menu/RegionSwitcher/RegionSwitcher';

import Menu from './menu/Menu/Menu';

import Catalog from './main/Catalog/Catalog';

import Searchbar from './main/Searchbar/Searchbar';

import Links from './main/Links/Links';

import './Header.css';

export default function Header()

{

return (

<header>

<div className="content">

<div className="header-row row-one">

<RegionSwitcher />

<Menu />

</div>

<div className="header-row row-two">

<div className="header-logo">

<a href="/"><img src="logo.svg" alt="Logo" width="80px" style={{display: 'block'}}/></a>

</div>

<Catalog />

<Searchbar />

<Links />

</div>

</div>

</header>

);

}

Листинг 3.3 – код файла шапки

import Rights from "./Rights/Rights";

import SpecialWords from "./SpecialWords/SpecialWords";

import SocialNetworks from "./SocialNetworks/SocialNetworks";

import './Footer.css';

export default function Footer() {

return (

<footer>

<div className="content">

<Rights />

<SpecialWords />

<div className="footer-logo">

<img src="logo.svg" alt="Logo" width="80px" />

</div>

<SocialNetworks />

</div>

</footer>

);

}

Листинг 3.4 – код файла подвала

import { useSelector } from 'react-redux';

import Good from './Good/Good';

import Card from './Card/Card';

import './Basket.css';

export default function Basket() {

const products = useSelector(state => state.basket.basket);

const resultProducts = products.map(product => {

return (

<Good key={product.\_id} product={product} />

);

});

return (

<div className="main-basket">

<div className="basket-goods">

{resultProducts.length > 0 ? resultProducts : <h2 style={{fontSize: 36, fontWeight: 500}}>Нет товаров</h2>}

</div>

<Card products={products} />

</div>

);

}

Листинг 3.5 – код файла корзины

import { useSelector } from 'react-redux';

import FavProduct from './product/favProduct';

export default function Fav() {

const favProducts = useSelector(state => state.fav.fav);

const resultProducts = favProducts.map(product => {

return (

<FavProduct key={product.\_id} value={product} className={"fav-goods\_good"} />

);

})

return (

<div className="main-fav">

<div className="fav-goods">

{resultProducts.length > 0 ? resultProducts : <h2 style={{fontSize: 26, fontWeight: 500}}>Нет товаров</h2>}

</div>

</div>

);

}

Листинг 3.6 – код файла избранного

import Breadcrumbs from './Breadcrumbs/Breadcrumbs';

import PDPProduct from './PDPProduct/PDPProduct';

import Panel from './Panel/Panel';

import Description from './Description/Description';

import '../main/Main.css';

import './PDP.css';

export default function PDP(props) {

const product = props.product;

return (

<div className="main-pdp">

<Breadcrumbs value={`<a href="/catalog">Catalog</a> / ${product.category} / ${product.title}`} />

<div className="main-pdp\_control">

<PDPProduct product={product} />

<Panel product={product} />

</div>

<Description value={product.description} img={product.category + "/" + product.title + "-1.png"}/>

<div className="main-pdp\_thanks">

Спасибо, что выбираете нас :)

</div>

</div>

);

}

Листинг 3.7 – код файла товара

import { useState, useEffect } from "react";

import Filters from './Filters/Filters';

import ProductsList from './ProductsList/ProductsList';

import '../main/Main.css';

import './PLP.css';

export default function PLP() {

const [products, setProducts] = useState([]);

useEffect(() => {

fetch('http://localhost:3001/products')

.then(res => res.json())

.then(data => setProducts(data));

}, []);

if (products)

return(

<div className="main-plp">

<h2 className="plp-header">Каталог</h2>

<div className="plp-content">

<Filters />

<ProductsList products={products}/>

</div>

</div>

);

}

Листинг 3.8 – код файла страницы каталога PLP.js

import { useState, useEffect } from "react";

import SpecialEvent from "../components/main/SpecialEvent/SpecialEvent";

import PopularProducts from "../components/main/PopularProducts/PopularProducts";

import VideoReviews from "../components/main/VideoReviews/VideoReviews";

import NewProducts from "../components/main/NewProducts/NewProducts";

import Comments from "../components/main/Comments/Comments";

import Companies from "../components/main/Companies/Companies";

import '../components/main/Main.css';

export default function Main() {

const [products, setProducts] = useState([]);

useEffect(() => {

fetch('http://localhost:3001/products')

.then(res => res.json())

.then(data => setProducts(data));

}, []);

const popularProducts = products.filter(product => product.status === "pop");

const newProducts = products.filter(product => product.status === "new");

if (products)

return (

<div className="content">

<SpecialEvent />

<PopularProducts products={popularProducts} />

<VideoReviews />

<NewProducts products={newProducts} />

<Comments />

<Companies />

</div>

);

}

Листинг 3.9 – код файла главной страницы

import '../components/main/Main.css';

import './styles/page.css';

export default function Login() {

document.querySelector('title').innerText = "Puzrain - Авторизация";

return (

<div className="content">

<div className="main-login">

<h1>Авторизация</h1>

<div className="main-login\_form">

<form action="">

<label htmlFor="auth\_login">Имя пользователя или E-mail</label>

<input type="text" id="auth\_login" name="auth\_login" required />

<label htmlFor="auth\_password">Пароль</label>

<input type="password" id="auth\_password" name="auth\_password" required />

<input type="submit" value="Войти" />

</form>

<div className="main-form\_links">

<p>Нет аккаунта? <a href="/register">Регистрация</a></p>

</div>

</div>

</div>

</div>

);

}

Листинг 3.10 – код файла страницы авторизации

import '../components/main/Main.css';

import './styles/page.css';

export default function Register() {

document.querySelector('title').innerText = "Puzrain - Регистрация";

return (

<div className="content">

<div className="main-login">

<h1>Регистрация</h1>

<div className="main-login\_form">

<form action="http://localhost:3001/register" method="POST">

<label htmlFor="reg\_username">Имя пользователя</label>

<input type="text" id="reg\_username" name="reg\_username" required />

<label htmlFor="reg\_email">E-mail</label>

<input type="email" id="reg\_email" name="reg\_email" required />

<label htmlFor="reg\_password">Пароль</label>

<input type="password" id="reg\_password" name="reg\_password" required />

<input type="hidden" value={window.location.origin} name="host" />

<input type="submit" value="Создать" />

</form>

<div className="main-form\_links">

<p>Уже есть аккаунт? <a href="/login">Авторизация</a></p>

</div>

</div>

</div>

</div>

);

}

Листинг 3.11 – код файла страницы регистрации

import { useState, useEffect} from 'react';

import Comment from '../components/main/Comment/Comment';

import '../components/main/Main.css';

import './styles/page.css';

export default function Reviews() {

const [reviews, setReviews] = useState();

let result = [];

useEffect(() => {

fetch('http://localhost:3001/reviews')

.then(res => res.json())

.then(data => setReviews(data))

.catch(error => window.location.pathname = "/");

}, []);

document.querySelector('title').innerText = "Puzrain - Отзывы";

if (reviews) {

result = reviews.map(review => {

return (

<Comment key={review.\_id} comment={review} />

);

});

}

if (reviews)

return (

<div className="content">

<h2 style={{fontWeight: 500, fontSize: 30, marginBottom: 20}} className="reviews-header">Отзывы о наших товарах</h2>

<div className="main-reviews">

{result.length > 0 ? result : <h2 style={{fontSize: 26, fontWeight: 500}}>Нет отзывов</h2>}

</div>

</div>

);

}

Листинг 3.12 – код файла страницы отзывов

import '../components/main/Main.css';

import './styles/page.css';

export default function ErrorPage() {

document.querySelector('title').innerText = "Puzrain - Ошибка";

return (

<div className="content">

<div className="main-error">

<h1 style={{textAlign: 'center', fontSize: 100}}>404</h1>

<h2 style={{textAlign: 'center', fontSize: 60}}>Страница не найдена</h2>

<p style={{textAlign: 'center', marginTop: 20, fontSize: 20}}>Если вы считаете, что попали на эту страницу по ошибке, то обратитесь к нам по одному из <a href="/contacts">контактов</a></p>

</div>

</div>

);

}

Листинг 3.13 – код файла страницы ошибки адресации

import { configureStore } from "@reduxjs/toolkit";

import basketReducer from './basketSlice';

import favReducer from "./favSlice";

const basketMiddleware = (store) => (next) => (action) => {

const result = next(action);

if (action.type?.startsWith('basket/')) {

const basketState = store.getState().basket;

localStorage.setItem('basket', JSON.stringify(basketState));

}

return result;

};

const favMiddleware = (store) => (next) => (action) => {

const result = next(action);

if (action.type?.startsWith('fav/')) {

const favState = store.getState().fav;

localStorage.setItem('fav', JSON.stringify(favState));

}

return result;

};

export default configureStore({

reducer: {

basket: basketReducer,

fav: favReducer

},

middleware: (getDefaultMiddleware) => getDefaultMiddleware().concat(basketMiddleware).concat(favMiddleware)

});

Листинг 3.14 – код файла хранилища Redux

## 3.2. Серверная часть приложения

Серверная часть приложения построена с использованием Node.js, Express.js, MongoDB и Mongoose. База дынных MongoDB хранит информацию о пользователях, товарах, отзывах и категориях товаров. Каждый такая категория представляет собой коллекцию json-документов с соответствующими полями данных. Node.js в связке с Express.js строит get и post запросы для клиентской части, с помощью которых выбираются данные из базы данных. Со всеми этими данными помогает пакет mongoose. С помощью этого пакета требуется создать схемы данных по коллекциям и простыми методами получать данные из базы данных.

const express = require("express"),

cors = require("cors"),

morgan = require("morgan"),

bodyParser = require("body-parser");

const app = express();

const port = 3001;

const Category = require('./models/Category');

const Product = require('./models/Product');

const Review = require('./models/Review');

const User = require('./models/User');

app.use(cors());

app.use(morgan('dev'));

app.use(express.json());

app.use(bodyParser.urlencoded({ extended: true }));

app.get("/categories", async function (req, res) {

let categories;

try {

categories = await Category.find({});

} catch(error) {

throw error;

}

res.send(categories);

});

app.get("/products", async function (req, res) {

let products;

try {

products = await Product.find({});

} catch(error) {

throw error;

}

res.send(products);

});

app.get("/reviews", async function (req, res) {

let reviews;

try {

reviews = await Review.find({});

} catch(error) {

throw error;

}

res.send(reviews);

});

app.get("/create", function (req, res) {

const category = new Category({

title: req.query.title

});

category.save((error, result) => {

if (error) {

throw error;

}

res.redirect("/categories");

});

});

app.get("/crprod", (req, res) => {

const product = new Product({

title: req.query.title,

price: req.query.price,

company: req.query.company,

country: req.query.country,

quantity: req.query.quantity,

status: req.query.status,

category: req.query.category,

description: req.query.description

});

product.save((error, result) => {

if (error) {

throw error;

}

res.redirect("/products");

});

})

app.get("/", function (req, res) {

res.send("<h1>Main page</h1>");

});

app.post("/create\_review", async function (req, res) {

const href = req.body.product\_href;

const review = new Review({

name: req.body.review\_name,

city: req.body.review\_city,

comment: req.body.review\_comment,

productImg: req.body.review\_productImg

});

review.save((error, result) => {

if (error) {

throw error;

}

res.redirect(href);

});

});

app.post("/register", async function (req, res) {

const host = req.body.host;

const user = new User({

username: req.body.reg\_username,

email: req.body.reg\_email,

password: req.body.reg\_password

});

user.save((error, result) => {

if (error) {

res.redirect(host + "/login");

}

res.redirect(host + "/login");

});

});

app.get("/product", async function (req, res) {

let product;

try {

product = await Product.findOne({ \_id: req.query.id });

} catch(error) {

console.log(error);

}

res.send(product);

});

app.listen(port, () => {

console.log(`Server is running on port: ${port}`);

});

Листинг 3.15 – код главного файла сервера

const db = require('mongoose');

db.set('strictQuery', false);

db.connect('mongodb://localhost:27017/puzrain', {

useNewUrlParser: true,

useUnifiedTopology: true

});

db.connection.on("error", console.error.bind(console, "MongoDB connection error:"));

module.exports = db;

Листинг 3.16 – код файла подключения к базе данных

const db = require('./../db/db');

const productSchema = new db.Schema({

title: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

price: {

type: Number,

required: true

},

company: {

type: String,

required: true,

maxlength: 255,

minlength: 2,

trim: true

},

country: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

quantity: {

type: Number,

required: true

},

status: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

category: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

description: {

type: String,

required: true,

maxlength: 2000,

minlength: 3,

trim: true

},

created: {

type: Date,

default: Date.now()

}

}, { collection: 'products' });

module.exports = db.model('Product', productSchema);

Листинг 3.17 – код схемы коллекции товаров

const db = require('./../db/db');

const reviewSchema = new db.Schema({

name: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

city: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

comment: {

type: String,

required: true,

maxlength: 100,

minlength: 3,

trim: true

},

productImg: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

created: {

type: Date,

default: Date.now()

}

}, { collection: 'reviews' });

module.exports = db.model('Review', reviewSchema);

Листинг 3.18 – код схемы коллекции отзывов

const bcrypt = require('bcryptjs');

const SALT\_WORK\_FACTOR = 10;

const db = require('./../db/db');

const userSchema = new db.Schema({

username: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

email: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

password: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

created: {

type: Date,

default: Date.now()

}

}, { collection: 'users' });

userSchema.pre('save', function(next) {

let user = this;

bcrypt.genSalt(SALT\_WORK\_FACTOR, function(err, salt) {

if (err) return next(err);

bcrypt.hash(user.password, salt, function(err, hash) {

if (err) return next(err);

user.password = hash;

next();

});

});

});

module.exports = db.model('User', userSchema);

Листинг 3.19 – код схемы коллекции пользователей

const db = require('./../db/db');

const categorySchema = new db.Schema({

title: {

type: String,

required: true,

maxlength: 255,

minlength: 3,

trim: true

},

created: {

type: Date,

default: Date.now()

}

}, { collection: 'categories' });

module.exports = db.model('Category', categorySchema);

Листинг 3.20 – код схемы коллекции категорий

{

"name": "server",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node server.js",

"dev": "nodemon server.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"bcryptjs": "^2.4.3",

"body-parser": "^1.20.1",

"cors": "^2.8.5",

"express": "^4.18.2",

"mongodb": "^4.12.1",

"mongoose": "^6.8.0",

"morgan": "^1.10.0",

"remove": "^0.1.5"

},

"devDependencies": {

"nodemon": "^2.0.20"

}

}

Листинг 3.21 – код файла package.json

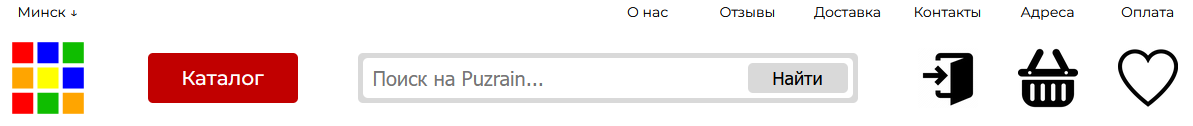
## 3.3. Выводы

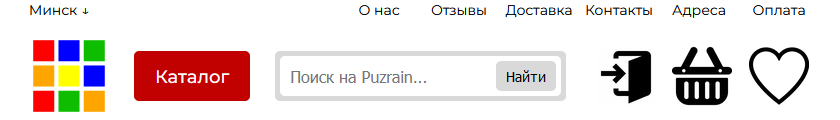
При реализации различных частей приложения возникали трудности с получением данных из базы с коллекциями, однако это всё быстро исправлялось и в итоге имеется рабочее приложение с действительно неплохой и отзывчивой клиентской частью, и с сервером, который построен на самых современных технологиях.

# 4. Тестирование веб-приложения

## 4.1. Адаптивный дизайн веб-сайта

Так, как это веб-приложение, то следует учесть, что такого рода продукты не строятся для девайсов с маленьким разрешением дисплея. Однако для удобства использования был реализован адаптив по планшетные устройства.

 Как пример можно привести адаптив шапки приложения. Вот шапка при разрешении 1600x900:

Рисунок 4.1 – шапка приложения при разрешении 1600x900

Вот дизайн шапки при разрешении 1000x900:

Рисунок 4.2 – шапки приложения при разрешении 1000x900

## 4.2. Крооссбраузерность веб-приложения

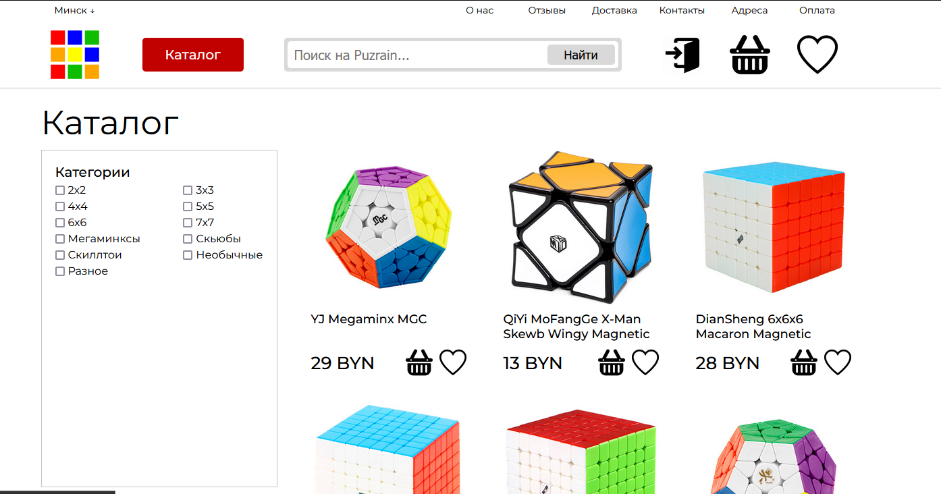
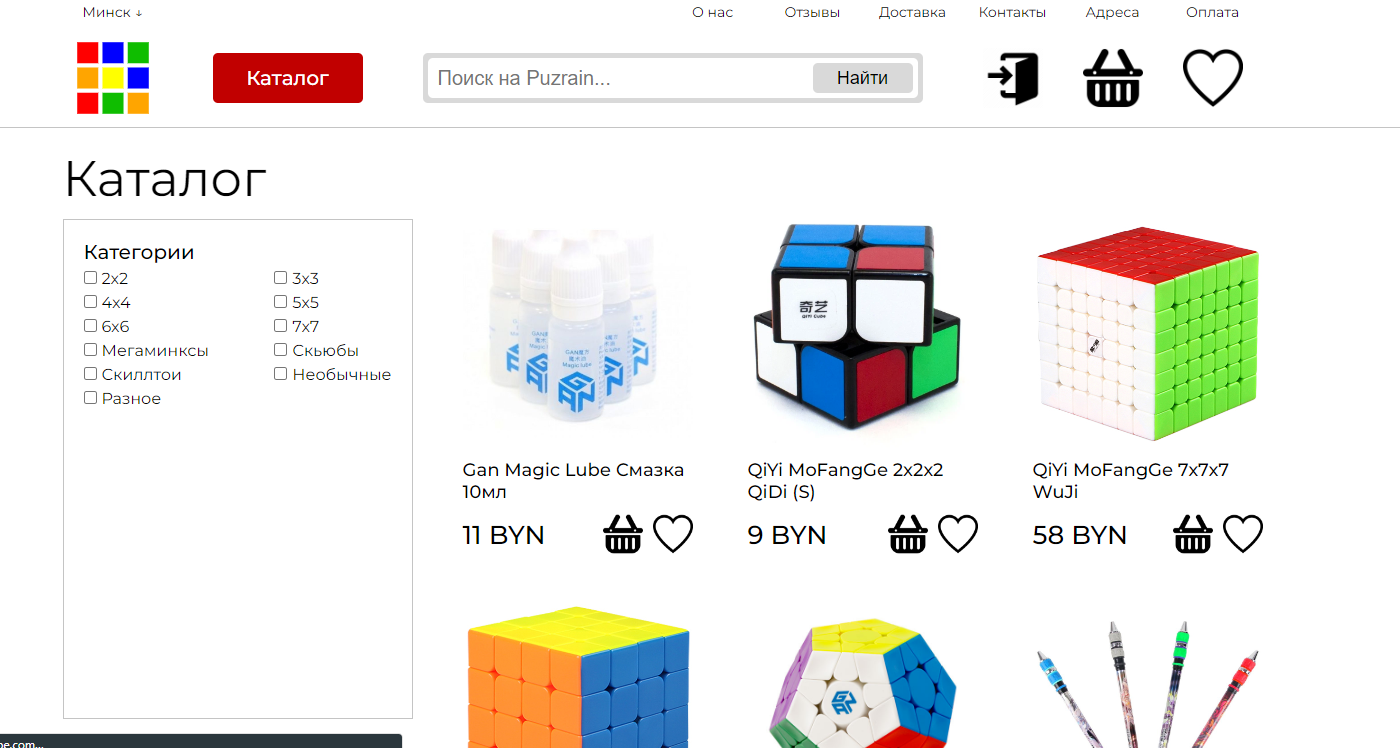
 Веб-приложение работает отлично во всех современных браузерах, без сбоя стилей и с поддержкой всех современных методов вёрстки страницы самого приложения.

Рисунок 4.3 – Страница каталога в браузере Firefox

Рисунок 4.4 – страница каталога в браузере Chrome

## 4.3. Запрос данных из БД

Все нужные данные для различных страниц получается и базы данных благодаря fetch-запросам. Сами эти запросы бывают либо get (запрос при помощи адресной строки) и post (запрос при помощи формы данных).

const [products, setProducts] = useState([]);

useEffect(() => {

fetch('http://localhost:3001/products')

.then(res => res.json())

.then(data => setProducts(data));

}, []);

Листинг 4.1 – код запроса товаров из базы данных

<form action="http://localhost:3001/create\_review" method="POST">

<label htmlFor="review\_name">Имя</label>

<input type="text" name="review\_name" id="review\_name" required />

<label htmlFor="review\_city">Город</label>

<input type="text" name="review\_city" id="riview\_city" required />

<label htmlFor="review\_comment">Ваш комментарий</label>

<textarea name="review\_comment" id="riview\_comment" rows={5} maxLength={100} title="Максимальная длина - 100 символов" required></textarea>

<input type="hidden" value={props.img} name="review\_productImg" />

<input type="hidden" value={"http://localhost:3000/reviews"} name="product\_href" />

<input type="submit" value="Оставить отзыв" />

</form>

Листинг 4.2 – код добавления отзыва о товаре

## 4.4. Вывод

В ходе тестирования веб-приложения в разных браузерах с разными значениями расширения экрана я пришёл к выводу, что приложение отображается схоже в современных браузерах, таких, как: Chrome, Firefox, Microsoft Edge, Opera, Safari, Brave и подобные. Из этого следует, что код для адаптивности и кроссбраузерности был написан правильно.

# Заключение

В ходе создания веб-приложения я следовал системе MERN (название представляет собой аббревиатуру из следующих технологий: MongoDB, Express.js, ReactJS и Node.js).

Вся клиентская часть была построена с помощью классического React-приложения, HTML, CSS и Redux. Все компоненты написаны по правилам современной вёрстки. С помощью Redux я реализовал глобальное состояние для некоторых компонентов. Благодаря ему мне не надо было создавать и мучаться с состояниями компонентов, а просто обращаться к самому глобальному состоянию.

Серверная часть была написана довольно быстро. Понимание того, как работают технологии Node.js и Express.js вместе приходит быстро. С помощью пакета mongoose были реализованы схемы данных в бд, а уже благодаря этим схемам мы производим выборку и отправление данных на клиентскую часть.

Подытожив можно сказать, что я успешно окончил курс «Технологии frontend-программирования», и даже изучил больше.

# Список использованных источников

1. SuperToys – интернет-магазин головоломок [Электронный ресурс] / Сайт superotys.by. ­– 2022. – Режим доступа: https://supertoys.by. – Дата доступа: 20.10.2022.
2. СССSTORE – русский интернет-магазин головоломок [Электронный ресурс] / Сайт cccstore.ru. ­– 2022. – Режим доступа: https://cccstore.ru. – Дата доступа: 20.10.2022.
3. Flexbox – изучение веб-вёрстки [Электронный ресурс] / Сайт developer.mozilla.org. ­– 2022. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/CSS/CSS\_layout/Flexbox. – Дата доступа: 20.10.2022.
4. Как работает Flexbox наглядно [Электронный ресурс] / Сайт tproger.ru. ­– 2022. – Режим доступа: https://tproger.ru/translations/how-css-flexbox-works. – Дата доступа: 20.10.2022.
5. CSS Flexbox. Полный курс [Электронный ресурс] / Сайт youtube.com. ­– 2022. – Режим доступа: https://www.youtube.com/watch?v=XXlw7TUxRVY. – Дата доступа: 20.10.2022.
6. Основный понятия Grid Layout [Электронный ресурс] / Сайт developer.mozilla.org. ­– 2022. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/CSS/CSS\_Grid\_Layout/Basic\_Concepts\_of\_Grid\_Layout. – Дата доступа: 20.10.2022.
7. 7 основных понятий CSS Grid Layout с примерами, которые помогут начать работу с гридами [Электронный ресурс] / Сайт tproger.ru. ­– 2022. – Режим доступа: https://tproger.ru/articles/7-osnovnyh-ponjatij-css-grid-layout-s-primerami-kotorye-pomogut-nachat-rabotu-s-gridami. – Дата доступа: 20.10.2022.
8. CSS Grid. Полный курс [Электронный ресурс] / Сайт youtube.com. ­– 2022. – Режим доступа: https://www.youtube.com/watch?v=yTLqfUxwdDk. – Дата доступа: 20.10.2022.
9. @media запросы CSS [Электронный ресурс] / Сайт developer.mozilla.org. ­– 2022. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/CSS/@media. – Дата доступа: 20.04.2022.
10. JSON [Электронный ресурс] / Сайт developer.mozilla.org. ­– 2022. – Режим доступа:https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\_Objects/JSON. – Дата доступа: 20.10.2022.
11. Работа с JSON [Электронный ресурс] / Сайт developer.mozilla.org. ­– 2022. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON. – Дата доступа: 20.03.2022.
12. React [Электронный ресурс] / Сайт reactjs.org. ­– 2022. – Режим доступа: https://reactjs.org/docs/getting-started.html. – Дата доступа: 20.10.2022.
13. Redux [Электронный ресурс] / Сайт youtu.be/C0fBnil\_Im4. ­– 2022. – Режим доступа: https://youtu.be/C0fBnil\_Im4. – Дата доступа: 20.10.2022.
14. MongoDB [Электронный ресурс] / Сайт mongodb.com. ­– 2022. – Режим доступа: https://www.mongodb.com/home. – Дата доступа: 20.10.2022.
15. Mongoose [Электронный ресурс] / Сайт mongoosejs.com/. ­– 2022. – Режим доступа: https://mongoosejs.com/. – Дата доступа: 20.10.2022.