

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

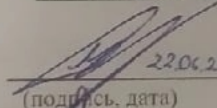
Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»
Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

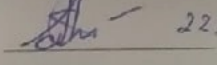
на тему

ИНТЕРНЕТ-МАГАЗИН С ИСПОЛЬЗОВАНИЕМ REACT
ТЕХНОЛОГИЙ

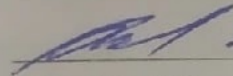
Студент

 22.06.22 Петров Михаил Викторович
(подпись, дата) (ФИО полностью)

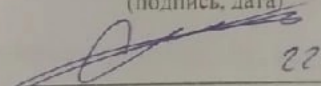
Руководитель

 22.06.22 Бикташев Р.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

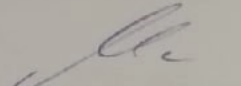
 23.06.22 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

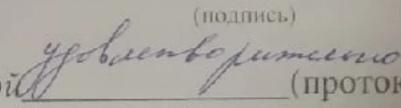
Рецензент

 22.06.22 С.В. Печерский
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.22 г. № 13)

Заведующий кафедрой

 М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от

24 июня 2022 г. № 11)

Секретарь ГЭК

Е.И. Гурин

Пенза 2022

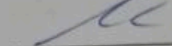
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой

 М.А. Митрохин

«26» апреля 2022 г.

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

1. Петров Михаил Викторович гр.17ВО2 факультета вычислительной техники.
2. Тема работы «Интернет-магазин с использованием react технологий»
Тема утверждена приказом ПГУ №0357/о от "26" апреля 2022 г.
3. Руководитель работы к.т.н., доцент ВТ ПГУ Бикташев Р.А.

1. Задание на работу (назначение разработки, исходные данные и т.п.) Интернет-магазин с использованием react технологий предназначен для быстрого и точного взаимодействия клиента с пользовательским интерфейсом приложения. Система должна иметь клиент-серверную архитектуру. Клиентская часть отвечает за отображение пользовательского интерфейса приложения.

2. Перечень подлежащих разработке вопросов

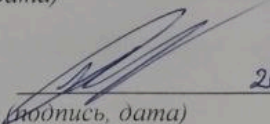
Требуется произвести анализ предметной области, сформировать техническое задание, выбрать инструменты для разработки клиентской части пользовательского интерфейса, выполнить проектирование системы, разработать клиентскую часть системы. Клиентская часть системы должна выполнять функции: обеспечивать добавление, удаление товара в потребительскую корзину; отображение товара на главной странице магазина; отображение истории покупок; добавление товара в избранное. Также выполнить задание по БЖД и осуществить экономическое обоснование разработки системы.

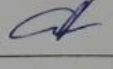
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	8%	26.04.2022 – 29.04.2022	
2.	Составление технического задания	6%	30.04.2022 – 02.05.2022	
3.	Выбор инструментов разработки	8%	03.05.2022 – 06.05.2022	
4.	Проектирование системы	12%	07.05.2022 – 12.05.2022	
5.	Разработка серверной части системы	40%	13.05.2022 – 01.06.2022	
6.	Выполнение экономического обоснования	10%	02.06.2022 – 06.06.2022	
7.	Выполнение задания по БЖД	8%	07.06.2022 – 10.06.2022	
8.	Оформление пояснительной записки	8%	11.06.2022 – 14.06.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель работы  26.04.22
(подпись, дата)

Задание к исполнению принял студент  26.04.22
(подпись, дата)

Работу к защите допустить
Декан факультета  Л.Р. Рогова
(подпись, дата)
23.06.2022

Реферат

Пояснительная записка содержит 107 листов, 36 рисунков, 4 таблиц, 6 источников, 3 приложения.

ФРОНТЕНД, КЛИЕНТ, API, IDE, REACT.JS, JAVASCRIPT, WINDOWS, САЙТ, ИНТЕРНЕТ.

Цель работы – разработать интернет-магазин на основе библиотеки *react*.

Разработка велась на основе библиотеки *react* в среде разработки *Visual Studio Code*.

В процессе разработки был проведен анализ требований к разрабатываемому продукту, анализ существующих современных средств проектирования программных продуктов.

В результате проделанной работы был разработан Интернет-магазин с использованием *react* технологий.

В экономической части дается технико-экономическое обоснование целесообразности разработки системы.

В разделе безопасности жизнедеятельности описаны психофизиологические факторы, воздействующие на оператора ЭВМ.

					ПГУ 1.090501.10.001 ПЗ			
Изм.	ЛистЛ	№ докум.№	Подпись	Дата				
Разраб.		Петров М.В.		22.06.22	Интернет-магазин с использованием <i>react</i> технологий	Лит.	Лист	Листов
Провер.		Бикташев Р.А.		19.06.22			2	107
Н. Контр.		Бычков А.С.		23.06.22				
Реценз.		Потеряев С.		22.06.22	Пояснительная записка	ФВТ, каф. ВТ гр. 17В02		
Утверд.		Митрохин М.А.						

Содержание

Перечень сокращений и обозначений.....	7
Введение.....	8
1 Интернет-магазин с использованием react технологий.....	9
1.1 Обзор предметной области.....	9
1.2 Обзор интернет-магазина	10
1.2.1 Определение интернет-магазина	10
1.2.2 Классификация	11
1.2.3 Анализ популярности	12
1.3 Постановка задачи.....	13
1.4 Проектирование системы	14
1.4.1 Выбор инструмента для разработки приложения.....	14
1.4.2 Выбор программного продукта для разработки	17
1.5 Разработка приложения для интернет-магазина.....	20
1.6 Тестирование и отладка.....	37
2 Техничко-экономическое обоснование разработки системы	49
2.1 Введение.....	49
2.2 Расчёт затрат на разработку программного обеспечения	51
2.3 Расчёт цены разработанной программы	52
2.4 Расчёт капитальных вложений	53
2.5 Расчёт эксплуатационных расходов.....	53
2.6 Определение расходов на интернет трафик	56
2.7 Итоги подсчётов	56
3 Безопасность жизнедеятельности. Эргономика рабочего места.....	58
3.1 Характеристика условий труда программиста.....	58
3.2 Требования к параметрам воздушной среды.	60
3.3 Требования к уровню шума и вибрации.....	60
3.4 Требования к монтажу производственного оборудования.....	60
3.5 Требования к освещению рабочих мест.	61

3.6 Расчет освещения рабочего места программиста.	61
Заключение.	62
Список использованных источников	63
Приложение А – <i>UML</i> -диаграммы	64
Приложение Б - Листинг программы.....	70
Приложение В - Презентация	100

Перечень сокращений и обозначений

API (Application programming interface) – интерфейс приложения для программирования

DOM (Document Object Model) - это независимый от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому *HTML*

IDE (Integrated Development Environment) – интегрированная, единая среда разработки

JSX (JavaScript XML) - это расширение синтаксиса *JavaScript*, которое позволяет использовать *HTML*-подобный синтаксис для описания структуры интерфейса

SDK (Software development kit) – набор средств разработки

ПК – персональный компьютер

ОС – операционная система

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

Введение

В наши дни Интернет является причиной возникновения нового вида экономики, что способствует расширению рыночных отношений между продавцом и покупателем. Рынок интернет-торговли представляет собой совокупность механизмов, позволяющих осуществлять процесс купли-продажи товаров и услуг посредством Интернета.

Современные рыночные отношения требуют от компаний быстро и эффективно создавать инструменты продвижения своего продукта для привлечения клиентов, чтобы воспользоваться именно их продуктом или услугой. Для решения данной проблемы необходимо воспользоваться современными инструментами для разработки интернет-магазина. Данная работа предлагает использование *react* технологий с целью быстрой и эффективной разработки конечного продукта, что способствует экономии времени и ресурсов для заказчиков.

Разработка данного программного обеспечения представляет собой совокупность использования таких инструментов *react* и *Figma*. Поэтому необходимо проанализировать существующие инструменты данного типа.

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

1 Интернет-магазин с использованием react технологий

1.1 Обзор предметной области

Одним из важнейших разделов интернет торговли является разработка приложений. На сегодняшний день для разработки приложения необходимо провести комплекс мероприятий, начиная от анализа рынка и конкурентов и заканчивая выбором инструментов разработки и языка программирования.

На данный момент существуют три окончательно сформированных типах интернет торговли.

Web-витрина – оформленный дизайнерами прайс-лист торговой компании, который способен обрабатывать заказы. Такой тип интернет торговли подходит для малого и среднего бизнеса. Основными недостатками данной системы является отсутствие интерактивных интерфейсов управления и не связаны напрямую с бизнес-приложениями компании [1].

Интернет-магазин – полноценная система продажи с индивидуальными бизнес-схемами взаимодействия с потребителем, которая включает в себя бизнес-логику для взаимодействия с процессами Интернет-торговли. Подходит для средних и крупных бизнесов.

Торговая Интернет-система – полностью интегрированы online и offline процессы компании. Делается это с помощью интеграции внешнего Интернет-магазина в бизнес-приложения, имеющиеся в компании, с системами складского учета.

Web-витрина обладает неполным функционалом для ведения полноценной Интернет-торговли, так же взаимодействие с ней происходит посредством передачи файлов.

Для торговой интернет-системы необходима инфраструктура бизнеса, в которую нужно интегрировать данный продукт. Так же данный тип интернет-торговли имеет высокую стоимость разработки и длительная разработка проекта.

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

Таким образом, выстраивается проблема дороговизны создания определенных типов интернет-торговых систем и ограниченный спектр инструментов у других. Решением в данной работе предлагается создание интернет-магазина на основе *react* технологий, так как это является самым оптимальным вариантом в плане скорости разработки, ее стоимости, а так же обширным функционалом. Но сначала следует рассмотреть саму технологию создания интернет-магазина.

1.2 Обзор интернет-магазина

1.2.1 Определение интернет-магазина

Интернет-магазин — сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или через мобильное приложение, сформировать заказ на покупку, выбрать способ оплаты и доставки заказа, оплатить заказ. Типичный интернет-магазин позволяет клиенту просматривать ассортимент продуктов и услуг фирмы, просматривать фотографии или изображения продуктов, а также информацию о технических характеристиках продукта и ценах [2].

Особенности интернет-магазина разнообразны, но выделяются основные критерии, с помощью которых происходит его создание:

- выбор ниши – правильный выбор ниши способствует успешному развитию бизнеса, в противном случае, если ниша перенасыщена, то стоит задуматься над тем, чем можно заинтересовать клиента и как выделиться среди конкурентов;
- правильная *CMS* сайта – это программа для администрирования и управления процессами Интернет-магазина. От нее зависит скорость отображения страниц и правильное функционирование всех компонентов сайта;
- веб-дизайн – правильное расположение смысловых блоков, правильная эргономика и оформление сайта способствует удерживанию

клиента на данном хостинге;

- продвижение в сети – раскрутка интернет-магазина одна из ключевых особенностей успешного развития бизнеса, грамотный интернет-маркетинг способен многократно увеличить продажи товаров и услуг;

- юзабилити сайта – пользователю должно быть понятно как оформить заказ, где найти интересующий его товар, произвести оплату и какие сроки доставки заказа;

- карточки товара – страница, на которой должна выводиться подробная информация о товаре, чтобы у покупателя не возникало лишних вопросов;

- сопутствующие товары – один из способов увеличения продаж. К выбранному товару или услуге можно дополнительно предложить максимально подходящий к ним товар или услугу. Способствует увеличению среднего чека;

- интернет-магазин способствует быстрому и успешному развитию бизнеса. Но для создания такого продукта необходимо проанализировать нишу и выбрать наиболее подходящие инструменты для разработки данного продукта.

1.2.2 Классификация

Интернет-магазины подразделяются на несколько типов, различных по техническим особенностям и различных по видам продаж. К таким интернет-магазинам можно отнести:

- интернет-магазины, которые подразделяются по типам ведения бизнеса:

1) онлайн - магазины;

2) торговые точки классический тип ведения бизнеса, в котором есть наличие складов, сотрудников, ведение бухгалтерского учета;

3) аутсорсинговые конторы. Компании посредники, которые не занимаются хранением и поставкой товаров, а делегируют эти операции в ка-

честве аутсорса сторонним компаниям.

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

4) система дропшипига - магазины, в которых отсутствует товар, при этом организовывается продажа поставщика;

- интернет-магазины подразделяются по объемам продаж:

1) розничные интернет-магазины. Продают потребителям определенное количество продукции;

2) оптовые интернет-магазины занимаются продажами товаров в больших объемах, стоимость которых зависит от количества покупаемого товара;

- интернет-магазины подразделяются по видам продаж:

1) *B2B* - товар продается в другие коммерческие структуры;

2) *B2C* - товар продается конечным потребителям;

- интернет-магазины подразделяются в зависимости от способа их заработка:

1) товары продаются онлайн через сайты производителей или их представителей. В таком случае целью сайта является продажа или реклама товара производителя;

2) партнерская система. Фирма, которая создала сайт, рекламирует и продает товар производителя. Производитель платит проценты с продаж таким фирмам;

3) магазины по продаже информации. Сайты, целью которых является продажа информации или определенного контента, которые они размещают на своей платформе. К примеру таких сайтов можно отнести *OnlyFans* и *boosty* [3].

1.2.3 Анализ популярности

Интернет-магазин используется во многих сферах жизни человека, таких как:

- развлечения: интернет-магазин очень необходим в данной сфере. Без него было бы невозможно приобретать игровые скины, онлайн-игры,

дополнения к играм, игровые атрибуты, мерч и так далее;

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		10

- образование: современные интернет-магазины позволяют приобретать онлайн курсы для повышения квалификации, можно покупать разного рода литературу, что повышает уровень образования;

- наука: интернет-магазин позволит ученым быстро и эффективно находить редкое оборудование, которое в реальном мире было бы найти намного сложнее, чем на сайте;

- медицина: интернет-магазин позволит людям с ограниченной подвижностью не выходя из дома приобретать необходимый им товар. Так же поможет людям, которые не могут обходиться без жизненно необходимых препаратов;

- архитектура: существуют интернет-магазины с онлайн конструкторами домов, что позволяет наиболее точно определить то, какой именно дом захочет собрать клиент;

- армия: интернет-магазины давно нашли свое место для использования в вооруженных силах различных стран. Программисты создают различные сайты, дающие возможность приобретать необходимую технику и вооружение с быстрой скоростью.

На основе этих выводов можно считать более чем возможным использование интернет-магазина, так как продажа товара через сайт значительно упрощает жизнь клиентам. Наиболее подходящим для разработки такого рода приложения является *IDE*, так как она позволяет быстро и эффективно заниматься разработкой данного проекта.

1.3 Постановка задачи

Необходимо разработать программу «Интернет-магазин с использованием *react* технологий».

В данной программе будет реализована часть, отображение товара на главной странице магазина. Возможности программы должны предоставлять

добавление товара в корзину, оформление заказа, отображение истории

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

покупок, добавление товара в избранное, отображение добавленных товаров в избранное, отображение слайдера на главной странице. Для решения этих задач необходимо разработать и реализовать простой, удобный и понятный пользовательский интерфейс.

Программа должна быть многомодульной: необходимо поделить её на логические модули. Это способствует упрощению добавления в программу новых функций при дальнейшей разработке, а также упростит обнаружение и исправление ошибок при её отладке и тестировании.

Программа должна быть разработана для работы в ОС *Microsoft Windows 10*.

1.4 Проектирование системы

1.4.1 Выбор инструмента для разработки приложения

В наши дни самыми доступными и распространенными инструментами для разработки веб приложений являются библиотеки и фреймворки. Они различаются между собой тем, что библиотеки – это набор близкой функциональности. Фреймворки в свою очередь сам решает, в какой момент времени от вызывается. Называется это инверсией управления. На сегодняшний день существует множество библиотек и фреймворков, рассмотрим основные из них.

React - это библиотека *JavaScript*, созданная разработчиками *Facebook* и *Instagram*.

Согласно опросу *Stack Overflow Survey 2017*, *React* был признан самой популярной технологией среди разработчиков. *React* также имеет честь быть самым популярным проектом *JavaScript*, согласно количеству звезд на *GitHub*.

С помощью *React* можно создать интерактивный интерфейс с использованием декларативного подхода, в котором вы можете

контролировать состояние приложения, говоря: «Представление должно выглядеть так». Он использует компонентную модель, в которой компоненты

					ПГУ 1.090501.10.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

являются повторно используемыми элементами пользовательского интерфейса, и каждый компонент имеет свое собственное состояние [4].

jQuery - это библиотека, которая сделала JavaScript более доступным а *DOM*-манипуляцией проще, чем раньше.

Плавная кривая обучения *jQuery* и простой синтаксис породили на стороне клиента новое поколение новых разработчиков. Несколько лет назад *jQuery* считался прочным решением для создания надежных веб-сайтов с поддержкой кросс-браузерности. Основные функции *jQuery*, такие как манипулирование *DOM* на основе селекторов *CSS*, обработка событий и создание вызовов *AJAX*, подпитывали его популярность.

Однако все изменилось, и среда *JavaScript* постоянно развивается. Некоторые функции *jQuery* были включены в новую спецификацию *ECMAScript*. Более того, новые библиотеки и фреймворки, используемые сегодня, имеют собственный способ связывания *DOM*, и поэтому простые методы манипуляции с *DOM* больше не требуются. Популярность *jQuery* находится на спаде, но я не вижу, чтобы эта библиотека исчезла в ближайшее время.

D3 (или *D3.js*) - мощная библиотека *JavaScript* для создания интерактивных визуализаций с использованием веб-стандартов, таких как *SVG*, *HTML* и *CSS*.

В отличие от других библиотек визуализации, *D3* предлагает лучший контроль над окончательным визуальным результатом.

D3 работает, связывая данные с *DOM* и затем преобразуя их в документ. Она также имеет собственную экосистему, которая состоит из плагинов и библиотек, которые расширяют ее базовые функциональные возможности. Библиотека существует с 2011 года, и в ней есть множество документации и учебников, которые помогут вам начать работу.

Vue.js - это легкий *JavaScript* фреймворк, который активно развивается в этом году.

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

Это самый популярный *JavaScript* фреймворк на *GitHub* с точки зрения звезд *GitHub*. *Vue* утверждает, что это менее упрямый фреймворк и, таким образом более удобный в использовании. Синтаксис шаблона на основе *HTML* *Vue* связывает отрисовываемый *DOM* с данными экземпляра.

Фреймворк предлагает опыт, похожий на *React*, с его виртуальными *DOM* и компонентами повторного использования, которые можно использовать для создания как виджетов, так и целых веб-приложений. Кроме того, вы также можете использовать синтаксис *JSX* для непосредственного написания функций рендеринга. Когда состояние изменяется, *Vue.js* использует систему реактивности, чтобы определить, что изменилось и перерисовывает минимальное количество компонентов. *Vue.js* также поддерживает интеграцию других библиотек во фреймворк без особых хлопот.

Ember.js является фронт-энд фреймворком на основе шаблона *Model-View-ViewModel (MVVM)*.

Фреймворк использует стандартный подход к конфигурации, который популярен среди серверных фреймворков таких как *Ruby on Rails* и *Laravel*. *Ember.js* включает в себя общие идиомы и лучшие практики фреймворков, чтобы вы могли получать приложение без особых усилий.

Стек *Ember* обычно включает:

- *ember CLI*: предоставляет основные варианты каркаса приложения и поддерживает сотни надстроек;
- *ember Data*: библиотека сохранения данных, которая может быть настроена для работы с любым сервером;
- *ember Inspector*: расширение доступное для *Chrome* и *Firefox*;
- *liquid Fire*: дополнение для переходов и анимаций;

На основании всех вышеперечисленных плюсов и минусов для исполь-

зования в данной работе была выбрана библиотека *React*, так как рассматривались простота обращения и совместимость с большинством инструментов разработки приложений.

1.4.2 Выбор программного продукта для разработки

Наиболее популярное ПО для разработки *web*-приложений:

- visual Studio Code*;
- webStorm*;
- sublime Text*.

Visual Studio Code — отличный высокопроизводительный легковесный редактор, предлагающий хорошую поддержку завершения проекта. Работает на *Node JS*. Содержит плагины для *VIM* и *Emacs*.

В число его плюсов входит следующее:

- работает на базе открытого исходного кода;
- отлично работает *Mac, Linux, Windows*;
- включает большой функционал. В него входит обозреватель решений, отладчик, область расширений и управление исходным кодом;
- поддержка терминала внутри окна;
- прекрасно подходит для разработки в ядре *.Net*.

У продукта есть и определенные недостатки:

- минималистичный, поэтому сложен для восприятия некоторыми разработчиками;
- обладает небольшим функционалом, поэтому не подходит для реализации больших проектов.

WebStorm - Интегрированная среда разработки на *JavaScript, CSS & HTML* от компании *JetBrains*, разработанная на основе платформы *IntelliJ IDEA*. *WebStorm* обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями.

Он является отличным софтом, у которого основным преимуществом

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

являются его функции поддержки кодирования, интеллектуальный редактор, функции поиска и навигации, поддержка интеграции популярных инструментов командной строки, встроенные шаблоны и широкие возможности настройки.

В число его плюсов входит следующее:

- подойдёт для решения любой задачи;
- поддерживает множество форматов данных;
- в сборке есть готовые шаблоны проектов;
- платформа *IntelliJ* с открытым исходным кодом;
- унифицированный интерфейс;
- инструменты качества кода.

У продукта есть и определенные недостатки:

- сложно разобраться;
- нужно подстраивать систему под требования программы;
- есть баги, которые висят с первой версии.

Sublime Text — это кроссплатформенный текстовый редактор, разработанный для пользователей, которые ищут эффективный, но минималистский инструмент для редактирования кода. Редактор, конечно же, прост, в котором отсутствуют панели инструментов или диалоговые окна.

Sublime Text предлагает множество функций, которые упрощают компиляцию кода.

Goto Anything — это удобная функция, которая позволяет легче получать доступ к файлам. Для перехода к аспектам скомпилированного кода, таким как символы, строки или слова, требуется всего несколько действий.

Сопоставление скобок — это функция, которая позволяет быстро определить неправильное сопоставление. Редактор напрямую выделяет соответствующие наборы скобок.

Множественное выделение — это удобный инструмент, который позволяет быстро менять строки кода на ходу. Изменяйте имена переменных

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

или даже файлов.

Наличие мощного *Python API* в *Sublime* выделяет текстовый редактор среди конкурентов. Мощный встроенный *API* позволяет *Sublime* достигать большей функциональности, позволяя плагинам расширять встроенную функциональность.

В число его плюсов входит следующее:

- *sublime Text* — это легкий текстовый редактор, который подойдет любому программисту. Программа сделана со скоростью, находящейся в ее основе. Особенность программы в ее скорости и отзывчивости пользовательского интерфейса;

- в редакторе доступно множество плагинов, которые интегрируются в одном месте;

- полностью настраиваемый — текстовый редактор создан, чтобы позволить конечному пользователю легко «поиграть» с ПО на свой лад. *Sublime* позволяет настраивать множество функций, включая: привязки клавиш, меню, фрагменты, макросы и многие другие. Кроме того, изменяйте внешний вид, настроив свои темы для ПО;

- кроссплатформенная поддержка — в редакторе доступна на большинстве распространенных настольных клиентов, включая *Windows*, *macOS* и *Linux*;

- *sublime* с открытым исходным кодом, соответственно бесплатный. Но в то же время, ПО также можно купить — по желанию. Важно отметить, что бесплатная версия работает просто отлично;

- с редактором, вы можете комфортно переключаться между различными файлами. К тому же, благодаря функции *Goto Anything*, доступ к которой получаете непосредственно с клавиатуры с помощью клавиш *Ctrl* или *Command + P*;

- простота в использовании. Редактор подходит для любого пользователя, независимо от уровня его опыта.

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

На основании вышеописанных преимуществ и недостатков, а также выбранной библиотеки *React*, в качестве инструмента для разработки была выбрана *IDE Visual Studio Code*, так как по опыту разработок, данная программная платформа лучше всего взаимодействует с библиотекой *React*. А наличие библиотеки открытого и бесплатного контента облегчает наполнение визуальной составляющей приложения. *Visual Studio Code* в совокупности с *React* выделяется на фоне конкурентов по качеству, удобству, скорости разработки и оптимизации, и бесплатен в использовании.

1.5 Разработка приложения для интернет-магазина

Приложение интернет-магазин с использованием react технологий состоит в совокупности из двух частей – визуальной и программной.

Визуальная часть:

- главная страница;
- страница избранных товаров;
- корзина;
- история покупок;
- страница оформления заказов.

Программная часть:

- управление;
- скрипты;
- интерфейс.

Для создания нового проекта на библиотеки *react* необходимо выполнить следующие действия.

Установить необходимую программную платформу *Node.js* через сайт <https://nodejs.org/en/> (рисунок 1).

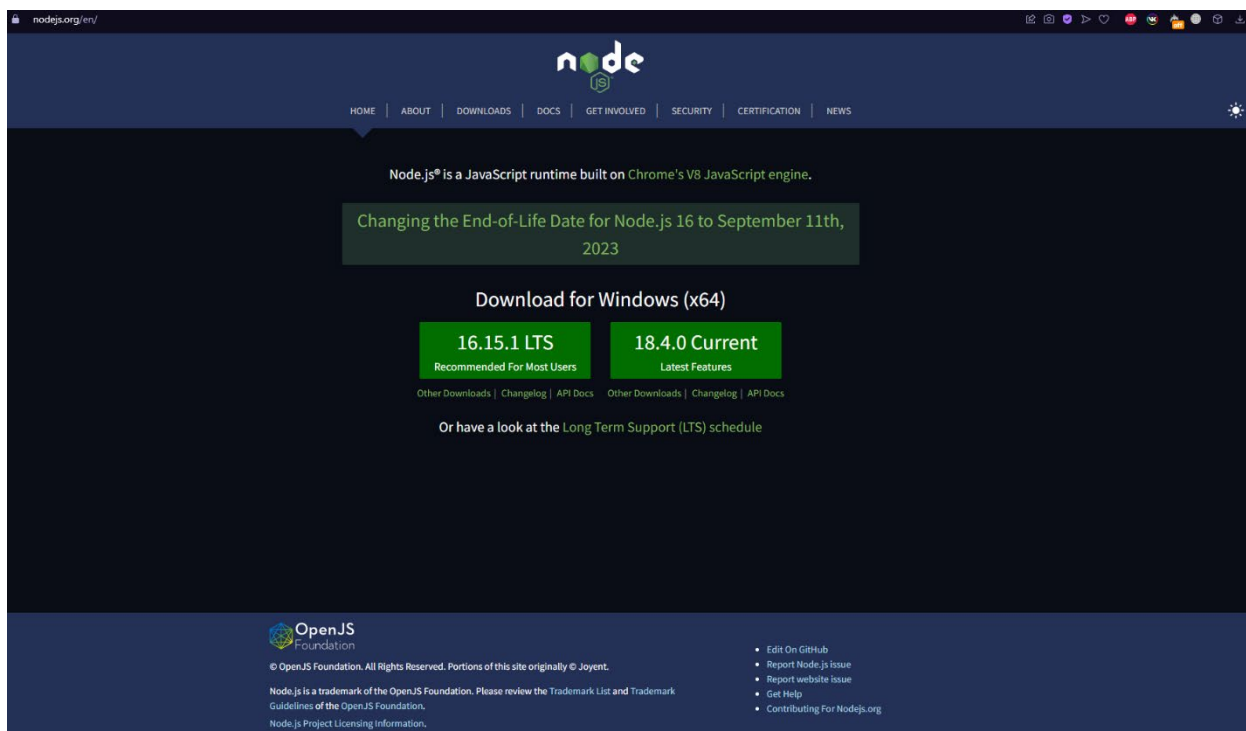


Рисунок 1 – Сайт для установки *node.js*

В открывшемся окне выбрать подходящую версию (рисунок 2).

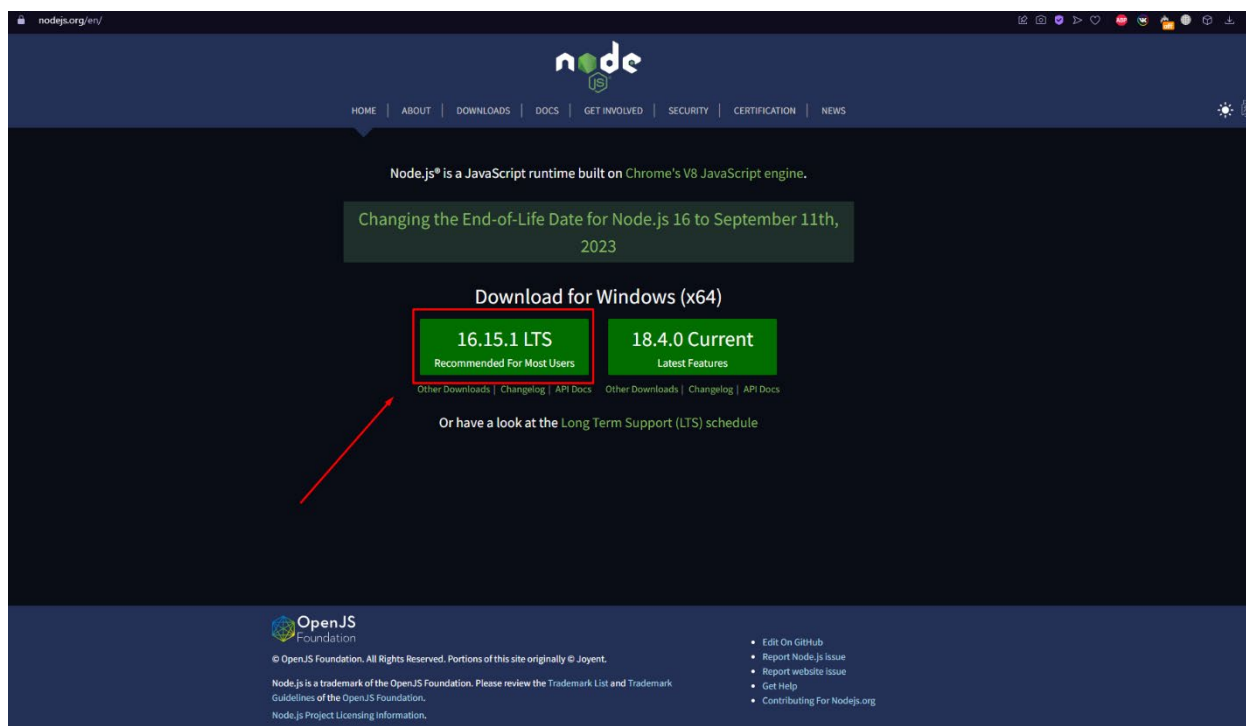


Рисунок 2 – Выбор версии *node.js*

Далее в открывшемся установщике необходимо нажать кнопку next (рисунок 3).

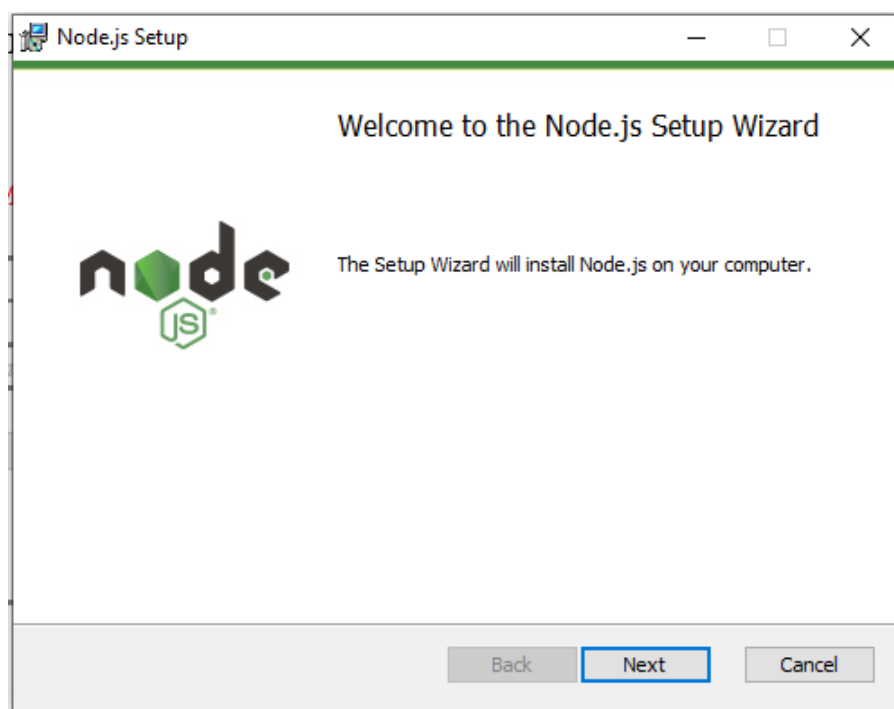


Рисунок 3 – Установка *node.js*

Далее необходимо согласиться с лицензией (рисунок 4).

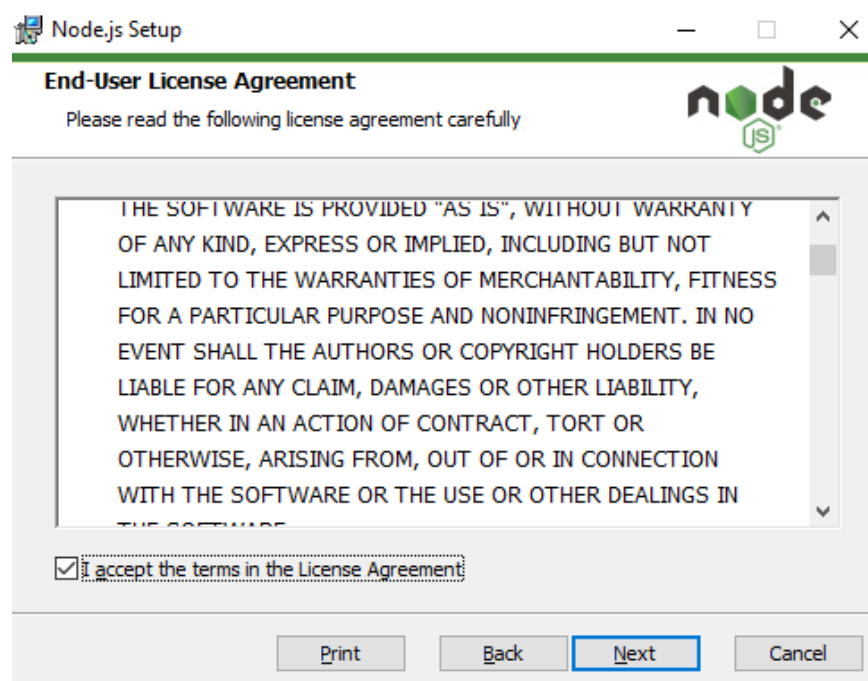


Рисунок 4 – Соглашение с лицензией

После чего необходимо выбрать путь установки (рисунок 5).

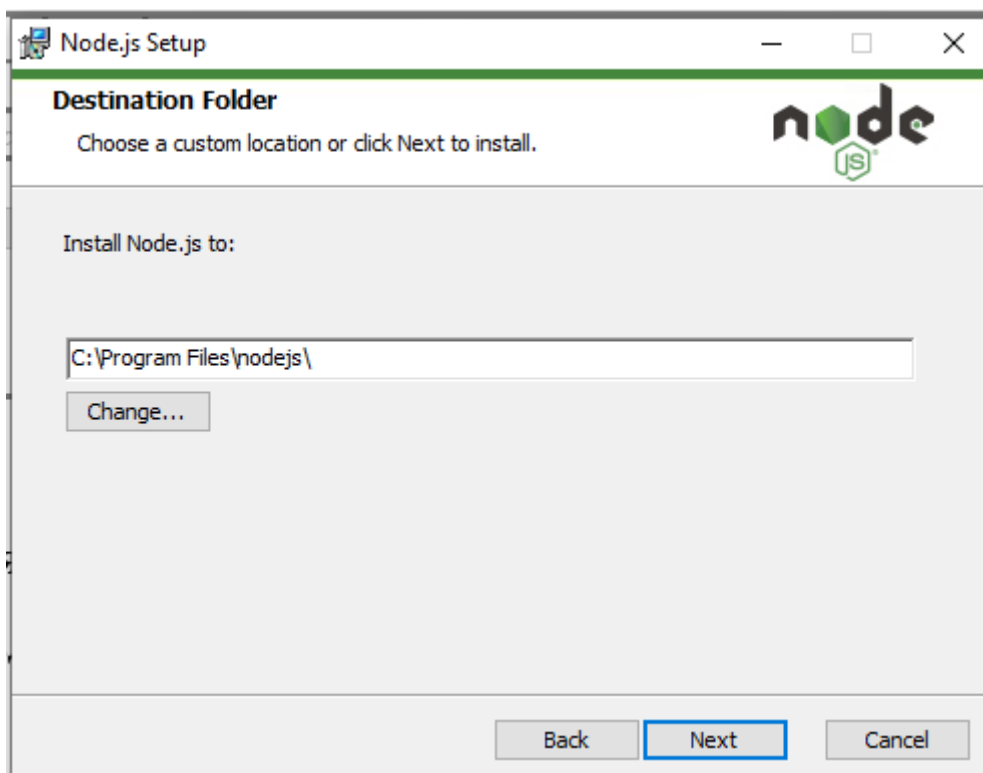


Рисунок 5 – Путь установки

Далее мы выбираем свойства установки по умолчанию (рисунок 6).

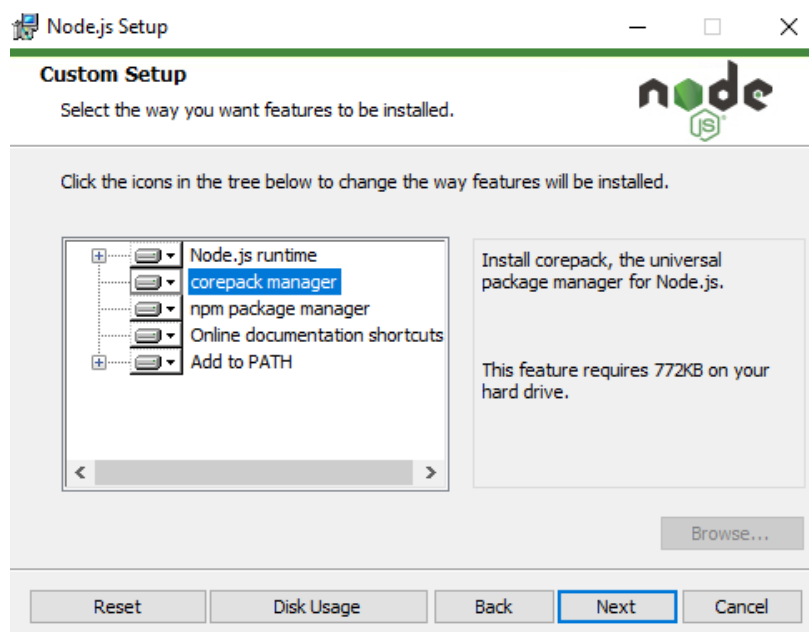


Рисунок 6 – Установка по умолчанию

Далее идет окно с дополнительными инструментами, которые не

требуются в данном проекте (рисунок 7).

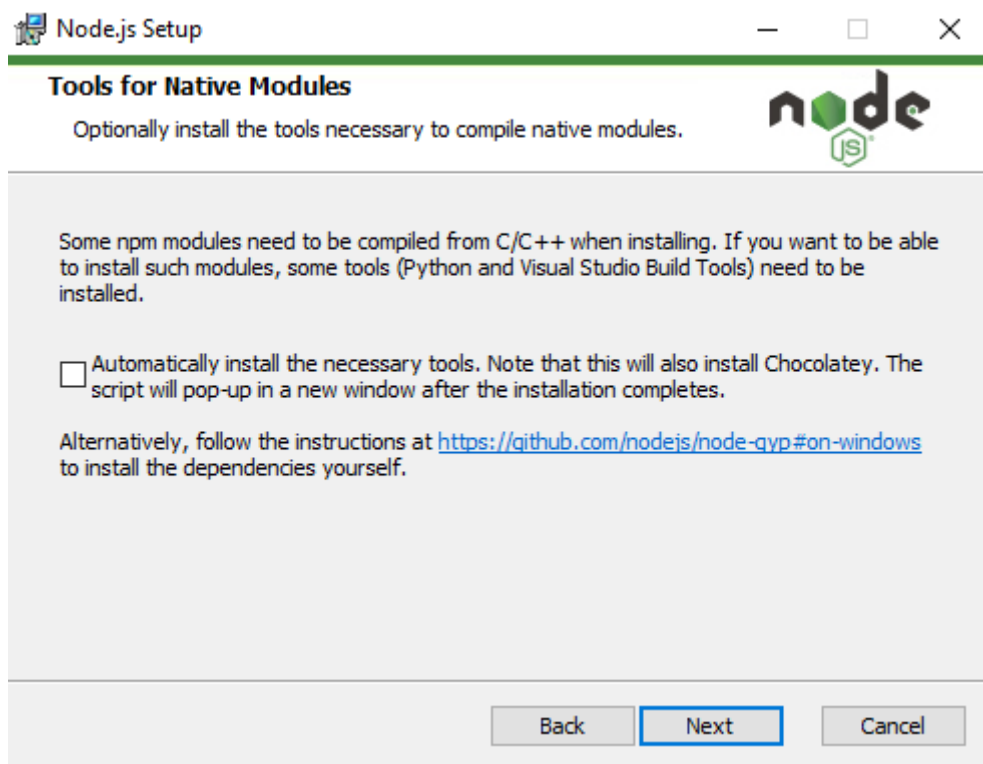


Рисунок 7 – Установка дополнительных инструментов

После этого необходимо нажать на кнопку *Install* для установки ПО (рисунок 8).

После установки *node.js* необходимо проверить успешную установку ПО. Для этого необходимо открыть консоль windows с помощью команды *win+r* и прописать команду *cmd* (рисунок 9).

В консоли необходимо прописать команду *node -v*, После чего отобразится актуальная версия *node.js*, если после написания команды отобразилась версия, значит все работает правильно (рисунок 10);

После успешной установки *node.js* можно приступить к созданию приложения при помощи библиотеки *react*. *Node.js* является средой выполнения *javascript*. С ее помощью *js* код можно выполнять не только в браузере (раньше можно было только в нем). Так же в нее входят встроенные библиотеки, которые была разработаны сообществом программистов. Называются они *npm* (менеджер пакетов *Node*).

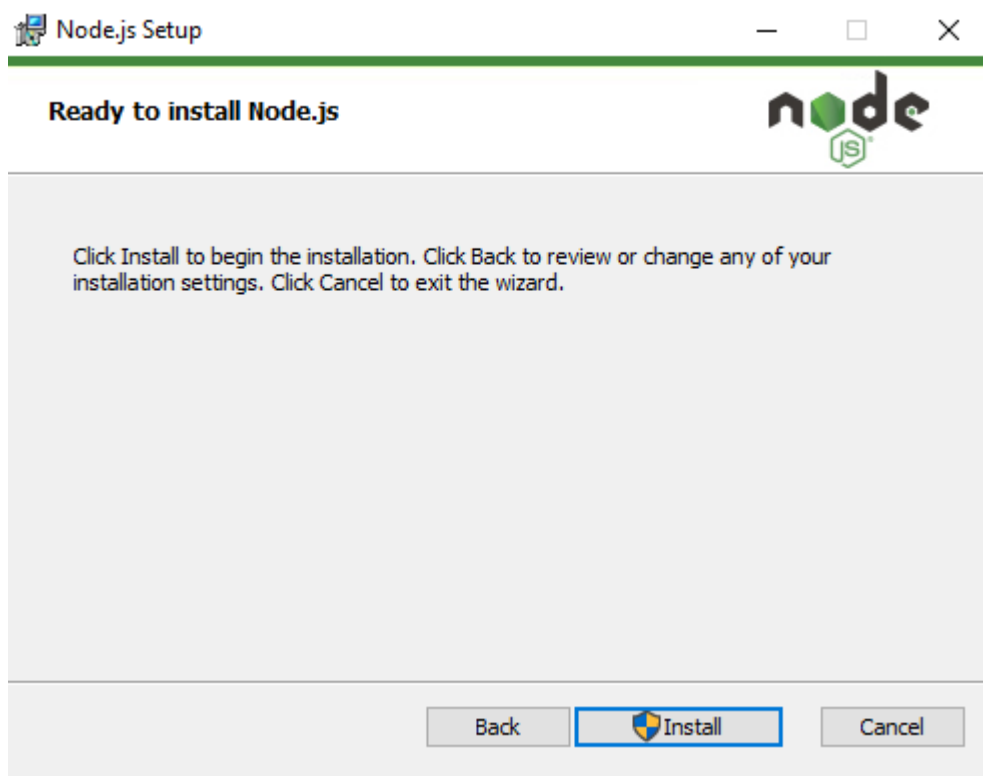


Рисунок 8 – Установка *node.js*

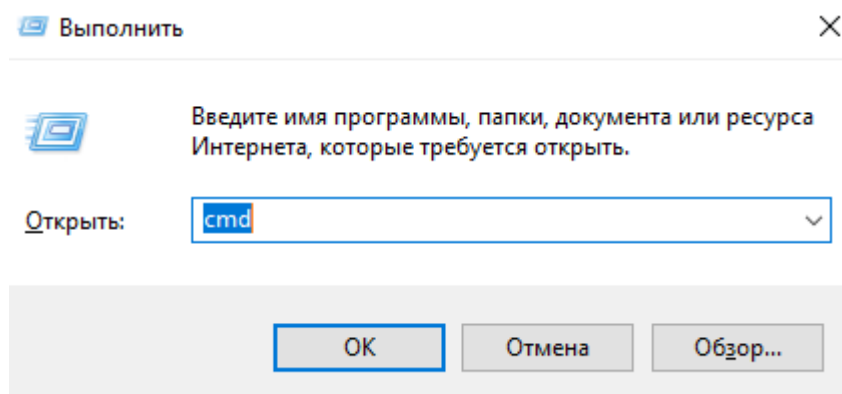
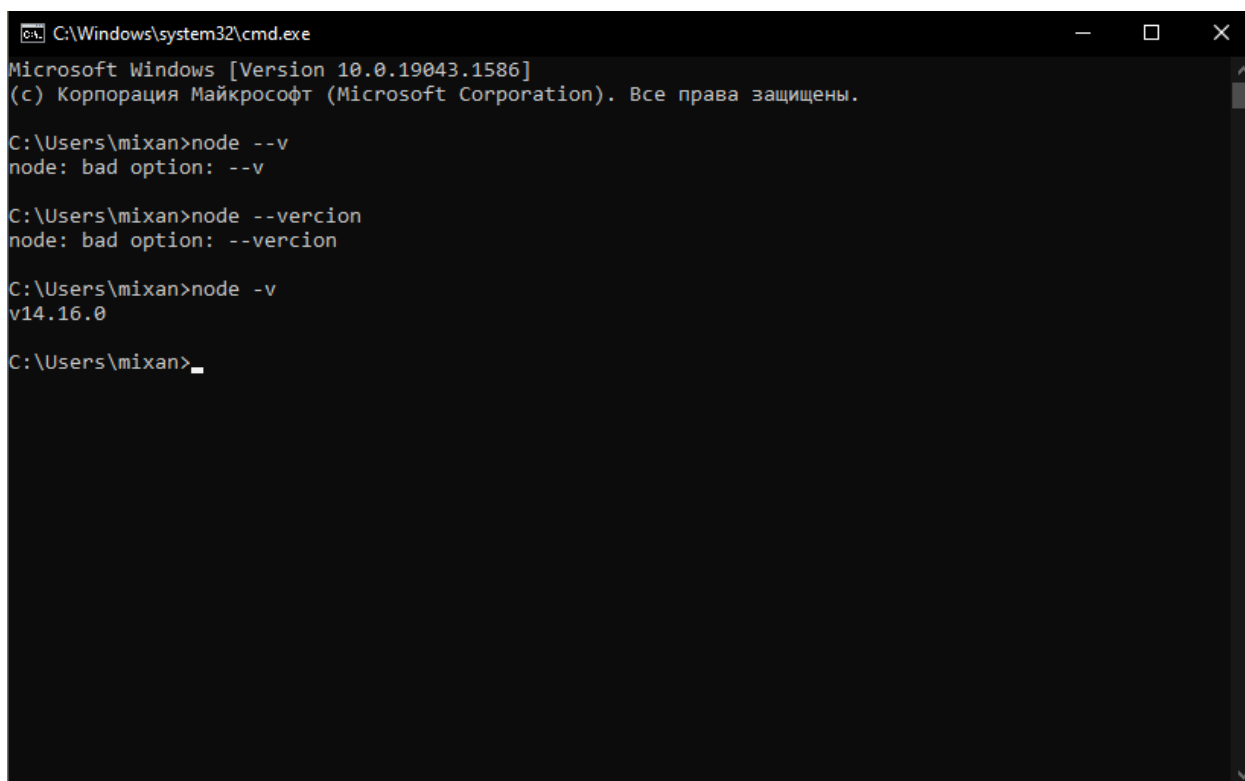


Рисунок 9 – Открытие консоли

Новая актуальная версия Node.js выпускается каждые 6 месяцев. Одна — в октябре, другая — в апреле. Их называют «текущими версиями».

Версии с нечётными номерами (v9, v11, v13) выпускаются каждый октябрь. Актуальны они сравнительно недолго, они не считаются готовыми для применения в продакшне. Их можно считать бета-версиями платформы.

Версии с чётными номерами (v8, v10, v12) выходят каждый апрель.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1586]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\mihan>node --v
node: bad option: --v

C:\Users\mihan>node --vercion
node: bad option: --vercion

C:\Users\mihan>node -v
v14.16.0

C:\Users\mihan>_
```

Рисунок 10 – Отображение версии *node.js*

После выпуска подобной версии прекращается выпуск обновлений для предыдущей «чётной» версии. Хотя эти версии и гораздо стабильнее «нечётных», активная работа над ними продолжается ещё 6 месяцев. В это время их можно считать находящимися в состоянии релиз-кандидата.

Для создания интернет-магазина с использованием библиотеки *react*, необходимо воспользоваться *npm*.

Чтобы создать шаблонное приложение *react*, необходимо выполнить следующие действия.

Для начала необходимо открыть терминал *windows* и прописать команду *npm install -g create react-app* (рисунок 11).

Данная команда устанавливает все необходимые пакеты из *npm*.

После установки нужных пакетов необходимо прописать команду *create-react-app weather*, где *weather* является названием приложения (рисунок 12).

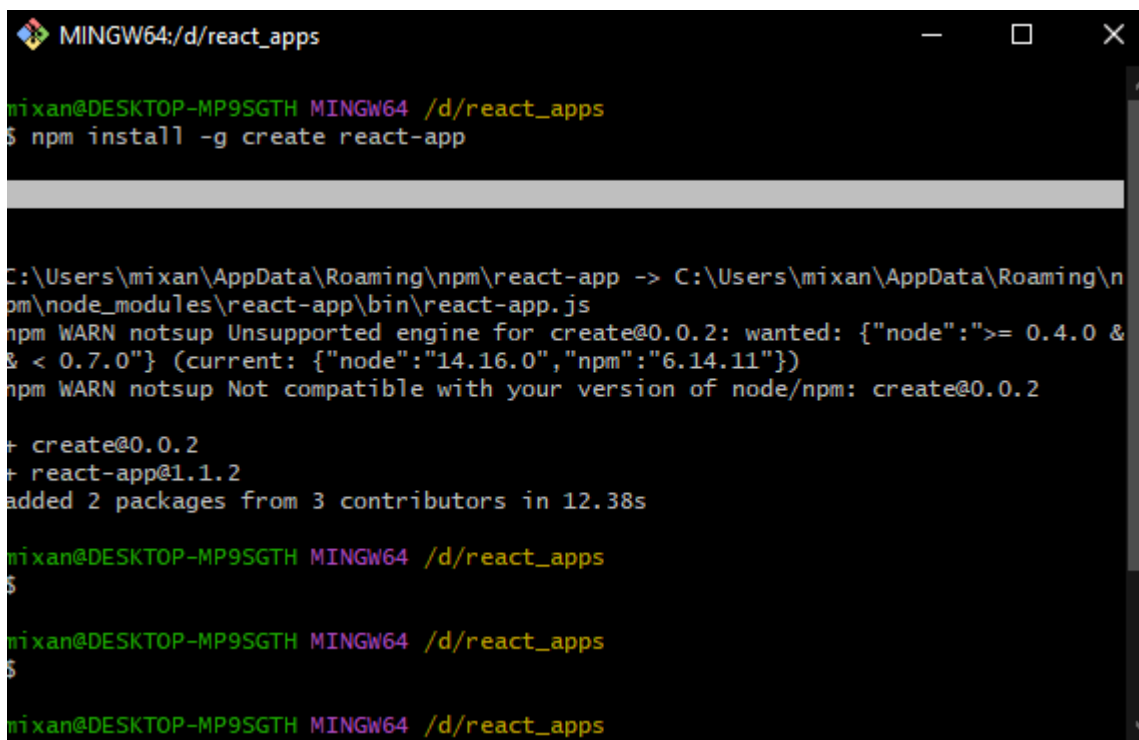


Рисунок 11 – Установка пакетов



Рисунок 12 – Создание шаблонного приложения

После того, как произвелась установка необходимых пакетов, можно запускать приложения. Для этого необходимо перейти в данную директорию при помощи команды *cd weather* (рисунок 13).

					ПГУ 1.090501.10.001 ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

```
npm start
  Starts the development server.

npm run build
  Bundles the app into static files for production.

npm test
  Starts the test runner.

npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd weather
  npm start

Happy hacking!

mixan@DESKTOP-MP9SGTH MINGW64 /d/react_apps
$ cd weather

mixan@DESKTOP-MP9SGTH MINGW64 /d/react_apps/weather (master)
$ |
```

Рисунок 13 – Переход в созданную директорию

После преехода в директорию необходимо написать команду *npm start* для запуска созданоого шаблонного проекта (рисунок 14).

```
npm start
weather@0.1.0 start D:\react_apps\weather
react-scripts start

(node:14028) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:14028) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...

Compiled successfully!

You can now view weather in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.3:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Рисунок 14 – Запуск проекта

После написания команды *npm start* в консоле будет выведен порт, на котором будет отображаться созданное шаблонное приложение (рисунок 15).

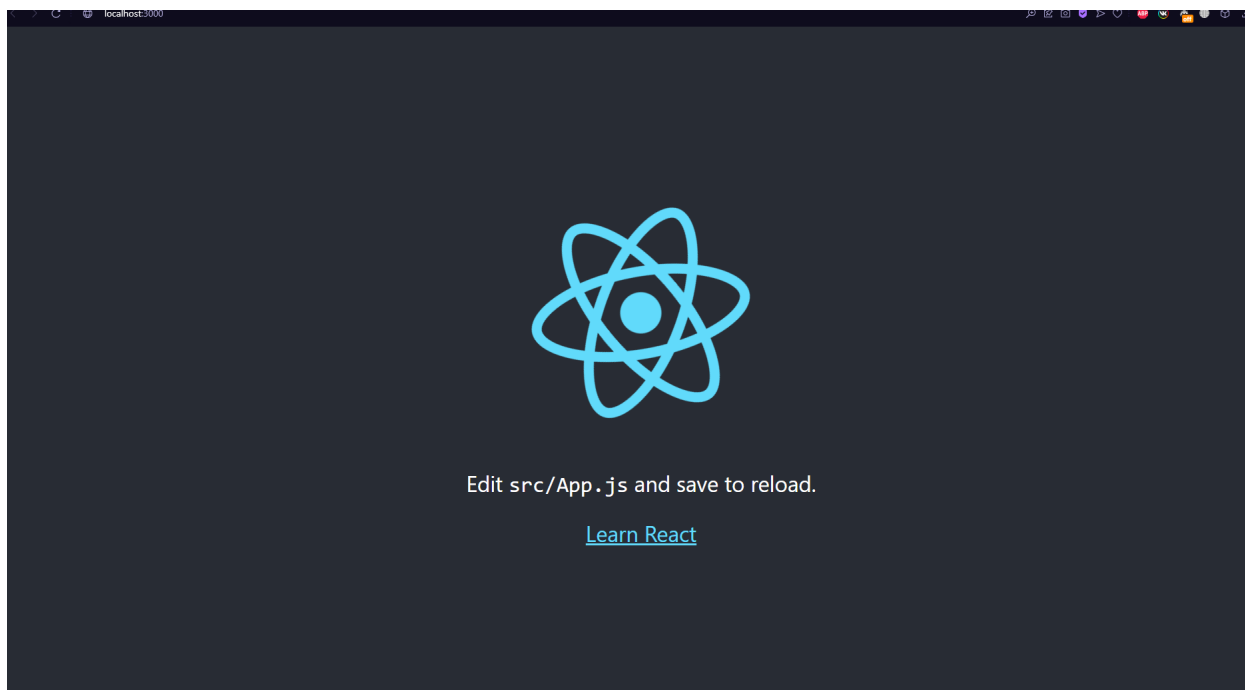


Рисунок 15 – Шаблонное приложение

Теперь необходимо проанализировать только что созданное *React*-приложение для того, чтобы поверх него написать собственное. Для этого необходимо воспользоваться *IDE Visual Studio Code*. (рисунок 16).

На стартовой странице *IDE* можно создать необходимые файлы, создать необходимую папку в любом месте компьютера, либо открыть существующую. Так же есть такая возможность, как клонирование любого репозитория. Для этого необходимо знать *url*-адресс репозитория, чтобы его клонировать. В *IDE* так же отображается история последних открытых проектов, как показано на рисунке 16. Нам необходимо открыть проект *weather*, поэтому нужно открыть существующую папку (рисунок 17).

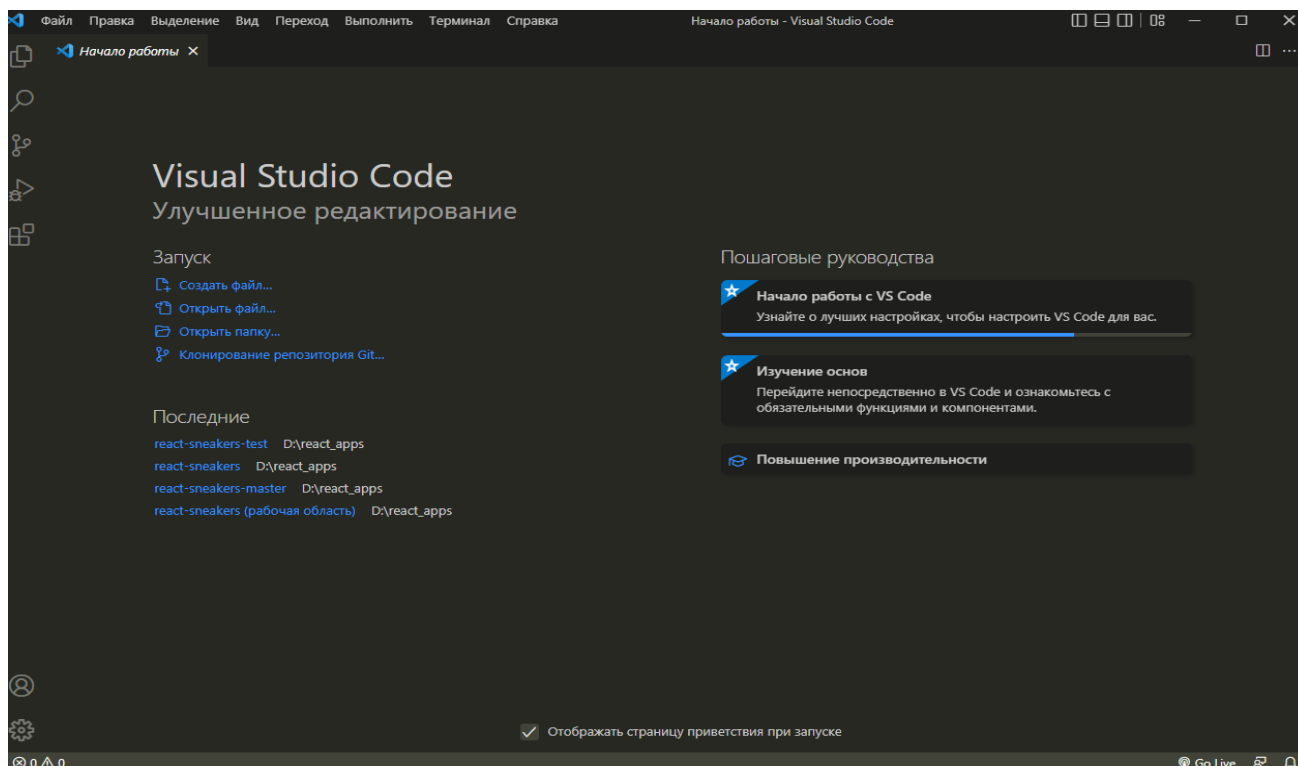


Рисунок 16 – Начальная страница *Visual Studio Code*

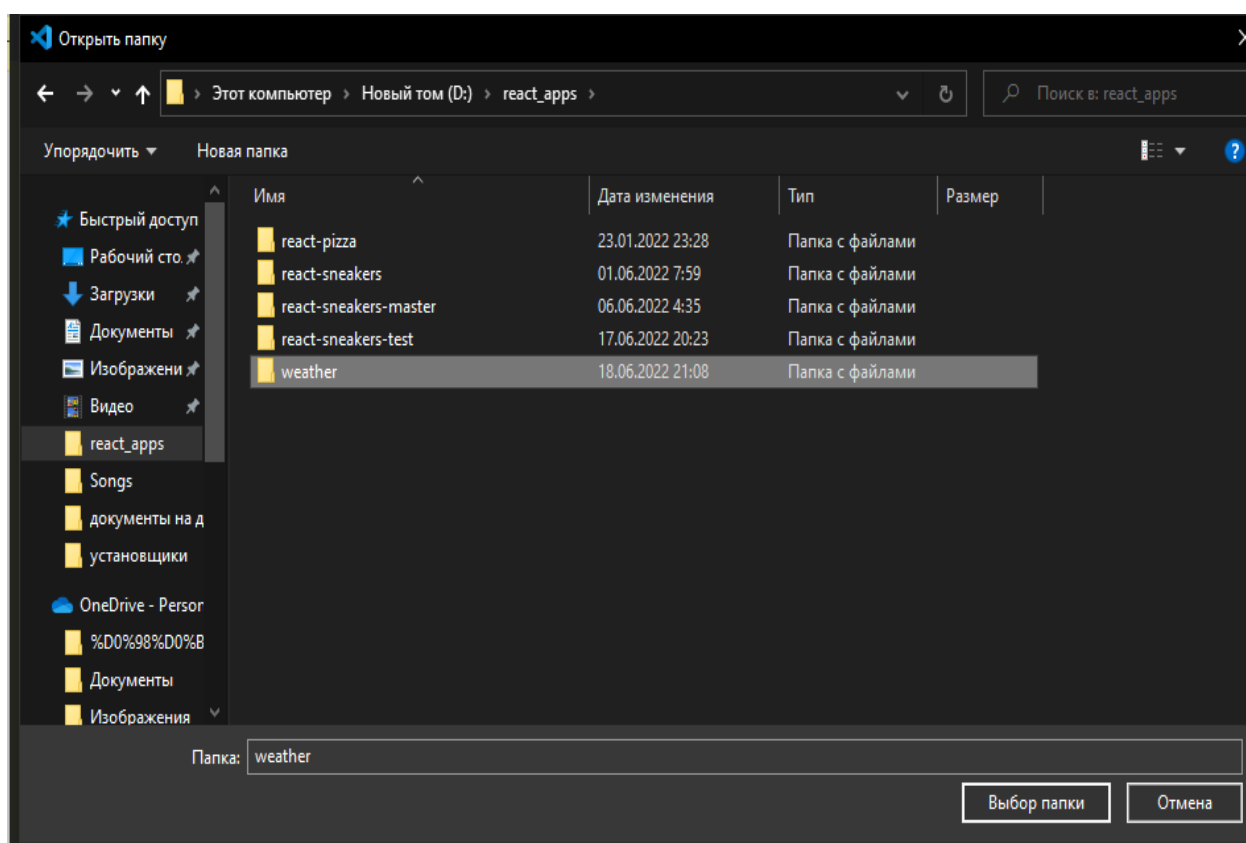


Рисунок 17 – Открытие выбранной папки

После открытия нужно папки, она отобразится в редакторе кода (рисунок 18).

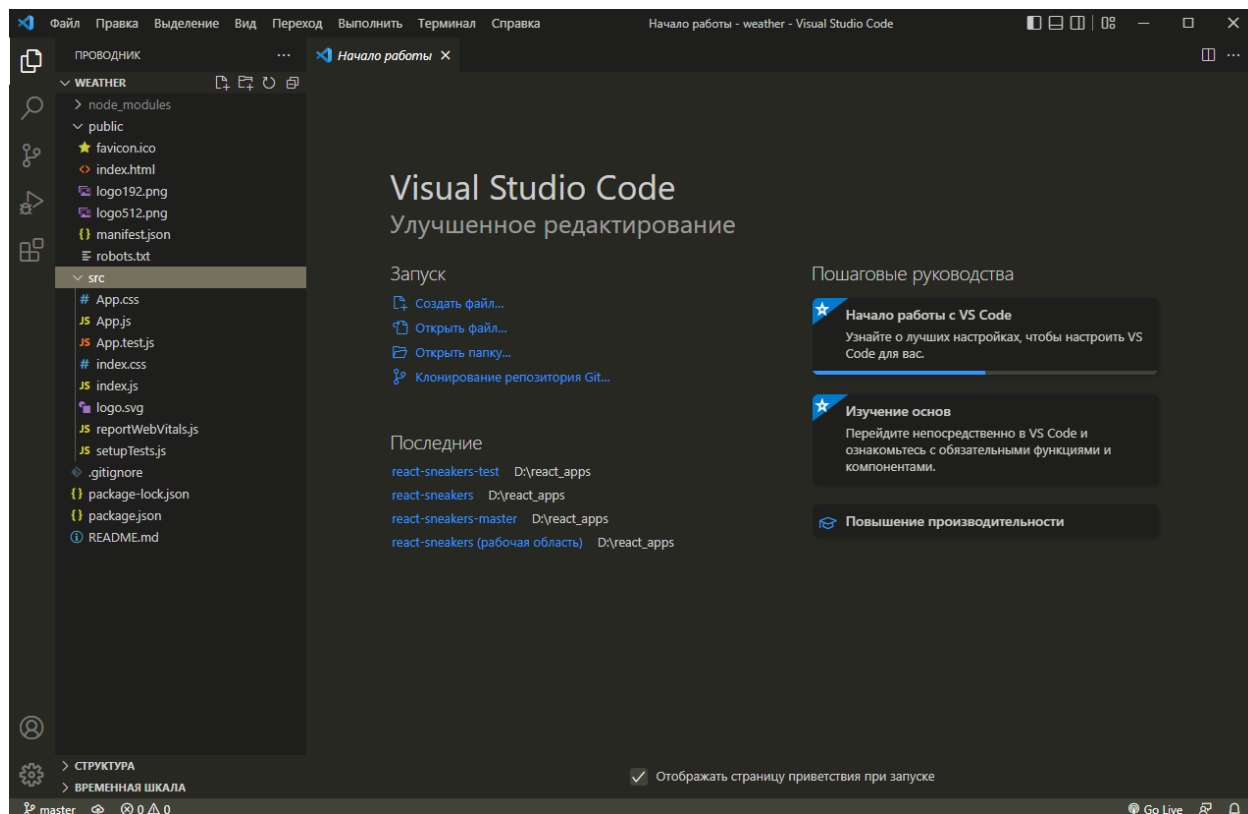


Рисунок 18 – Отображение папки *weather* в редакторе

В левой части редактора кода расположено структура приложения. Необходимо рассмотреть основные компоненты проекта, чтобы понять логику работы приложения.

Для этого откроем файл *src/App.js* (рисунок 19). Для того чтобы создать компонент *React*, необходимо сначала создать класс, являющийся наследником *React.Component*. Именно эту задачу решает строка *class App extends Component*. Все компоненты *React* должны содержать реализацию метода *render*, в котором выполняется рендеринг компонента, формирование описания его визуального представления. Этот метод должен вернуть *HTML*-разметку для вывода её на страницу [5].

Файл *index.html*. Находясь в папке проекта, нужно открыть файл, который находится по адресу *public/index.html* (рисунок 20).

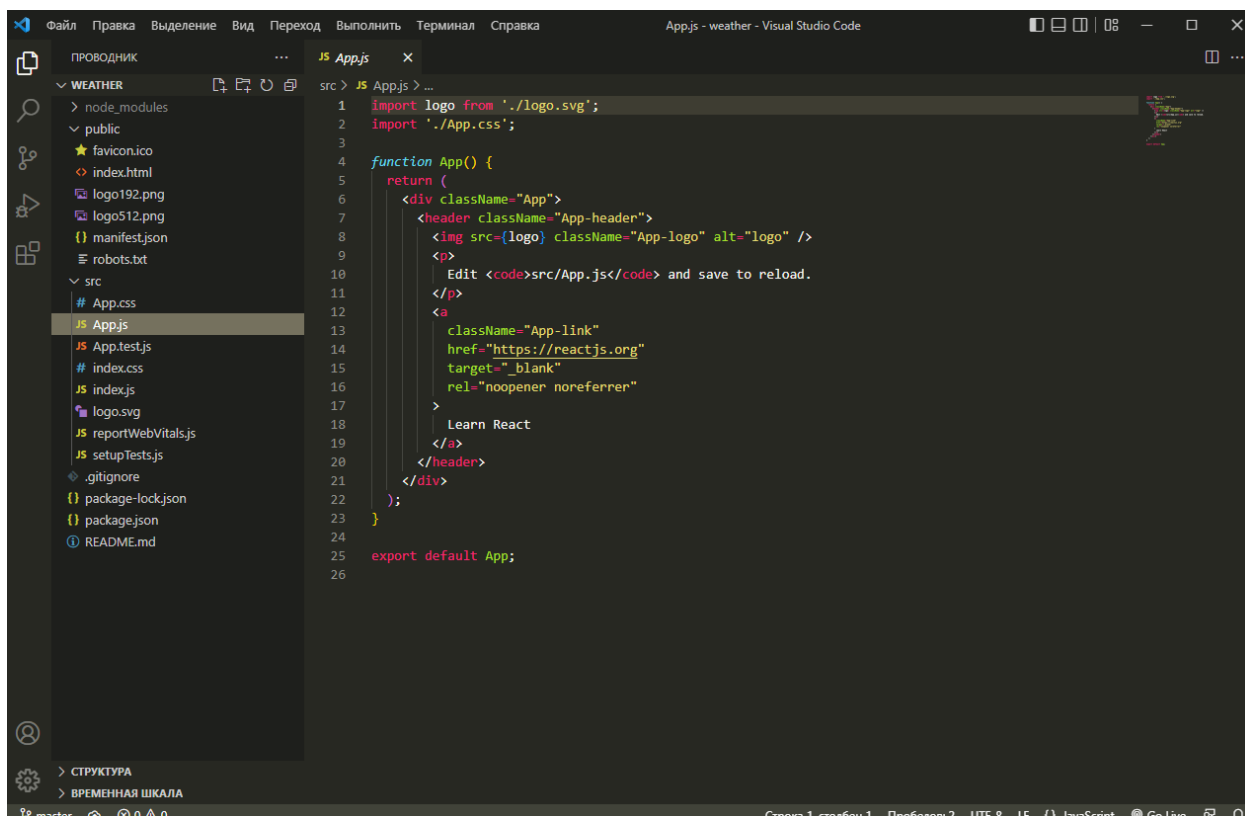


Рисунок 19 – Файл *App.js*

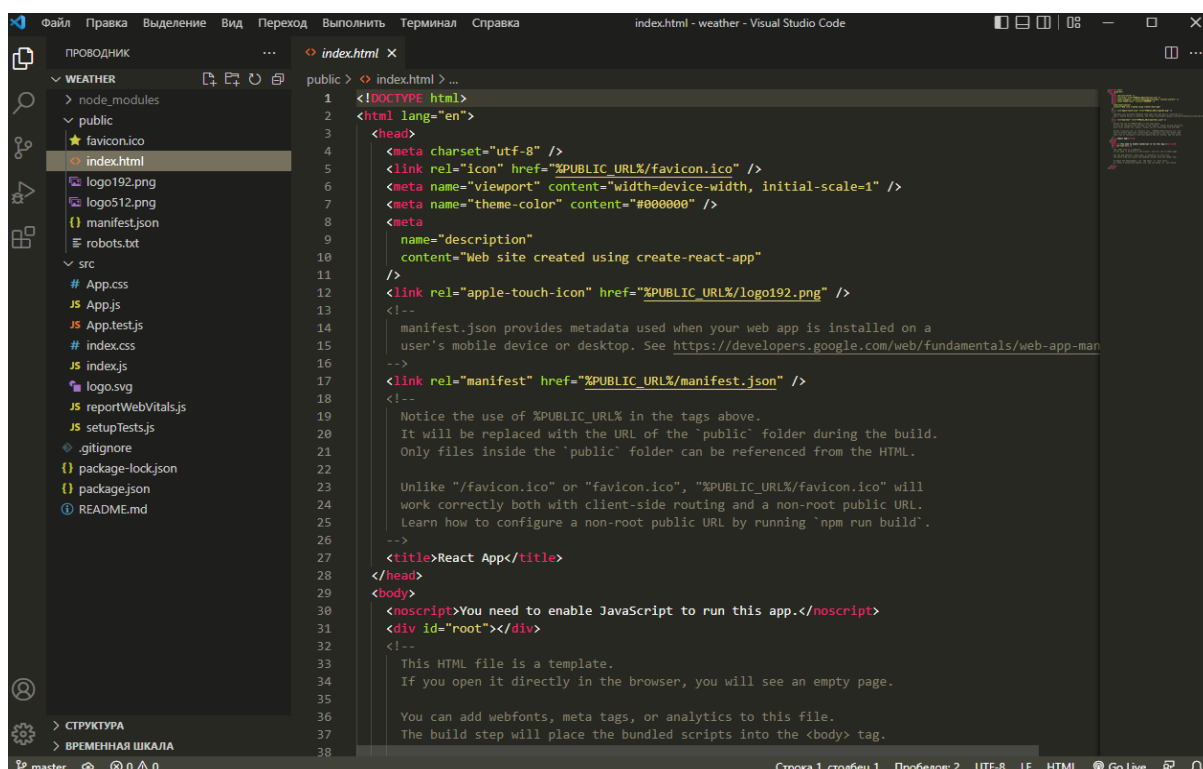


Рисунок 20 – Файл *index.html*

Основное внимание необходимо уделить строке `<div id="root">`. Этот элемент будет отображать наше react-приложение путем его замены, а все остальное останется неизменным.

Теперь откроем файл `src/index.js` (рисунок 21).

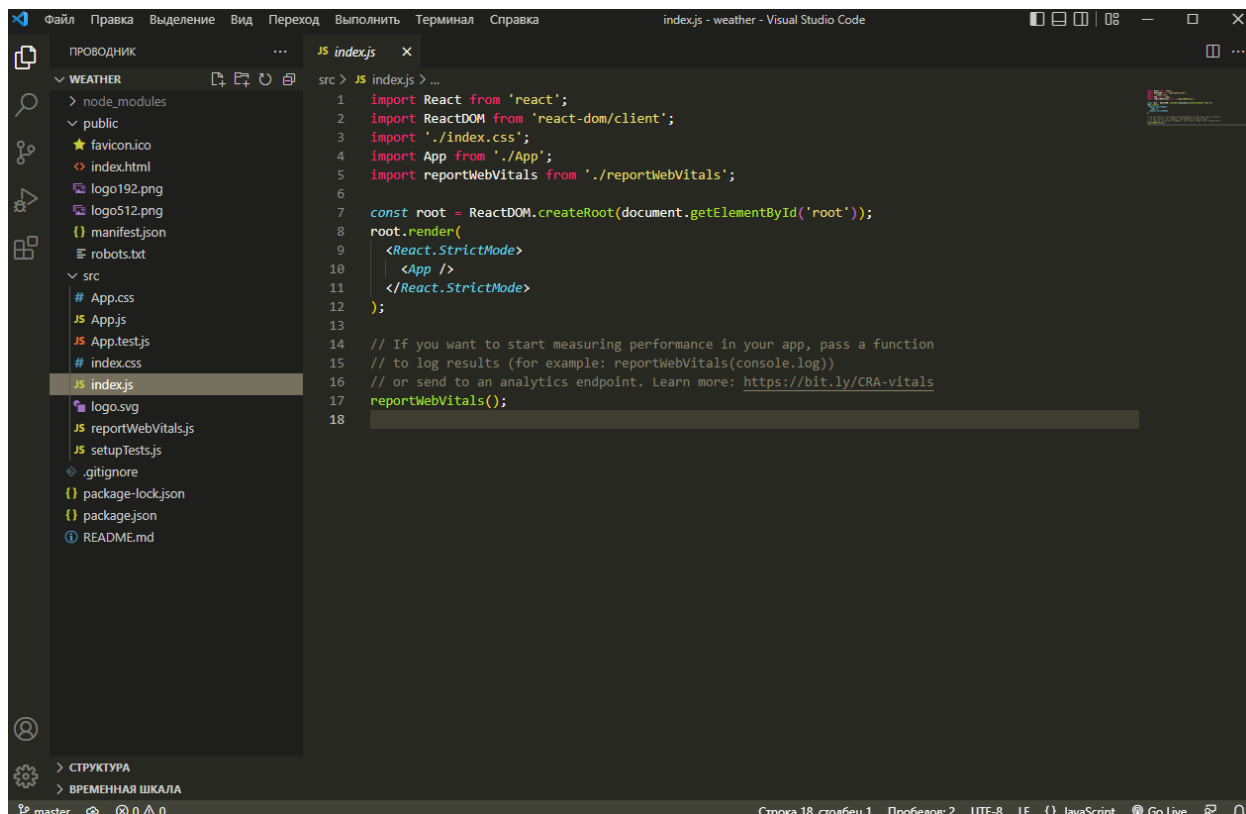


Рисунок 21 – Файл `index.js`

Вот строка кода, которая ответственна за вывод того, что называется «*React*-приложением», на страницу:

`-ReactDOM.render(<App />, document.getElementById('root'))`.

Эта строка сообщает *React* о том, что нужно взять компонент *App* и поместить его в *div*-элемент *root*, который был определён в только что рассмотренном файле `index.html`.

Разберёмся теперь с конструкцией `<App />`. Она очень похожа на *HTML*-код, но это — образец *JSX*-кода, который представляет собой особый синтаксис *JavaScript*, используемый *React*. Эта конструкция начинается с заглавной буквы *A*, на то, что это именно `<App />`, а не `<app />`. Это так из-за

используемого в *React* соглашения по именованию сущностей. Такой подход позволяет системе различать обычные *HTML*-теги и компоненты *React*. Если имена компонентов не будут начинаться с заглавной буквы, *React* не сможет вывести их на страницу.

Проведя анализ шаблонного приложения и разобравшись в его логике, можно приступить к написанию основного, для этого необходимо составить иерархию модулей в приложении (рисунок 22).

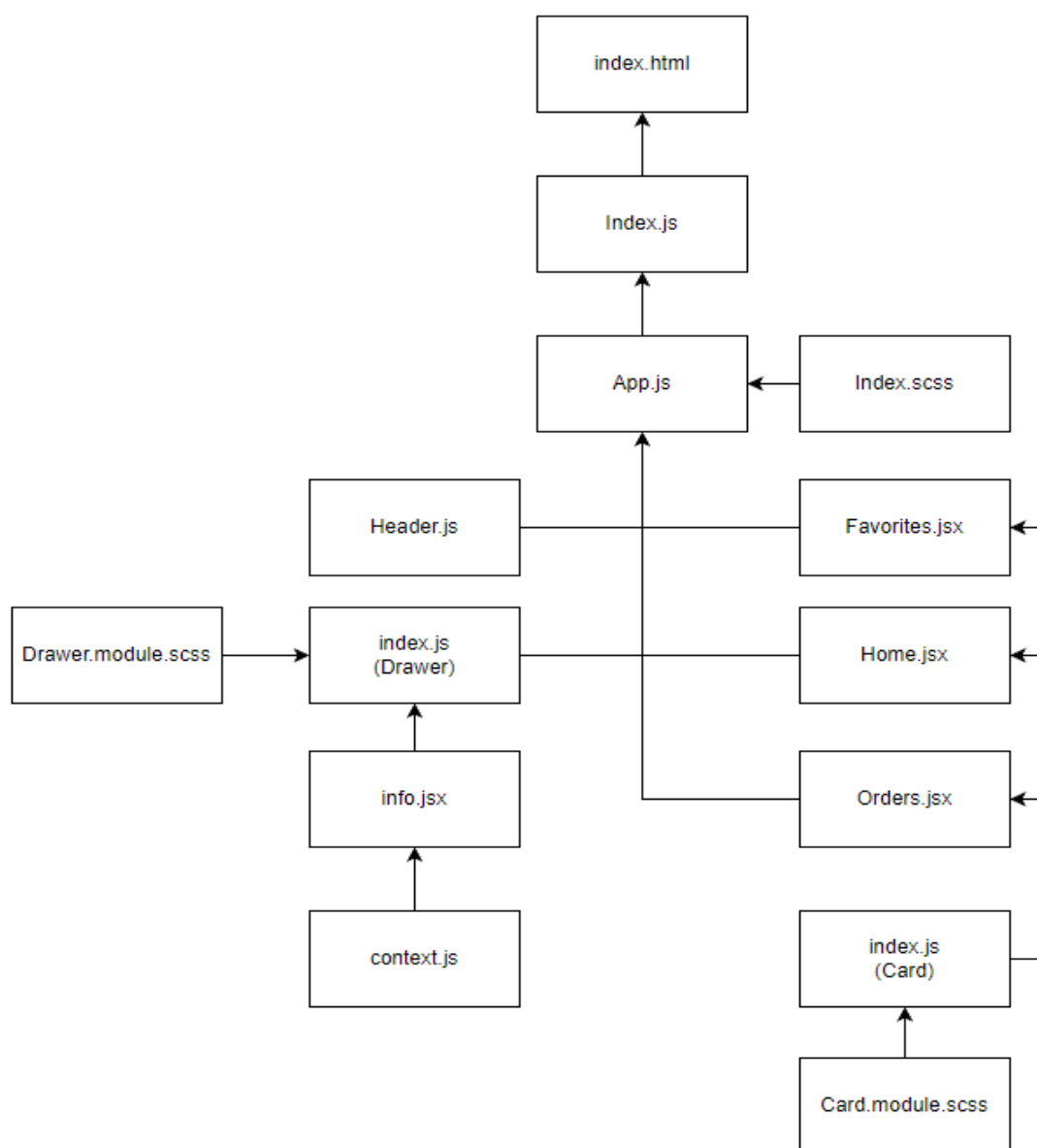


Рисунок 22 – Иерархия модулей в приложении

Для того, чтобы разобраться в иерархии, нужно дать определение модулей и их функциональное назначение.

Модуль — функционально законченный фрагмент программы.

Модульное программирование — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам.

Модульное приложение должно отвечать следующим требованиям:

- весь код модуля располагается в одной папке. Чтобы полностью удалить модуль из программы достаточно удалить соответствующую папку. Удаление модуля не нарушает работоспособности других модулей, но лишает приложение части функциональности;

- модули не зависят друг от друга. Модификация любого модуля не влияет на работу других модулей. Допускается зависимость модулей от «ядра» системы;

- ядро системы содержит публичное *API*, предоставляющее модулям средства ввода/вывода и набор компонентов для создания *UI* [6] .

В приложении модули можно подразделить на следующие типы:

- модули, отвечающие за дизайн и правильное отображение всех смысловых блоков. К таким модулям относятся:

- 1) *card.module.scss*;
- 2) *drawer.module.scss*;
- 3) *index.scss*;

- модули, отвечающие за функционал отдельных блоков в приложении.

К таким модулям относятся:

- 1) *index.js (Drawer)*;
- 2) *header.js*;
- 3) *index.js (Card)*;
- 4) *info.jsx*;

- модули, отвечающие за работу приложения в целом. К таким модулям

относятся:

- 1) *app.js*;
- 2) *index.js*;
- 3) *index.html*;
- 4) *home.jsx*;
- 5) *favorite.jsx*;
- 6) *orders.jsx*.

Разберем каждый из модулей более подробно:

- *app.js* – модуль, отвечающей за сборку всего приложения. Он делится на 3 основные части:

- 1) импорт сторонних модулей;
- 2) *react*-компонент *App*;

3) инструкция *export* для экспортирования *App* в него импортируются модули приложения, отвечающие за функционал, после чего они собираются в один компонент и отправляются на экспорт;

- *index.js* – модуль, отвечающий за рендер *DOM* дерева. Он принимает *App.js* для его отрисовки;

- *index.html* - модуль который отображает все приложение. Название этого модуля изменять нельзя;

- *home.jsx* – модуль, в котором отрисовывается первоначальная страница приложения;

- *favorite.jsx* – модуль, в котором отрисовывается страница закладок пользователя;

- *orders.jsx* – модуль, в котором отрисовывается страница истории покупок пользователя;

- *card.module.scss*, *Drawer.module.scss*, *index.scss* – модули, в которых описываются стили соответствующих модулей приложения;

- *idex.js (Drawer)* – модуль, отвечающий за функционал корзины интернет – магазина;

- *header.js* – модуль, отвечающий за отображение шапки интернет-магазина;

- *index.js (Card)* – модуль, отвечающий за функционал карточек товара, которые в свою очередь, экспортируются на все страницы магазина.

1.6 Тестирование и отладка

Тестирование и отладка проводились с использованием персонального компьютера, имеющего следующие характеристики:

- сетевой адаптер 10/100Мбит;
- клавиатура;
- мышь;
- видеокарта: *NVIDIA GeForce RTX 2060*;
- процессор: *AMD Ryzen 7 2700 3.20 GHz*;
- оперативная память: *16 GB*;
- операционная система: *Windows 10 X64*;
- **Память:** не менее *10 GB*.

Страница, которая изначально показывается пользователю – домашняя страница интернет-магазина, откуда ведутся основные действия. Первое что увидит пользователь – навигацию магазина, на котором изображен логотип интернет-магазина, корзина с отображением суммы стоимости товара, который в ней находится, иконка избранных товаров, иконка пользователя (Рисунок 23).

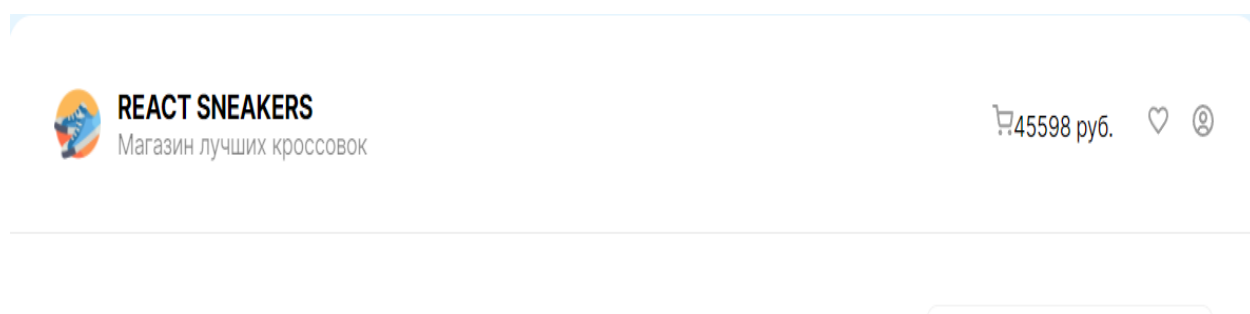


Рисунок 23 – Навигация интернет-магазина

Освоившись с навигацией интернет-магазина, пользователь начинает ознакомление с основной частью страницы, на которой расположен поиск и список товаров магазина (рисунок 24).

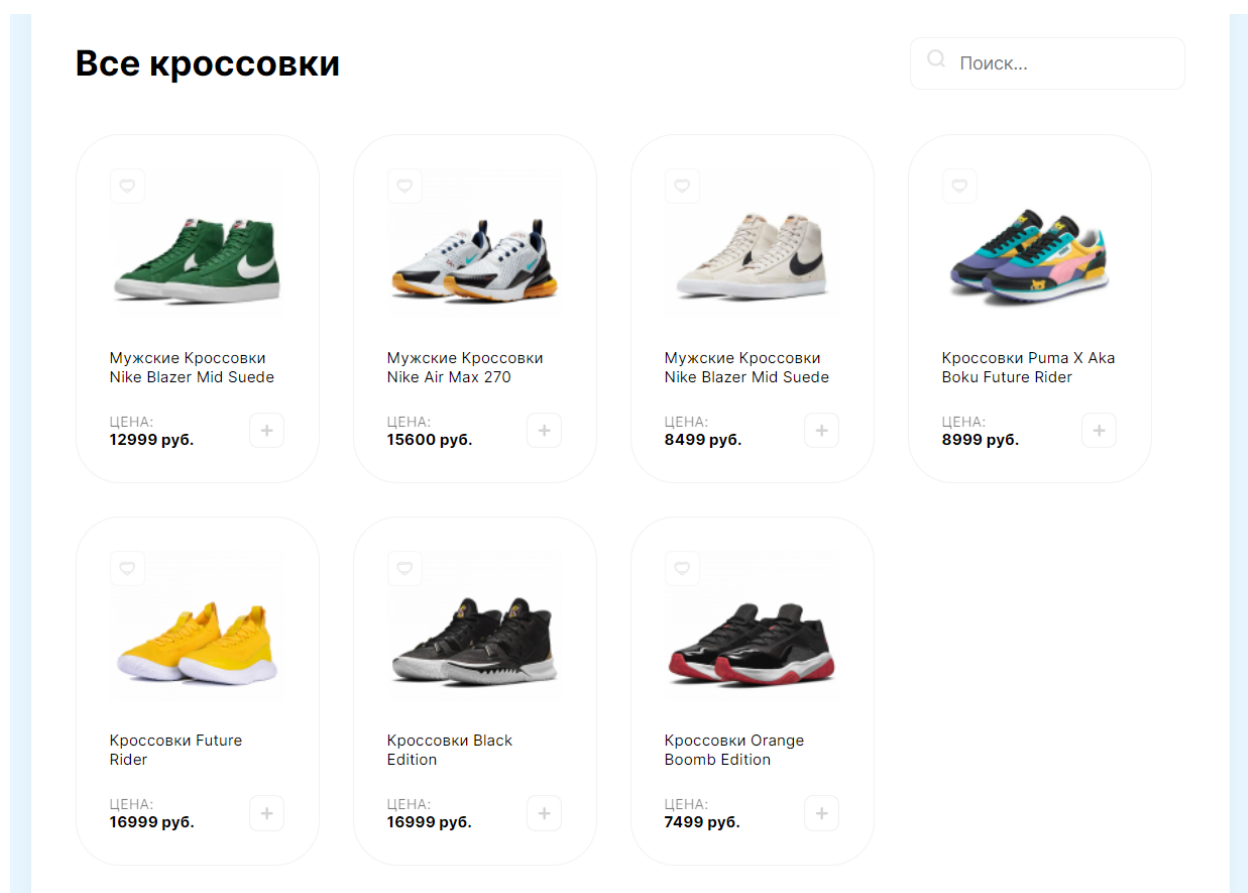


Рисунок 24 – Навигация интернет-магазина

Поиск расположен в правой части сайта. Для того, чтобы найти определенную модель товара, необходимо произвести ввод ключевых слов товара пользователем, затем выведутся подходящие товары под данный параметр, чтобы очистить поле поиска, необходимо нажать на кнопку clear (рисунок 25, 26).

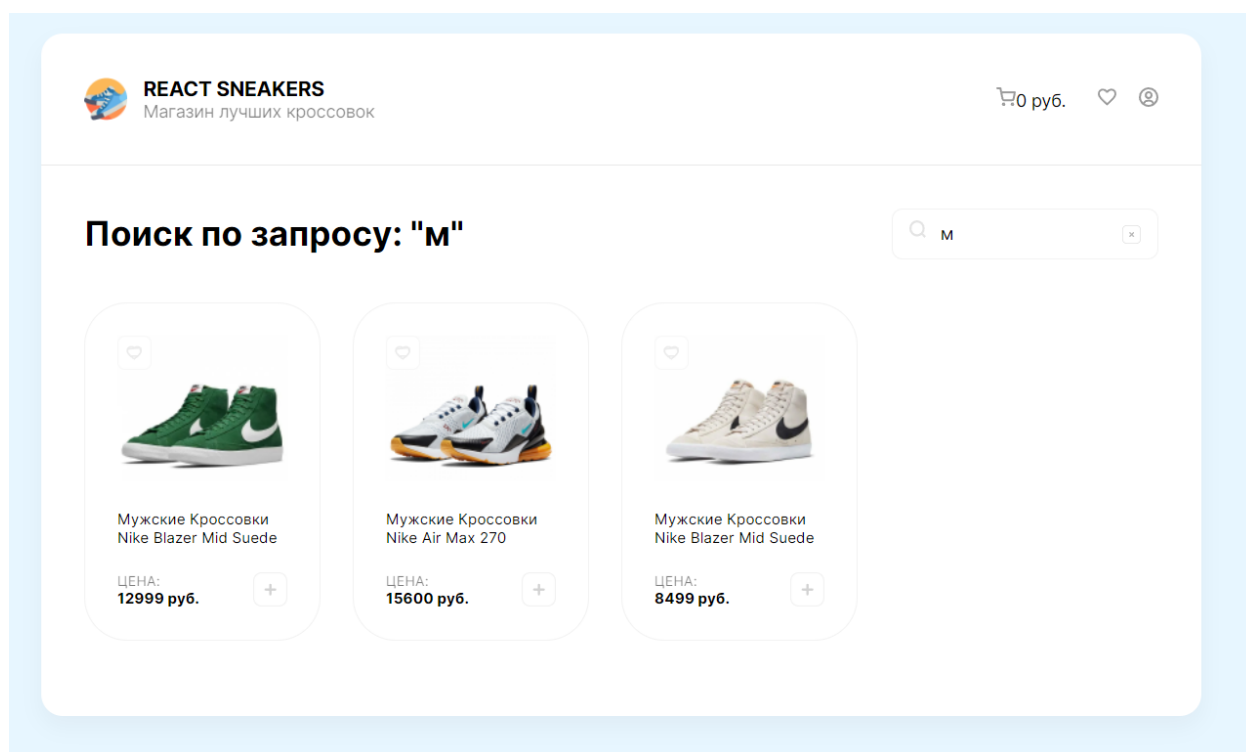


Рисунок 25 – Поиск товара по ключевым словам

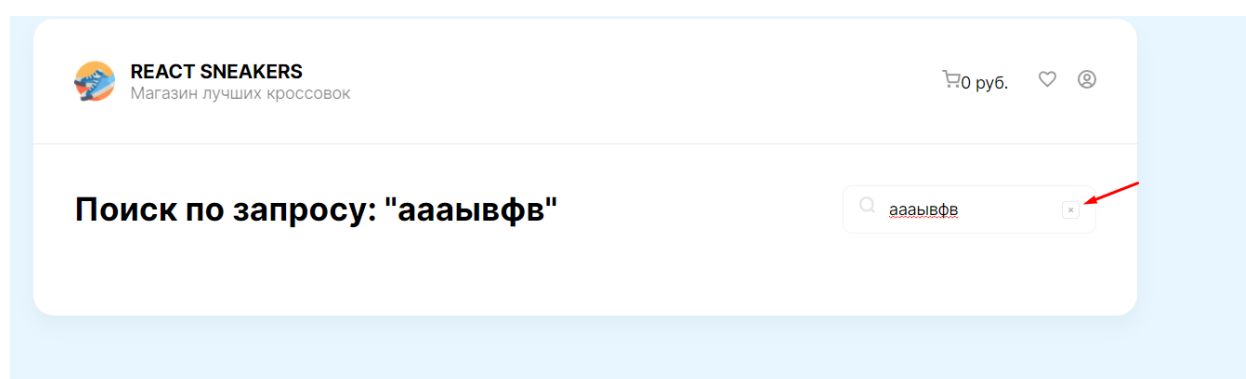


Рисунок 26 – Кнопка очистки текста

Ниже поиска расположен товар, который можно добавлять в корзину и в избранное, при добавлении в корзину кнопка плюса преобразуется в галочку с зеленым цветом. Чтобы добавить товар в избранное, необходимо нажать на иконку сердечка в левом верхнем углу товара, чтобы убрать товар из избранного, нужно еще раз нажать на эту же кнопку. (рисунок 27, 28).

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

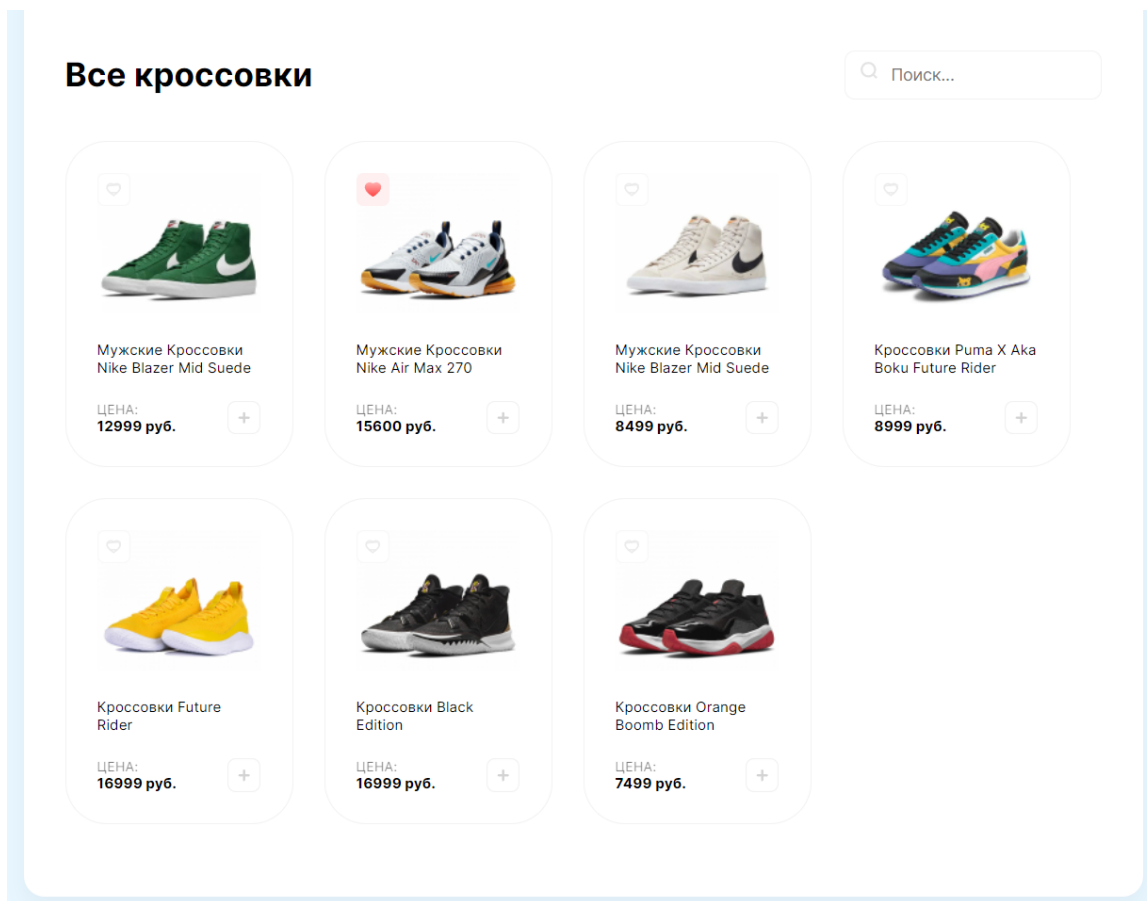


Рисунок 27 – Добавление товара в избранное

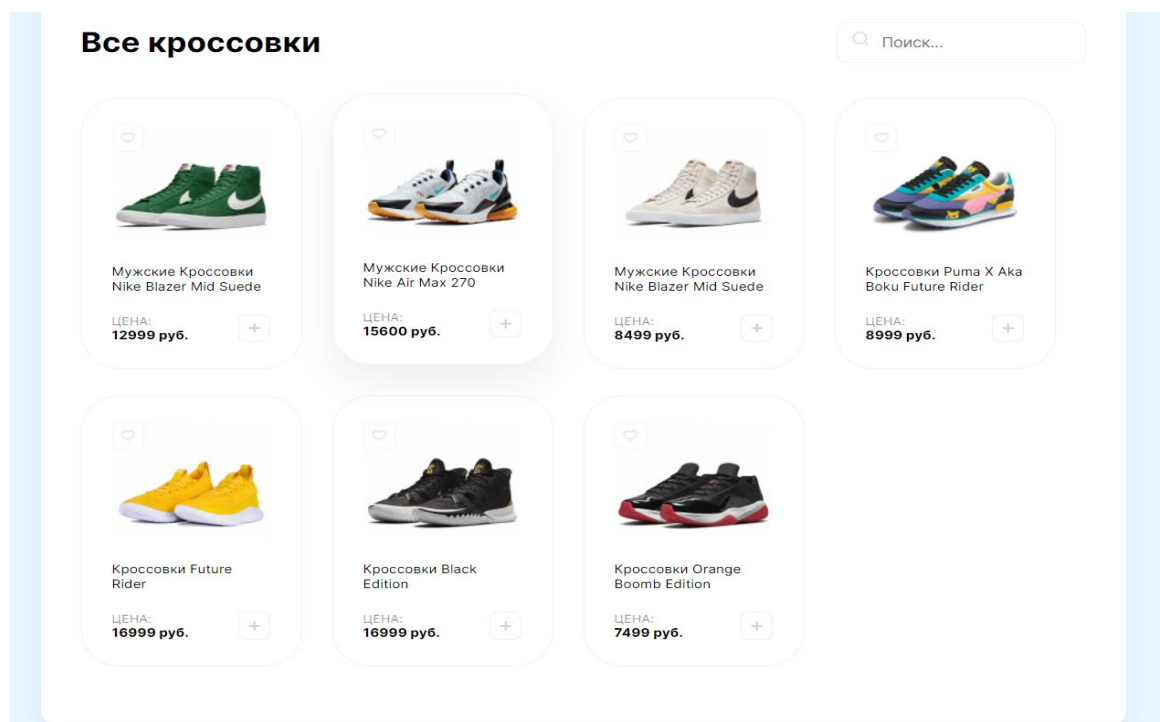


Рисунок 28 – Удаление товара из избранного

Добавление товара происходит путем нажатия пользователем иконки галочки в правой нижней части товара, после чего в верхней правой части сайта будет выведена суммы стоимости товаров, которые были добавлены в корзину (рисунок 29).

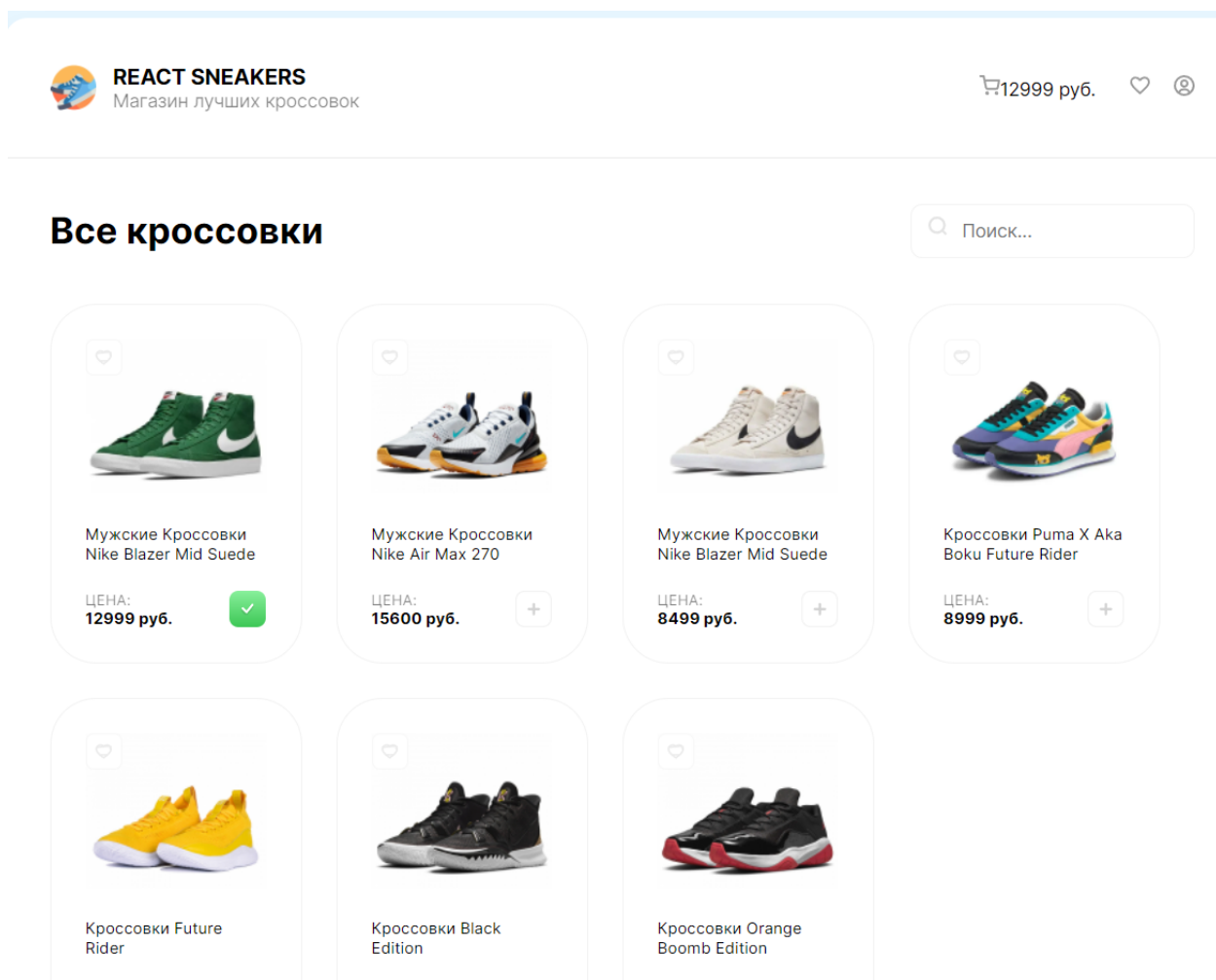


Рисунок 29 – Добавление товара в корзину

Для того, чтобы удалить товар из потребительской корзины, необходимо перейти в корзину путем нажатия иконки в правом верхнем углу сайта, после чего откроется корзина со списком товара, который был добавлен в нее (рисунок 30).

Корзина

×



Мужские Кроссовки
Nike Blazer Mid Suede
12999 руб.

×

Итого: **12999 руб.**

Налог 5%: **649.95 руб.**

Оформить заказ→

Рисунок 30 – Открытие потребительской корзины

					<i>ПГУ 1.090501.10.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		40

Удалить товар из корзины можно путем нажатия кнопки крестика рядом с иконкой товара. Для оформления заказа необходимо нажать на зеленую кнопку оформить заказ, после чего заказ будет оформлен (рисунок 31).

Корзина



Заказ оформлен!

Ваш заказ #18 скоро будет передан
курьерской доставке

← Вернуться назад

Рисунок 31 – Оформление заказа

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		41

При удалении товара появится окно с тем, что товара в корзине нет. Чтобы вернуться на главную страницу магазина, необходимо нажать на зеленую кнопку «Вернуться назад» (рисунок 32).

Корзина



Корзина пустая

Добавьте хотя бы одну пару кроссовок, чтобы сделать заказ.

← Вернуться назад

Рисунок 32 – Пустая корзина

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		42

С главной страницы магазина пользователь может перейти на страницу закладок, для этого ему нужно нажать на иконку закладок в правом верхнем углу экрана (рисунок 33).

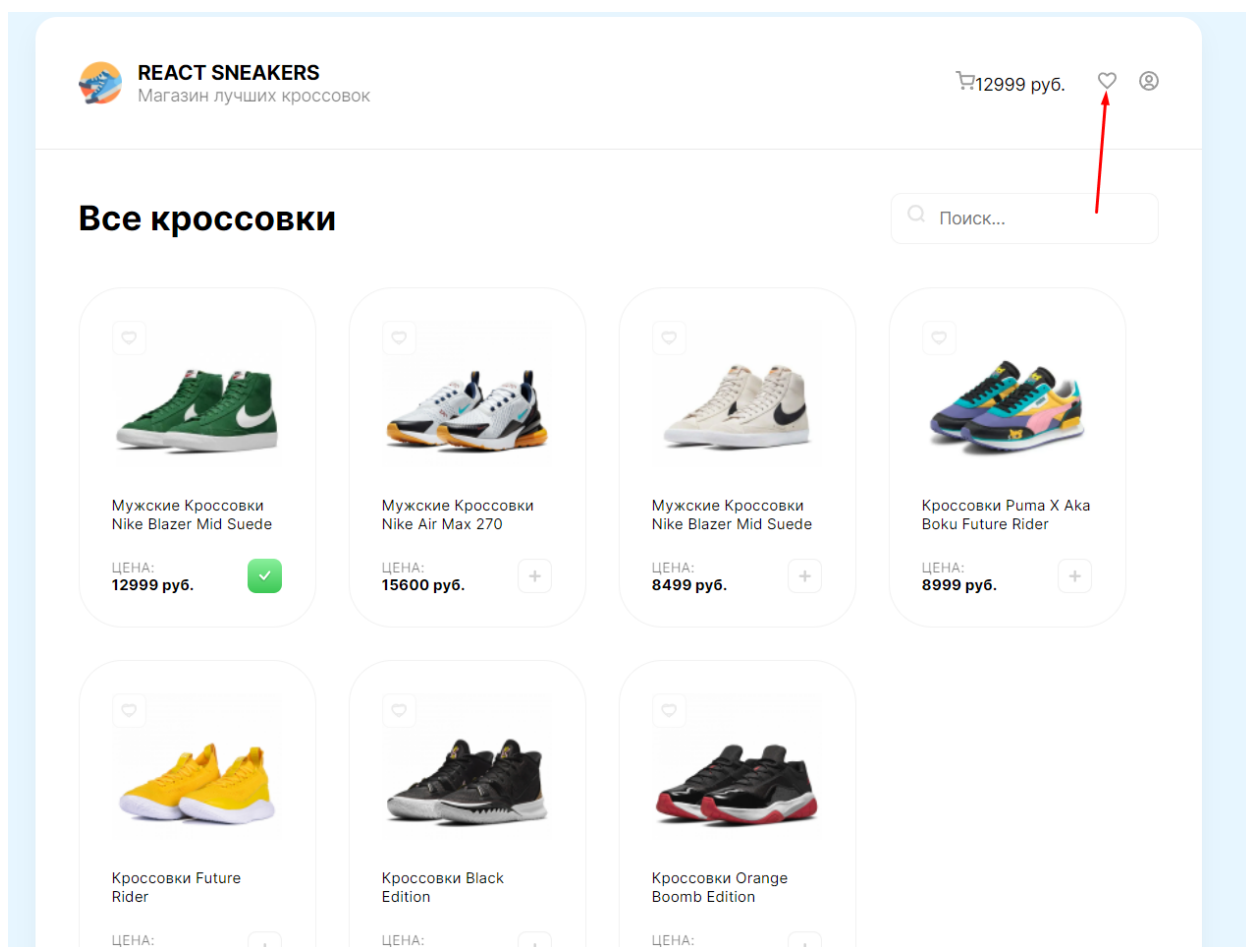


Рисунок 33 – Иконка избранных товаров

В закладках отображаются избранные товары пользователем. В них можно удалить избранные товары из списка. Для этого необходимо пользователю нажать на иконку избранного товара на определенном товаре в списке, после чего он уберется из списка. Чтобы вернуться на главную страницу магазина, нужно нажать на логотип магазина в левом верхнем углу сайта (рисунок 34, 35).

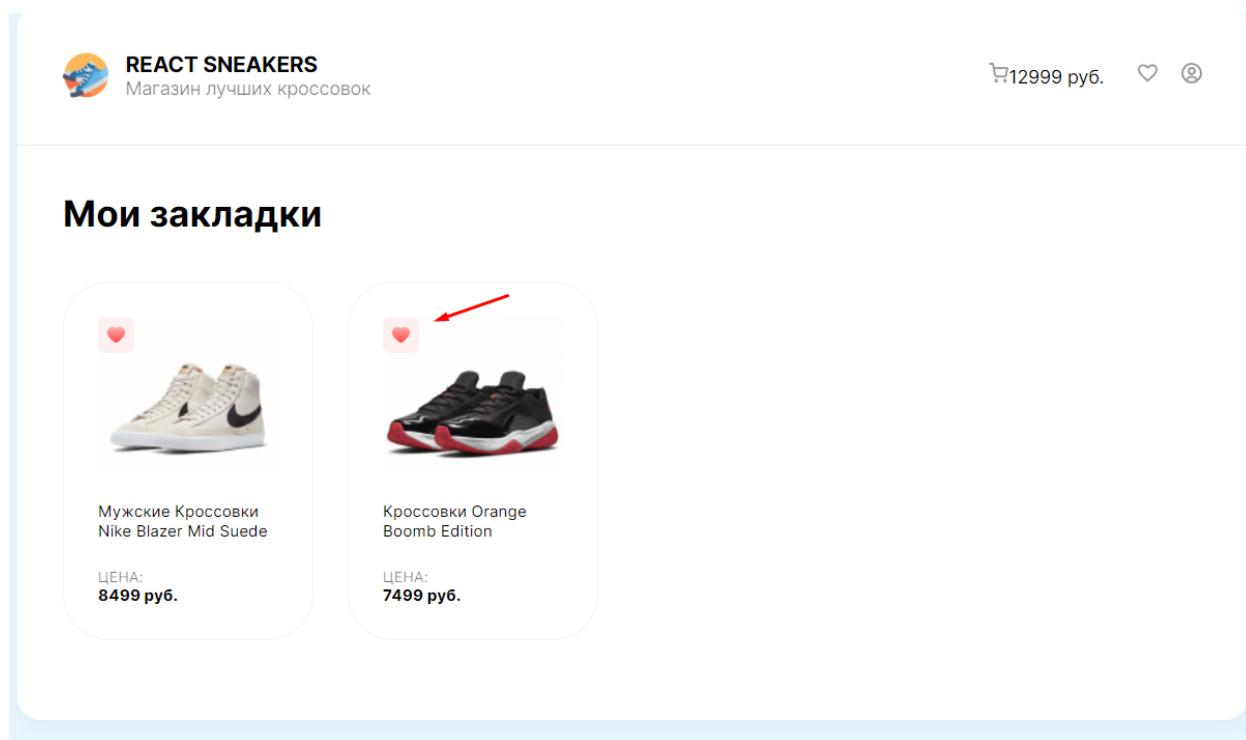


Рисунок 34 – Удаление выбранного товара из избранного

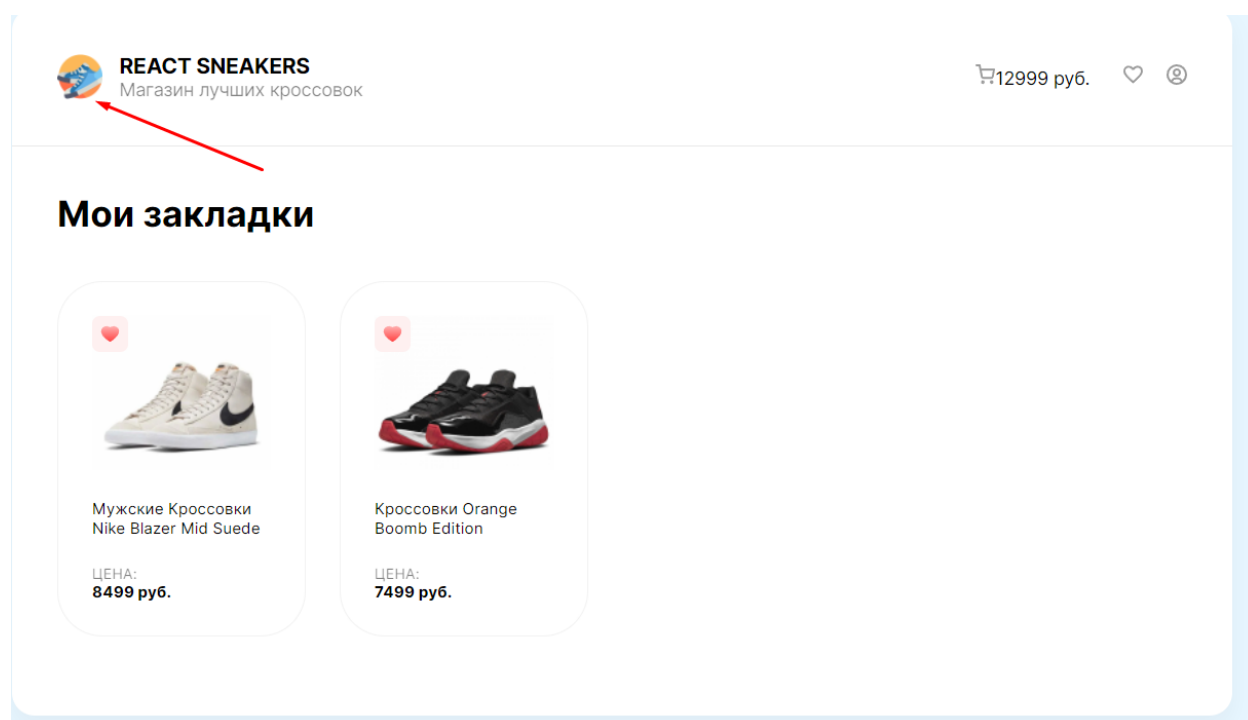


Рисунок 35 – Переход на главную страницу магазина

С главной страницы магазина можно перейти в историю покупок пользователем. Для этого необходимо нажать на иконку истории покупок в

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		44

левом правом углу сайта, чтобы выйти на главную страницу, необходимо нажать на логотип сайта в левом верхнем углу сайта (рисунок 36, 37).

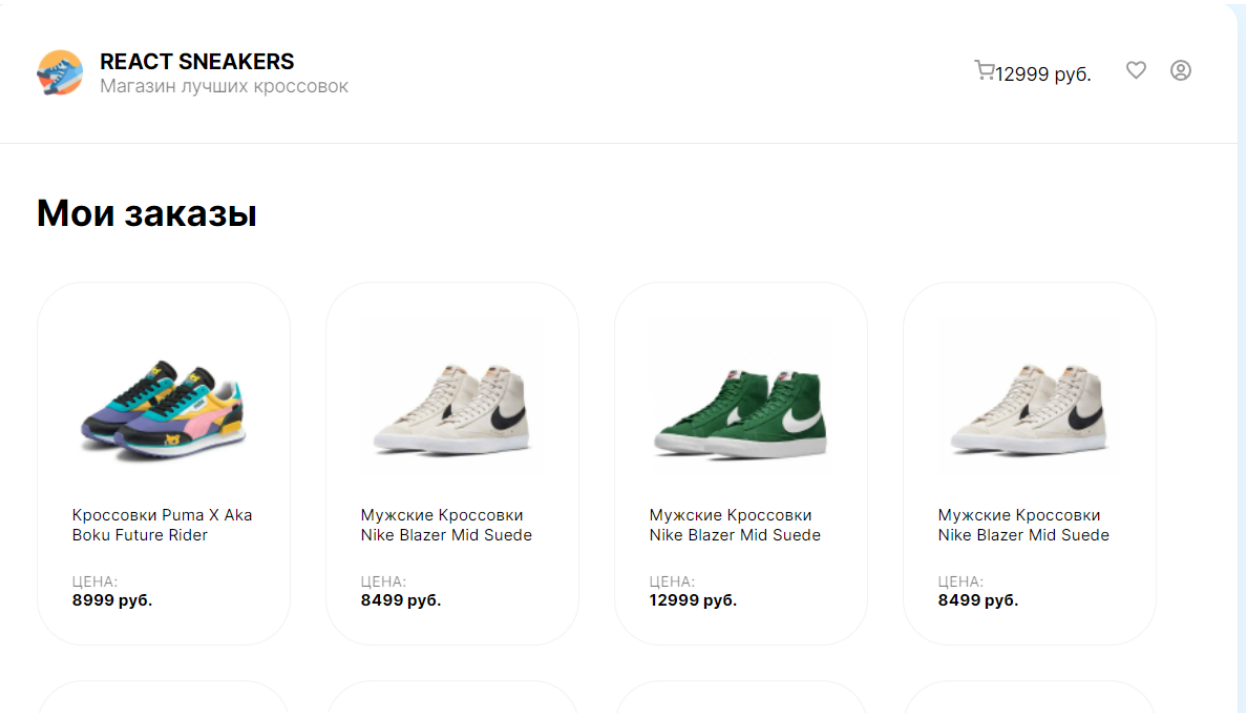


Рисунок 36 – История покупок

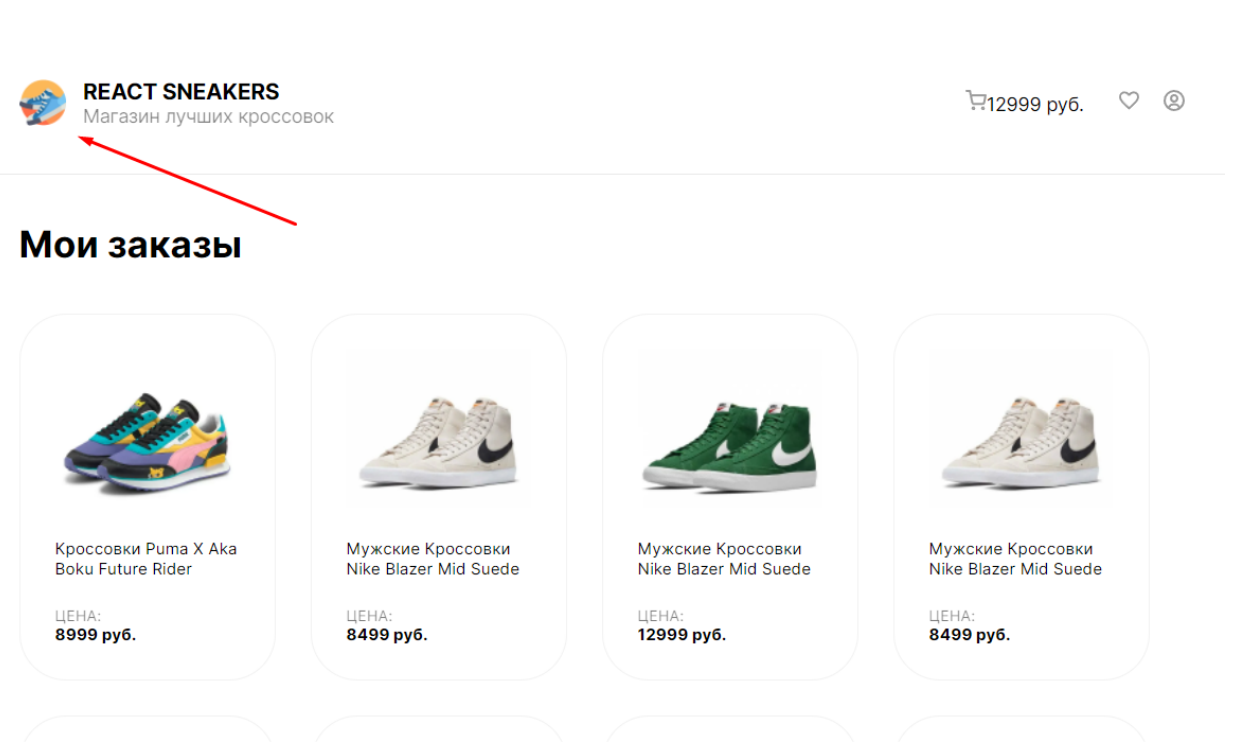


Рисунок 37 – Переход на главную страницу магазина

Таким образом, программа выполняет все функции, описанные ранее, в

полном объеме, сайт имеет интуитивный и понятный дизайн, что позволяет пользователю находиться на нем максимально комфортно продолжительное время.

					<i>ПГУ 1.090501.10.001 ПЗ</i>	<i>Лист</i>
						46
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

2 Технико-экономическое обоснование разработки программы

2.1 Введение

Целью технико-экономического обоснования разработки является количественное и качественное доказательство экономической целесообразности разработки и усовершенствования программы, а также определение организационно-экономических условий ее эффективного функционирования. Эффективность программного модуля определяется его качеством и эффективностью процесса разработки и сопровождения. Качество программного изделия определяется тремя составляющими:

- с точки зрения пользователя данного программного продукта;
- с позиции использования ресурсов и их оценки;
- по выполнению требований на программное изделие.

Программное изделие должно быть разработано так, чтобы оно выполняло свои функции без лишних затрат ресурсов (оперативной памяти ПК, машинного времени, пропускной способности каналов передачи данных и др. – на стадии функционирования; времени разработки и денежных ресурсов – на стадии использования программного изделия). Содержание технико-экономического обоснования разработки программы заключается в следующем:

- расчет технико-экономических показателей и выбор базы сравнения;
- определение трудоемкости и стоимости программного обеспечения;
- расчет цены ПО;
- расчет капитальных и эксплуатационных затрат на разработку;
- определение показателей финансово-экономической эффективности.

Приведенные в данном разделе дипломного проекта результирующие

таблицы, перечни данных, позволяют сопоставить результаты разработки и затраты на нее, чтобы сделать вывод об эффективности проекта. Исходные данные для расчета экономических показателей приведены в таблице 1.

Таблица 1 — Исходные данные для расчета экономических показателей

Обозначение	Наименование показателя	Единицы измерения	Значение показателя		
СЭВМ	Стоимость ЭВМ и телефона	тыс. руб	70		
ДМ	Среднее количество рабочих дней в месяце	дни	25		
ρ_n	Норматив рентабельности	-	0,24		
ω_d	Коэффициент, учитывающий дополнительную заработную плату разработчика программы	-	0,2		
ω_c	Коэффициент, учитывающий начисления органам социального страхования	-	0,27		
ω_n	Коэффициент, учитывающий накладные расходы организации	-	1,35		
qI	Количество I-задач, решаемых потребителем	Зад. год	60		
$tM.B.I$	Время решения I-ой задачи разработанной программой	Маш. час	1		
nn	Количество организаций, которые приобретут данную программу	шт.	1г	2г	3г
			4	5	8
$ZЭЛ$	Тариф за 1 кВт/час	Руб.	3,44		
$ТС$	Срок службы разработанной программы	год	5		
$НДС$	Налог на добавленную стоимость	%	20		
TP	Количество рабочих дней в году	дн	264		
NCM	Количество смен работы ЭВМ	-	1		
tCM	Продолжительность смены	ч	8		
α	Простои ЭВМ	%	10		
P	Мощность, потребляемая ЭВМ	кВт	0,4		
NCP	Среднее количество ремонтов в год	-	2		
SD	Стоимость деталей, заменяемых при ремонте	Руб.	2500		

2.2 Расчёт затрат на разработку программного обеспечения

Суммарные затраты на разработку проекта рассчитываются по следующей формуле:

$$S_{PI} = S_{3PI} + S_{HAK}, \quad (1)$$

где S_{3PI} – затраты по заработной плате инженера-программиста;

S_{HAK} – накладные расходы.

Затраты по заработной плате инженера-программиста рассчитываются по формуле:

$$S_{3PI} = O_{3PI} * (1 + \omega_c) * (1 + \omega_d) * tp_i, \quad (2)$$

где O_{3PI} – основная заработная плата инженера-программиста за месяц (20000 руб.);

tp_i – время, необходимое для разработки программы программистом i -го разряда (чел.-мес.);

ω_d – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

ω_c – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика.

Система разрабатывалась 30 дней, если учесть, что в одном месяце 25 рабочих дней, то:

$$tp_{12} = 30/25 = 1,2 \text{ (чел./Мес)}. \quad (3)$$

Таким образом, затраты по заработной плате инженера-программиста составят:

$$S_{3PI} = 10000 * (1 + 0,268) * (1 + 0,20) * 1,2 = 18258,52 \text{ (руб.)}$$

Накладные затраты рассчитываются с учетом ω_n – коэффициента, определяющего уровень накладных расходов организации по формуле:

$$S_{HAK} = O_{3PI} * \omega_n * tp_{12}, \quad (4)$$

$$S_{HAK} = 20000 * 1,35 * 1,2 = 32400 \text{ (руб.)}.$$

Таким образом, суммарные затраты на разработку системы составляют:

$$S_{PI} = 18258,52 + 32400 = 50658,52 \text{ (руб.)}.$$

2.3 Расчет цены разработанной программы

Оптовая цена разработанной программы определяется по следующей формуле:

$$Z_{\Pi} = S_{\Pi\Pi} + \Pi, \quad (5)$$

где Z_{Π} – оптовая цена (цена разработчика) (руб.);

$S_{\Pi\Pi}$ – суммарные затраты на разработку программы (руб.);

Π – прибыль, рассчитанная по формуле:

$$\Pi = \rho_n * S_{\Pi\Pi}, \quad (6)$$

где ρ_n – норматив рентабельности, учитывающий прибыль организации, разрабатывающей данную программу в долях ко всем затратам данной организации на разработку программы.

Итак,

$$Z_{\Pi} = S_{\Pi\Pi} * (1 + \rho_n) \quad (7)$$

$$Z_{\Pi} = 50658,52 * (1 + 0,24) = 62815,92 \text{ (руб.)}.$$

Розничная цена программы рассчитывается с учетом налога на добавленную стоимость (НДС = 20%) по формуле:

$$Z_{\Pi p} = Z_{\Pi} + H_{\text{НДС}} = Z_{\Pi} + (Z_{\Pi} * 20) / 100 = Z_{\Pi} * (1 + 0,20), \quad (8)$$

$$Z_{\Pi p} = 62815,92 * (1 + 0,20) = 75379,1 \text{ (руб.)}.$$

Выручка от продаж при условии пп – количество организаций, желающих приобрести программу, составит:

$$B = Z_{\Pi p} * n_n \quad (9)$$

$$B1 = 75379,1 * 4 = 301516,41 \text{ (руб.)},$$

$$B2 = 75379,1 * 5 = 376895,52 \text{ (руб.)},$$

$$B3 = 75379,1 * 8 = 603032,83 \text{ (руб.)}.$$

2.4 Расчет капитальных вложений

Капиталовложения, связанные с работой ЭВМ рассчитываются по формуле:

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		50

$$K_{ЭВМ} = C_{ЭВМ} + S_3, \quad (10)$$

где $C_{ЭВМ}$ – стоимость ЭВМ (руб.);

S_3 – стоимость запасных частей (руб.).

Итак, произведем расчет коэффициентов, входящих в формулу расчета величины капиталовложений:

$$S_3 = 0,15 * C_{ЭВМ}; \quad (11)$$

$$S_3 = 0,15 * 70000 = 10500 \text{ (руб.)};$$

Капиталовложения в ЭВМ и телефона составляет:

$$K_{ЭВМ} = 70000 + 10500 = 80500 \text{ (руб.)}.$$

2.5 Расчет эксплуатационных расходов

Эксплуатационные расходы на ЭВМ рассчитываются по формуле:

$$E = (T_{MB} * E_ч) + Zn/Tc, \quad (12)$$

где T_{MB} – машинное время для решения задач с помощью разработанной программы, (маш.час/год);

$E_ч$ – эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ;

Zn – цена, по которой продается программа (руб.);

Tc – срок службы программы (г).

Полезный фонд времени работы ЭВМ рассчитывается по формуле:

$$T_{ПОЛ} = T_{ОБЩ} * t_{CM} * N_{CM} * (1 - \alpha/100), \quad (13)$$

где $T_{ОБЩ}$ – общий фонд времени работы ЭВМ (дни); $T_{ОБЩ} = T_P$; N_{CM} – количество смен работы ЭВМ;

t_{CM} – время одного рабочего дня (час);

α – простои ЭВМ (в % от общего фонда времени работы ЭВМ). Полезный фонд времени работы ЭВМ получим:

$$T_{ПОЛ} = 264 * 8 * 1 * (1 - 10/100) = 1900,8 \text{ (Маш. час /год)}.$$

Машинное время для решения задач с помощью данной программы рассчитывается по формуле:

$$T_{MB} = qI * t_{MBI}, \quad (14)$$

где qI – количество I -задач, решаемых потребителем в год(шт.);

t_{MBI} – время решения I -ой задачи, разработанной программой (Маш. час).

$$T_{MB} = 60 * 1 = 60 \text{ (Маш. час / год)}.$$

Эксплуатационные расходы, приходящиеся на 1 час работы ЭВМ, оцениваются по формуле:

$$E_{\text{ч}} = (A_o + S_{\text{зн}} + S_{\text{эл}} + R_{\text{pm}}) / T_{\text{пол}}, \quad (15)$$

где A_o – амортизационные отчисления (руб.);

$S_{\text{зп}}$ – затраты по заработной плате инженера в год (руб./год);

$S_{\text{эл}}$ – стоимость потребляемой энергии (руб.);

R_{pm} – затраты на ремонт ЭВМ (руб.);

$T_{\text{пол}}$ – полезный годовой фонд работы ЭВМ, (маш.час/год).

Амортизационные отчисления рассчитываются с учетом нормы амортизации ($A_n = 12,5 \%$);

$$A_o = A_n * K_{\text{ЭВМ}} = 0,125 * 80500 = 10062 \text{ (руб.)}. \quad (16)$$

Затраты по заработной плате инженера за год рассчитывается по формуле:

$$S_{\text{зп}} = (1 + \omega_{\text{с}}) * (1 + \omega_{\text{д}}) * O_{\text{зп}} * 12, \quad (17)$$

где $\omega_{\text{с}}$ – коэффициент, учитывающий начисления органам социального страхования на заработную плату разработчика программы, в долях к сумме основной заработной плате разработчика;

$\omega_{\text{д}}$ – коэффициент, учитывающий дополнительную заработную плату разработчика программы, в долях к сумме основной заработной платы;

$O_{\text{зп}}$ – основная заработная плата инженера за месяц 8-го разряда (20000 руб.).

Рассчитываем годовые затраты по заработной плате и социальным отчислениям для инженера:

$$S_{\text{зп}} = (1 + 0,268) * (1 + 0,20) * 20000 * 12 = 365184 \text{ (руб./ год)}$$

Стоимость потребляемой энергии оценивается по формуле:

$$S_{\text{эл}} = P * T_{\text{пол}} * Z_{\text{эл}}, \quad (18)$$

где P – мощность, потребляемая ЭВМ (кВт);

$T_{пол}$ – полезный годовой фонд работы ЭВМ (маш.час/год);

$Z_{эл}$ – тариф за 1 кВт/час (руб. /кВт).

Итак, произведем расчет стоимости потребляемой энергии:

$$S_{эл} = 0,4 * 1900,8 * 3,44 = 2202,7 \text{ (руб.)}.$$

Затраты на ремонт ЭВМ вычисляются по формуле:

$$R_{PM} = N_{CP} * S_D, \quad (19)$$

где N_{CP} – среднее количество ремонтов в год;

S_D – стоимость деталей, заменяемых при одном ремонте, в среднем.

$$R_{PM} = 2 * 2500 = 5000 \text{ (руб.)}.$$

Произведем вычисление эксплуатационных расходов, приходящихся на 1 час работы ЭВМ:

$$E_q = (10062 + 365184 + 2202,7 + 5000) / 1900,8 = 201 \text{ (руб./час)}.$$

Далее вычислим эксплуатационные расходы на ЭВМ:

$$E = (60 * 201) + 62815,92 / 5 = 24653,184 \text{ (руб.)}.$$

2.6 Определение расходов на интернет трафик

Для облегчения работы программиста и повышения его производительности в проекте предусмотрено использование глобальной сети интернет.

В настоящее время на пензенском рынке интернет очень много провайдеров предлагающие свои услуги. Цены, предлагаемые основными провайдерами, сведем в таблицу 2, для анализа и экономически целесообразного выбора поставщика услуги по интернет трафику.

Таблица 2 — Таблица интернет трафиков и провайдеров

Название провайдера	Предлагаемый тариф	Стоимость тарифа(руб.)	Скорость доступа(Мбит/с)
Ростелеком	Пакет-Для Впечатлений	699	100
МТС	Безлимитище	450	100
Дом.РУ	Скорость света	550	100

Т.к. вышеперечисленные провайдеры предлагают безлимитные интернет трафики с одинаковой скоростью доступа в сеть интернет, то экономически целесообразно выбрать трафик с самой низкой стоимостью. Выбираем ПАО "МТС", тариф "Безлимитище", с абонентской платой в месяц 450 рублей.

2.7 Техничко-экономические показатели разработки

Обобщенные технико-экономические показатели разработки программы сведены в таблицу 3

Таблица 3 — Техничко-экономические показатели разработки

Показатель	Значение
Капитальные вложения (руб.)	80500
Эксплуатационные расходы (руб.)	24653,184
Затраты на разработку ПО (руб.)	50658,52
Затраты на интернет трафик (руб.)	450
Затраты на заработную плату программисту (руб.)	30432
Суммарно (руб.)	186693,704

3 Безопасность жизнедеятельности. Эргономика рабочего места программиста

3.1 Характеристика условий труда программиста

В наше время программистам приходится работать почти со всей электроникой, окружающей нас, начиная от компьютеров и телефонов и заканчивая умными холодильниками и беспилотными машинами. Из-за этого всего от программистов требуется увеличенная трата физической и умственной энергии. Что бы не допускать травм или других калечащих эффектов. Необходимо комплексно решать вопросы в сферах оборудования рабочих мест, контроля уровня шума, светом и воздушной средой.

В наши дни работа с ПК и телефонами интенсивно применяется во всех сферах работы человека. Такая деятельность подвергает человека влиянию ряда опасных электромагнитных излучений (СВЧ, УВЧ, ВЧ), шума, статики и других.

Взаимодействие с ПК и телефонами сопровождается увеличенной умственной нагрузкой на пользователя, постоянной зрительной работой и работой с клавиатурой. Несомненная важность имеет правильное оборудование и размещение рабочих мест пользователей и разработчиков.

Также в процессе своей деятельности работникам следует помнить о правильном соблюдении режимов труда и отдыха. При пренебрежении данного режима у рабочих могут возникать различные симптомы, выражающиеся в головных болях, нарушениях сна, раздражительности, усталости, болях в пояснице, шее и руках.

Одними из критериев, по которым определяется пригодность рабочих пространств работников, являются требования к параметрам воздушной среды, требования к уровню шума и вибрации, требования к освещенности рабочих мест, требования к производственному оборудованию и требования к использованию телефонами.

					<i>ПГУ 1.090501.10.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		55

3.2 Требования к параметрам воздушной среды

Микроклимат может меняться в широких диапазонах, необходимым условием для работы является поддержание постоянной температуры тела благодаря свойству терморегуляции, способности организма регулировать отдачу тепла в окружающей среде.

Для рабочих помещений допустимые и оптимальные значения параметров микроклимата приведены в таблице 4.

Таблица 4 — Климат рабочих помещений

Период года	Зона	Температура воздуха, С	Скорость движения воздуха, м/с	Относительная влажность воздуха, %
Холодный	Оптимальная	18-21	до 0,1	60-40
Переходный	Допустимая	17-21	до 0.3	до 75
Теплый ($t > 100$ С)	Оптимальная	20-25	до 0.3	60-40
Жаркий	Допустимая	До 28 в 13 часов самого жаркого месяца	до 0.5	до 75

Воздух, находящийся в рабочих помещениях работников ЭВМ, должен быть профильтрован от пыли, загрязнений и микроорганизмов.

Температура воздуха в помещении должна регулироваться с учетом теплового потока от оборудования. Предпочтительным выбором оборудования должна быть техника с малой электрической мощностью. Оборудование техники должно быть установлено так, чтобы тепловой поток не был направлен на рабочих. Следует ограничивать количество техники, находящейся в помещении.

Пол в помещениях должен быть ровный, из нескользкого материала, удобным для чистки и влажной уборки. Покрытия рабочих пространств должны обладать антистатическими свойствами и сохранять свои параметры в процессе эксплуатации. В помещениях должна проводиться влажная уборка.

3.3 Требования к уровню шума и вибрации

Шум оказывает не благоприятные последствия на труд человека. При длительной работе при шуме человек может начать испытывать раздражительность, головные боли, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах.

Так же под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением, ухудшается речевая коммуникация.

Главными источниками шума в рабочих комнатах с установленными в них ЭВМ являются принтеры, вентиляторы, кондиционеры, вентиляционные системы, трансформаторы, множительная техника.

Уровень шума на рабочем месте программистов не должен превышать 50 дБА. Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами. Уровень вибрации в помещениях вычислительных центров может быть снижен путем установки оборудования на специальные виброизоляторы. Существует возможность использования акустических подвесных потолков.

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		57

Оборудование, уровень издаваемого шума которого превышает допустимые нормы, необходимо монтировать вне рабочих помещений.

3.4 Требования к монтажу производственного оборудования

Для обеспечения электропитанием ЭВМ и компьютеров в комнатах должна быть смонтирована отдельная сеть электроснабжения.

Периферийные устройства компьютеров Должны подключаться исключительно к линии электроснабжения компьютерной сети.

Запрещается включение в линии электроснабжения компьютерной сети энергопотребителей, не относящихся к компьютерам (вентиляторы, лампы и т.п.).

Запрещается использовать оборудование, не подключенное к защитному заземлению, если оно предусмотрено конструкцией оборудования.

Запрещается использовать технику с открытыми корпусами, если это не предусмотрено правилами работы этого оборудования.

Проведение обслуживания рабочих мест программистов должно проводиться специально подготовленным персоналом, имеющим необходимую квалификацию техника (инженера).

Электрические розетки системы электропитания необходимо располагать так, чтобы кабели электрооборудования на рабочих местах не пересекали его, мешая работе.

3.6 Требования к освещению рабочих мест

Одним из важнейших параметров для оптимальных условий труда является уровень световой среды помещения. Это рационализация организации совокупности естественного и искусственного освещения на

рабочих местах.

Осуществление искусственного освещения комнат с рабочими местами должно осуществляться с помощью световых проемов, ориентированных на север, либо северо-восток.

В том случае, если ориентация производится в другие стороны, необходимо предусмотреть наличие средств регулирования интенсивности поступающего естественного освещения.

3.6 Расчет освещения рабочего места программиста

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения. Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

- по спектральному составу света они близки к дневному, естественному свету;
- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
- более длительный срок службы.

Расчет освещения производится для комнаты площадью 15м², ширина которой 5м, высота - 3 м. Воспользуемся методом светового потока. Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

$$F = E * K * S * Z / n, \quad (20)$$

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		59

где F - рассчитываемый световой поток, Лм;

E - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет $E = 300 \text{ Лк}$;

S - площадь освещаемого помещения (в нашем случае $S = 20 \text{ м}^2$);

Z - отношение средней освещенности к минимальной (обычно принимается равным 1,1...1,2, пусть $Z = 1,2$); K - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае $K = 1,5$);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; характеризуется коэффициентами отражения от стен (PC) и потолка (PII), значение коэффициентов PC и PII были указаны выше: $PC = 40\%$, $PII = 60\%$. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = S / h * (A + B), \quad (21)$$

где S - площадь помещения, $S = 20 \text{ м}^2$;

h - расчетная высота подвеса, $h = 3 \text{ м}$;

A - ширина помещения, $A = 4 \text{ м}$;

B - длина помещения, $B = 5 \text{ м}$.

Подставив значения получаем:

$$I = 20/3 * (4+5) = 60.$$

Зная индекс помещения I , по таблице находим $n = 0,22$.

Подставим все значения в формулу для определения светового потока F :

$$F = 300 * 1,5 * 20 * 1,2 / 0,22 = 49090 \text{ Лм}.$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световойпоток которых $F = 4320$ Лк.

Рассчитаем необходимое количество ламп по формуле:

$$N = F/F_{\text{л}}, \quad (22)$$

где N - определяемое число ламп;

F - световой поток, $F = 49090$ Лм;

$F_{\text{л}}$ - световой поток лампы, $F_{\text{л}} = 4320$ Лм.

$$N = 49090/4320 = 11 \text{ штук.}$$

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		61

Заключение

В процессе выполнения данной выпускной квалификационной работы проведены анализ требований, проектирование и разработка, тестирование Интернет магазина с использованием *react* технологий.

В процессе анализа требований, были сформированы требования к системе. Проведён анализ вариантов использования. Разработана модель предметной области, представленная диаграммой классов. В результате выполнения выпускной квалификационной работы реализован Интернет магазина с использованием *react* технологий.

Проект реализован в среде разработки *Visual Studio Code* с использованием сторонней библиотеки *react*, в качестве дополнительного инструмента разработки. На основе анализа, полученных результатов, можно сделать вывод, что все функции приложения отрабатываются корректно в штатном режиме.

Таким образом, требования технического задания были выполнены в полном объёме и в установленные сроки.

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

Список использованных источников

1. Основы интернет-магазина [Электронный ресурс]: Режим доступа: <https://docs.yandex.ru/docs/view?tm=1655106109&tld=ru&lang=ru&name> (дата обращения 01.05.22).
2. Определение интернет-магазина Википедия [Электронный ресурс]: Режим доступа: [https://ru.wikipedia.org/wiki/Интернет-магазин#:~:text=Интернет-магазин%20\(англ.%20online%20shop%20или,u%20доставки%20заказа%20С%20оплатить%20заказ.](https://ru.wikipedia.org/wiki/Интернет-магазин#:~:text=Интернет-магазин%20(англ.%20online%20shop%20или,u%20доставки%20заказа%20С%20оплатить%20заказ.) (дата обращения 20.05.22).
3. Классификация интернет магазинов [Электронный ресурс]: Режим доступа: <https://saitrus.ru/web-studio/blog/razrabotka-sajtov/klassifikatsiya-internet-magazinov> (дата обращения 29.05.22).
4. Основные библиотеки и фреймворки *JavaScript* [Электронный ресурс] URL: <https://code.tutsplus.com/ru/articles/essential-javascript-libraries-and-frameworks-you-should-know-about--cms-29540> (дата обращения 11.05.22)
5. *React.js*: понятное руководство для начинающих [Электронный ресурс]: Режим доступа: <https://habr.com/ru/company/ruvds/blog/> (дата обращения 01.05.22).
6. Архитектура модульных *React + Redux* приложений [Электронный ресурс]: Режим доступа: <https://habr.com/ru/post/326484/> (дата обращения 01.05.22).

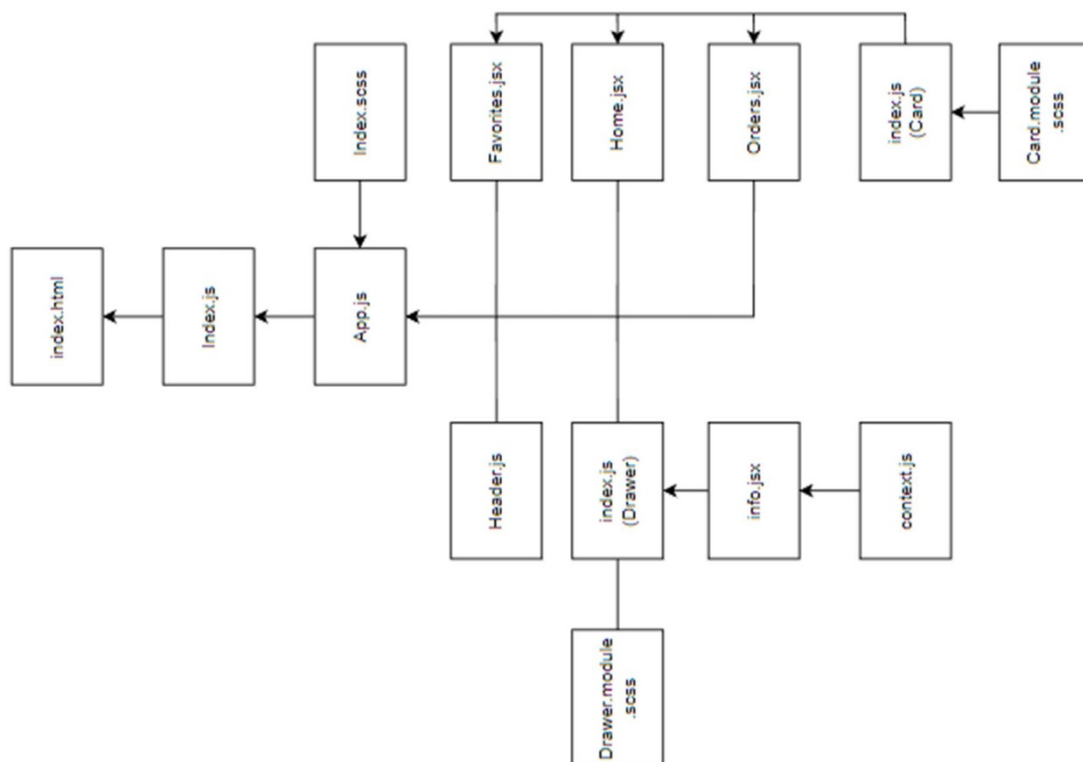
Приложение А

(обязательное)

Интернет-магазин с использованием *react* технологий

UML-диаграммы

					ПГУ 1.090501.10.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		



ПГУ 1.090501.10.006 Д18									
Программа, материал и оборудование для проведения УИЕ-обучения									
ПГУ, кафед. БТ, зр. 17802									

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.10.001 ПЗ

Лист

69

Приложение Б

(обязательное)

Интернет-магазин с использованием *react* технологий

Листинг программы

					ПГУ 1.090501.10.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

```

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-
scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta name="description" content="Web site created using
create-react-app" />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png"
/>

    <!--

      manifest.json provides metadata used when your web app is
      installed on a

      user's mobile device or desktop. See
      https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC_URL% in the tags above.

      It will be replaced with the URL of the `public` folder
      during the build.

      Only files inside the `public` folder can be referenced
      from the HTML.

```

Unlike `"/favicon.ico"` or `"favicon.ico"`,
`"%PUBLIC_URL%/favicon.ico"` will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running ``npm run build``.

-->

```
<link rel="preconnect" href="https://fonts.gstatic.com" />
```

```
<link
```

```
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700;800&display=swap"
```

```
rel="stylesheet"
```

```
/>
```

```
<title>React Sneakers</title>
```

```
</head>
```

```
<body>
```

```
<noscript>You need to enable JavaScript to run this app.</noscript>
```

```
<div id="root"></div>
```

```
<!--
```

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

```
import React from 'react';
```

```
import ContentLoader from 'react-content-loader';
```

```
import { ReactComponent as Btn } from '../assets/btn-plus.svg';
```

```
import { ReactComponent as Checked } from '../assets/btn-checked.svg';
```

```
import {ReactComponent as LikedIcon} from  
'../assets/liked.svg'
```

```
import {ReactComponent as UnlikedIcon} from  
'../assets/unliked.svg'
```

```
import AppContext from '../context';
```

```
import styles from './Card.module.scss';
```

```
function Card({
```

```
  id,
```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

```

    title,

    imageUrl,

    price,

    onFavorite,

    onPlus,

    favorited = false,

    loading = false,
  )) {

    const { isItemAdded } = React.useContext(AppContext);

    const [isFavorite, setIsFavorite] = React.useState(favorited);

    const obj = { id, parentId: id, title, imageUrl, price };

    const onClickPlus = () => {

      onPlus(obj);

    };

    const onClickFavorite = () => {

      onFavorite(obj);

      setIsFavorite(!isFavorite);

    };

    console.log(process.env.PUBLIC_URL+imageUrl);

    return (

```

					ИГУ 1.090501.10.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

```

<div className={styles.card}>

  {loading ? (

    <ContentLoader

      speed={2}

      width={155}

      height={250}

      viewBox="0 0 155 265"

      backgroundColor="#f3f3f3"

      foregroundColor="#e3e3e3">

        <rect x="1" y="0" rx="10" ry="10" width="155"
height="155" />

        <rect x="0" y="167" rx="5" ry="5" width="155"
height="15" />

        <rect x="0" y="187" rx="5" ry="5" width="100"
height="15" />

        <rect x="1" y="234" rx="5" ry="5" width="80"
height="25" />

        <rect x="124" y="230" rx="10" ry="10" width="32"
height="32" />

      </ContentLoader>

    ) : (

      <>

        {onFavorite && (

          <div className={styles.favorite}
onClick={onClickFavorite}>

```

```

        { /* <img src={isFavorite ? 'img/liked.svg' :
'img/unliked.svg'} alt="Unliked" /> */}

        {isFavorite ? <LikedIcon/>: <UnlikedIcon/>}

    </div>

    )}

    <img width="100%" height={135}
src={process.env.PUBLIC_URL+'/'+imageUrl} alt="Sneakers" />

    <h5>{title}</h5>

    <div className="d-flex justify-between align-center">

        <div className="d-flex flex-column">

            <span>Цена:</span>

            <b>{price} руб.</b>

        </div>

        {onPlus && (

            //<Btn onClick={onClickPlus}
className={styles.plus}/>

            isItemAdded(id) ? <Checked/> : <Btn
onClick={onClickPlus} className={styles.plus}/>

            // <img

            //   className={styles.plus}

            //   onClick={onClickPlus}

            //   src={isItemAdded(id) ? 'img/btn-checked.svg'
: 'img/btn-plus.svg'}

            //   alt="Plus"

            // />

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		76

```

        })

        </div>

    </>

    })

</div>

);

}

export default Card;

import React from 'react';

import axios from 'axios';

import Info from '../Info';

import { useCart } from '../../hooks/useCart';

import styles from './Drawer.module.scss';

import { ReactComponent as RemoveBtn } from '../../assets/btn-remove.svg';

import { ReactComponent as Arrow } from
'../../assets/arrow.svg';

import { ReactComponent as EmptyCart } from
'../../assets/arrow.svg';

import emptycart from '../../assets/empty-cart.jpg';

```

```

const delay = (ms) => new Promise((resolve) =>
setTimeout(resolve, ms));

function Drawer({ onClose, onRemove, items = [], opened }) {

  const { cartItems, setCartItems, totalPrice } = useCart();

  const [orderId, setOrderId] = React.useState(null);

  const [isOrderComplete, setIsOrderComplete] =
React.useState(false);

  const [isLoading, setIsLoading] = React.useState(false);

  const onClickOrder = async () => {

    try {

      setIsLoading(true);

      const { data } = await axios.post('/orders', {

        items: cartItems,

      });

      setOrderId(data.id);

      setIsOrderComplete(true);

      setCartItems([]);

      for (let i = 0; i < cartItems.length; i++) {

        const item = cartItems[i];

        await axios.delete('/cart/' + item.id);

        await delay(1000);

      }

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		78

```

    } catch (error) {

        alert('Ошибка при создании заказа :(');

    }

    setIsLoading(false);

};

return (

    <div className={` ${styles.overlay} ${opened ?
styles.overlayVisible : ''}`}>

        <div className={styles.drawer}>

            <h2 className="d-flex justify-between mb-30">

                Корзина

                <RemoveBtn onClick={onClose} className="cu-p"/>

                {/*  */}

            </h2>

            {items.length > 0 ? (

                <div className="d-flex flex-column flex">

                    <div className="items flex">

                        {items.map((obj) => (

                            <div key={obj.id} className="cartItem d-flex
align-center mb-20">

                                <div

```

```

        style={{ backgroundImage:
`url (${process.env.PUBLIC_URL+'/' +obj.imageUrl})` }}

        className="cartItemImg"></div>

<div className="mr-20 flex">

    <p className="mb-5">{obj.title}</p>

    <b>{obj.price} pyб.</b>

</div>

<RemoveBtn onClick={() => onRemove(obj.id)}
className="removeBtn" alt="Remove"/>

{/* <img

    onClick={() => onRemove(obj.id)}

    className="removeBtn"

    src="img/btn-remove.svg"

    alt="Remove"

/> */}

</div>

)}}

</div>

<div className="cartTotalBlock">

    <ul>

        <li>

            <span>Итого:</span>

            <div></div>

            <b>{totalPrice} pyб. </b>

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		80


```

        }

        image={isOrderComplete ? 'img/complete-order.jpg' :
'../../assets/empty-cart.jpg'}

        />

    )}

</div>

</div>

);
}

```

```

export default Drawer;

import React from 'react';

import logo from '../assets/logo.png';

import { Link } from 'react-router-dom';

import { ReactComponent as Cart } from '../assets/cart.svg';

import { ReactComponent as Heart } from '../assets/heart.svg';

import { ReactComponent as User } from '../assets/user.svg';

import { useCart } from '../hooks/useCart';

function Header(props) {

    const { totalPrice } = useCart();

    return (

```

```

<header className="d-flex justify-between align-center p-
40">

  <Link to="/">

    <div className="d-flex align-center">

      <img width={40} height={40} src={logo} alt="Logotype"
/>

      <div>

        <h3 className="text-uppercase">React Sneakers</h3>

        <p className="opacity-5">Магазин лучших
кроссовок</p>

      </div>

    </div>

  </Link>

  <ul className="d-flex">

    <li onClick={props.onClickCart} className="mr-30 cu-p">

      <Cart width={18} height={18}/>

      {/*  */}

      <span>{totalPrice} руб.</span>

    </li>

    <li className="mr-20 cu-p">

      <Link to="/favorites">

        <Heart width={18} height={18}/>

        {/*  */}

      </Link>

```

```

        </li>

        <li>

            <Link to="/orders">

                <User width={18} height={18}/>

                { /*  */ }

            </Link>

        </li>

    </ul>

</header>

);

}

export default Header;

import React from 'react';

import AppContext from '../context';

import { ReactComponent as Arrow } from '../assets/arrow.svg';

const Info = ({ title, image, description }) => {

    const { setCartOpened } = React.useContext(AppContext);

    return (

        <div className="cartEmpty d-flex align-center justify-center
flex-column flex">

```

```

        <img className="mb-20" width="120px" src={image}
alt="Empty" />

        <h2>{title}</h2>

        <p className="opacity-6">{description}</p>

        <button onClick={() => setCartOpened(false)}
className="greenButton">

            <Arrow alt="Arrow"/>

            { /*  */ }

            Вернуться назадд

        </button>

    </div>

    );

};

export default Info;

import React from 'react';

import AppContext from '../context';

export const useCart = () => {

    const { cartItems, setCartItems } =
React.useContext(AppContext);

    const totalPrice = cartItems.reduce((sum, obj) => obj.price +
sum, 0);

    return { cartItems, setCartItems, totalPrice };

```

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		85

```

};

import React from 'react';

import Card from '../components/Card';

import AppContext from '../context';

function Favorites() {

  const { favorites, onAddToFavorite } =
    React.useContext(AppContext);

  return (

    <div className="content p-40">

      <div className="d-flex align-center justify-between mb-40">

        <h1>Мои закладки</h1>

      </div>

      <div className="d-flex flex-wrap">

        {favorites.map((item, index) => (

          <Card key={index} favorited={true}
            onFavorite={onAddToFavorite} {...item} />

        ))}

      </div>

    </div>

  );

}

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

```

export default Favorites;

import React from 'react';

import { ReactComponent as Search } from '../assets/search.svg';

import { ReactComponent as Clear } from '../assets/btn-
remove.svg';

import Card from '../components/Card';

function Home({
  items,
  searchValue,
  setSearchValue,
  onChangeSearchInput,
  onAddToFavorite,
  onAddToCart,
  isLoading,
}) {
  const renderItem = () => {
    const filteredItems = items.filter((item) =>
      item.title.toLowerCase().includes(searchValue.toLowerCase()),
    );
    return (isLoading ? [...Array(8)] : filteredItems).map((item,
      index) => (

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

<Card

  key={index}

  onFavorite={ (obj) => onAddToFavorite(obj) }

  onPlus={ (obj) => onAddToCart(obj) }

  loading={isLoading}

  {...item}

/>

));

};

return (

  <div className="content p-40">

    <div className="d-flex align-center justify-between mb-40">

      <h1>{searchValue ? `Поиск по запросу: "${searchValue}"` : 'Все кроссовки'}</h1>

      <div className="search-block d-flex">

        <Search alt="Search" className="mt-10"/>

        {/*  */}

        {searchValue && (

          <Clear onClick={() => setSearchValue('')}
className="clear cu-p" alt="Clear" />

          // <img

          //   onClick={() => setSearchValue('')}

          //   className="clear cu-p"

```

```

        //   src="img/btn-remove.svg"

        //   alt="Clear"

        // />

    })

    <input onChange={onChangeSearchInput}
value={searchValue} placeholder="Поиск..." />

    </div>

</div>

    <div className="d-flex flex-wrap">{renderItems()}</div>

</div>

);

}

export default Home;

import React from 'react';

import axios from 'axios';

import Card from '../components/Card';

import AppContext from '../context';

function Orders() {

    const { onAddToFavorite, onAddToCart } =
React.useContext(AppContext);

    const [orders, setOrders] = React.useState([]);

    const [isLoading, setIsLoading] = React.useState(true);

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		89

```

React.useEffect(() => {

  (async () => {

    try {

      const { data } = await
axios.get('https://60d62397943aa60017768e77.mockapi.io/orders');

      setOrders(data.reduce((prev, obj) => [...prev,
...obj.items], []));

      setIsLoading(false);

    } catch (error) {

      alert('Ошибка при запросе заказов');

      console.error(error);

    }

  })();

}, []);

return (

  <div className="content p-40">

    <div className="d-flex align-center justify-between mb-
40">

      <h1>Мои заказы</h1>

    </div>

    <div className="d-flex flex-wrap">

```

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		90

```

        {(isLoading ? [...Array(8)] : orders).map((item, index)
=> (
            <Card key={index} loading={isLoading} {...item} />
        ))}
    </div>
</div>

);
}

```

```

export default Orders;

import React from 'react';

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

import axios from 'axios';

import Header from './components/Header';

import Drawer from './components/Drawer';

import AppContext from './context';

```

```

import Home from './pages/Home';

import Favorites from './pages/Favorites';

import Orders from './pages/Orders';

```

```

function App() {

    const [items, setItems] = React.useState([]);

```

					ИГУ 1.090501.10.001 ПЗ	Лист
						91
Изм.	Лист	№ докум.	Подпись	Дата		

```

const [cartItems, setCartItems] = React.useState([]);

const [favorites, setFavorites] = React.useState([]);

const [searchValue, setSearchValue] = React.useState('');

const [cartOpened, setCartOpened] = React.useState(false);

const [isLoading, setIsLoading] = React.useState(true);

React.useEffect(() => {

    async function fetchData() {

        try {

            const [cartResponse, favoritesResponse, itemsResponse] =
await Promise.all([

axios.get('https://60d62397943aa60017768e77.mockapi.io/cart'),

axios.get('https://60d62397943aa60017768e77.mockapi.io/favorites
'),

axios.get('https://60d62397943aa60017768e77.mockapi.io/items'),

]);

            setIsLoading(false);

            setCartItems(cartResponse.data);

            setFavorites(favoritesResponse.data);

            setItems(itemsResponse.data);

        } catch (error) {

            alert('Ошибка при запросе данных :(');

```

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		92

```

        console.error(error);

    }

}

fetchData();

}, []);

const onAddToCart = async (obj) => {

    try {

        const findItem = cartItems.find((item) =>
Number(item.parentId) === Number(obj.id));

        if (findItem) {

            setCartItems((prev) => prev.filter((item) =>
Number(item.parentId) !== Number(obj.id)));

            await
axios.delete(`https://60d62397943aa60017768e77.mockapi.io/cart/${
findItem.id}`);

        } else {

            setCartItems((prev) => [...prev, obj]);

            const { data } = await
axios.post('https://60d62397943aa60017768e77.mockapi.io/cart',
obj);

            setCartItems((prev) =>

                prev.map((item) => {

                    if (item.parentId === data.parentId) {

                        return {

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		93

```

        ...item,

        id: data.id,

    };

    }

    return item;

  })),

);

}

} catch (error) {

    alert('Ошибка при добавлении в корзину');

    console.error(error);

}

};

const onRemoveItem = (id) => {

    try {

        axios.delete(`https://60d62397943aa60017768e77.mockapi.io/cart/${id}`);

        setCartItems((prev) => prev.filter((item) =>
        Number(item.id) !== Number(id)));

    } catch (error) {

        alert('Ошибка при удалении из корзины');

        console.error(error);

    }
}

```

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

};

const onAddToFavorite = async (obj) => {

  try {

    if (favorites.find((favObj) => Number(favObj.id) ===
Number(obj.id))) {

      axios.delete(`https://60d62397943aa60017768e77.mockapi.io/favori
tes/${obj.id}`);

      setFavorites((prev) => prev.filter((item) =>
Number(item.id) !== Number(obj.id)));

    } else {

      const { data } = await axios.post(

        'https://60d62397943aa60017768e77.mockapi.io/favorites',

        obj,

      );

      setFavorites((prev) => [...prev, data]);

    }

  } catch (error) {

    alert('Не удалось добавить в фавориты');

    console.error(error);

  }

};

```

```
const onChangeSearchInput = (event) => {
```

					ПГУ 1.090501.10.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

        setSearchValue(event.target.value);

    };

    const isItemAdded = (id) => {

        return cartItems.some((obj) => Number(obj.parentId) ===
Number(id));

    };

    return (

        <AppContext.Provider

            value={{

                items,

                cartItems,

                favorites,

                isItemAdded,

                onAddToFavorite,

                onAddToCart,

                setCartOpened,

                setCartItems,

            }}>

            <div className="wrapper clear">

                <Drawer

                    items={cartItems}

                    onClose={() => setCartOpened(false)}

```

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		96

```

        onRemove={onRemoveItem}

        opened={cartOpened}

    />

    <Header onClickCart={() => setCartOpened(true)} />

    <Switch>

        <Route path="/" exact>

            <Home

                items={items}

                cartItems={cartItems}

                searchValue={searchValue}

                setSearchValue={setSearchValue}

                onChangeSearchInput={onChangeSearchInput}

                onAddToFavorite={onAddToFavorite}

                onAddToCart={onAddToCart}

                isLoading={isLoading}

            />

        </Route>

        <Route path="/favorites" exact>

            <Favorites />

        </Route>

        <Route path="/orders" exact>

```

					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		97

```

        <Orders />

    </Route>

</Switch>

</div>

</AppContext.Provider>

);

}

export default App;

import React from 'react';

const AppContext = React.createContext({});

export default AppContext;

import React from 'react';

import { BrowserRouter as Router } from 'react-router-dom';

import ReactDOM from 'react-dom';

import './index.scss';

import 'macro-css';

import App from './App';

```

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		98

```

ReactDOM.render(

  <React.StrictMode>

    <Router>

      <App />

    </Router>

  </React.StrictMode>,

  document.getElementById('root'),

);

```

					ПГУ 1.090501.10.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

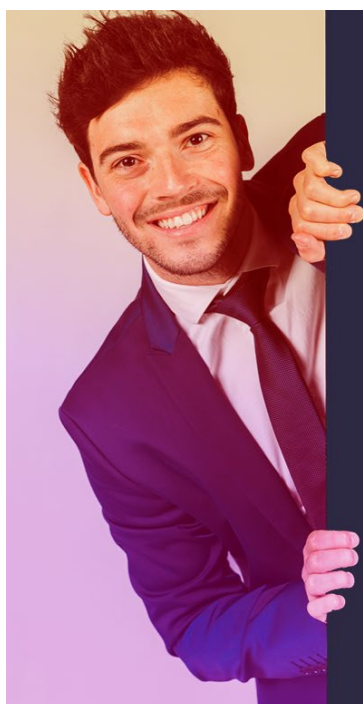
Интернет-магазин с использованием *react* технологий

Презентация

					ПГУ 1.090501.10.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

ИНТЕРНЕТ-МАГАЗИН С ИСПОЛЬЗОВАНИЕМ REACT ТЕХНОЛОГИЙ

Разработал студент группы
17B02 Петров Михаил



ВСТУПЛЕНИЕ

Интернет-магазин по
продаже кроссовок был
разработан с целью
демонстрации
возможности создания
пользовательских
интерфейсов

					ПГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		101

ОСНОВНЫЕ РАЗДЕЛЫ ПРЕЗЕНТАЦИИ

001

ПРОБЛЕМА VS РЕШЕНИЯ

Describe here this section

002

НАШ ПРОДУКТ

Describe here this section

003

НАШИ ПЛАНЫ

Describe here this section

004

ДЕМОНСТРАЦИЯ ПРОДУКТА

Describe here this section

ПРОБЛЕМА



Venus has a beautiful name and is the second planet from the Sun, but its atmosphere is extremely poisonous

КОНЕКУРЕНТЫ

Сайт с устаревшим не-
юзабельным интерфейсом

МЫ

Качественный
продуманный сайт с
дружественным
интерфейсом

РЕШЕНИЕ



В качестве решения предлагается
разрабатывать пользовательские
интерфейсы на react

REACT SNEAKERS

Крупнейший магазин в
мире по продаже
брендовых кроссовок



COMPANY LOGO
Магазин продажи кроссовок

SWOT ANALYSIS

S

STRENGTHS

Saturn is composed of
hydrogen and helium

W

WEAKNESSES

Mercury is the closest
planet to the Sun

O

OPPORTUNITIES

Venus is the second
planet from the Sun

T

THREATS

Despite being red, Mars
is actually a cold place

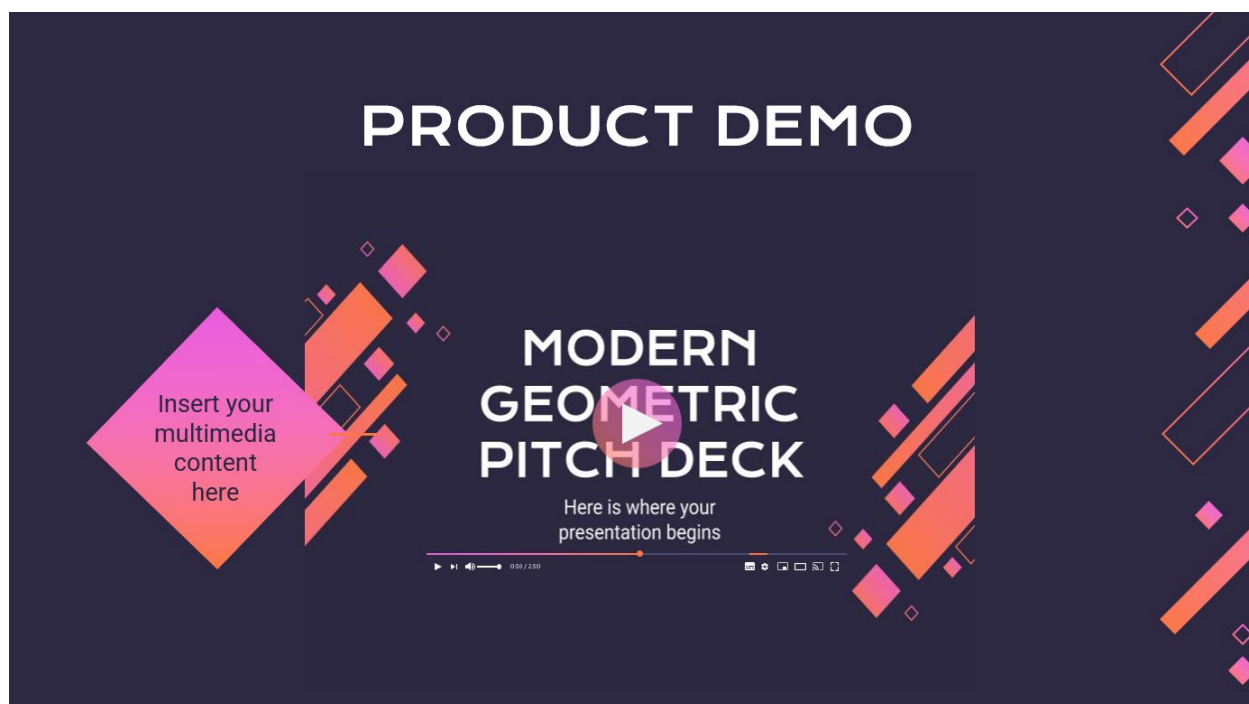
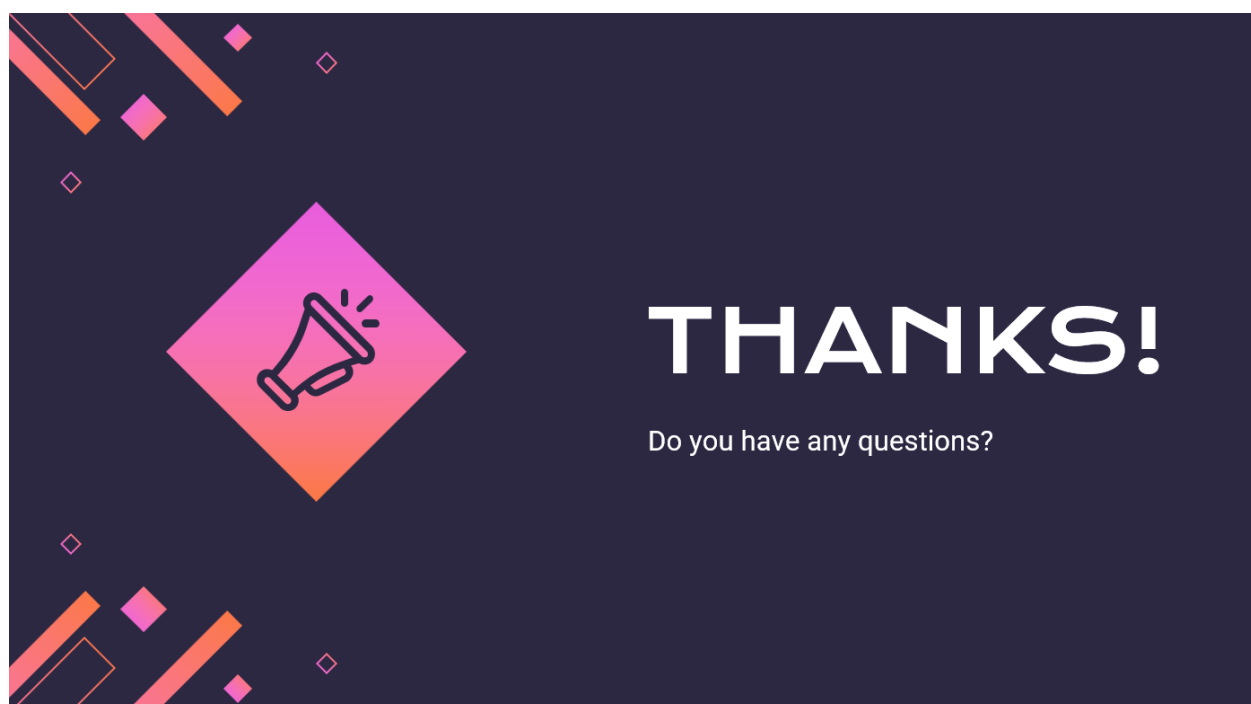
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.10.001 ПЗ

Лист

104





					ИГУ 1.090501.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		106

					<i>ПГУ 1.090501.10.001 ПЗ</i>	<i>Лист</i>
						<i>107</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		