**Введение**

**Тема проекта:** темой проекта выбрана «Разработка веб-сайта для продажи офисной техники и сетевого оборудования компании».

**Цель проекта:** разработать веб-сайт для продажи офисной техники и сетевого оборудования компании.

**Какую цель решает:** утилизация устаревшего или ненужного оборудования.

В настоящее время, работая в крупном call-центре, столкнулись с проблемой избавления от устаревшего и ненужного для целей компании оборудования. В связи с переоснащением компьютерной техники, а также с реорганизацией офисного пространства компании в нескольких офисах освободилось большое количество материальных ценностей в виде офисных системных блоков, мониторов и сетевого оборудования в виде серверов, различных конфигураций.

По этим причинам необходимо найти удобное, современное, общедоступное решение для реализации большого числа наименований техники с различными характеристиками.

**Задачи:**

1. Проанализировать рынок на предмет имеющихся инструментов по ликвидации оборудования (продажа/утилизация) и принятие решения по выбору варианта реализации оборудования.
2. Анализ и требования предъявляемые к разрабатываемому веб-сайту.
3. Разработка веб-сайта по продаже офисной техники и сетевого оборудования компании.
4. Заключение выводов по проделанной работе, поиск вариантов по улучшению и масштабированию продукта.

**Инструменты для разработки:** Visual Studio Code, Java Script, CSS, React, GitHub.

**Состав команды:** Сотников Андрей Валентинович (Frontend-разработчик, Дизайнер, Верстальщик).

**Инструменты разработки**

**Выбор среды разработки**

В первую очередь необходимо определиться со средой разработки программного продукта (IDE). IDE - IDE, или Integrated Development Environment — интегрированная среда разработки, набор ПО для создания кода. В него входят специальный редактор для кодинга и ряд инструментов, которые помогают запускать, тестировать и отлаживать код [1].

На рынке можно найти множество современных интегрированных сред разработки (IDE) с различными возможностями и разной ценой. Многие IDE имеют открытый исходный код или являются бесплатными для использования и настройки. Ниже приведены некоторые критерии, которые следует учитывать при выборе IDE[2].

Так проект является учебным, то основным критерием при выборе среды разработки является ее бесплатность. Помимо этого при выборе IDE нужно учитывать следующее[2]:

1. Язык программирования

Язык программирования, на котором вы хотите писать, часто диктует выбор IDE. Специализированные IDE имеют функции автоматизации, которые особенно подходят для синтаксиса конкретных языков. С другой стороны, мультиязычные IDE поддерживают несколько языков.

1. Операционная система

Хотя большинство IDE имеют несколько версий для различных операционных систем, они могут лучше работать на определенных платформах. Например, некоторые IDE могут оптимально работать на платформе Linux, но могут быть медленными или сложными в использовании на других платформах.

1. Автоматизация функций

Три общие функции большинства IDE – это редактор исходного кода, автоматизация сборки и отладчик. Дополнительные функции могут варьироваться и могут включать следующее:

* Улучшения пользовательского интерфейса редактора кода
* Автоматическое тестирование функций
* Поддержка развертывания кода с помощью интеграции плагинов
* Поддержка рефакторинга кода
* Поддержка упаковки приложений

1. Настройка IDE

Некоторые IDE включают возможность настройки рабочих процессов в соответствии с потребностями и предпочтениями разработчика. Вы можете загружать и использовать плагины, расширения и дополнения, чтобы настраивать свой процесс программирования.

Языком программирования будет использоваться JavaScript.

Написать код JavaScript можно и в обычном «Блокноте» — достаточно сохранить результат с расширением .js, и он не только подключится к сайту, но даже заработает. Однако использование такого ограниченного редактора сильно замедлит разработку. Поэтому лучше взять что-то специализированное. Есть несколько основных критериев, по которым нужно выбирать редактор для работы с кодом на JavaScript[3].

Во-первых, нужна подсветка синтаксиса: названия, операторы и специальные символы должны отображаться по-разному, чтобы можно было быстро прочитать код.

Во-вторых, нужно автодополнение: когда вы вводите название функции или переменной, редактор должен предложить несколько вариантов команд, которые вы, вероятно хотите ввести. Желательно, чтобы он также показывал подсказки к выбранному компоненту — например, необходимые аргументы, тип переменной, краткое описание и так далее.

В-третьих, нужна возможность скрывать отдельные блоки. При работе над большими проектами это полезно, потому что не придётся скролить огромные фрагменты кода, чтобы найти нужный.

Желательно, чтобы подсвечивались ошибки — тогда вы сможете их сразу заметить и исправить.

Существует большое количество редакторов, которые соответствуют этим требованиям и подойдут для работы с JavaScript. Ниже представлены 5 инструментов[3]^

**WebStorm**

Оригинальный релиз: 2010

Тип: Интегрированная среда разработки (IDE)

Совместимость с системой: Рекомендуется последняя 64-разрядная версия Windows, macOS или Linux

WebStorm является одним из наиболее распространенных инструментов разработки для JavaScript и, вероятно, наилучшей на текущий момент IDE для фронтенд-разработки. Этот программный редактор специализируется на JavaScript и TypeScript. Представители компании JetBrains утверждают, что «WebStorm делает процесс разработки более приятным, автоматизируя рутинные задачи и облегчая решение сложных задач».

На данный момент WebStorm считается одним из самых интеллектуальных JavaScript-редакторов для создания веб-, мобильных, серверных и настольных приложений. Он поддерживает разработку фронтендов с использованием Angular, React или Vue.js; серверных или бэкенд-приложений с помощью Node.js и Meteor; мобильных приложений с Ionic, Cordova или React Native; а также настольных приложений с Electron. Эта IDE базируется на открытой платформе IntelliJ.

WebStorm является одним из наиболее поддерживаемых и регулярно обновляемых вариантов на рынке, что позволяет ему быть первым, кто представляет новые возможности. Называя его «самой интеллектуальной JavaScript IDE», разработчики отмечают его множество функций:

* Редактирование «на лету»,
* Безопасный рефакторинг,
* Отличная навигация,
* Завершение кода и качественный анализ,
* Быстрое документирование,
* Встроенный предварительный просмотр HTML,
* Широкий выбор плагинов и шаблонов,
* Интеграция с системой контроля версий,
* Встроенный HTTP-клиент,
* Встроенный терминал,
* Быстрая навигация и поиск,

Многие разработчики считают функции автоматической отладки и тестирования кода WebStorm лучшими на рынке, а также отмечают отличную функциональность интеллектуальных шаблонов.

Плюсы:

* Надежные обновления и поддержка пользователей,
* Возможность настройки с использованием плагинов,
* Бесплатность для студентов и проектов с открытым исходным кодом,
* Интеллектуальное автоисправление,
* Поставляется с предварительной интеграцией с Angular, TypeScript, Vue и React.

Минусы:

* Возможно потребление памяти устройства,
* Иногда медленная загрузка,
* Сложность отслеживания графиков и ребаз Git,
* Избранное и закладки могли бы быть более полезными.

**PyCharm**

Оригинальный релиз: 2014

Тип: Интегрированная среда разработки (IDE)

Совместимость с системами: Microsoft Windows 8 и выше, macOS 10.14 и выше, любой дистрибутив Linux с поддержкой Gnome, KDE или Unity DE (PyCharm недоступен для некоторых дистрибутивов Linux).

Хотя PyCharm не является лучшим бесплатным инструментом для разработки на JavaScript, платная версия Professional заслуживает внимания. Эта отличная среда разработки сайтов предназначена для программистов на Python, но она также отлично справляется с практически всеми популярными языками и фреймворками, включая JavaScript, AngularJS, CoffeeScript, CSS, HTML, Node.js, TypeScript и шаблонные языки.

Среди функциональных возможностей PyCharm:

* Завершение кода;
* Автоматический поиск ошибок;
* Интеллектуальная навигация по проекту;
* Встроенный отладчик;
* Интеграция с популярными системами контроля версий;
* Поддержка Jupyter Notebook, Anaconda и других библиотек.

Плюсы:

* Отличное завершение кода;
* Хорошие возможности отладки;
* Удобство работы с системами контроля версий (СКВ);
* Работа с базами данных;
* Хорошее автодополнение SQL;
* Работа с кодом на удаленной машине и его отладка;
* Автоматическое исправление зависимостей;
* Встроенная оболочка.

Минусы:

* Платный;
* Основное направление — Python;
* Иногда возникают проблемы при отладке.

**IntelliJ IDEA**

Оригинальный релиз: 2001

Тип: Интегрированная среда разработки (IDE)

Совместимость с системой: Рекомендуется последняя 64-разрядная версия Windows, macOS или Linux

IntelliJ IDEA — это Java-ориентированная IDE, разработанная компанией JetBrains с целью увеличения производительности разработчиков. Эта эргономичная и интеллектуальная среда способна автоматизировать повторяющиеся задачи программирования, сокращая время разработки. В момент своего выпуска в 2001 году, это была одна из первых Java IDE, интегрировавших функции, такие как навигация по коду и рефакторинг.

Одной из великолепных особенностей IntelliJ IDEA является автоматическое предоставление соответствующих предложений в любом контексте: мгновенное и продуманное завершение кода, анализ кода «на лету» и эффективные инструменты рефакторинга.

По словам представителей JetBrains, «цель IntelliJ IDEA — предоставить готовый опыт работы с критически важными инструментами, такими как интегрированные системы контроля версий и поддержка множества языков и фреймворков, без необходимости установки плагинов».

Хотя IntelliJ IDEA специализируется на Java, она также обеспечивает интеллектуальную помощь при написании кода для различных языков, включая JavaScript.

Особенности:

* Встроенные статические анализаторы кода
* Инструменты развертывания и отладки
* Расширенное завершение кода
* Интегрированный контроль версий
* Простая интеграция с Git
* Редактор баз данных
* UML-дизайнер с обширными возможностями

Плюсы:

* Интуитивно понятный пользовательский интерфейс
* Точное автодополнение на основе DOM-модели
* Функции доступности, включая программу чтения с экрана
* Встроенный контроль версий
* Множество плагинов и расширений для опробования
* Удобное автодополнение
* Отличные функции режима отладки

Минусы:

* Довольно медленное индексирование
* Крутая кривая обучения
* Бесплатная версия не поддерживает JavaScript
* Довольно дорогостоящая
* Потребление значительных ресурсов компьютера
* Довольно медленная встроенная консоль

**PhpStorm**

Оригинальный релиз: 2009

Тип: Интегрированная среда разработки (IDE)

Совместимость с системой: Рекомендуется последняя 64-разрядная версия Windows, macOS или Linux

PhpStorm представляет собой универсальную кроссплатформенную IDE, созданную компанией JetBrains. Хотя она изначально была разработана для работы с PHP, в ней реализован ряд выдающихся функциональностей, которые делают ее весьма эффективной IDE для JavaScript. Кроме того, расширенные возможности PhpStorm позволяют использовать различные плагины для еще большего функционала.

PhpStorm предоставляет нулевую конфигурацию для отладки, автоматический рефакторинг, предотвращение ошибок, подсветку синтаксиса и анализ кода «на лету». В ее функционал входит сотни проверок кода, которые ускоряют процесс исправления ошибок и повышают качество кода.

Эта IDE также оборудована встроенными средствами для тестирования, профилирования и предоставляет унифицированный интерфейс для работы с популярными системами контроля версий, такими как Git, Mercurial и Perforce. Кроме того, PhpStorm предлагает инструменты для эффективной работы с базами данных и языком SQL в рамках проекта.

Особенности:

* Помощь в работе с кодом (включая работу с базами данных и SQL)
* Автозавершение
* Подсветка синтаксиса
* Отладка
* Встроенное форматирование кода
* Подсказки для переменных и методов
* Удобная навигация по коду

Плюсы:

* Прост в использовании при отладке новой кодовой базы
* Эффективная интеграция с внешними инструментами, такими как Docker и XDebug, облегчающая работу с кодом
* Отличная поддержка плагинов
* Высокая чувствительность к обнаружению ошибок
* Простое подключение локальной базы данных к проекту на PHP
* Активный интеллект снижает количество допущенных ошибок при разработке приложения

Минусы:

* Может быть немного медленным и требовательным к ресурсам компьютера
* Иногда развертывание занимает много времени, но быстро отлаживается
* Несколько дороговат для новичков
* Отсутствует бесплатная версия, только 30-дневная пробная

**Visual Studio Code**

Оригинальный релиз: 2015

Тип: Редактор кода

Совместимость с системой: Рекомендуется последняя 64-разрядная версия Windows, macOS или Linux

Visual Studio Code представляет собой высокооптимизированный редактор кода от Microsoft, предназначенный для поддержки различных операций разработки, включая отладку, контроль версий и выполнение задач. Этот редактор является одним из наиболее популярных и надежных инструментов для работы с кодом на JavaScript.

В соответствии с информацией на сайте компании, Visual Studio Code нацелен на предоставление разработчику только тех инструментов, которые необходимы для быстрого цикла «сборка кода — отладка», оставляя более сложные рабочие процессы для более мощных IDE, таких как Visual Studio IDE.

Помимо поддержки JavaScript, Visual Studio Code обеспечивает возможности работы с более чем 30 языками программирования, включая C#, HTML5, PHP и Ruby.

Ключевые особенности Visual Studio Code включают:

* IntelliSense, предоставляющий разработчикам предложения по коду, подсказки и описания параметров;
* возможности рефакторинга кода;
* средства отладки;
* встроенные команды Git;
* функция in-line peak;
* Task Runner для создания и настройки задач с использованием Gulp, Grunt или MSBuild;
* стек вызовов, установка точек останова, следящие переменные.

Visual Studio Code легкий, легко настраиваемый, бесплатный и не требует высокопроизводительного компьютера. Он также обладает активным сообществом разработчиков и качественной технической поддержкой. Если вы готовы использовать плагины для расширения функционала VS Code, то этот инструмент вполне подходит для разработки на JavaScript.

Плюсы:

* Бесплатный
* Низкое потребление памяти
* Интеграция с консолью
* Управление задачами
* Поддержка JavaScript-фреймворков, включая React и Node.js
* Программа чтения с экрана для обеспечения доступности
* Расширяемость и настраиваемость

Минусы:

* Отсутствие поддержки некоторых популярных языков программирования
* Ограниченный контроль исходного кода
* Ограниченные встроенные инструменты и функции тестирования, но их можно добавить с помощью плагинов
* Крутая кривая обучения
* Иногда может быть медленным
* Требует времени для настройки с использованием расширений и плагинов

На основании вышеописанного, в качестве IDE выбор был сделан в пользу Visual Studio Code.

**Выбор языка для фронтенд разработчика.**

Фронтенд пишут при помощи трех ключевых технологий, но языком программирования считают только один из них — JavaScript.

**HTML**

Это язык гипертекстовой разметки. Он наверняка знаком по урокам информатики в школе. Раньше, когда ресурсы в интернете были простыми и представляли собой обычные страницы, его использовали для создания сайтов.

Но и сейчас HTML лежит в основе интернет-страниц, где отвечает за расположение различных элементов: таблиц, графики, текстовой информации и т.д.

HTML состоит из тэгов. Это команды, которые говорят браузеру, как правильно отображать содержимое.

Метод его работы прост. Когда вы вводите название сайта в адресную строку, браузер отправляет запрос к нужному ресурсу. В ответ он получает файл в формате HTML. А браузер преобразует набор команд в визуальное представление страницы.

Можно сказать, что HTML — каркас любого веб-документа и своеобразная азбука веб-разработчика. И фронтендеры, и бэкендеры обычно знают основы языка разметки.

**CSS**

Это каскадные таблицы стилей и вторая базовая технология в современной веб-разработке. Если HTML — это каркас сайтов, то CSS отвечает за внешний вид.

Сайты, написанные на «голом» HTML, имели четкую структуру, но были некрасочными и напоминали обычные документы со ссылками и таблицами. CSS позволил радикально улучшить их оформление.

Каскадные таблицы стилей подключают к основе на HTML, чтобы задать единые стандарты внешнего вида. Именно CSS дает возможность установить общий размер заголовков, цвета фона и текста, размер кнопок и т.д.

Некоторые из вариантов оформления можно задать и с помощью тэгов HTML, но это делать гораздо сложнее. А CSS позволяет один раз создать стандарты оформления и подключить их ко всем веб-страницам.

**JavaScript**

Это самый популярный и практически безальтернативный язык программирования для фронтенда современных сайтов. По индексу TIOBE на март 2024 JavaScript занимает шестое место. Этот рейтинг оценивает общую популярность языков программирования.

А вот согласно исследованию GitHub по итогам 2023 года JavaScript занимает первое место среди веб-разработчиков. На нем пишут 19% респондентов. Вместе с его вариацией TypeScript они занимают треть рынка.

Причина популярности проста — JavaScript позволяет создавать интерактивные сайты, то есть реагирующие на действия пользователя. До его появления нужно было перезагружать страницу, чтобы увидеть изменения.

Внедрение JavaScript стало настоящим прорывом в веб-разработке. Этот язык программирования работает на основе скриптов или сценариев, которые исполняет браузер. Например, пользователь может нажать на кнопку и активировать всплывающее окно. Через эту форму можно заполнить данные и фронтенд сайта самостоятельно их обработает, без запросов на сервер.

Это позволяет, с одной стороны, уменьшить время ожидания ответа, с другой — снизить нагрузку на трафик и аппаратные мощности.

Все, что вы видите в интерфейсе сайтов: кнопки, всплывающие формы, меню, автозаполнение данных — все это реализовали на основе JavaScript. Этот язык программирования применяют и для серверной (бэкенд) разработки, но значительно реже.

Для фронтендеров в JavaScript разработали множество фреймворков — каркасов приложений и библиотек — готовых участков кода. Знание одного из них — практически обязательное условия для трудоустройства. Самые популярные технологии: Angular, React и Vue.

JavaScript считают одним из относительно простых языков программирования. Его можно изучить примерно за год. А зарплаты фронтенд-разработчиков — одни из самых высоких в отрасли.

В отличие от фронтенда, у бэкенда выбор технологий гораздо богаче. Мы расскажем о самых популярных и востребованных языках для решения разных задач [4].

**PHP**

PHP один из старейших языков для веб-разработки. Его создали в 1995 году, и он до сих пор не сдает позиций — 76% сайтов в интернете написаны именно на нем. Это доминирование обусловлено в основном готовыми движками — CMS, например, WordPress.

В индексе TIOBE на март 2024 года PHP занимает двенадцатое место. А в исследовании GitHub — шестое место среди всех языков программирования и третье среди технологий для бэкенда.

Сейчас PHP позволяет создавать сайты самой разной сложности. От простых страниц на основе готовых конструкторов до таких монстров как Wikipedia, BlaBlaCar, Facebook или «Вконтакте».

PHP завоевал такую популярность за счет своей простоты и совместимости со всеми технологиями, включая распространенные базы данных и веб-сервера.

Этот язык программирования активно развивается, последняя версия 8.3 вышла в 2023 году. Также у него большое комьюнити, которое занимается поддержкой и продвижением технологии.

PHP — узкоспециализированный язык, его используют только для веб-разработки. Зато в этой сфере он отличается максимальной демократичностью: его применяют как в крупных проектах, так и при создании небольших коммерческих сайтов.

Самые популярные фреймворки для веб-разработки на PHP: Laravel и Symfony.

PHP, как и JavaScript, язык программирования с динамической типизацией. Это делает его менее надежным, зато более простым в освоении. Его считают одним из самых легких для новичков [4].

**Python**

В отличие от PHP, Python — один из самых многофункциональных языков программирования. Его используют не только в веб-разработке, но и в анализе данных, машинном обучении, создании десктопных приложений и даже игр.

По индексу TIOBE на март 2024 года Python занимает почетное первое место по популярности. А в исследовании GitHub — третье место среди всех языков программирования и четвертое среди технологий для бэкенда.

В веб-разработке его применяют для создания серверных приложений. Например, в Google на Python написали некоторые из компонентов поискового движка. Также на нем работает серверная часть Dropbox и YouTube.

Python — интерпретируемый язык программирования, то есть его не нужно компилировать для запуска. Файл с кодом можно запустить на любой машине, на которой установили интерпретатор. А за счет все той же динамической типизации код на нем лаконичный и не громоздкий.

За счет своей популярности Python относительно легко изучать, большинство ошибок и сложных моментов можно быстро найти в Google. У него обширное сообщество, которое готово помогать новичкам.

Еще один плюс этого языка программирования — многофункциональность. Веб-разработчику легко найти другой проект или вовсе сменить сферу, перейдя от создания серверных приложений, например, к машинному обучению.

У Python множество фреймворков и библиотек. Веб-разработчики обычно используют Django, Pyramid или Flask [4].

**Java**

Java — язык программирования для крупных корпораций, финансовых компаний и банков. На нем пишут бэкенд для высоконагруженных приложений с повышенными требованиями к безопасности.

По индексу TIOBE на март 2024 года Java занимает четвертое место по популярности. А в исследовании GitHub — второе место среди всех языков программирования и первое среди технологий для бэкенда.

На этом языке программирования написали серверные приложения для Amazon, Goldman Sachs и Netflix.

Его создала компания Oracle в 1995 году. Java задумывался как более «дружелюбная» альтернатива С и С++, в итоге став таким же надежным, но менее сложным.

Тем не менее, Java тяжелее освоить, чем другие популярные языки программирования для бэкенда: Python или PHP. Но это окупается большей надежностью кода.

Java — компилируемый язык. Программы на нем сначала переводят в байт-код, который затем исполняет виртуальная машина. За счет этого развивается хорошая скорость и повышается безопасность. Компилятор верифицирует код и проверяет на предмет ошибок.

Еще одна особенность Java — многословность. Код на нем занимает больше строк, а синтаксис строже и сложнее. Порог входа в этот язык программирования выше, чем у аналогов.

У Java есть несколько фреймворков, самый популярный и объемный — Spring [4].

**Ruby**

Это язык программирования общего назначения, но в основном его применяют для веб-разработки. Его создали в 1995 году с целью произвести на свет простой язык, с которым приятно работать. Но популярность Ruby завоевал только в 2005 году, когда появился фреймворк Ruby on Rails, предназначенный специально для веб-приложений.

По индексу TIOBE на март 2024 года Ruby занимает восемнадцатое место по популярности. А в исследовании GitHub — двенадцатое место среди всех языков программирования и восьмое среди технологий для бэкенда.

Ruby отличает лаконичный синтаксис и динамическая типизация. Этот язык программирования применяют для e-commerce и создания стартапов, так как на нем можно быстро реализовать сложные решения.

Его почти всегда используют в связке с фреймворком Ruby on Rails [4].

Исходя из приведенных выше сравнений, выбор сделан в пользу JavaScript. Так же немаловажным фактором являетсязарплата программитов.

По данным «Хабр Карьеры» медианные зарплаты программистов на JavaScript[5]:

* 72 000 рублей — у джунов
* 169 000 рублей — у миддлов
* 278 000 рублей — у сеньоров
* 308 000 рублей — у тимлидов

**Выбор JavaScript-фреймворка**

JavaScript-фреймворки — это абстракции, которые предоставляют предопределенные функции для ускорения процесса создания приложений, делая его более легким и простым для веб-дизайна. Фреймворки предназначены для обеспечения общей функциональности, которая может быть изменена кодом написанным пользователем. Это полезно при попытке разработать детальные пользовательские интерфейсы или управлять огромными объемами данных. Адаптивность фреймворков это ключевая причина для их выбора.

Как упоминалось выше, в последние годы JavaScript расширил свои возможности по работе с программированием на стороне сервера в дополнение к клиентскому. Он имеет 2 типа фреймворков:

* Front-end-фреймворки (сторона клиента)
* Back-end-фреймворки (сторона сервера)

Ниже, мы рассмотрим наиболее используемые Front-end-фреймворки и их преимущества и ограничения [6].

**Angular**

Angular это мощный JavaScript-фреймворк с открытым исходным кодом для разработки одностраничных приложений (SPA) [6].

Плюсы:

* двустороннее связывание данных;
* логика отделена от отображения при использовании MVC, и это нравится разработчикам;
* простые манипуляции с DOM; и
* возможность повторного использования компонентов.

Минусы:

* некоторые функции сложные для кастомизации; и
* может иметь сложные сценарии.

Особенности Angular:

* привязка данных для синхронизированного обновления представления и модели;
* директивы для добавления интерактивности;
* внедрение зависимостей для HTTP-запросов и передачи данных между компонентами;
* шаблоны для создания UI элементов;
* поддержка для создания прогресивных веб-приложений с современными веб-возможностями;
* механизм обнаружения изменений, который может обновлять определенные части приложения без перезагрузки всех компонентов, используя механизмы по умолчанию и OnPush.

**ReactJS**

React — это JavaScript-библиотека, используемая для создания пользовательских интерфейсов. Используется для одностраничных и мобильных приложений и известен своей высокой производительностью используя Virtual DOM, что сокращает время необходимое для повторного отображения представления при изменении состояния[6].

Плюсы:

* позволяет повторное использование компонентов;
* решает вопросы связанные с HTML для динамических веб-приложений, и использует виртуальный DOM для более быстрого рендеринга; и
* использует функциональные и классовые компоненты, у которых есть свои особенности.

Минусы:

* используется JSX, который может иметь крутую кривую обучения;
* JSX, привязка данных, и крючки являются обязательными.

Особенности React:

* декларативный UI для удобочитаемости и легкой отладки;
* виртуальный DOM для эффективного рендеринга UI;
* архитектура на основе компонентов для модульной разработки;
* cинтаксис разметки JSX для упрощения обслуживания компонентов React;
* высокая производительность благодаря односторонней привязке данных и элементам управления Flux;
* используется для создания нативных приложений в React native.

**Vue.js**

Vue.js — это прогрессивный JavaScript-фреймворк для создания пользовательских интерфейсов и одностраничных приложений. Он использует синтаксис шаблонов на основе HTML для рендеринга пользовательского интерфейса и поддерживает одностороннее связывание данных для высокой производительности и небольшого размера[6].

Плюсы:

* маленький размер, легко адаптируемый, и расширяется с помощью других библиотек;
* легкообучаем и использует простые HTML и JavaScript кодирующие конструкции; и
* реактивность создается API Composition, который не требует декоратора.

Минусы:

* небольшая система плагинов;
* языковой барьер, так как крупные комьюнити пользователей разрабатывают плагины и документацию на китайском языке, что делает его сложным для тех, кто не понимает языка; и
* излишняя гибкость может создавать риски и различные подходы к программированию в рамках одной команды.

Особенности Vue.js:

* декларативный рендеринг и архитектура на основе компонентов;
* основная библиотека разработана таким образом, чтобы быть простой в освоении и использовании, но при этом настраиваемой и расширяемой;
* Composition API для управления состоянием с помощью Redux без внешних зависимостей;
* однофайловый компонент для удобства использования во всем приложении; и
* встроенный компонент Transitions/Animations для создания эффектов в HTML-документах.

**Next.js**

Next.js — это фронтенд-фреймворк React для создания пользовательских интерфейсов и статических веб-сайтов. В нем предусмотрена автоматическая генерация страниц на основе файловой структуры и инкрементная статическая генерация для обновления статических страниц[6].

Плюсы:

* адаптивный дизайн — гибкость и восприимчивость ко всем мультимедийным устройствам;
* позволяет создавать статические веб-сайты быстро с динамическими функциональными возможностями; и
* Next.js отлично подходит для SEO, так как он предлагает рендеринг на стороне сервера.

Минусы:

* слабая система плагинов;
* время создания значительно больше при работе с несколькими страницами; и
* маршрутизация может быть ограничением.

Особенности Next.js:

* Static Generation для генерации и кэширования HTML-страниц во время сборки; и
* серверный рендеринг для генерации страниц с динамическим контентом при каждом запросе.

Исходя из перечисленных достоинств и недостатков принято решение использовать в работе React. ReactJS хорошо подходит для создания пользовательского интерфейса благодаря компонентной архитектуре и большому сообществу.

**Разработка веб-сайта**

**Подготовка среды для разработки**

Как оговаривалось в предыдущей главе, в качестве IDE будем использовать Visual Studio Code. Процесс установки не сложен. С офиициального сайта скачиваем нужный дистрибутив, согласно имеющейся операционной системы (Windows в моем случае) [7]. Устанавливаем ПО из скачанного файла. При необходимости производим настройки согласно FAQ [8].

Также для удобства разработки, как договорились выше, будем использовать фреймворк React. Для создания проекта необходимо сделать несколько шагов [9]:

* Установить **node** и **npm**
* Создать новую папку и открыть ее в редакторе кода
* Запустить в терминале команду npx create-react-app diplom\_sotnikov\_av, где diplom\_sotnikov\_av - название приложения

Node или Node.js — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и Espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом [10].

Npm — крупнейший в мире реестр программного обеспечения. Разработчики открытого исходного кода со всех континентов используют npm для обмена и заимствования пакетов, а многие организации используют npm для управления частной разработкой [11].

После этого скрипт создаст папку с названием, которое было указано, а также выполнит установку и настройку необходимых для работы React модулей. Кроме того, будет автоматически создан первый компонент в файле App.js. Для того чтобы увидеть результат работы скрипта, надо перейти в папку diplom\_sotnikov\_av и запустить в терминале команду npm start[9].

Для хранения и управления версиями будем использовать GitHub. Для чего создадим новый репозиторий и привяжем его к нашему локальному проекту, выполнив команды:

* git remote add origin <https://github.com/sav91987/Diplom.git>;
* git branch -M main;
* git push -u origin main.

В дальнейшем будем использовать следующие основные команды при работе с проектом:

* git add . – добавление новых изменений для отслеживания;
* git commit -m "some commit" – запись коммита;
* git push – отправка изменений на удаленный репозиторий.

Среда для разработки подготовлена, можно приступать к созданию веб-сайта.

**Компонент App.js**

Первый и основной компонент появляется на этапе создания проекта в React. Код компонента представлен на рисунке 1.

На строках 1-9 происходит ряд импортов:

* Импорт необходимых компоненты из 'react-router-dom' для маршрутизации.
* импорт CSS для стилизации App.js.
* импорт компонентов, которые будут описаны позднее (Cart, Catalog, MainPage, ProductPage, RegistrationPage) из соответствующих файлов.
* импортирует DataContext
* данные из файла JSON.

Строка 11 – функциональный компонент App. Это основной компонент приложения.

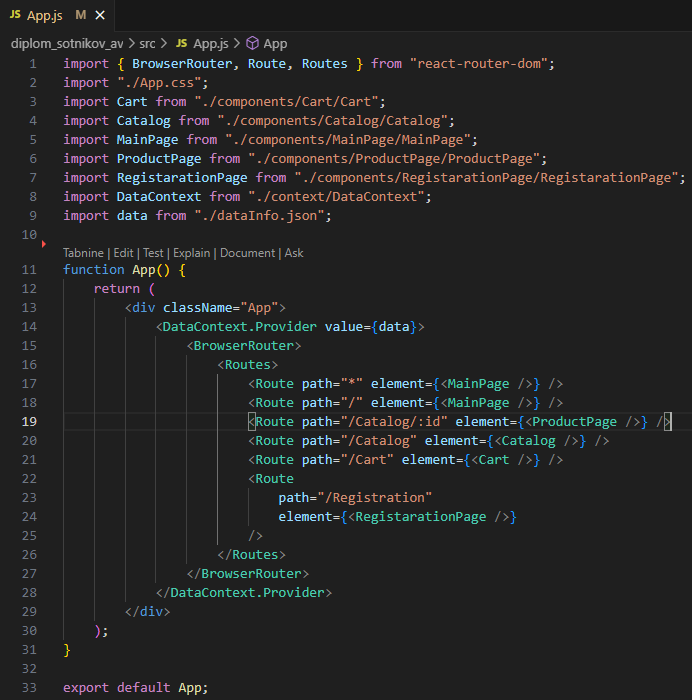


Рисунок 1 – Код компонента App.js

Строка 12 – оператор возврата - возвращает JSX, который определяет структуру приложения.

Строка 14 - DataContext.Provider - оборачивает все приложение, предоставляя данные всем дочерним компонентам через API контекста React. В качестве параметра используется data имортированная на строке 9. Здесь находятся данные по товарам. Это имитация прихода данных с сервера.

Строки с 15 по 27 – BrowserRouter - компонент включает функциональность маршрутизации в приложении. Маршруты и компоненты маршрута определяют конфигурацию маршрутизации для приложения. Каждый маршрут указывает путь path и соответствующий компонент для рендеринга.

Пути маршрутизации (path):

* "\*" и "/" оба отображают компонент MainPage.
* "/Catalog/:id" отображает компонент ProductPage (":id" — параметр).
* "/Catalog" отображает компонент Catalog.
* "/Cart" отображает компонент Cart.
* "/Registration" отображает компонент RegistrationPage.

Строка 33 - компонент App экспортируется как экспорт по умолчанию, что делает его доступным для использования в других частях приложения.

Этот код (рисунок 1) устанавливает базовую структуру для приложения React с возможностями маршрутизации, позволяя осуществлять навигацию между различными страницами/компонентами на основе пути URL. Данные из файла JSON становятся доступными во всем приложении с помощью React Context API.

Для стилизации компонента используется App.css. Код представлен в Приложении 1 – App.css.

**Компонент главной страницы сайта MainPage.js**

Скриншот главной страницы сайта представлен в приложении 2 – Главная страница сайта. Код компонента MainPage.js представлен на рисунке 2.

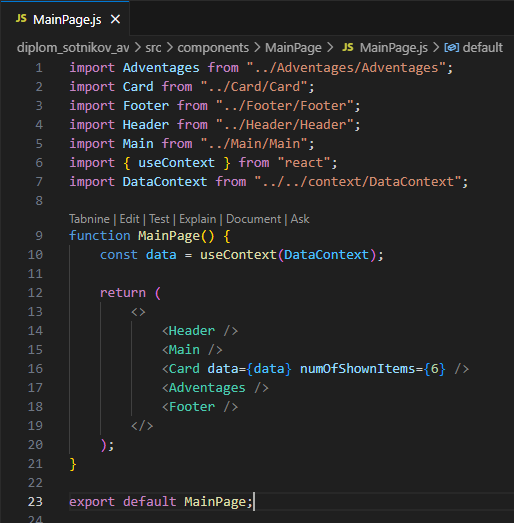


Рисунок 2 – Код компонента MainPage.js

Строки с 1 по 7 - импорт необходимых компонентов и хуков:

* компоненты Adventages, Card, Footer, Header и Main из соответствующих файлов
* хук useContext из React
* DataContext из файла контекста

Строки с 9 по 21 - функциональный компонент MainPage. Это основной компонент, представляющий структуру главной страницы.

Строка 10 - использование контекста. Эта строка использует хук useContext для доступа к данным из DataContext. Это позволяет компоненту использовать общие данные в приложении.

Код DataContext.js паредставлен на рисунке 3. Цель этого кода — создать общий контекст, который можно использовать во всем приложении. DataContext можно использовать для предоставления и потребления данных через различные компоненты без явной передачи props через каждый уровень дерева компонентов.

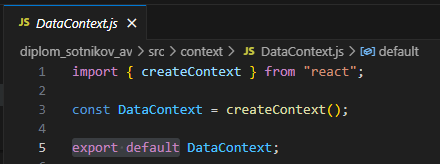


Рисунок 3 – Код DataContext.js

Строки с 12 по 20 возвращают структуру компонента в виде React- фрагмента (<>), содержащего несколько дочерних компонентов:

* <Header />: хэдэр страницы
* <Main />: основная область содержимого
* <Card data={data} numOfShownItems={6} />: компонент Card (карточка товара), который получает данные контекста и свойство для отображения 6 элементов
* <Adventages />: компонент отображющий блок с преимуществами компании
* <Footer />: футер страницы

Строка 23 экспортирует компонент MainPage как экспорт по умолчанию, что позволяет импортировать его и использовать в других частях приложения.

В целом, этот код устанавливает основную структуру страницы, используя различные компоненты и контекст для управления данными. Это типичная настройка для главной страницы.

Далее разберем отдельно каждый компонент, входящий в MainPage.js

**Компонент хедера сайта Header.js**

Код компонета Header.js представлен на рисунках 4 и 5. Это компонент отвечает за отображение шапки (header) на всех страницах сайта.

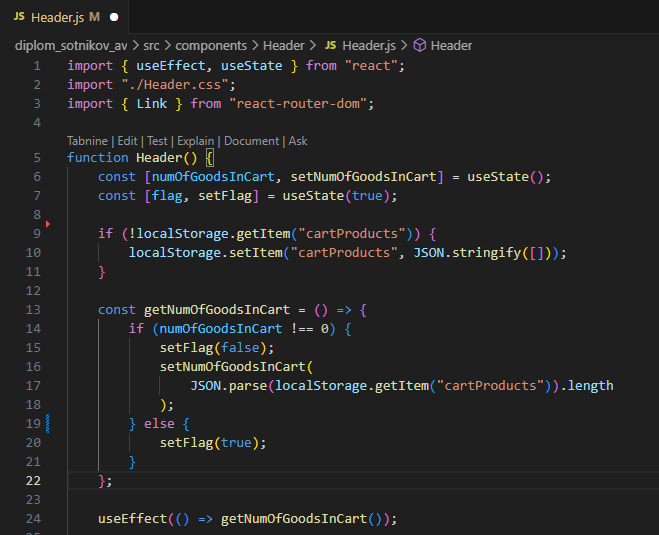


Рисунок 4 – Первая часть кода компонента Header.js

Строки 1-3 – импорты:

* useEffect и useState — хуки React для работы с состоянием и побочными эффектами.
* Link используется для навигации между страницами в приложении на React с использованием react-router-dom.
* Header.css — файл со стилями для компонента.

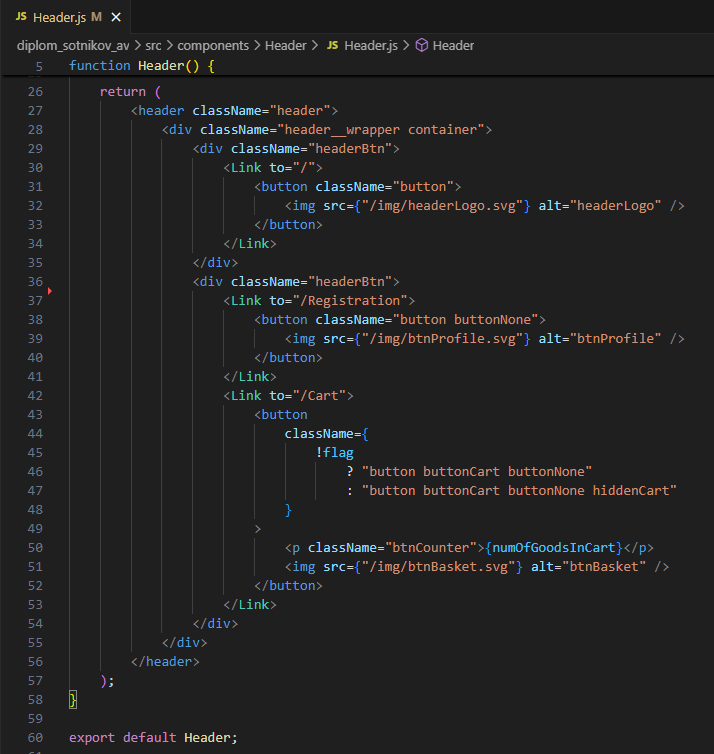


Рисунок 5 – Вторая часть кода компонента Header.js

Строки 6 и 7 определяют два состояния:

* numOfGoodsInCart — количество товаров в корзине.
* flag — логическое значение, которое используется для контроля отображения кнопки корзины.

Строки 9-11 - инициализация localStorage - проверяется, есть ли в localStorage данные о продуктах в корзине. Если нет, создается пустой массив cartProducts.

Строки с 13 по 22 - функция для получения количества товаров в корзине. Эта функция считывает количество товаров из localStorage и обновляет состояние numOfGoodsInCart. Если в корзине есть товары, flag устанавливается в false, иначе — в true.

Хук useEffect вызывается при первом рендере компонента, чтобы получить количество товаров в корзине.

Строки с 26 по 56 (рисунок 5) - возвращаемый JSX. Главный элемент — это <header>, который содержит кнопку с логотипом, кнопку для перехода на страницу регистрации и кнопку для перехода в корзину.

Кнопка корзины отображает количество товаров (numOfGoodsInCart) и меняет свой класс в зависимости от значения flag (если корзина пуста, она скрыта).

Link переадресует на главную страницу сайта.

Этот компонент обеспечивает функциональность отображения шапки сайта с кнопками для перехода на главную страницу, страницу регистрации и страницу корзины, а также отображает количество товаров в корзине.

Для стилизации компонента используется Header.css. Код представлен в Приложении 3 – Header.css.

**Компонент футера сайта Footer.js**

Код компонета Footer.js представлен на рисунках 6 и 7. Это компонент отвечает за отображение подвала (footer) на всех страницах сайта.

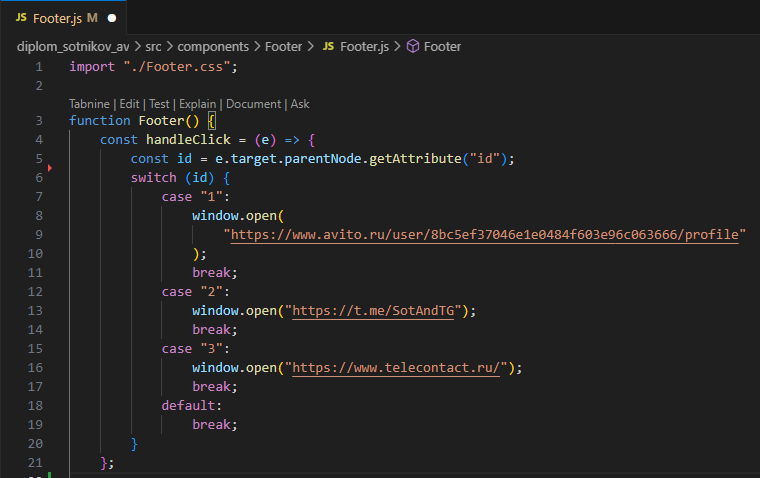


Рисунок 6 – Первая часть кода компонента Footer.js

Строка 1 - Импорт стилей: Этот код импортирует CSS-файл Footer.css

Строка 3 - Объявление компонента: Здесь мы определяем функциональный компонент Footer.

Строки с 4 по 21 - Обработчик клика. Функция handleClick вызывается при клике на кнопки с иконками. Она берет идентификатор кнопки (через parentNode, которая указывает на родительский элемент кнопки) и открывает соответствующую ссылку в новой вкладке.

switch (id): В зависимости от значения id, открывается соответствующий URL:

* id="1" → открывает Avito
* id="2" → открывает Телеграм
* id="3" → открывает сайт Telecontact



Рисунок 7 – Вторая часть кода компонента Footer.js

Строки с 23 по 60 - JSX-Разметка. Внутри компонента возвращается JSX-элемент, который представляет собой подвал страницы:

* <footer className="footer">: главный элемент подвала с классом для стилизации.
* <div className="footer\_\_wrapper container">: обертка для контента.
* <p className="footer\_\_text">: текст авторского права.
* <div className="footer\_\_icons">: контейнер для кнопок с иконками.

Каждый <button> элемент имеет идентификатор, обработчик клика и включает изображение для соответствующей социальной сети или сервиса.

Строка 63 - Экспорт компонента: Это позволяет использовать компонент Footer в других частях приложения.

Этот компонент создает подвал с кнопками для перехода по ссылкам, и при нажатии на каждую кнопку открывается новая вкладка с заданным URL. Стиль оформляется через внешний CSS файл.

Для стилизации компонента используется Footer.css. Код представлен в Приложении 4 – Footer.css.

**Компонент Main.js главной страницы сайта.**

Код компонента Main.js представлен на рисунке 8. Этот компонент рендерит основную секцию страницы с изображением, заголовками и кнопкой перехода к каталогу товаров. Вот подробнее о каждой части:

Строки 1 и 2 – импорты:

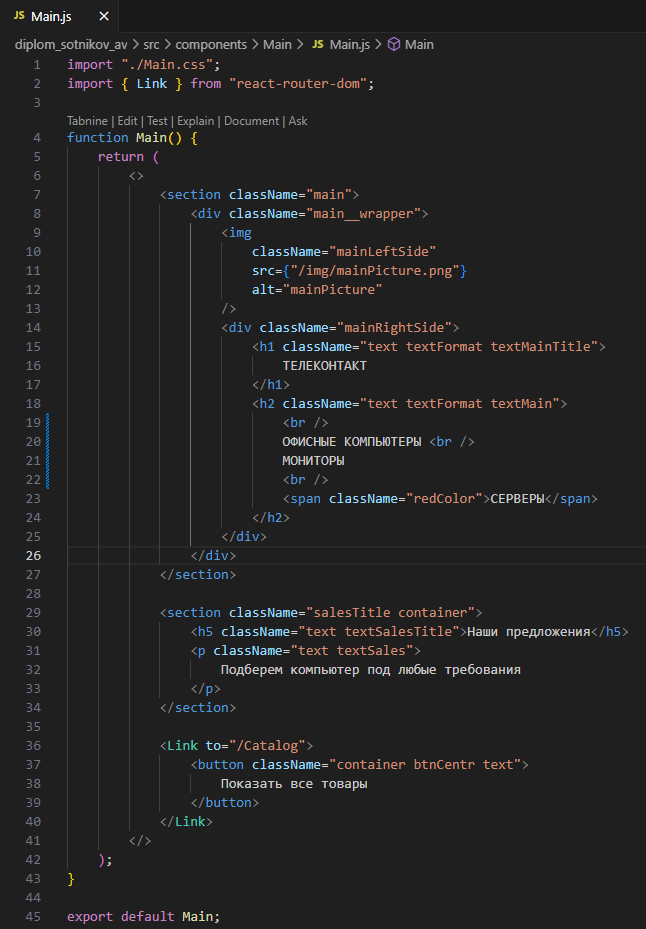


Рисунок 8 - Код компонента Main.js

* Импорт стилей: Здесь импортируется CSS-файл Main.css для стилизации компонента.
* Импорт Link: Компонент Link из библиотеки react-router-dom используется для создания ссылок между страницами приложения без перезагрузки.

Строка 4 - Определение компонента Main.

Строки 5-42 - Объявление компонента: Мы определяем функциональный компонент Main. Используется синтаксис фрагмента <> для группировки элементов без добавления дополнительных узлов в DOM.

Секция с классом main: В этом разделе размещается основное содержимое страницы.

Изображение mainPicture.png – главная картинка блока

Используются заголовки <h1> и <h2> для отображения названия "ТЕЛЕКОНТАКТ" и описания предложений, таких как "ОФИСНЫЕ КОМПЬЮТЕРЫ", "МОНИТОРЫ" и "СЕРВЕРЫ". Последний пункт выделен красным цветом с помощью класса redColor.

Секция salesTitle. Этот раздел представляет собой заголовок и описание, которые дают пользователю информацию о том, что предлагается в каталоге.

Заголовок и текст: Заголовок уровня h5 для "Наши предложения" и параграф с описанием.

Внутри компонента Link создается кнопка, которая перенаправляет пользователя на страницу каталога товаров (/Catalog) при нажатии.

Экспорт компонента позволяет импортировать и использовать компонент Main в других частях приложения.

Компонент Main создает основную визуальную часть главной страницы сайта с изображением, заголовками и кнопкой для перехода к каталогу товаров. Все элементы стилизуются с помощью CSS и организованы в секциях для лучшей структуры.

Для стилизации компонента используется Main.css. Код представлен в Приложении 5 – Main.css.

**Компонент Card.js**

Код компонета Card.js представлен на рисунках 9 и 10.



Рисунок 9 – Первая часть кода компонента Main.js



Рисунок 10 – Вторая часть кода компонента Main.js

Код представляет собой компонент для отображения карточек товаров с возможностью добавления их в корзину и перехода на страницу товара.

Строки 1 и 2 – импорты:

* Link: Импортируется из библиотеки react-router-dom для создания ссылок на другие страницы в приложении, сохраняя историю браузера и предотвращая полную перезагрузку страницы.
* Card.css: Импортируется файл стилей для стилизации компонентов.

Строка 4 - Определение компонента. Компонент Card принимает два пропса:

* data: Массив объектов товаров.
* numOfShownItems: Количество товаров, которое должно быть показано на странице.

Строки 5 и 6 - Подготовка данных. Создается копия массива данных data для работы с ним. Splice используется для обрезки массива с конца, оставляя только первые numOfShownItems элементов.

Строки с 8 по 11 - функция getCartArr отвечает за добавление товара в корзину. Она принимает один аргумент targetEl, который указывает на элемент (например, кнопку или изображение товара). Filter формирует массив с продуктами, у которых id соответствует id целевого элемента.

Строки с 13 по 25 - Работа с localStorage. Проверяется, есть ли в localStorage корзина (cartProducts). Если нет, создается новая корзина с добавляемым товаром. Если корзина уже существует, извлекаются её содержимое и добавляется новый товар. Устанавливает количество всех товаров в корзине по умолчанию равным 1. Использует Set для удаления дубликатов, сериализуя объекты для правильного сравнения.

Строки с 29 по 36 - Обработка клика по кнопке "В корзину". Функция addToCart срабатывает при нажатии кнопки добавления в корзину. Если клик был по изображению, передается родительский элемент, иначе — сам элемент. После добавления в корзину выводится алерт с сообщением "Добавлено в корзину!".

Строки с 38 по 48 - Обработка клика по карточке товара. Функция handleCardClick сохраняет информацию о выбранном товаре, чтобы отобразить его на странице продукта. При клике на элемент, находящий в теге id сравнивается с id товара в newData и в случае совпадения записывается в массив productArr.

Если в localStorage не хранится выбранный товар, он добавляется, а если уже есть, то сначала удаляется старая запись и добавляется новая.

Строки с 49 по 93 – Рендеринг. Возвращается JSX, который построен на основе данных newData. Создаются карточки с изображениями товаров, названиями, описаниями и ценами. Каждая карточка включает в себя:

* Изображение продукта.
* Ссылку для перехода на страницу деталей.
* Кнопку для добавления в корзину.

Строка 94 - Экспортирует компонент Card, что позволяет использовать его в других частях приложения.

Код создает функциональный компонент для отображения карточек товаров с возможностью их добавления в корзину и перехода на отдельные страницы продуктов. Это основы современных подходов к созданию интерфейсов в React с использованием состояния и localStorage для временного хранения данных.

Для стилизации компонента используется Card.css. Код представлен в Приложении 6 – Card.css.

**Компонент Adventages.js**

Код компонента Adventages.js представлен на рисунках 11.

Строка 1 - Импорт стилей. Этот импорт загружает CSS-файл, содержащий стили для компонента Adventages.

Строка 3 - Определение компонента. Здесь мы определяем функциональный компонент Adventages.

Строки с 4 по 40 - JSX-разметка. Компонент возвращает JSX-разметку. Вся разметка заключена в HTML-теги и соединяется, чтобы создать структуру контента. Здесь у нас есть секция (<section>), которая содержит весь контент компонента. Внутри секции есть div, который служит оберткой для всего контента. Класс advantages\_\_wrapper и container используются для задания определенных стилей. Каждое преимущество представлено в отдельной карточке (div с классом advantages\_\_card). Это обеспечивает простой способ стилизовать и организовать визуальное представление информации.

Строка 43 - экспортируется компонент Adventages, что позволяет использовать его в других частях приложения.

В целом, данный компонент Adventages реализует секцию с преимуществами. Он включает изображения, заголовки и текст описания, каждое преимущество представлено в отдельной карточке, что делает информацию более структурированной и визуально привлекательной.

Для стилизации компонента используется Adventages.css. Код представлен в Приложении 7 – Adventages.css.

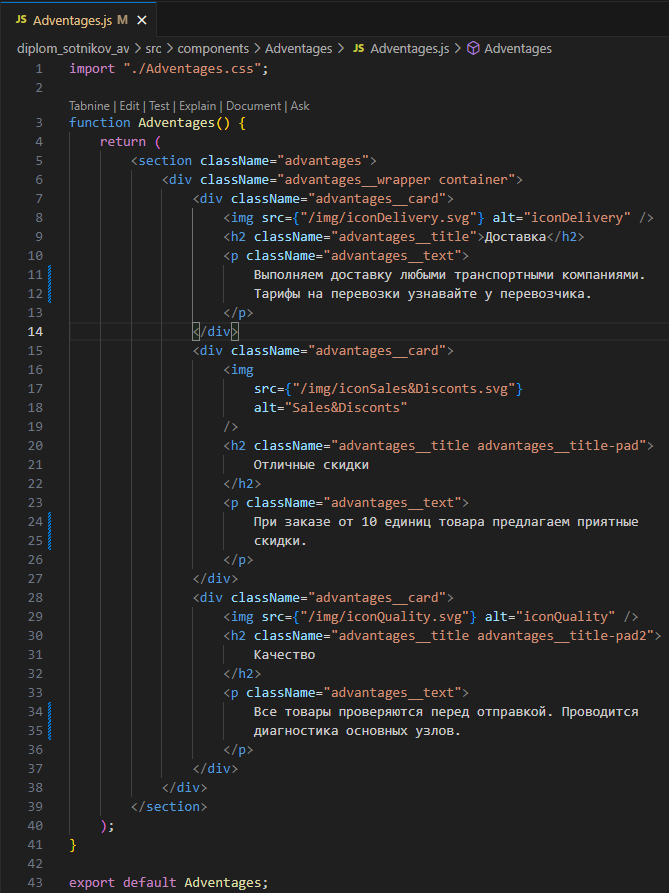


Рисунок 11 - Код компонента Adventages.js.

**Компонент каталога сайта Catalog.js**

Скриншот каталога сайта представлен в приложении 8. Код компонента Catalog.js представлен на рисунке 12.

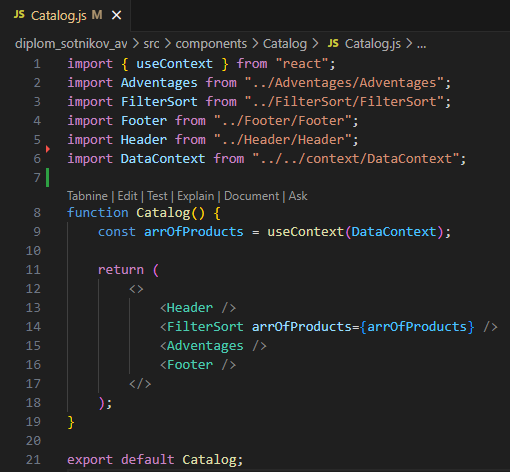


Рисунок 12 – Код компонента Catalog.js

Строки с 1 по 6 - Импортируемые модули:

1. useContext: Это хук из React, который позволяет использовать контекст API. Он дает доступ к значениям контекста в функциональных компонентах.
2. Компоненты:

* Adventages: компонент, который отображает преимущества компании, описывался ранее.
* FilterSort: Компонент для фильтрации и сортировки списка продуктов.
* Footer: подвал сайта, описывался ранее.
* Header: хэдер, описывался ранее.

1. DataContext: Это контекст, который предоставляет данные о продуктах, чтобы компонент FilterSort мог к ним обращаться.

Строка 8 - Определение компонента. Здесь мы определяем функциональный компонент Catalog.js.

Строка 9 - Здесь мы используем useContext для извлечения данных о продуктах (массив объектов) из DataContext. Это означает, что Catalog будет автоматически обновляться, когда данные в контексте изменяются.

Строки с 11 по 18 - В этом блоке возвращается JSX, который описывает структуру компонентов, которые будут отображаться на странице и были описаны выше.

Строка 21 - позволяет другим компонентам или модулям использовать Catalog.js.

Компонент Catalog является частью интерфейса интернет-магазина, где продукты могут быть отфильтрованы и отсортированы. Он использует контекст для управления доступом к данным о продуктах и включает основные компоненты интерфейса как хэдер, блок фильтрации, отображение преимуществ компании и футер сайта.

Далее разберем компонент FilterSort.js. Остальные компоненты были описаны выше.

**Компонент FilterSort.js.**

Код компонента FilterSort.js представлен на рисунке 13, 14, 15, 16 и 17.

Для сохранения места картинки svg на строчках 119 и 171 были свернуты.



Рисунок 13 – Первая часть кода компонента FilterSort.js.

Код компонента FilterSort используется для фильтрации и сортировки списка продуктов. Код работает с состоянием и обработкой событий, чтобы управлять отображением списка продуктов.

Строки 1-3 - Импортируемые модули

* useState: Хук, который позволяет добавлять состояние в функциональные компоненты React.
* "./FilterSort.css": Импорт стилей для обеспечения соответствующего оформления компонента.
* Card: Компонент, который отображает каждый элемент из списка отфильтрованных продуктов. Был описан ранее.

Строка 5 - Определение компонента. Здесь мы определяем функциональный компонент FilterSort.

Строка 6 нужна для перемотки страницы вверх после ее обновления.

Строки 7-9 – Состояния:

* const [sortOrder, setSortOrder] = useState(arrOfProducts) - Хранит отсортированные продукты. Изначально в нем все продукты arrOfProducts.
* const [selectedPrices, setSelectedPrices] = useState([]) - Хранит выбранные цены для фильтрации.
* const [arrSelectedCategory, setArrSelectedCategory] = useState([]) - Хранит выбранные категории для фильтрации.

Функция getIntersectionArr принимает два массива и параметр (param), чтобы отфильтровать продукты по цене или категории.

Если второй массив пуст, она просто устанавливает sortOrder равным первому массиву.

В зависимости от значения параметра (price или category), она фильтрует arr1 (массива продуктов) на основе значений arr2 (избранных фильтров) и обновляет состояние sortOrder.

Обработчик события handleClick предназначен для щелчков на чекбоксы, отвечающет за обновление выбранных размеров и фильтрацию по цене. Если чекбокс был активирован, выбранный размер добавляется в массив. Если отключен, то удаляется. Затем функция вызывает getIntersectionArr для фильт рации по цене в зависимости от выбранных категорий.

Обработчик handleClickCategory нужен для кликов по категориям. Если выбран элемент "ВСЕ ТОВАРЫ", состояние сбрасывается, и отображаются все продукты. В противном случае выбранная категория добавляется в массив и вызывается фильтрация. Этот обработчик также сбрасывает состояние чекбоксов по цене.

На рисунка 16 и 17 представлен возврат JSX. В возвращаемом JSX коде компонуются секции для отображения заголовка, фильтров и сортировок:

* <section className="top-head">: Содержит заголовок "КАТАЛОГ".
* <section className="filter-sort container">: Основной контейнер для фильтров:
* <details>: Содержит выпадающие элементы для выбора категорий.
* <div className="sort">: Содержит выпадающие элементы для выбора диапазонов цен с чекбоксами.
* <Card data={sortOrder} />: Компонент, который отображает отфильтрованные продукты. Передается массив sortOrder, который содержит текущие продукты после применения фильтров.



Рисунок 14 – Вторая часть кода компонента FilterSort.js.

FilterSort — это компонент, отвечающий за фильтрацию и сортировку списка продуктов на основе выбранных пользователем категорий и ценовых диапазонов. Код эффективно управляет состоянием и событиями, чтобы минимизировать количество отображаемых продуктов в зависимости от пользовательских предпочтений.

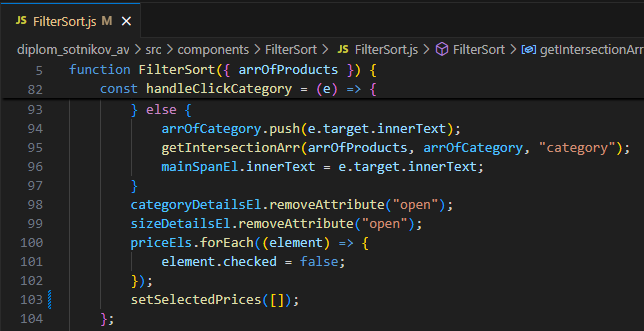


Рисунок 15 – Третья часть кода компонента FilterSort.js.

Для стилизации компонента используется FilterSort.css. Код представлен в Приложении 9 – FilterSort.css.



Рисунок 16 – Четвертая часть кода компонента FilterSort.js.



Рисунок 17 – Пятая часть кода компонента FilterSort.js.

**Компонент страницы продукта ProductPage.js**

Скриншот страницы продукта представлен в приложении 10. Код компонента ProductPage.js представлен на рисунке 18.

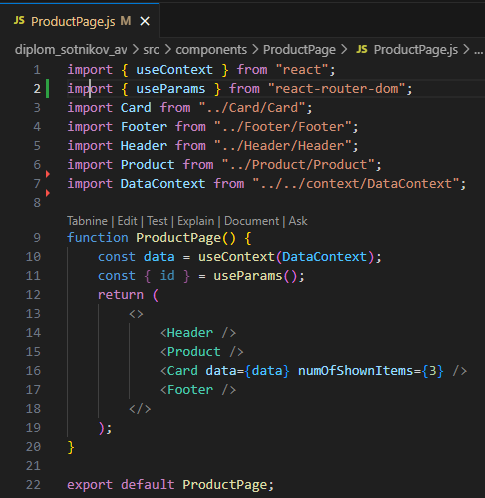


Рисунок 18 – Код компонента ProductPage.js

Код реализует страницу продукта с использованием контекста и маршрутизации.

Страницы 1-7 - Импортируем библиотеки и компоненты:

* useContext: Хук, который позволяет получить доступ к значениям контекста (в данном случае, DataContext).
* useParams: Хук, который позволяет извлекать параметры URL, такие как id продукта из маршрута.
* Далее идут импорты компонентов (Card, Footer, Header, Product), которые будут использоваться в разметке. Первые 3 были описаны ранее. К Product вернемся в следующей главе.

Строка 9 - Определение компонента. Здесь мы определяем функциональный компонент ProductPage. Внутри ProductPage мы сначала получаем данные из контекста с помощью useContext и сохраняем их в переменной data (строка 10).

useParams извлекает параметр id из URL, который можно использовать для загрузки конкретного продукта и обновлении страницы при смене продукта (строка 11).

Строки с 12 по 19 - Возврат JSX. В этом блоке кода возвращается JSX-разметка. Она включает:

* <Header />: хэдер сайта.
* <Product />: Компонент для отображения информации о конкретном продукте
* <Card data={data} numOfShownItems={3} />: Компонент для отображение карточек товара, который получает данные из контекста и будет отображать 3 элемента.
* <Footer />: футер сайта.

Строка 22 - Экспорт компонента. Компонент ProductPage экспортируется как модуль для использования в других частях приложения.

Код создает страница продукта. Он использует React Router для извлечения идентификатора продукта из URL и контекст API для доступа к данным о продуктах. Страница состоит из хедера, информации о продукте, компонента для отображения карточек других товаров, а также включает футер.

Далее будет описан, еще неохваченный ранее компонент, Product.js

**Компонент Product.js**

Код компонента Product.js представлен на рисунке 19, 20 и 21. Его основная задача — отображение страницы товара с изображениями, описанием и возможностью добавления товара в корзину.

Строки 1-3 – импорты библиотек:

* "./Product.css" - Импорт стилей для компонента.
* { Swiper, SwiperSlide } - Импорт компонентов Swiper для слайдера изображений.
* "swiper/css" - Импорт стилей Swiper.

Строки с 5 по 7 - Определение компонента. Здесь мы получаем данные о товаре из localStorage. Параметр scrollTo используется для возвращения к началу страницы.

Функция addToCart (с 9 по 25 строки) отвечает за добавление товара в корзину. Если корзина пуста, товар добавляется. Если в корзине уже есть товары, новый товар добавляется к существующим, а количество каждого товара устанавливается в 1. Затем происходит фильтрация дубликатов за счет Set и обновление localStorage.

Строки с 27 по 105 - возврат JSX.

Swiper используется для создания карусели изображений. Каждый SwiperSlide содержит изображение товара, которое берётся из объекта product. Данный компонент взят с ресурса <https://swiperjs.com> [10].

Далее идет раздел, который отображает категорию товара, название, описание и цену. Также есть кнопка для добавления товара в корзину, которая связывается с функцией addToCart.



Рисунок 19 – Первая часть кода компонента Product.js



Рисунок 20 – Вторая часть кода компонента Product.js

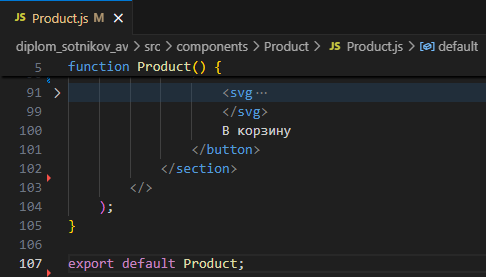


Рисунок 21 – Третья часть кода компонента Product.js

Код представляет собой функциональный компонент для отображения продукта с использованием слайдера для картинок и кнопкой добавления товара в корзину.

Для стилизации компонента используется Product.css. Код представлен в Приложении 11 – Product.css.

**Компонент страницы корзины товаров Cart.js**

Скриншот страницы корзины товаров представлен в приложении 12. Код компонента Cart.js представлен на рисунке 22.

Строки 1-3 – Здесь мы импортируем три компонента (Footer, Header, SelectedProduct) из соответствующих файлов.

Строки 5-13 - Определение компонента Cart. Здесь мы определяем функциональный компонент с названием Cart. Внутри этого компонента возвращается JSX, который описывает структуру пользовательского интерфейса. Этот интерфейс включает три компонента: Header, SelectedProduct, и Footer.

* <Header /> — хэдер сайта, описывался выше.
* <SelectedProduct /> — отображает выбранный продукт, включает детали продукта, цену и другие соответствующие элементы.
* <Footer /> — представляет подвал страницы, также описывался выше.

Строка 15 - Экспорт делает компонент Cart доступным для импорта в других файлах. Это позволяет использовать его в других частях приложения.

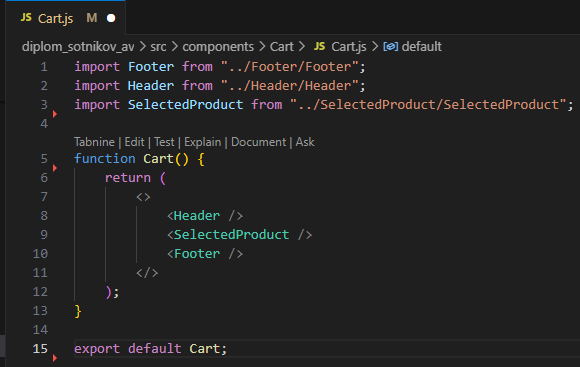


Рисунок 22 – Код компонента Cart.js

Компонент Cart служит контейнером для отображения хэдера, выбранных продуктов и подвала на странице. Он организует структуру интерфейса с использованием других компонентов и оформления.

Далее рассмотрим подробнее компонент SelectedProduct.

**Компонент SelectedProduct.js**

Код компонента SelectedProduct.js представлен на рисунке 23, 24, 25, 26 и 27. Для сокращения места svg на строке 115 был свернут.

Компонента SelectedProduct отвечает за отображение и управление выбранными товарами в корзине.

Строки с 1по 10 - Импорт и состояние. В начале мы импортируем необходимые хуки useEffect и useState из React для управления состояниями, а также CSS файл для стилизации компонента.

Определяем три состояния:

* totalCost — для хранения общей стоимости товаров в корзине.
* noGoodsInCartText — для управления видимостью сообщения о пустой корзине.

Загружаем выбранные товары из localStorage.

Строки с 12 по 19 – Функция handleChange обрабатывает изменение количества товара. Находит индекс товара в массиве selectedProducts, обновляет его количество и сохраняет изменения в localStorage. Затем вызывается функция getGrandTotal() для пересчета общей стоимости.

Строки с 21 по 28 - Функция clearCartClick очищает корзину, устанавливая пустой массив в localStorage, и обновляет состояние для отображения сообщения о пустой корзине. Через 2 секунды происходит перенаправление на страницу каталога.

Строки с 30 по 32 – Функция continueShoppingClick (продолжение покупок) просто перенаправляет пользователя на страницу каталога.

Строки с 34 по 41 – Функция getGrandTotal вычисляет общую стоимость всех товаров в корзине, проходя по массиву selectedProducts, и обновляет состояние totalCost.

Строки с 43 по 57 – Функция getCartArr удаляет товар из корзины по его ID. Если после удаления корзина оказывается пустой, показывается сообщение и происходит перенаправление. В противном случае страница перезагружается.

Строки с 59 по 67 – Функция removeItemFromCart определяет, какой элемент был нажат (кнопка, иконка (svg или path)), и вызывает функцию getCartArr для удаления соответствующего товара.

Строки с 69 по 80 – Функция payment имитирует оплату товара. Перед оплатой проверяет, что заполнены все необходимые поля и запускет функцию clearCartClick. В противном случае выдает сообщение о необходимости их запонения.

Строка 82 - Хук useEffect вызывается при первом рендере компонента, чтобы получить стоимость товаров в корзине.

Строки с 84 по 228 - Рендеринг компонента. Компонент возвращает JSX, который включает заголовок, список товаров, кнопки для очистки корзины и продолжения покупок, а также форму для ввода адреса доставки и кнопки для оплаты.

Если в корзине есть товары, они отображаются в виде списка. Если корзина пуста, отображается сообщение о том, что в корзине нет товаров.

Ввод адреса доставки и кнопка для отправки формы: обработчик onSubmit вызывает функцию payment, которая проверяет заполненность полей и обрабатывает платеж.

Компонент SelectedProduct предоставляет пользователям все необходимые инструменты для управления корзиной. Он обрабатывает изменения, удаление товаров, расчет общей стоимости и оформление заказа.

Для стилизации компонента используется SelectedProduct.css. Код представлен в Приложении 13 – SelectedProduct.css.

С 

Рисунок 23 – Первая часть кода компонента SelectedProduct.js.



Рисунок 24 – Вторая часть кода компонента SelectedProduct.js.

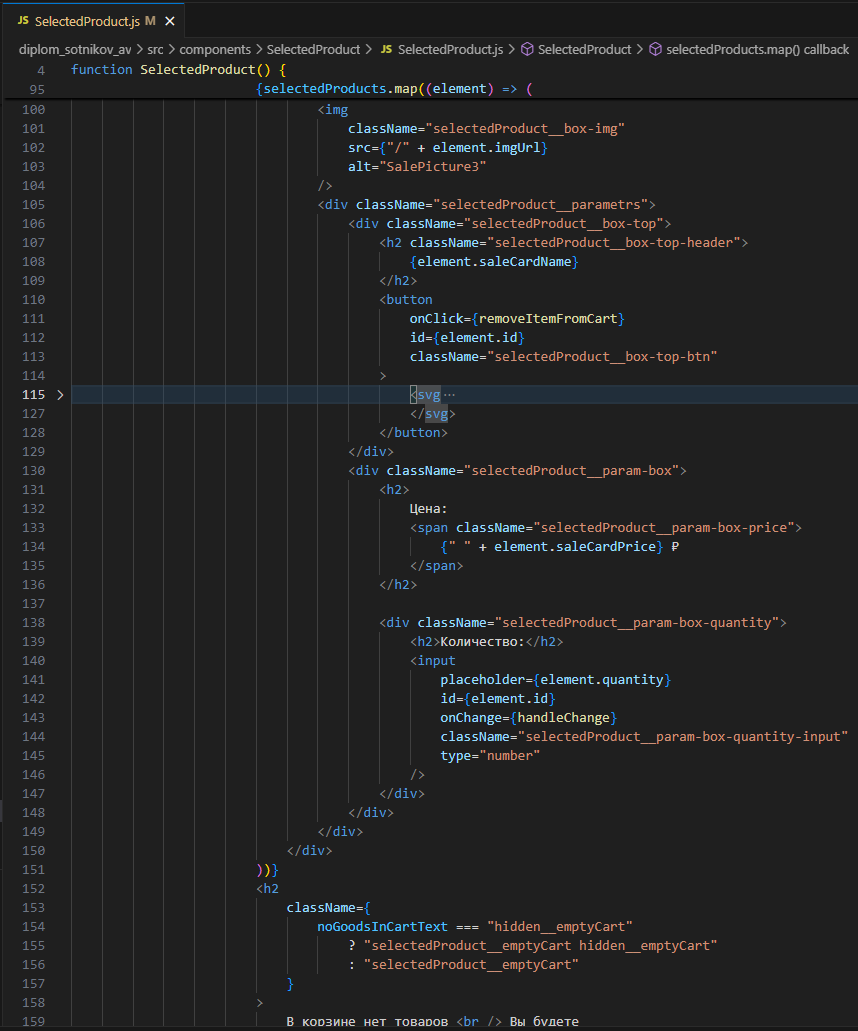


Рисунок 25 – Третья часть кода компонента SelectedProduct.js.



Рисунок 26 – Четвертая часть кода компонента SelectedProduct.js.

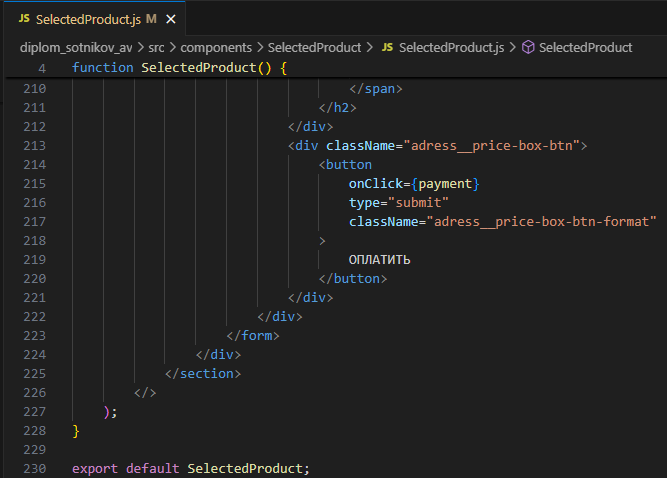


Рисунок 27 – Пятая часть кода компонента SelectedProduct.js.

**Компонент страницы регистрации пользователя RegistrationPage.js**

Скриншот страницы регистрации пользователя представлен в приложении 14. Код компонента RegistrationPage.js представлен на рисунке 28.

Строки 1-3 – Здесь мы импортируем три компонента (Footer, Header, RegistarationForm) из соответствующих файлов.

Строки 5-13 - Здесь мы определяем функциональный компонент с названием RegistrationPage. Внутри этого компонента возвращается JSX, который описывает структуру пользовательского интерфейса. Этот интерфейс включает три компонента Header, RegistrationForm, и Footer:

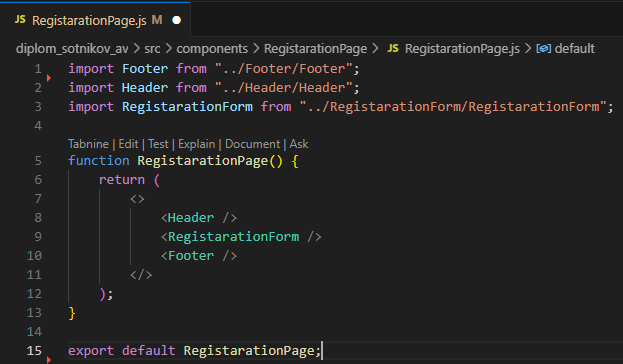


Рисунок 28 – Код компонента RegistrationPage.js.

* <Header /> — хэдер сайта, описывался выше.
* < RegistrationForm /> —отображает форму для заполнения информации о пользователе.
* <Footer /> — представляет подвал страницы, также описывался выше.

Строка 15 - Экспорт делает компонент RegistrationPage доступным для импорта в других файлах. Это позволяет использовать его в других частях приложения.

Компонент RegistrationPage служит контейнером для отображения хэдера, формы с личной информацией о пользователе и подвала на странице. Он организует структуру интерфейса с использованием других компонентов и оформления.

Далее рассмотрим подробнее компонент RegistrationForm.

**Компонент RegistrationForm.**

Код компонента RegistrationForm представлен на рисунке 29 и 30. Для экономии места svg на строках 85 – 104 был свернут.

Строки 1-4 - Импорт и объявление компонента. Здесь импортируется хук useState для управления состоянием, CSS-файл для стилизации компонента. RegistarationForm — это функциональный компонент, который возвращает JSX-элементы.

Функция registration обрабатывает событие регистрации.

Строки 5 – 11 - Создано состояние formData для управления значениями полей формы (имя, фамилия, email, пароль и пол).

Строки 13 – 19 – Функция handleChange обрабатывает изменения в полях ввода, обновляя состояние формы на основе имени элемента, используя деструктуризацию.

Строки 21 – 30 - В функции handleSubmit добавлено предотвращение отправки формы по умолчанию с помощью e.preventDefault(). Проверка на пустые поля теперь выполняется с использованием метода Object.values для проверки всех значений одновременно.

Строки с 32 по 138 - Возвращается JSX разметка. Используются контейнеры <section> для структурирования. Заголовок "РЕГИСТРАЦИЯ" оформлен в отдельном блоке. Cоздаётся форма с именем, фамилией, полом (мужской/женский) и данными для авторизации (email и пароль).

Строка 140 - Экспорт компонента для его использования в других частях приложения.



Рисунок 29 – Первая часть кода компонента RegistrationForm.



Рисунок 30 – Вторая часть кода компонента RegistrationForm.

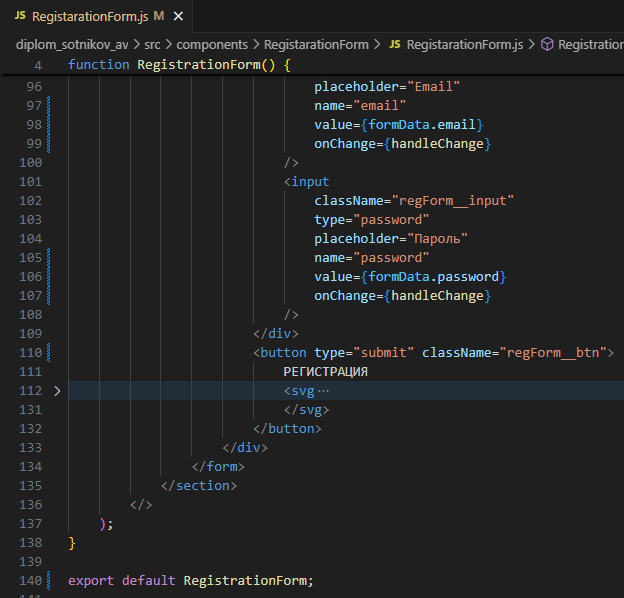


Рисунок 31 – Третья часть кода компонента RegistrationForm.

**Заключение**

В ходе выполнения проектного задания были достигнуты следующие цели:

1. Был проведен анализ современных средств для реализации продуктов компании, а именно торговые площадки, интернет-сервисы для размещения объявлений о товарах, фирмы по утилизации оборудования. В результате была выбрана разработка собственного веб-сайта для продажи офисной техники и сетевого оборудования компании.
2. В ходе выполнения проектного задания удалось изучить и освоить такие прикладные программные продукты как среда разработки Visual Studio Code, фреймворк React, распределенная система управления версиями Git, язык программирования JavaScript, формальный язык декодирования и описания внешнего вида документа CSS.
3. С использованием указанных продуктов был разработан веб-сайт по продаже офисной техники и сетевого оборудования компании, отвечающий требованиям удобства и простоты использования пользователем, а также обеспечения выполнения основной задачи данного проекта по реализации излишков техники и устаревшего оборудования.
4. По итогам выполнения проекта был выявлен перечень улучшений:

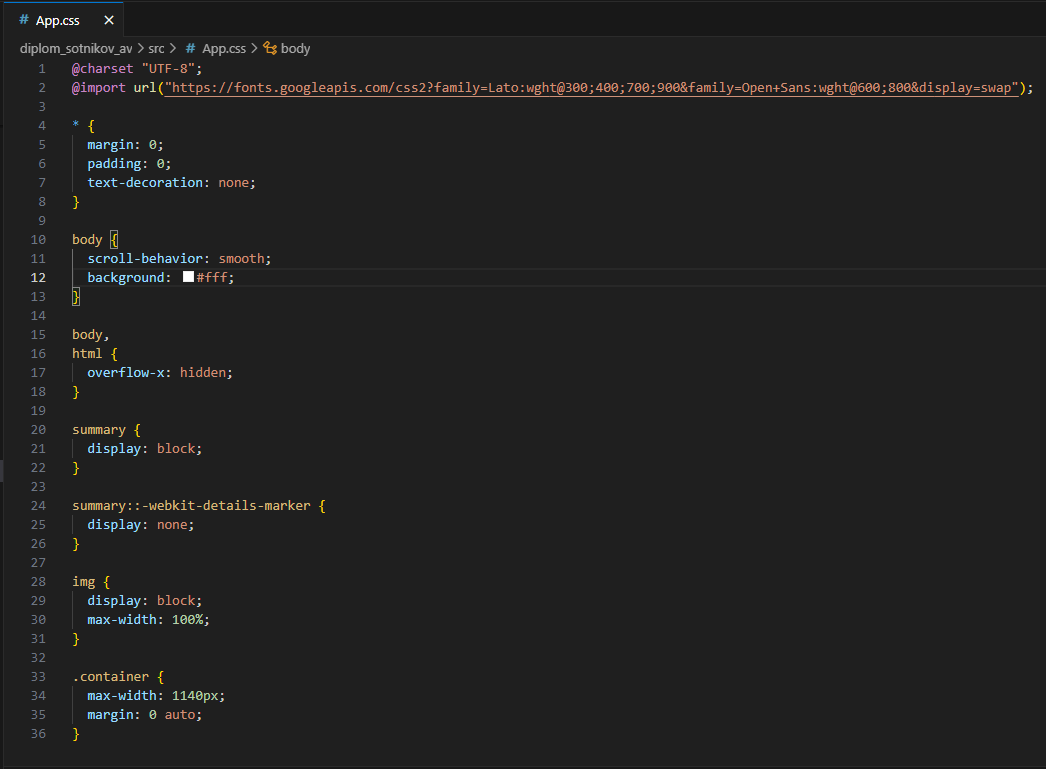
* добавление обратной связи при взаимодействии клиента с веб-сайтом в виде комментариев и оценки продукта.
* добавление страницы «О нас» для улучшения лояльности клиента
* возможность авторизации клиента для сохранения личной информации о клиенте.
* Использование препроцессора SASS для визуализации HTML.

**Список используемой литературы**

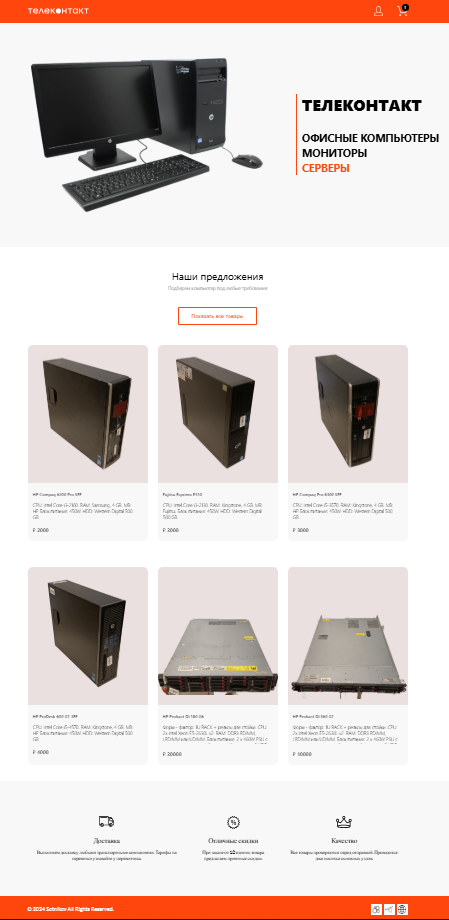
1. <https://blog.skillfactory.ru/glossary/ide/>
2. <https://aws.amazon.com/ru/what-is/ide/>
3. <https://blog.geekbrains.by/top-11-luchshih-ide-i-redaktorov-koda-javascript/>
4. <https://ru.hexlet.io/blog/posts/yazyki-programmirovaniya-dlya-sozdaniya-saytov>
5. <https://habr.com/ru/companies/habr_career/articles/746038/>
6. <https://aw.club/global/ru/blog/best-javascript-frameworks-2023>
7. <https://code.visualstudio.com/docs/?dv=win64user>
8. <https://code.visualstudio.com/docs/supporting/faq>
9. <https://gbcdn.mrgcdn.ru/uploads/asset/5796196/attachment/cc27f8217e3cf60e510430ecd67c128a.pdf>
10. <https://swiperjs.com/react#usage>

**Приложения**

Приложение 1 – App.css



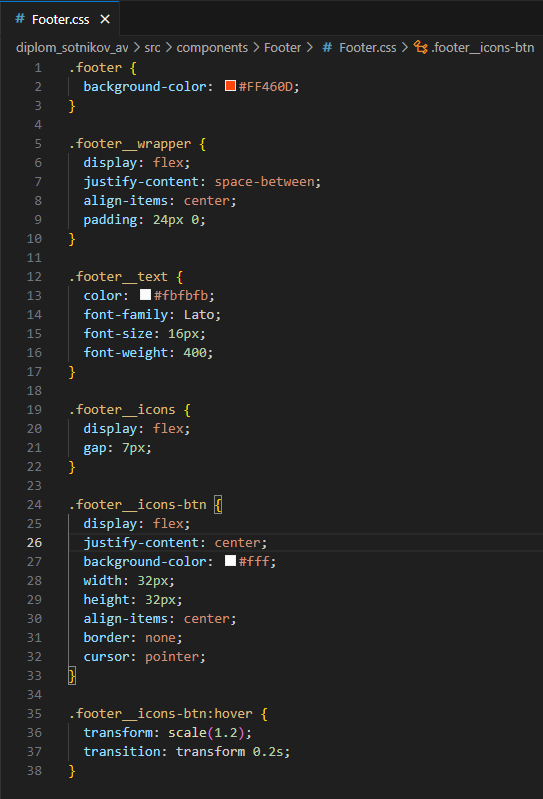
Приложение 2 – Главная страница сайта



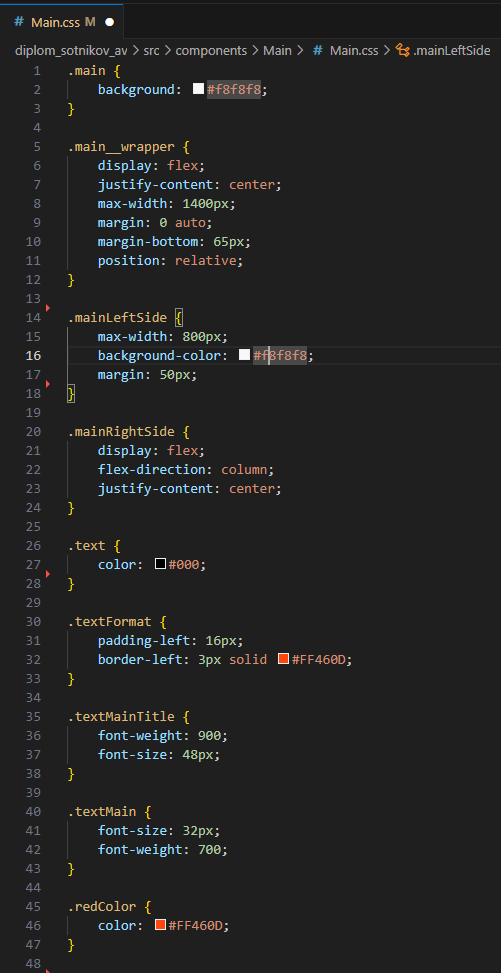
Приложение 3 – Header.css

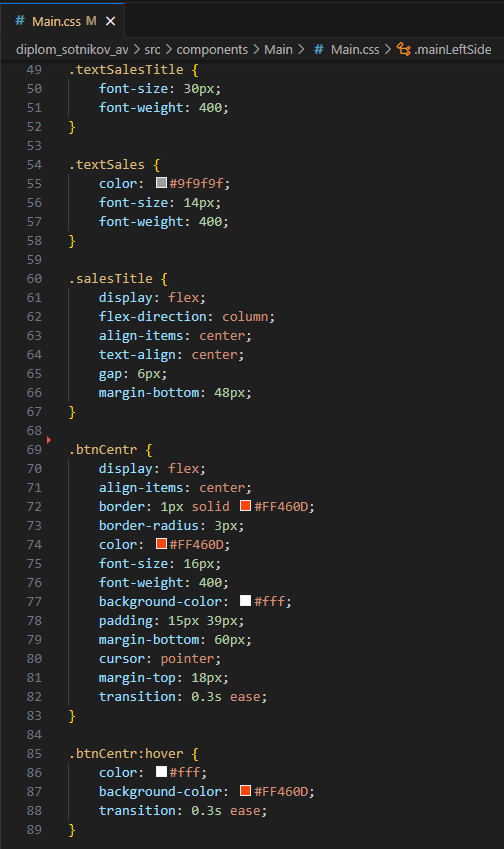


Приложение 4 – Footer.css.



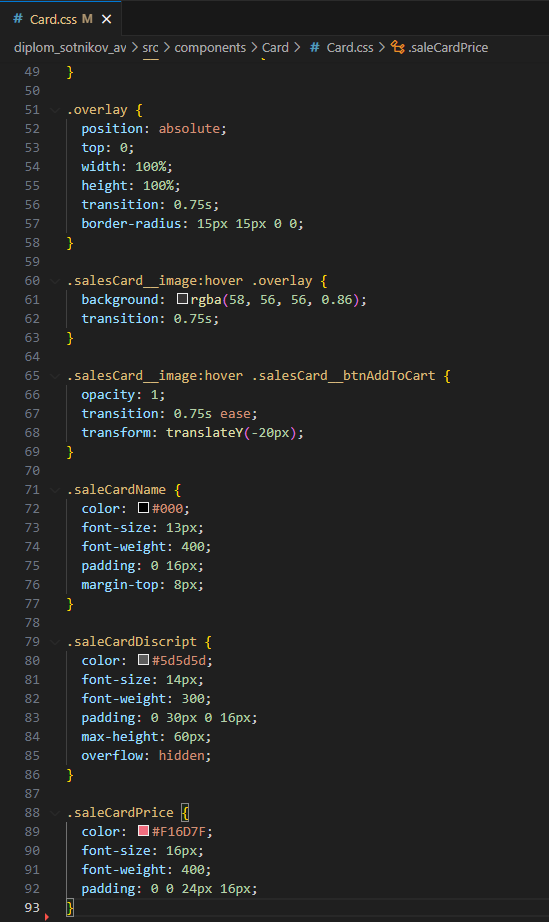
Приложение 5 – Main.css.



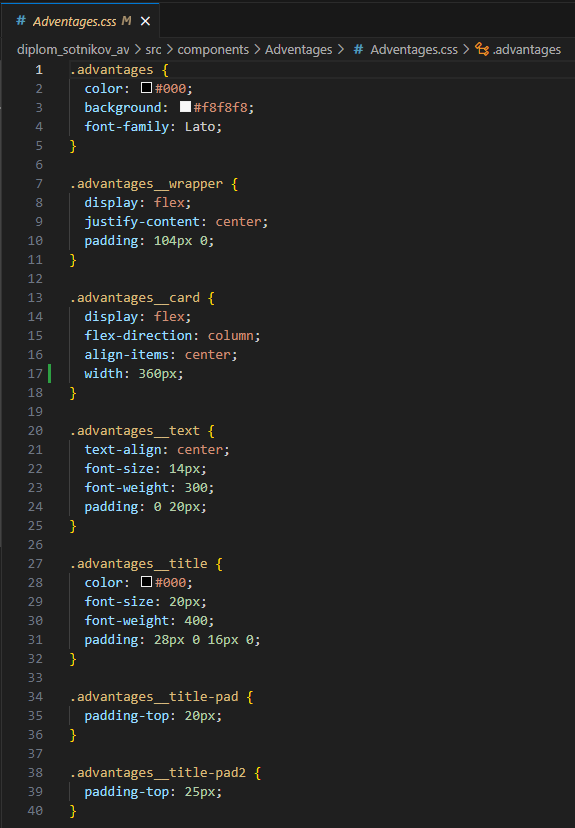


Приложение 6 – Card.css.

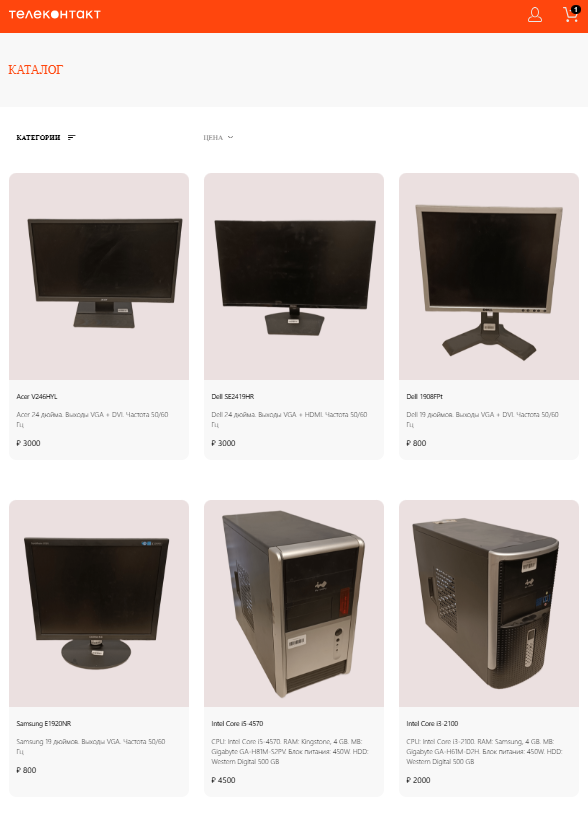


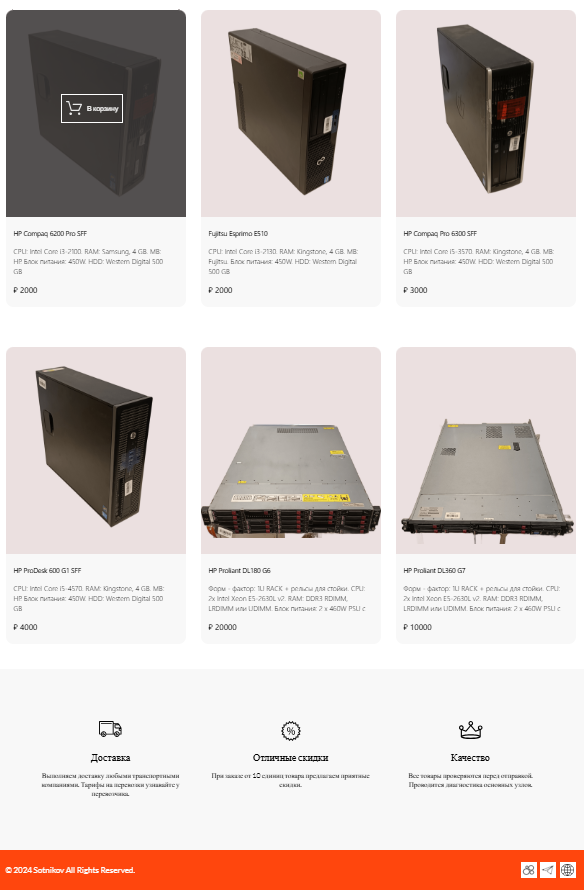


Приложение 7 – Adventages.css.

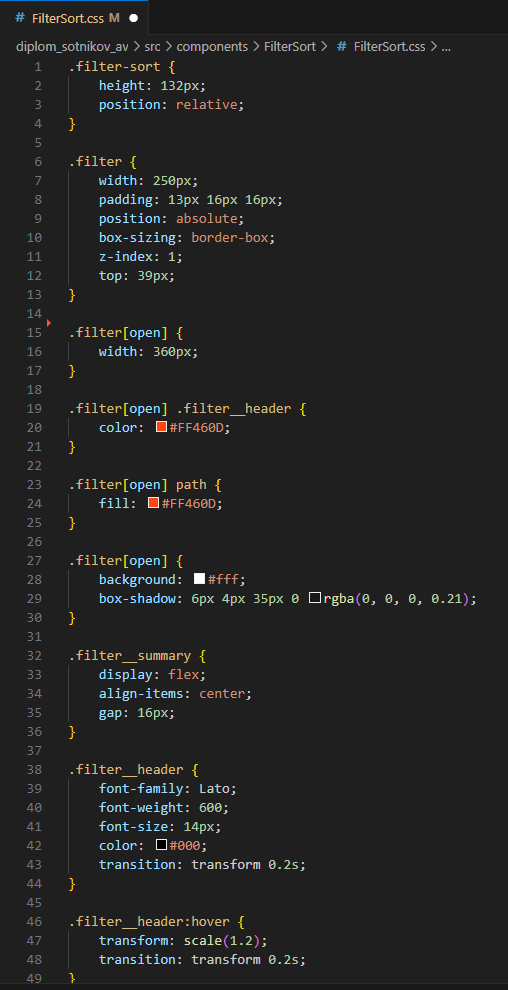


Приложение 8 – Страница каталога сайта

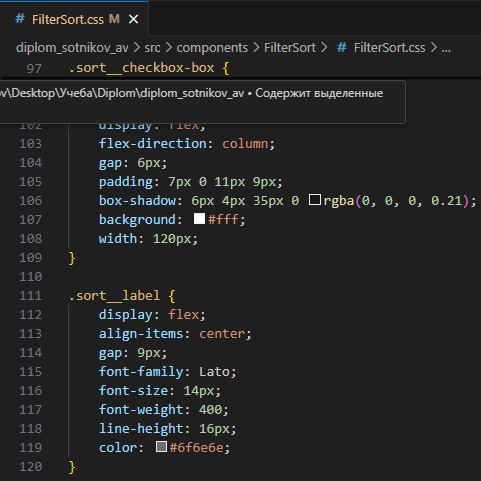




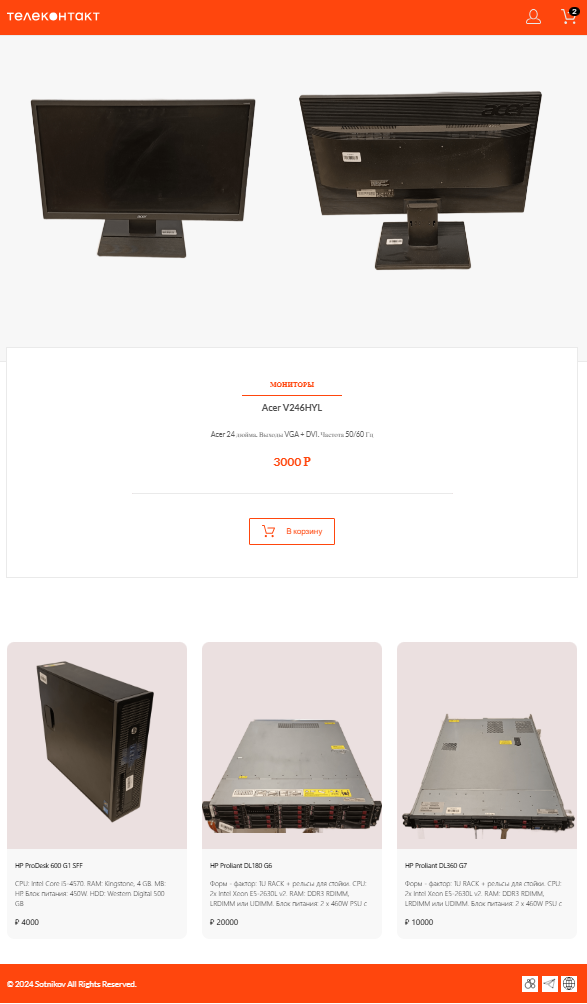
Приложение 9 – FilterSort.css



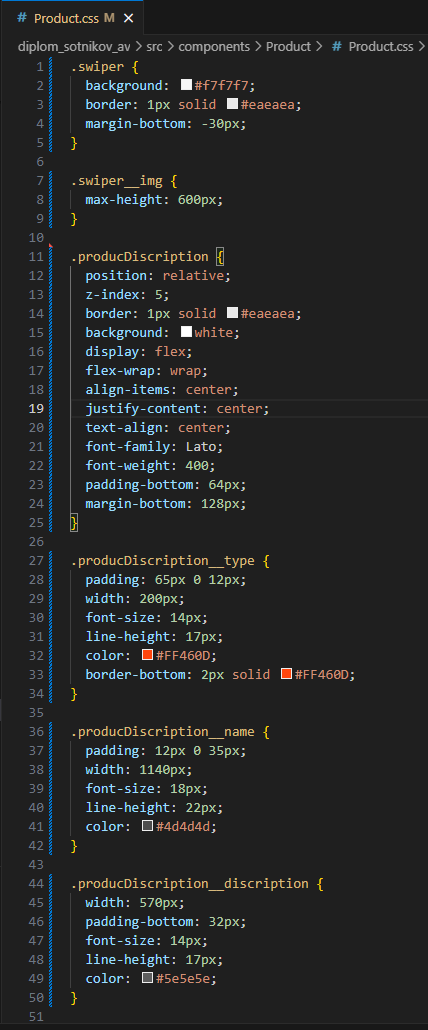


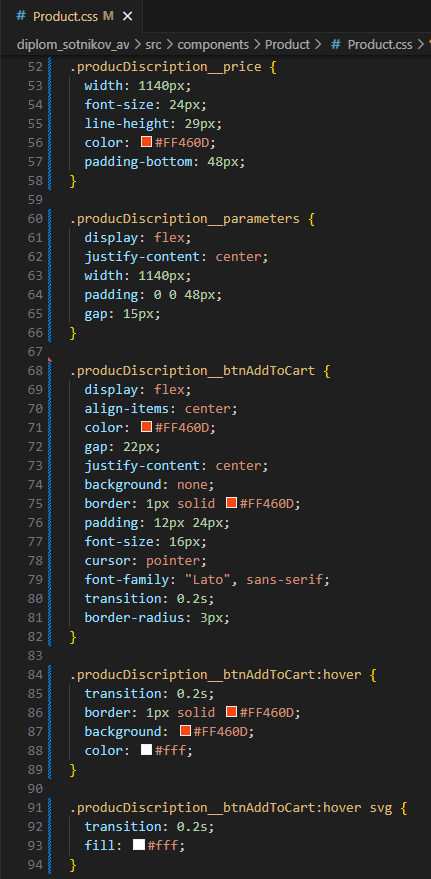


Приложение 10 – Страница продукта

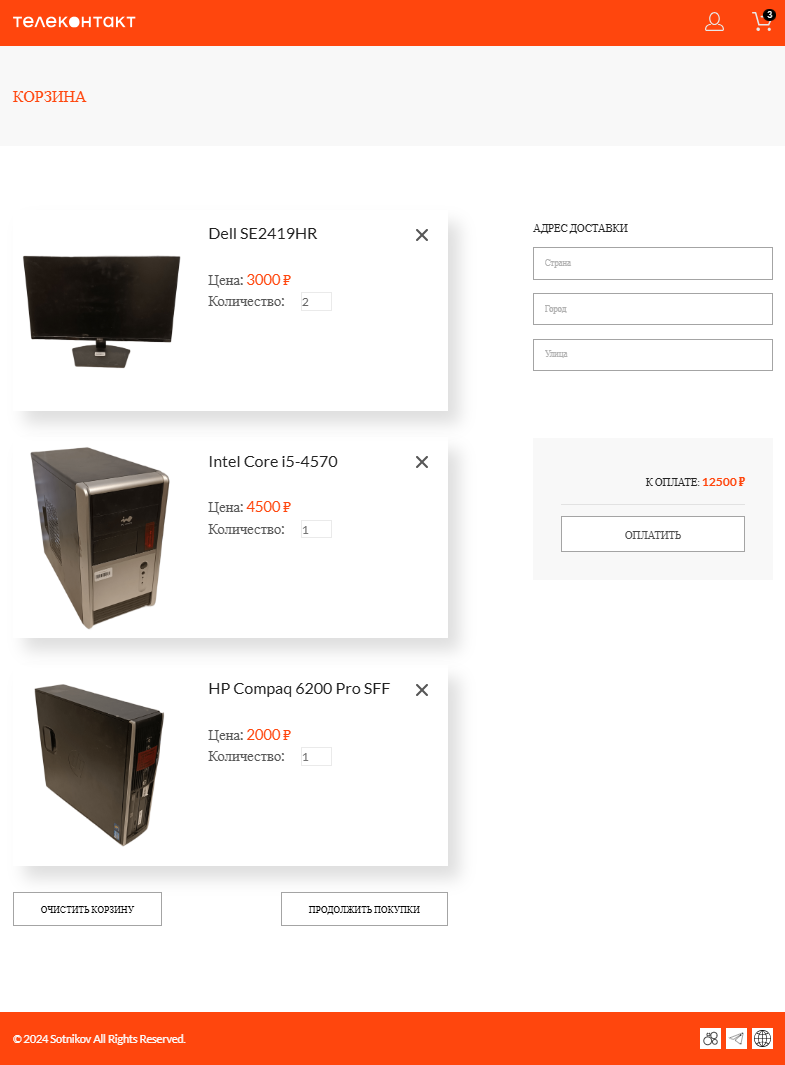


Приложение 11 – Product.css.





Приложение 12 – Страница корзины товаров



Приложение 13 – SelectedProduct.css.

