Департамент внутренней и кадровой политики Белгородской области

Областное государственное автономное профессиональное образовательное учреждение

«Белгородский индустриальный колледж»

**«К ЗАЩИТЕ ДОПУЩЕН»**

зам. директора по УР

\_\_\_\_\_\_\_\_\_\_/Выручаева Н.В./

«\_\_»\_\_\_\_\_\_20\_\_\_г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к дипломному проекту

Тема: **«Разработка информационного веб-сайта для ИП "Кутепов Д.А.»**

БИК О. 09.02.03 ДП. 111750. ПЗ

|  |  |  |
| --- | --- | --- |
| РАЗРАБОТАЛ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Ануфриев З.В./ |
| РУКОВОДИТЕЛЬ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Белянская Е.И./ |
| ЗАВ. ОТДЕЛЕНИЕМ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Беляева Г.Н./ |

Белгород 2021

СОДЕРЖАНИЕ

[Введение 3](#_Toc15570)

[1 Общая часть 5](#_Toc1345)

[1.1 Обзор рынка аналогичных программных продуктов 5](#_Toc8577)

[1.2 Планирование - как система личной эффективности 5](#_Toc16586)

[1.3 Обзор способов создания интернет-сайта 7](#_Toc18631)

[1.4 JavaScript - как язык для написания клиентской части 11](#_Toc31093)

[1.5 Выбор и обоснование библиотек 1](#_Toc24299)2

[2 Технологическая часть 1](#_Toc5051)3

[2.1 Описание среды программирования 1](#_Toc8869)3

[2.2 Реализация программного продукта 1](#_Toc5901)4

[2.3 Размещение программного продукта в сети Интернет](#_Toc6865) 37

[2.4 Руководство пользователя](#_Toc24383) 41

[3 Экономическая часть](#_Toc2745) 44

[4 Мероприятия по технике безопасности, противопожарной технике и охране труда при работе с вычислительной техникой](#_Toc9773) 54

[Заключение](#_Toc8601) 58

[Список использованных источников](#_Toc4017) 59

# ВВЕДЕНИЕ

Сегодня большинство видов бизнеса стараются перенести большую свою часть в интернет пространство. Это позволяет расширить круг потребителей, снизить расходы на организационные рутинные операции, а так же повысить престиж компании.

Актуальность дипломного проекта заключается в предоставлении приобретения продуктов людям с ограниченными возможностями, людям удалённым от магазина географически, а так же людям находящимся на самоизоляции в период пандемии.

Объектом исследования является разработка интернет-магазина мясных продуктов.

Предмет исследования дипломного проекта – это механизм работы создания интернет-сайтов на языке программирования JavaScript с использованием стека технологий Mern.

Целью данного проекта является разработка интернет–магазина мясных продуктов. Благодаря созданному программному продукту у людей появится возможность приобретать продукты питания прямо из дома, что особенно актуально в условиях пандемии.

Для достижения поставленной цели необходимо решить следующие задачи:

* Изучить планирование как систему личной эффективности
* Познакомиться с основными теоретическими понятиями разработки интернет-сайта;
* Изучить язык программирования: JavaScript;
* Изучить библиотеку React;
* Изучить работу менеджера состояний Redux;
* Изучить программную платформу Node js
* Изучить базу данных mongodb
* Продумать архитектуру и выполнить разработку программы;
* Раскрыть экономическую эффективность продукта.

В первой главе произведён анализ рынка на предмет аналогичных продуктов. Проанализирован рынок конкурентов с аналогичными программными продуктами. Проанализированы основные технологии позволяющие реализовать необходимый программный продукт. Обоснован выбор языка программирование и фреймворков выбранных для реализации дипломного проекта.

Во второй главе описана среда программирования. Описано создание общей структуры проекта. Описаны выбранные пакетные модули, а также некоторые другие детали программы.

В третьей главе раскрыта экономическая часть, перечислены основные затраты на реализацию интернет-магазина. Рассчитаны сроки окупаемости программного продукта

В четвертой главе описываются мероприятия по технике безопасности, противопожарной технике и охране труда при работе с вычислительной техникой.

Практическая значимость дипломного проекта заключается в том, что создание интернет магазина позволит уменьшить часть расходы компании на обслуживание клиентов, позволит людям совершать покупки не выходя из дома, а так же способно снизить уровень заражаемости в период пандемии.

## 1 ОБЩАЯ ЧАСТЬ

## Обзор рынка аналогичных программных продуктов

Современный рынок переоснащён программными продуктами различного направления и каждый из них имеет свои преимущества и недостатки. Проанализировав рынок на предмет аналогичных продуктов, основным недостатком был крайне громоздкий интерфейс, а так же долгое ожидание загрузки страниц, в связи с чем было принято решение разработать single page application (SPA) с минималистичным интерфейсом.

Разработанный программный продукт будет содержать следующий функционал:

* Удобная навигация по страницам приложения без их перезагрузки
* Возможность просматривать товар в каталоге;
* Возможность выбора количества товара в каталоге;
* Возможность добавления товара в корзину;
* Подсчёт общей стоимости товаров в корзине;
* Возможность удаления товаров из корзины;
* Возможность поиска товаров по названию;
* Возможность фильтрации товаров по категориям;
* Возможность заполнения формы, с дальнейшей отправкой данных о заказе на сервер.

## Роль интернета в современном бизнесе

Большинство современных компаний сегодня стараются перенести часть своего бизнеса в интернет. Это позволяет.

* Увеличить приток аудитории;
* Автоматизировать многие рутинные задачи;
* Повысить престиж и узнаваемость компании.

Разберём каждый вышеперечисленный пункт, подчеркнём значимость каждого из них для компании, что позволит сформировать более чёткое представление о деталях разрабатываемого программного продукта.

- Увеличение притока аудитории.

В эпоху информационного общества значительную часть информации человек получает в интернете. Это касается как макро составляющей жизни человека, например поиска работы или жилья, так и микро задач, например выбор магазина продуктов или одежды.

Создание сайта для магазина позволит привлечь клиентов ещё на этапе планирования их быта, а качество сайта напрямую влияет на вероятность попадания конкретной компании в список планов определённого круга потребителей.

После привлечение клиента, наступает этап знакомства знакомства клиента с сайтом, и уже от этого зависит уровень желание человека пользоваться услугами конкретной компании.

На качество опыта влияет множество факторов, таких как дизайн сайта, цветовая схема, удобство интерфейса и т.д. Чем качественней будут реализованы вышеперечисленные составляющие нашего сайта, тем больше клиентов будут выбирать именно нашу компанию.

- Автоматизация

Во многом именно этот пункт определяет экономическую необходимость перевода части бизнеса в интернет. Чем больше процессов автоматизировано в каком либо виде бизнеса, тем выше как правило окупаемость деятельности предприятия.

Главная цель любого предприятия – получение прибыли, и именно автоматизация помогает непосредственно снизить расходы внутри предприятия и мотивирует владельцев того или иного бизнеса вкладываться именно в автоматизацию.

Сегодня автоматизируют не только процессы непосредственного производства, но и многие информационные составляющие бизнеса.

Как правило такими задачами, как оформление клиентов во многих кампаниях занимаются отдельные люди. Замена этих людей программой позволяет сократить расходы на содержание части штата, занимающимися решением многих информационных задач.

Помимо этого многим людям доставляет дискомфорт общение с живым человеком. Минимизация живого общения из процесса обслуживания клиента, способно повысить качество опыта от обслуживания, а так же увеличить количество людей, готовых пользоваться нашими услугами.

- Престиж;

Как правило, видя незнакомую кампанию, человек ищет информацию о ней в интернете. От качества сайта зависит уровень доверия человека конкретному предприятию. Качественный сайт способен значительно увеличить уровень доверия к компании, а так же её узнаваемость, что позволит не только более эффективно привлекать новых клиентов, но и удерживать старых.

## Обзор способов создания интернет-сайта

Есть три основных способа создания интернет-сайта:

* Разработка в конструкторе сайтов;
* Разработка на CMS (особенно популярна WordPress);
* Самостоятельная разработка, в том числе с использованием популярных инструментов и фреймворков;

У каждого из способов есть как свои плюсы, так и недостатки.

* Конструктор

Конструктор – программное решение (чаще всего онлайн), позволяющее построить сайт по модульному принципу, когда разработчик собирает всю конструкцию с помощью готовых «кубиков», которые предоставляет конструктор. Такой подход позволяет создать сайт без знаний о веб-разработке Больше всего подходит написания простого сайт о себе или своей компании, без претензий на качество программного кода и скорость работы сайта. Сделать серьёзное корпоративное решение или интернет-магазин на конструкторе невозможно.

Плюсы:

* + Низкая цена. Почти все конструкторы изначально бесплатны, а стоимость подключаемых модулей очень низкая;
  + Простота использования. Для того, чтобы создать свой сайт, достаточно уметь пользоваться браузером.

Минусы:

* + Домен третьего или более высокого уровня. При использовании бесплатных аккаунтов в конструкторах, клиент получает размещение своего сайта только на домене не ниже третьего уровня, например, sitename.constructor.com. Такие домены не вызывают большого доверия у интернет-сообщества, а домены второго уровня (напр. sitename.com) стоят дороже, чем если их приобретать у регистраторов доменов напрямую;
  + Тяжеловесность сайта. Сайт, сделанный на конструкторе всегда будет загружаться дольше аналогичного сайта, сделанного на CMS или разработанного самостоятельно. Объясняется это тем, что конструктор содержит в себе огромное количество программного кода, который не относится к Вашему сайту, но необходим для построения его итогового внешнего вида;
  + Отсутствие SEO. Несмотря на заявления любого конструктора сайтов, провести полноценное SEO сайта, сделанного на конструкторе, не представится возможным по причине того, что SEO – это комплекс мероприятий, в том числе, по работе с кодом, индексации в поисковых системах, построению структуры сайта, и для этого всего требуется доступ к программному коду и разметке, что невозможно в конструкторе.
* CMS

CMS – это базовый каркас и набор дополнительных инструментов и надстроек, который позволяет не только создать веб-сайт или веб-приложение, но и поддерживать его работу, обновлять контент и взаимодействовать с пользователями. Все CMS имеют панель управления с относительно дружелюбным интерфейсом. Основной язык программирования – PHP. Любая CMS может позволить создать даже очень сложные решения, такие как интернет-магазины или большие корпоративные сайты с глубокой вложенностью страниц, но и у них есть свои особенности.

Плюсы:

* + Бесплатный доступ. Почти все CMS (кроме 1С-Битрикс) изначально бесплатны;
  + Удобное управление контентом. С помощью панели управления управлять сайтом легко и просто. При использовании CMS не требуется особых навыков для управления контентом;
  + Множество готовых решений. Существует масса модулей, плагинов, дополнений для решения различных задач;

Минусы:

* + Уязвимость сайта. Самый важный минус любой широко распространённой CMS. Даже сайты, сделанные на конструкторах порой лучше защищены от взлома;
  + Сложности с переносом. Хотя популярные CMS на данный момент имеют автоматизированные средства установки почти на любом хостинге, в случае необходимости переноса сайта или управления его положением, могут возникнуть трудности, так как придётся производить всю процедуру установки заново;
  + Затраты на дополнительный контент. Как и в случае с конструкторами, дополнительные модули и расширения сайта для CMS стоят денег, и в данном случае, уже больших, чем для конструкторов. Почти любой модуль, который относится к постоянно используемым, потребует либо приличных затрат в начале, либо платной ежемесячной подписки;
* Самостоятельная разработка

Самый творческий и свободный, но и самый трудоёмкий процесс. Написание сайта или веб-приложения требует серьёзных знаний не только по самим языкам программирования, но и пониманию архитектуры, бизнес-процессов клиента и многому другого. Самостоятельная разработка позволяет создавать проекты любой сложности и по любым пожеланиям клиента.

Плюсы:

* + Широкие возможности продвижения. В отличие от CMS и конструкторов, продвинуть в естественном поиске самостоятельно разработанный сайт намного легче;
  + Индивидуальный дизайн. Становится возможным разрабатывать уникальные дизайны;
  + Расширяемость. Так как имеется доступ к коду, облегчается возможность добавления нового функционала;

Минусы:

* + Цена. Спорный, но всё-таки минус чистого кода. Хотя в случае крупных проектов, разработка без конструкторов и CMS является самым эффективным и, порой, самым недорогим решением, так как в итоге не придётся получать не то, что хотелось, а в последствии не переделывать при каждой необходимости;
  + Наличие знаний. Самостоятельная разработка требует наличия большого количества знаний по языкам программирования, построению архитектуры веб-приложений, алгоритмам и структурам данных, бизнес-процессам и многому другому;
  + Временные затраты. Использование CMS и конструкторов позволяет сделать простой сайт быстрее, чем при разработке без них. В ситуациях, когда время играет большую роль, самостоятельная разработка становится бессмысленной;

Исходя из данных плюсов и минусов, а так же специфичности задачи, была выбрана самостоятельная разработка на языке JavaScript.

## JavaScript - как язык для написания клиентской части

Существует огромное множество языков программирования для написания клиентской части сайта. Но так или иначе, все они, языки программирования, транслируются в JavaScript.

Сам [JavaScript](https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/JavaScript) – это лёгкий, интерпретируемый, объектно-ориентированный язык с [функциями первого класса](https://en.wikipedia.org/wiki/First-class_functions), самый известный скриптовый язык для веб-страниц, но также [во многих не браузерных окружениях.](https://en.wikipedia.org/wiki/JavaScript#Uses_outside_web_pages) [Прототипно-ориентированный](https://en.wikipedia.org/wiki/Prototype-based_programming), мультипарадигменный язык сценариев, который поддерживает динамический, объектно-ориентированный, императивный и функциональный стили программирования. Является реализацией стандарта ECMAScript. Он даёт возможность реализовывать сложное поведение веб-страницы. JavaScritpt – легок в изучении и написании кода. У него огромное сообщество разработчиков, которое реализовало множество библиотек и фреймворков облегчающих процесс разработки интернет-сайта. Из их числа:

* React;
* Angular;
* Vue.js.

Angular - полноценный фреймворк для написания сложных MVC приложений. Насыщен различными возможностями, но довольно сложен в освоении. Хоть это и прекрасное решение для написания сложных систем, но для реализации небольших приложений и интернет-сайтов он не слишком хорошо подходит, требуя писать большое количество кода.

В то же время, React и Vue.js примерно одинаково просто позволяют писать приложения малых и средних размеров. Но из-за личных предпочтений был выбран React.

React – это JavaScript библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. Он разрабатывается и поддерживается [Facebook](https://ru.wikipedia.org/wiki/Facebook), [Instagram](https://ru.wikipedia.org/wiki/Instagram) и сообществом отдельных разработчиков и корпораций.

В основном React используется для создания SPA (Single Page Application). Его цель - предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, [Redux](https://ru.wikipedia.org/wiki/Redux) и GraphQL. React дает язык шаблонов и некоторые callback-функции для отрисовки HTML. Весь результат работы React - это HTML. Связки HTML/JavaScript, называемые компонентами, занимаются тем, что хранят свое внутреннее состояние в памяти, но в итоге просто возвращается сгенерированный HTML.

Одним из основных достоинств библиотеки является быстрое и «экономичное» применение изменений. Достигается это с использованием:

* ShadowDOM, который позволяет быстро производить перерисовку компонентов без тяжеловесного изменения обычного DOM;
* Использованием собственного эвристического алгоритма со сложностью O(n), для внесения изменений в ShadowDOM;
* С передачей разработчикам права, через методы жизненного цикла, контролировать процесс перерисовки.

## Выбор и обоснование библиотек

В данном дипломном проекте мы использовали технологии стека Mern. Библиотеки стека Mern позволяют разработать full-stack приложение используя один лишь JavaScript. Пройдёмся по каждой библиотеке и подчеркнём плюсы и минусы каждой их них.

1. Mongodb

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

Плюсы:

* маленькая схема
* легко масштабируемо
* бесплатно

Минусы:

* Размер данных в MongoDB обычно выше из-за, например, каждый документ имеет имена полей, которые хранятся в нем.
* меньше гибкости при запросе (например, нет JOIN)
* нет поддержки транзакций - поддерживаются определенные атомарные операции на одном уровне документа

# 2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 2.1 Описание среды программирования

Visual Studio Code - [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0), разработанный [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) для [Windows](https://ru.wikipedia.org/wiki/Windows), [Linux](https://ru.wikipedia.org/wiki/Linux) и [macOS](https://ru.wikipedia.org/wiki/MacOS). Позиционируется как «лёгкий» редактор кода для [кроссплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) разработки [веб](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)- и [облачных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) приложений. Включает в себя [отладчик](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA), инструменты для работы с [Git](https://ru.wikipedia.org/wiki/Git), [подсветку синтаксиса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0), [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и средства для [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Имеет широкие возможности для [кастомизации](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F): пользовательские темы, [сочетания клавиш](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%87%D0%B5%D1%82%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BB%D0%B0%D0%B2%D0%B8%D1%88) и [файлы конфигурации](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB_%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%B8). Распространяется бесплатно, разрабатывается как программное обеспечение с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), но готовые сборки распространяются под [проприетарной лицензией](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Встроенные возможности:

* Внутренний терминал;
* Сниппеты использующиеся для переиспользования кода;
* Автозавершение кода;
* Широкие возможности для кастомизации;
* Навигация по проекту и исходному коду: отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и использованиями методов;
* Рефакторинг: переименование, извлечение метода, введение переменной, введение константы и т. д;
* Поддержка систем контроля версий: общий пользовательский интерфейс для Mercurial, Git, Subversion, Perforce и CVS с поддержкой списков изменений и слияния.

Большинство возможностей расширяются с помощью плагинов из магазина плагинов (Visual Studio MarketPlace).

## 2.2 Реализация программного продукта

Перед началом разработки приложения необходимо было продумать структуру приложения. В неё входит:

* Менеджер состояний;
* Пользовательский интерфейс;
* Хранилище для данных;
* Сервер
* База данных

Для реализации менеджера состояния приложения была выбрана библиотека Redux. В состоянии приложения хранится корзина с добавленными в неё товарами.

* actions. Содержит действия посылаемые приложением для смены состояния. При реализации возникла потребность в двух событиях – добавление и удаление товаров из корзины. Программная реализация действий продемонстрирована на рисунке 2.1.

1. ADD\_PRODUCT\_TO\_CART. Добавляет товар в корзину
2. DELETE\_PRODUCT\_INTO\_CART. Удаляет товар из корзины;

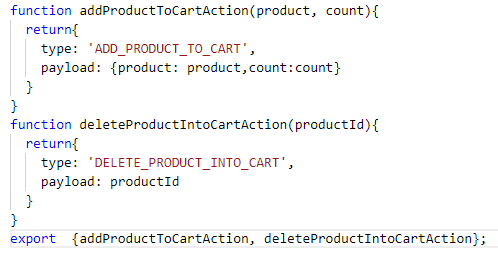


Рисунок 2.1 – Действия для изменения состояния приложения

* reducers. Содержит функции, которые изменяют состояние приложения в ответ и в соответствии с actions. Reducer-ы, обычно, разделяют по типу событий и области хранилища для изменения. На рисунке 2.2 показано как выглядит типовой reducer;

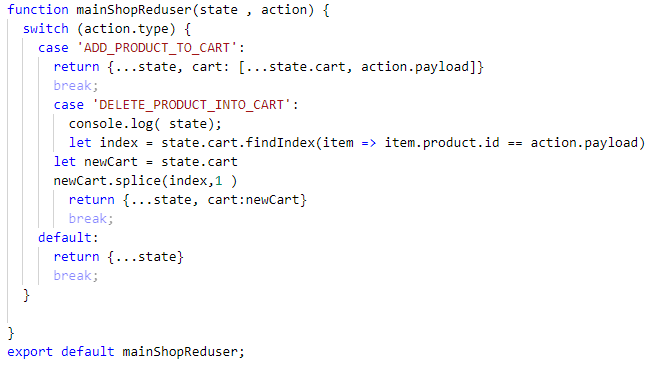


Рисунок 2.2 - reducer для обработки событий сохранения и удаления задачи

* store. Содержит файлы с конфигурацией хранилища приложения.

Для реализации взаимодействия менеджера состояний Redux и React-компонентов использовалась библиотека react-redux. Из этой библиотеки, для каждого компонента, которому требуется получить доступ к хранилищу, используются одна функция - connect. Аргументами этой функции являются другие две функции, которые реализуются самим пользователем библиотеки и компонент которому требуется получить доступ к хранилищу. Так же необходимо было обернуть корневой React-компонент в компонент Provider, главной задачей которого являлось обеспечение дочерним React-компонентам доступа к хранилищу.

Actions и reduser находятся в папках с одноимённым названием и импортируются в корневой компонент.

Для реализации пользовательского интерфейса были созданы компоненты с помощью фреймворка React в сочетании с библиотекой reactstrap:

* Root. Корневой компонент, отвечающий за отображения всего SPA. Программная корневого компонента продемонстрирована на рисунке 2.3;



Рисунок 2.3 – Разметка корневого компонента

* Header. Компонент используется в корневом компоненте Root. Шапка сайта. Пользовательский интерфейс продемонстрирован на рисунке 2.4. Необходима для осуществления навигации. Компонент подписан на изменения состояния приложения с помощью метода store.subscribe для корректного отображения счётчика товаров в корзине при их добавлении и удалении. Программная реализация подписки на состояние продемонстрирована на рисунке 2.5. Разметка компонента показана на рисунке 2.6.



Рисунок 2.4 - пользовательский интерфейс компонента Header

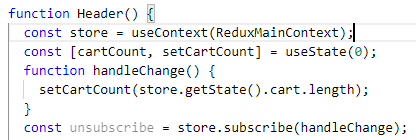


Рисунок 2.5 – Реализация подписки на состояние



Рисунок 2.6 – разметка компонента Header

* Menu. Необходимо для изменения фильтрации и поиска товаров. Принимает состояние фильтрации категорий из корневого компонента, сроку поиска, а так же функции обрабатывающие изменение фильтрации. Пользовательский интерфейс компонента Меню показан 2.7. Разметка компонента показана на рисунке 2.8.

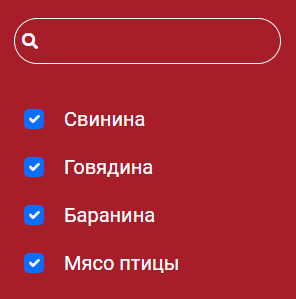


Рисунок 2.7 - Пользовательский интерфейс компонента Menu

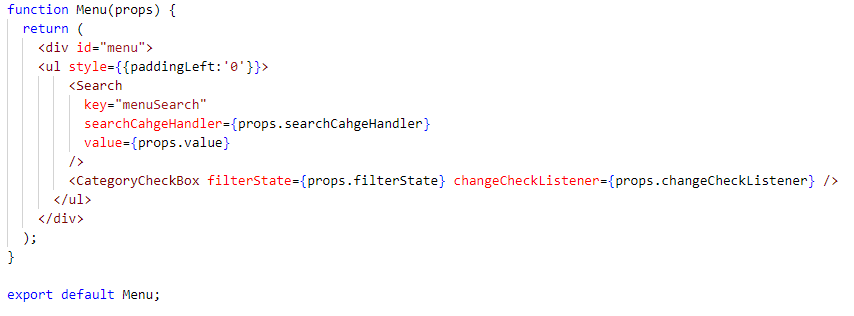


Рисунок 2.8 – Реализация компонента Menu

Так же внутри компонента меню отдельно были объявлен компонент поиска Search и компонент с флагами фильтрации CategoryCheckBox. Кроме меню данные компоненты нигде не используются, и созданы для возможности их повторного использования при модернизации приложения. На рисунках 2.9 и 2.10 показана программная реализация компонентов CategoryCheckBox и Search.



Рисунок 2.9 – Реализация компонента CategoryCheckBox

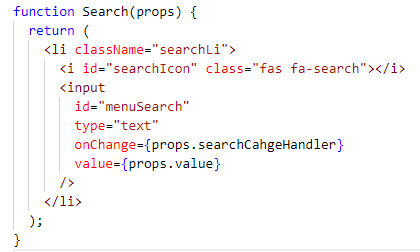


Рисунок 2.10 – Реализация компонента Search

Так как состояние фильтрации вынесено в корневой компонент, для последующей передачи его в каталог, функционал этих компонентов так же вынесен в корневой компонент, и в дальнейшем передаётся через пропсы. Функционал изменения компонентов фильтрации продемонстрирована на рисунке 2.11.

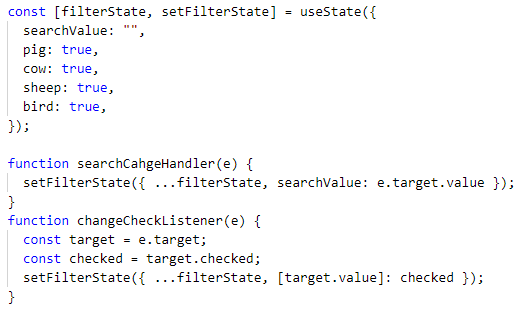


Рисунок 2.11 – Функционал изменения компонентов фильтрации

* ProductList. Компонент, который необходим для реализации отображения каталога. Разметка компонента продемонстрирована на рисунке 2.12. Этот компонент принимает состояние фильтрации, а так же массив объектов, содержащих лист с продуктами, которые нужно отобразить. На одной странице отображается 9 карточек продуктов. Функционал компонента продемонстрирован на рисунке 2.12. При рендере компонента ProductList начинается перебор элементов полученных в пропсе prodList и сравнение их категории с состоянием фильтрации, которое так же передаётся через пропсы от корневого элемента. Помимо этого, так же осуществляется поиск строки фильтрации в названии продукта. Отображаются только те продукты которые удовлетворяют всем вышеперечисленным условиям. Пользовательский интерфейс компонента ProductList продемонстрирован на рисунке 2.14

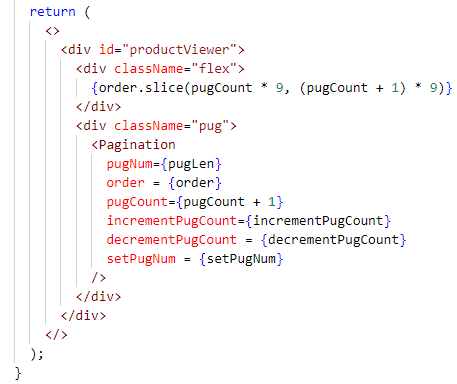


Рисунок 2.12 - Разметка компонента ProductList



Рисунок 2.13 - Функционал компонента ProductList



Рисунок 2.14 - пользовательский интерфейс компонента ProductList

Сам список продуктов хранится в файле prodist.json в формате JSON.

В файле ProductList.js так же созданы ещё 2 компонента, которые на данный момент используются только в нём. Это компонент Pagination и компонент ProductCard.

ProductCard был вынесен в отдельны компонент, так как содержит повторяющийся код. ProductCard принимает цену, название и ссылку на картинку продукта и создаёт карточку товара. Компонент имеет 2 состояния inCart и count. Состояние inCart хранит булевую переменную, которая отвечает на наличие данного товара в корзине. От этого состояние зависит отрисовка кнопки и возможность изменения количества товара в счётчике. Состояние count хранит количество товара установленное в счётчике, используется при добавлении товара в корзину. Введя несколько операторов if, была исключена возможность добавление товара в корзину при установке значения счётчика на 0, установке значения счётчика меньше нуля или установке значения счётчика на нечисловое значение. Функционал и разметка компонента ProductCard показана на рисунках 2.15 и 2.16.

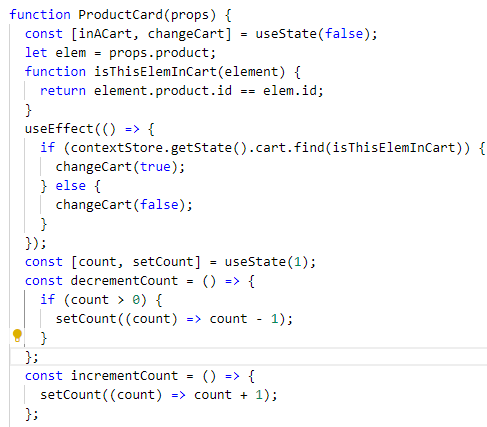
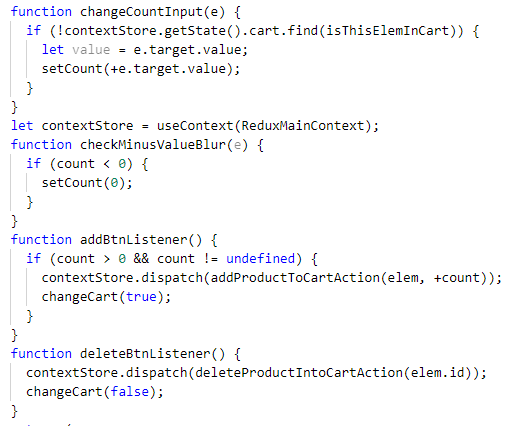
 

Рисунок 2.15 – функционал компонента ProductCard



Рисунок 2.16 – Разметка ProductCard

Компонент Pagination отвечает за пагинацию, которая нужна для удобной навигации, и избавление браузера клиента от необходимости загружать фото всех товаров сразу. Реализация и пользовательский интерфейс компонента Pagination показана на рисунках 2.17 и 2.18.



Рисунок 2.17 – Реализация компонента пагинации



Рисунок 2.18 - Пользовательский интерфейс компонента пагинации

* About. Отображает описание условий оплаты и доставки, слайдер, фрейм с местоположением магазина на карте, а так же контактную информацию. Разметка и отображение компонента About продемонстрирована на рисунках 2.19 и 2.20

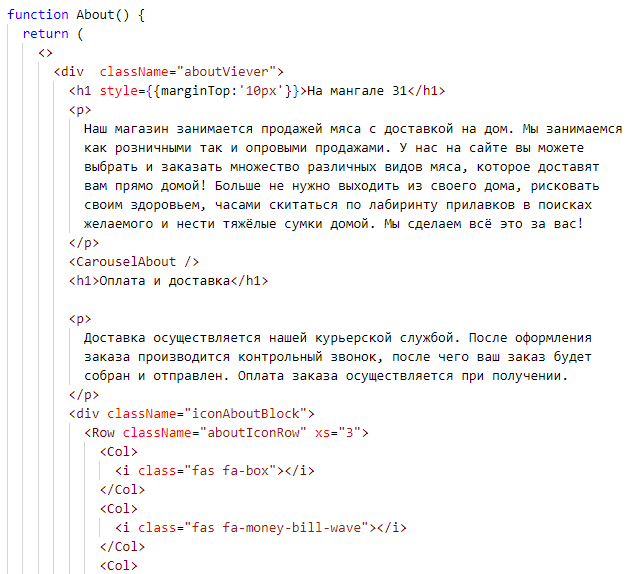


Рисунок 2.19 – Разметка компонента About

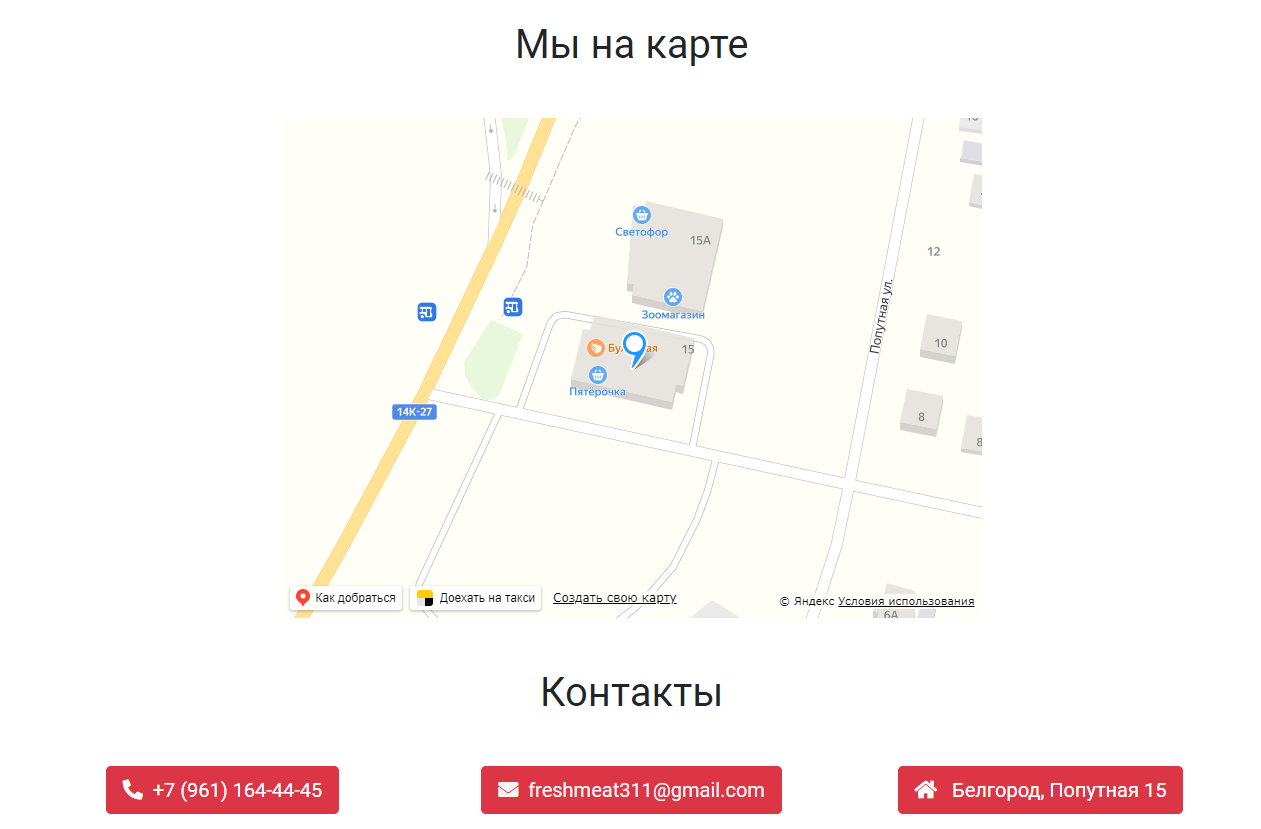
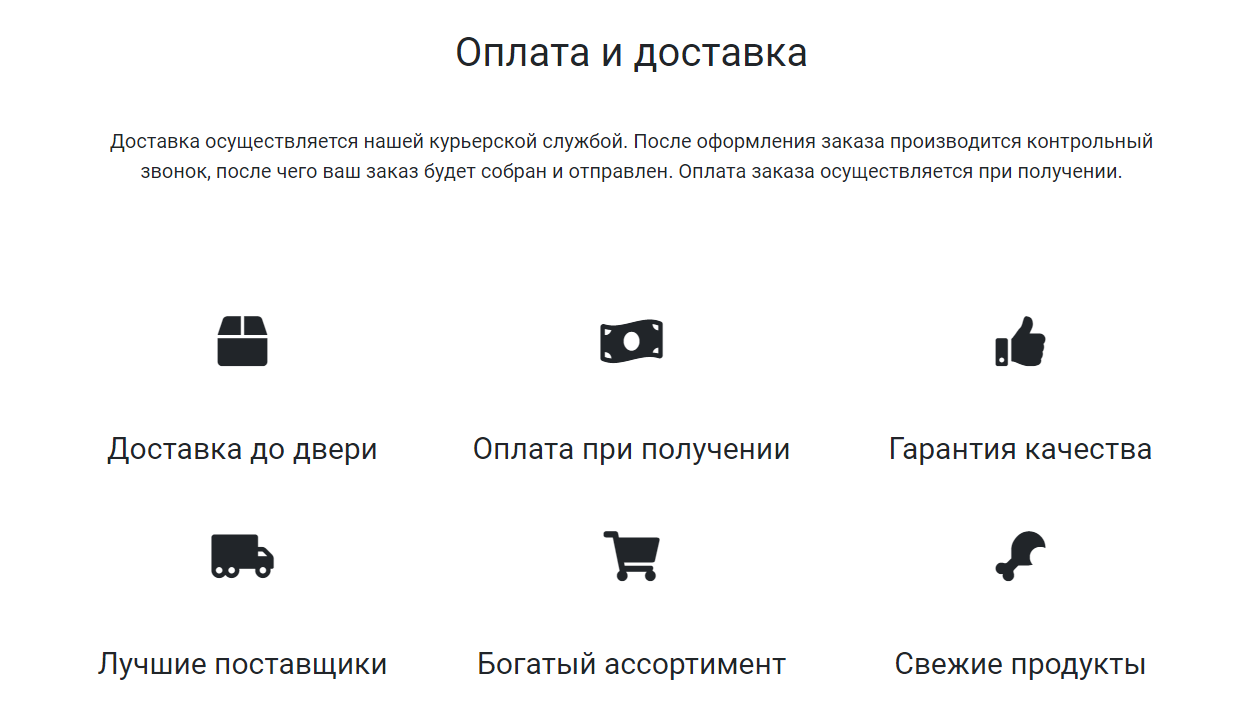


Рисунок 2.20 – Отображение компонента About

* Cart. Этот компонент необходим для отрисовки товаров добавленных в корзину. Он получает продукты находящиеся в корзине из общего состояния приложения и отображает их в виде таблицы, подсчитывая общую стоимость как каждой позиции, так и всех товаров, находящихся в корзине. Так же после таблицы находится кнопка, позволяющая пользователю перейти к форме оформления заказа. Если количество товаров в корзине равно нулю, то отображается текст, сообщающий о том что корзина пуста.

В таблице отображается картинка товара, название, цена, количество, стоимость. Пользовательский интерфейс компонента Корзина показан на рисунке 2.21.

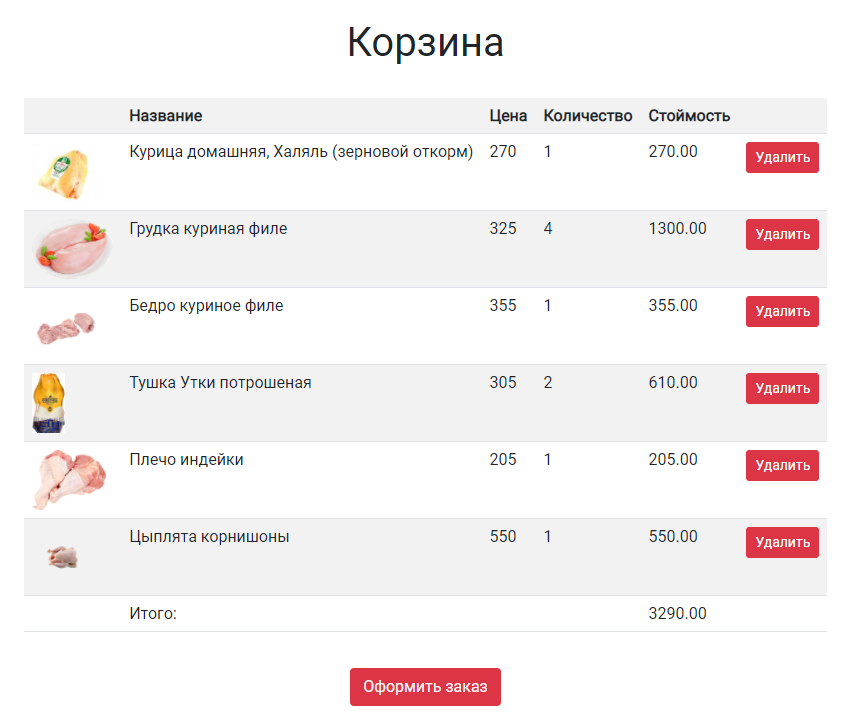


Рисунок 2.21 – Пользовательский интерфейс компонента Корзина

В файле Cart.js помимо компонента Cart так же хранятся ProductPosition и DeleteBtn. ProductPosition хранит строку с информацией о продукте и кнопкой удаления товара из корзины. Код компонента ProductPosition был вынесен в отдельный компонент так как это шаблонный, повторяющийся код, который мы используем несколько раз. В компонент DeleteBtn был вынесен функционал удаления так как с доработкой нашего SPA этот компонент может быть повторно использован, а так же при выносе кнопки удаления в отдельный компонент, появляется возможность более удобно реализовать изменения рендера компонента кнопки при удалении товара из корзины. Реализация компонентов DeleteBtn и ProductPosition продемонстрирована на рисунке 2.22 и 2.23. Функционал и разметка компонента Cart показана на рисунках 2.24 и 2.25

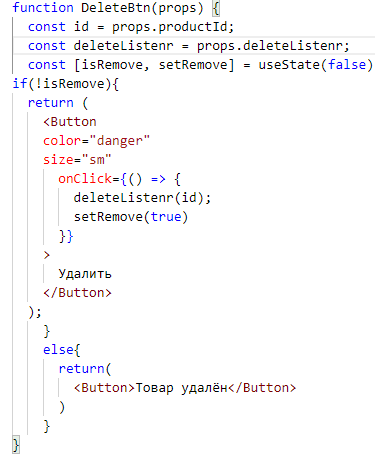


Рисунок 2.22 – Реализация компонента DeleteBtn

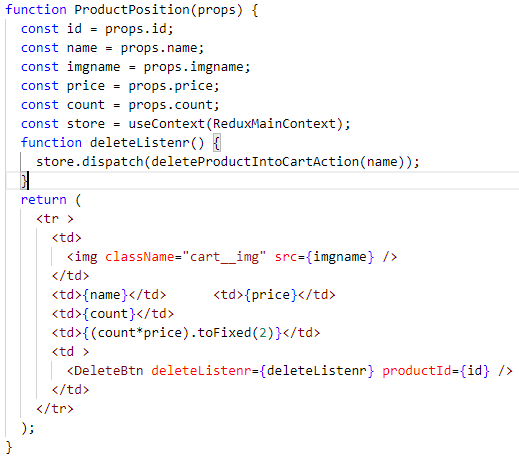


Рисунок 2.23 - Реализация компонента ProductPosition

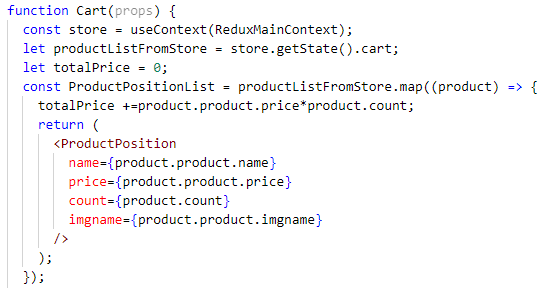


Рисунок 2.24 - Функционал компонента Cart

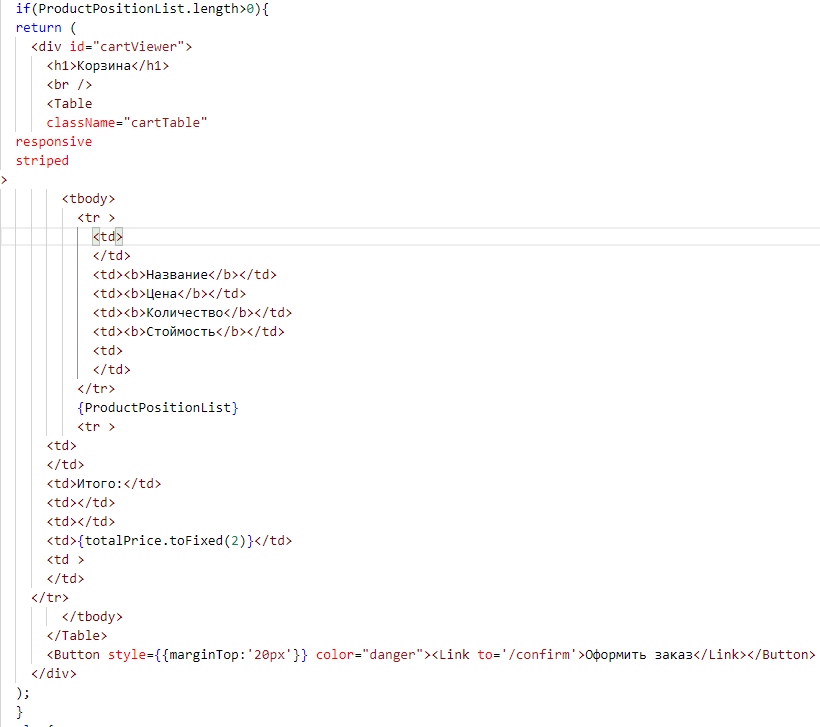


Рисунок 2.25 - Разметка компонента Cart

* OrderConfirm. Компонент отображает форму оформления заказа, собирает данные о клиенте и корзину из состояния приложения в 1 объект и отправляет его на сервер. Попасть на страницу в формой можно только из корзины, при наличии в ней товаров. Это сделано, что бы не перегружать пользовательский интерфейс, и исключить возможность оформления заказа с пустой корзиной. Пользовательский интерфейс и функционал компонента OrderConfirm показан на рисунках 2.26 и 2.27.

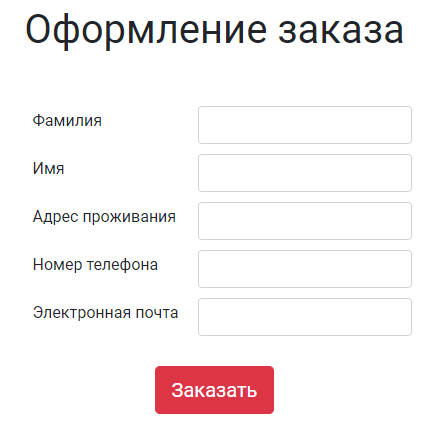


Рисунок 2.26 – Пользовательский интерфейс компонента OrderConfirm



Рисунок 2.27 – Функционал компонента OrderConfirm

При формировании объекта заказа, перед отправкой на сервер определяются поля, хранящие данные о клиенте, значения для которых берётся из полей формы, корзину, данные о которой берутся из redux-хранилища, поле isSended показывает статус отправки заказа и по умолчанию принимает значение false.

Для максимально удобного общения с сервером был создан пользовательский React-хук useHttp. Он возвращает 2 состояния и 2 метода. loading – состояние хранящее булевую переменную. По нему можно определить, когда сервер обработал запрос и прислал ответ. Error – массив, хранящий сообщения об ошибках. clearError – метод отчищающий массив с ошибками. Request – метод принимающий адрес, метод, тело запроса и заголовки. Необходим для отправки асинхронных запросов на сервер. Реализация пользовательского хука показана на рисунке 2.28.

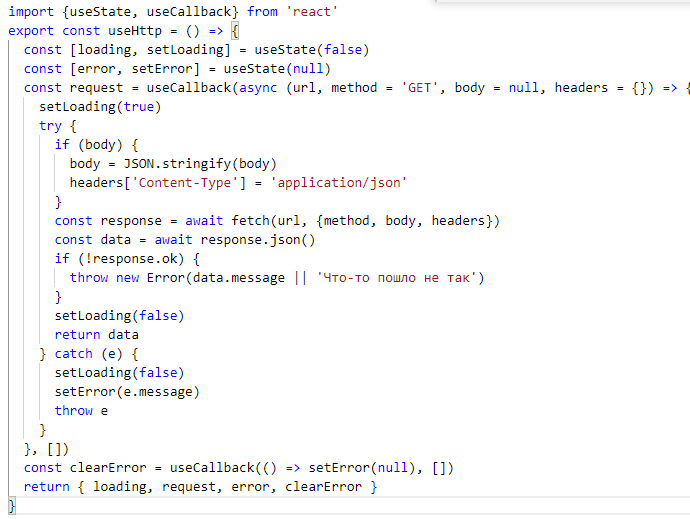


Рисунок 2.28 – Реализация пользовательского хука

Все React-компоненты хранятся в папке «toddo». Каждый компонент находится в папке с соответствующим компоненту названием. Рядом с каждым компонентом, находится файл со стилями, которые дальнейшем в каждый компонент. Содержимое папки todo показано на рисунке 2.29.



Рисунок 2.29 – Содержимое папки todo

После запуска команды npm run build всё SPA собирается в папку build.

В корневой папке проекта находятся серверные файлы. Для работы сервера используется фреймворк express. Запуск приложения осуществляется на системном порту, или порту указанном в конфигурационном файле, или на порту 8080 если системный порт не определён. Запуск сервера показан на рисунке 2.30.



Рисунок 2.30 – запуск сервера

Так же на сервере используется пакет config, который позволяет вынести конфигурационные переменные в отдельный файл. Помимо этого, использование конфигурационного файла позволяет “спрятать” некоторую информацию от чужих глаз. Например ссылку подключения к базе данных. Пример получения данных с помощью пакета config продемонстрирован на рисунке 2.31.



Рисунок 2.31 – Пример получения данных с помощью пакета config

Сервер обрабатывает 1 вариант GET запроса и 1 вариант POST запроса.

GET запрос отрабатывается в серверном файле index.js и возвращает index.html из папки client/build.

Для обработки запросов по путям была создана папка routes. Файл обработки добавления заказов был вынесен в файл order.routers.js. Он импортирует файл Order.js из папки models. Подключение роутера обработке заказов показано на рисунке 2.32.



Рисунок 2.32 - Подключение роутера обработке заказов

В папке models хранятся модели, описывающие типы и флаги полей перед занесением их в базу данных. Модель заказа показана на рисунке 2.33.

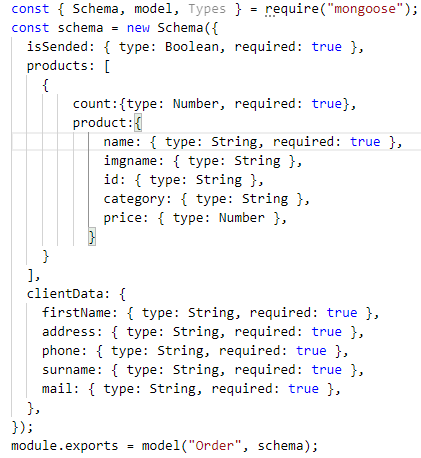


Рисунок 2.33 – Модель заказа

После получения запроса сервер получает основные поля заказа из тела запроса, после чего заносит их в базу данных. В случае успешного добавления заказа в базу данных сервер присылает ответ “Заказ добавлен”. В случае ошибки, которая например может возникнуть, если тип поля объекта, при занесении в БД не соответствует типу поля модели или если не определено обязательное поле, то сторона клиента получит сообщение об ошибке. Роутер обработки добавления заказа показан на рисунке 2.34.



Рисунок 2.34 – Роутер обработки добавления заказа.

В качестве базы данных была выбрана Mongodb Atlas. Она проста в обращении, входит в стек технологий Mern, а так же позволяет бесплатно разместить рабочую удалённую базу данных. Пример записи в базе данных продемонстрирован на рисунке 2.35.

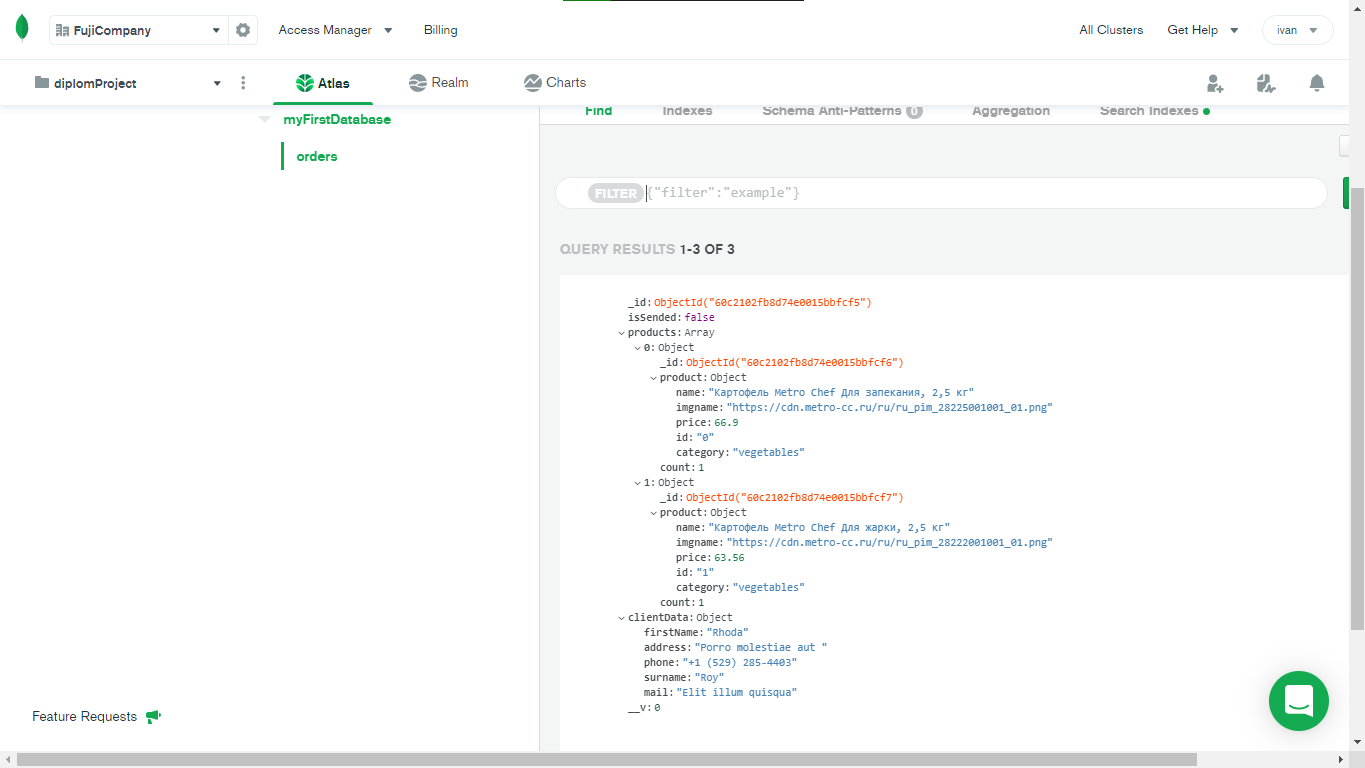


Рисунок 2.35 - Пример записи в базе данных.

Все части нашего приложения написаны на JavaScript, за счёт чего очень просто наладить коммуникацию между разными частями приложения. В этом и заключается главное преимущество стека технологий Mern.

Так же после оформления заказа, приходит уведомление на почту, указанную пользователем. В письме содержится информация о заказе. Модуль оправки находится в файле mailer.js. При написании модуля использовалась библиотека nodemailer. Код и пример письма изображён на рисунках 2.36 и 2.37.



Рисунок 2.36 – код модуля оправки сообщений

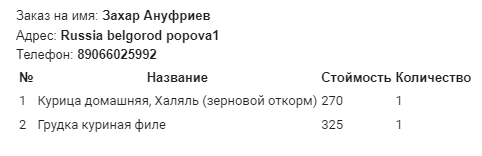


Рисунок 2.37 – пример сообщения с информацией о заказе

## 2.3 Размещение программного продукта в сети Интернет

Чтобы пользователь мог воспользоваться информационным продуктом, его нужно разместить в сеть интернет. Для этого существуют хостинги.

Хостинг – это круглосуточно включенный компьютер со специализированными программами и постоянным доступом в интернет, на жёстком диске которого размешен Web-сайт.

Существуют несколько типов хостинга:

* Виртуальный хостинг

Подходит для средних и небольших сайтов. Сама о себе площадка представляет сервер на котором размещено огромное количество сайтов, где все пользователи имеют одинаковые возможности и ограниченное установленное программное обеспечение.

Плюсы:

* + Дешевизна. Обычно такие сервисы берут не самую большую цену за обслуживание, что позволяет практически каждому пользователю или же компании не тратясь содержать свой Web-продукт;
  + Упрощённая панель с удобным интерфейсом загрузки файлов созданию резервной копии сайта и др функциями;
  + Простота в использовании. Обычно это небольшой набор удобно расположенных функций, чтобы не путать неопытных пользователей.

Минусы:

* + ограниченное место на жёстком диске;
  + отсутствие гарантии стабильной работы сайта. Чему служат перебои и медленная загрузка из-за нагрузок на сервер;
  + Возможность взлома и заражения вирусами, из-за огромного количества Web-сайтов расположенных на этом сервере.
* Виртуальный выделенный сервер

Предполагает выделение части удалённого сервера для каждого клиента. Такой вид хостинга подходит для средних и крупных Web-сайтов. Стоимость же будет значительно выше чем для виртуального хостинга.

Плюсы:

* + Для каждой виртуальной машины устанавливается отдельная ОС;
  + Гарантированные значения вычислительных мощностей;
  + Есть возможность для установки предпочитаемого ПО;
  + Полная независимость от других клиентов.

Минусы:

* + Необходимы навыки системного администратора;
  + Также остаётся вероятность взлома и заражения вирусами, но гораздо безопаснее виртуального хостинга.
* Выделенный сервер

Это отдельный удалённый сервер, который может контролировать сам пользователь. В обязанности же компании-хостера будет входить лишь администрирование аппаратного обеспечения. Подходит для масштабных интернет проектов.

Плюсы:

* + Отдельный сервер под вашим полным контролем;
  + Безопасность высокого уровня.

Минусы:

* + Высокая стоимость
  + Так как настройки и управление требует специализированных знаний, требуется наличие опытного системного администратора.
* Колокейшн

В общем этот хостинг похож на предыдущий с тем отличием, что хост провайдер предоставляет лишь выделенный интернет канал и специализированное помещение для установки оборудования.

* Облачный сервер

Это вид хостинга отличается от выделенного тем что, у него отсутствует конкретные тарифы по используемым ресурсам. Он определяется облачными технологиям. Облачные технологии – это одновременное использование ресурсов нескольких серверов. Хостинг использует ресурсы целой сети серверов, такая сеть может объединять более 100 единиц и все взаимосвязаны. При выходе из строя хотя бы одного сервера, вместо него работает следующий, а при увеличении нагрузки подключаются сервера, которые разгружают сеть, что гарантирует почти 100% доступность сайта.

Плюсы:

* + По необходимости можно увеличить все необходимые параметры путём задействования дополнительных мощностей.
  + Гарантированный круглосуточный доступ к сайту, благодаря использованию сети серверов.
  + Минусов у облачного сервера нет.
* Бесплатный хостинг

Это вид хостинга, обычно предоставляющийся крупными хост-провайдерами в ознакомительных целях. Такие хостинги крайне ненадёжны и ограничены сроком использования. Подходит только для пользователей, которые начинают писать сайт

Для публикации нашего проекта была выбрана платформа Heroku. Она поддерживает множество языков программирования, в том числе nodejs и предоставляет бесплатный домен. В файле package.json расположены команды, которые сервер начинает выполнять для установки проекта. Скрипты установки и запуска проекта показаны на рисунке 2.36.

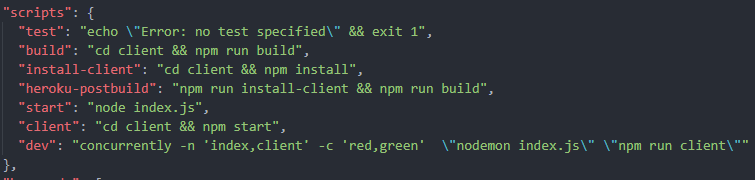


Рисунок 2.38 - Скрипты установки и запуска проекта

Проект был опубликован по адресу “https://freshmeat.herokuapp.com/”. Процесс публикации был простым и комфортным.

## 2.4 Руководство пользователя

Интернет-сайт предназначен для продажи продуктов. Для заказа продуктов для начала необходимо выбрать их количество и добавить их в корзину, как показано на рисунке 2.37.



Рисунок 2.39 - создание задачи

После добавления необходимых товаров в корзину, пользователь может перейти на страницу с корзиной и при желании удалить ненужные товары, после чего перейти на страницу оформления заказа, как показано на рисунке 2.38.

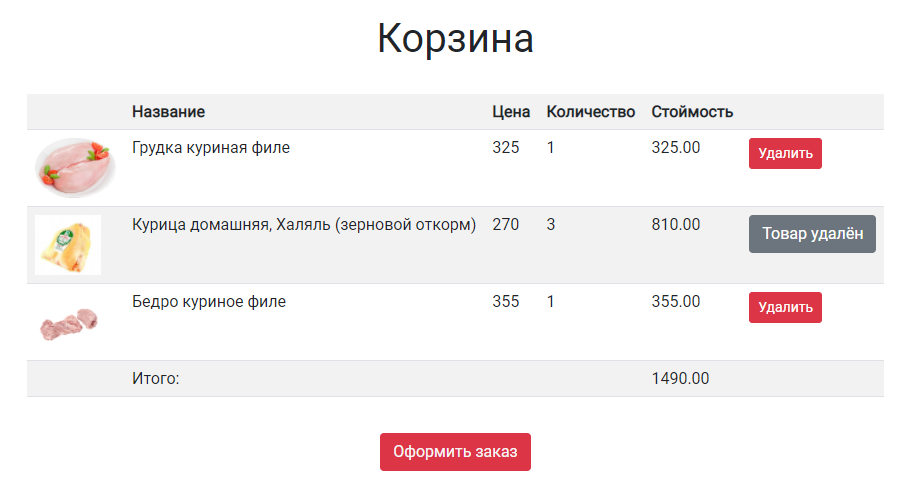


Рисунок 2.40 – Корзина

После клика по кнопке “Оформить заказ” пользователь попадает на форму оформления заказа. После заполнения всех полей и клика на кнопку “Заказать”, пользователь получает сообщение от сервера об исходе добавления заказа Это показано на рисунке 2.39.

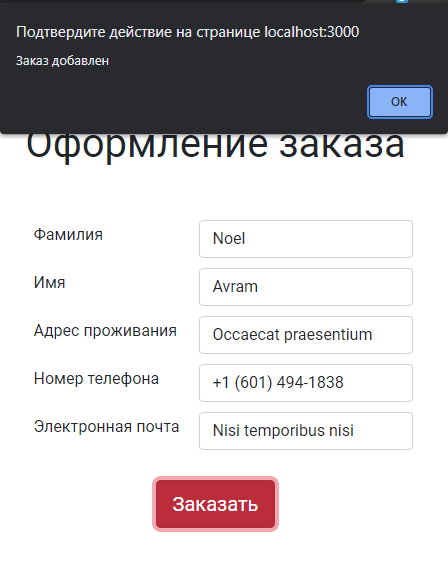


Рисунок 2.41 – Ответ от сервера при добавлении заказ

Так же интернет-сайт позволяет производить фильтрацию и поиск продуктов в каталоге. Для этого пользователю необходимо отметить флаги с соответствующими категориями и ввести слово в строку поиска. Это показано на рисунке 2.40.

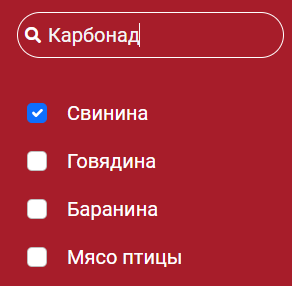


Рисунок 2.42 – Поиск и фильтрация

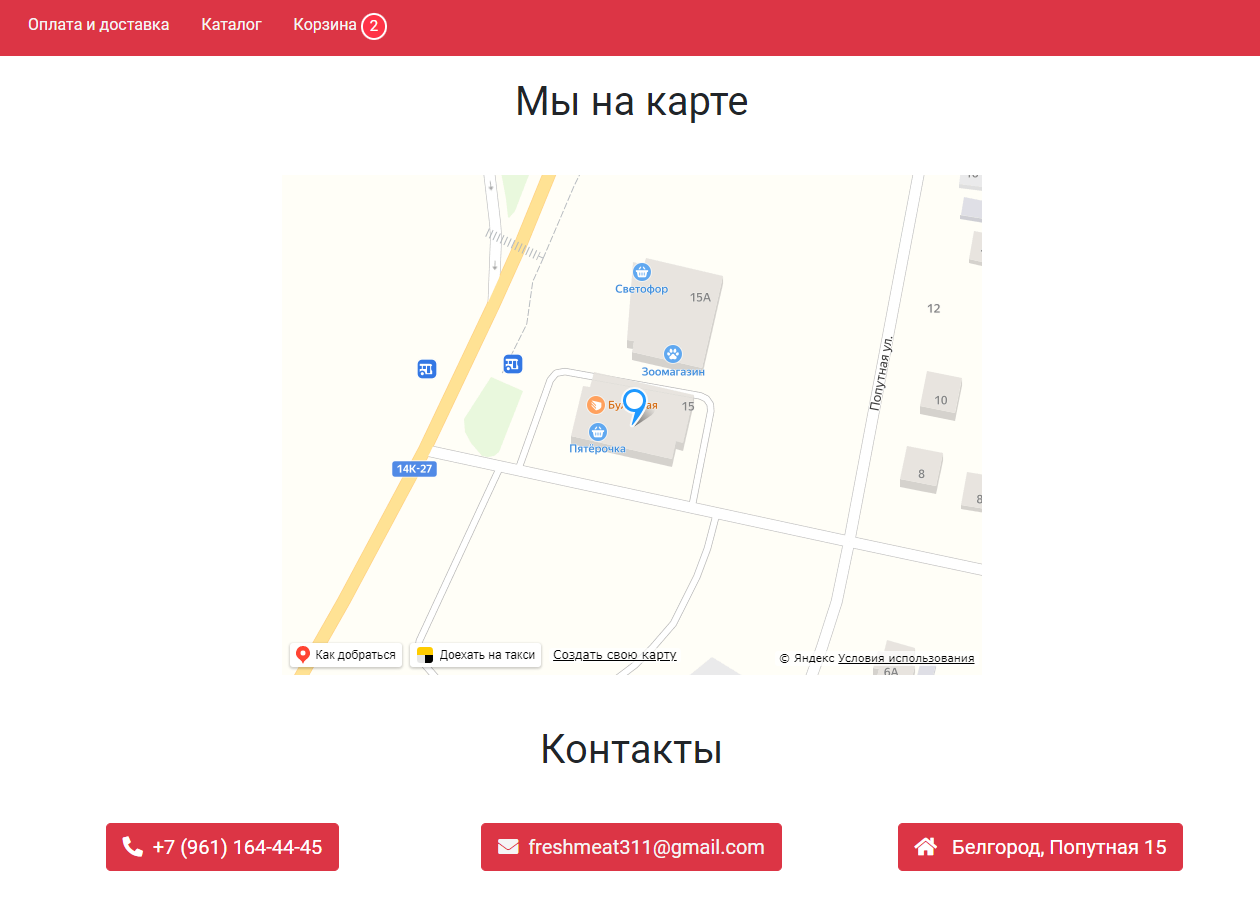
Так же через меню пользователь может попасть на страницу с информацией о компании. Это показано на рисунке 2.41. 

Рисунок 2.43 – Компонент “О нас”

# 3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

В экономической части описываются затраты разработчика на теоретические исследования, постановку задачи, проектирование, разработку алгоритмов и программ, отладку, опытную эксплуатацию, оформление документов.

Себестоимость представляет собой сумму затрат на разработку и расходов на содержание и эксплуатацию оборудования, используемого при создании программного продукта. Расчёт себестоимости произведём по следующим видам затрат:

а) Материальные затраты;

б) Затраты на оплату труда разработчика;

в) Отчисления на социальные нужды разработчика;

г) Амортизация;

д) Расходы на содержание и эксплуатацию ПЭВМ;

е) Прочие затраты.

Фактическая трудоёмкость по стадиям проектирования представлена в таблице 3.1.

Таблица 3.1 – Затраты времени на различных стадиях разработки ПО

|  |  |  |  |
| --- | --- | --- | --- |
| Стадия НИР | Содержание работ | Трудоёмкость | |
| Дни | % |
| 1 | 2 | 3 | 4 |
| 1. Техническое задание | Подбор и изучение литературы, анализ составления вопроса, согласование с руководителем и утверждение технического задания и плана работ. Обоснование принципиальной возможности решения поставленной задачи. Постановка задачи. | 2 | 6,5 |
| 2. Эскизный проект | Теоретическая разработка темы. Предварительная разработка структуры входных и выходных данных. Разработка общего описания алгоритма решения задачи. | 2 | 8,1 |
|  |  |  |  |
| Продолжение таблицы 3.1 | | | |
| 1 | 2 | 3 | 4 |
| 3. Технический проект | Проектирование. Определение основных блоков, классов, объектов Разработка сценария взаимодействия объектов. | 2 | 9,7 |
| 4. Рабочий проект | Написание и отладка программ. Тестирование и сборка системы. | 13 | 53,1 |
| 5. Внедрение | Подготовка инструкций пользователям, написание, оформление и защита отчёта (дипломного проекта). | 5 | 22,6 |
| Итого: |  | Tоб=24 | 100 |

Материальные затраты (МЗ) для разработки программного продукта складываются из затрат на расходные материалы, которые берутся по факту и определяются исходя из реальной стоимости (таблица 3.2):

Таблица 3.2 – Расчёт стоимости расходных материалов

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Кол-во | Цена, ед., руб. | Всего, руб. |
| 1 | 2 | 3 | 4 |
| Картридж | 1 | 250,00 | 250,00 |
| СD-R диск | 1 | 30,00 | 30,00 |
| Бумага | 1 | 200,00 | 200,00 |
| Ручка шариковая | 1 | 20,00 | 20,00 |
| Дипломная папка | 1 | 280,00 | 280,00 |
| Итого: | 5 | 780,00 | 780,00 |

Заработная плата исполнителя работ по созданию ПП складывается из основной заработной платы (ОЗПр) разработчика за время разработки ПП и дополнительной заработной платы. (ДЗПр) (формула 3.1).

ЗПобщ.р = ОЗПр + ДЗПр, (3.1)

где ОЗПр - основная заработная плата разработчика, руб.

Основная заработная плата программиста за период разработки программы вычисляется по формуле 3.2:

ОЗПр = ЧТСр **×** Тпр, (3.2)

где ЧТСр - заработная плата программиста за один час работы, руб.

Заработная плата программиста за один час работы определяется исходя из заработной платы разработчика за месяц по формуле 3.3:

ЧТСр = ЗПр/м / (N **×** Тд), (3.3)

где ЗПр/м - заработная плата программиста за месяц, руб. (принять МРОТ по Белгородской области на текущий год).

N - количество рабочих дней в месяце, дни (принять 22 дня).

Тд - продолжительность рабочей смены, час. (принять 8 час.)

ДЗПр - дополнительная заработная плата разработчика.

ЧТСр = 12 130 / (22 \* 8) = 12 130 / 176 = 68,92 руб.

Вычисляется заработная плата программиста за период разработки программы по формуле 3.2:

ОЗПр = 68,92 \* (24 \* 8) = 68,92 \* 192 =13 232,64 руб.

Дополнительная заработная плата включает выплаты, предусмотренные действующим законодательством за неотработанное время. Рассчитывается в процентах от основной заработной платы (13%).

ДЗПр = 13 232,64 \* 13/100 = 1720,24 руб.

Заработная плата исполнителя работ по созданию ПП вычисляется по формуле 3.1:

ЗПобщ.р = 13 232,64 + 1720,24 = 14 952,88 руб.

Отчисления на социальные нужды (ОСНр) устанавливаются в процентах от расходов на оплату труда. (30%)

ОСНр = 14 952,88 \* 30 / 100 = 4 485,86 руб.

Сумма амортизации за период разработки ПП вычисляется линейным методом по формуле 3.4:

, (3.4)

где На – годовая норма амортизации, % рассчитывается по формуле 3.5;

(3.5)

где Тн – нормативный срок службы ПК, год;

Соб – балансовая стоимость ПЭВМ, руб.;

Тм – машинное время, затрачиваемое на создание ПП, час.

Фд – годовой фонд рабочего времени оборудования, час. Определяется по формуле 3.6:

Фд = ((365-С-В-Пр) **×** 8-ППр **×** 1) **×** S **×** (1-а/100), (3.6)

где 365 – количество календарных дней в году;

С, В, Пр – количество нерабочих дней в году: субботних, воскресных и праздничных в расчетном периоде;

ППр – продолжительность рабочей смены, ч;

S – количество смен работы в сутки;

ППр – количество предпраздничных дней в году.

а – процент потерь времени на ремонт оборудования (принять а = 3-5%)

На ис = НМА /12, (3.7)

где НМА – балансовая стоимость инструментальных средств, руб.;

12 – 12 месяцев нормативный срок службы НМА.

Таблица 3.3 – Стоимость оборудования

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование оборудования | Количество, шт. | Первоначальная стоимость, руб. | Общая стоимость, руб. |
| Компьютер | 1 | 31 000,00 | 31 000,00 |
| Принтер | 1 | 5 000,00 | 5 000,00 |
| Итого: | | | 36 000,00 |

Стоимость инструментальных средств (таблица 3.4) включает в себя стоимость системного программного обеспечения, использованного при разработке проекта в размере износа за этот период.

Таблица 3.4 ­– Стоимость инструментальных средств

|  |  |
| --- | --- |
| Программное обеспечение | Стоимость, руб. |
| MacOS Catalina 10.15.2 | ­– |
| Microsoft Office 2018 | 2 800,00 |
| Итого: | 2 800,00 |

На об = 1 / 4 **×** 100% = 25%

На ис = 2 800 / 12 = 233,33 руб.

Фд = ((365 - 118) **×** 8 - 6 **×** 1) **×** 1**×** (1 - 0,05) = 1871 час.

Вычисляем сумму амортизации за период разработки ПП линейным методом по формуле 3.4:

Тм = 22 × 8 = 176

Ао = (25% × 36 000,00 × 176) / (100 % × 1871 ) = 846,61 руб.

Основой для расчёта расходов на содержание и эксплуатацию ПЭВМ относящихся к данной ПП является себестоимость 1-го машино-часа работы ПЭВМ, которая включает:

а) Основную заработную плату работников, обеспечивающих функционирование ПЭВМ. К их числу относятся, например, инженер-электрик, инженер по обслуживанию ПЭВМ, системный программист, оператор и т.д.

Заработная плата обслуживающего персонала рассчитывается по формуле 3.8:

ОЗПоп = 12 **×** ЗПоп / n, (3.8)

где ЗПоп – заработная плата обслуживающего персонала по категориям работников, руб./мес. (принять МРОТ по Белгородской области на текущий год);

n – количество обслуживаемых ПЭВМ (принять = 10 шт).

ОЗПоп = 12 × 12 130,00 /10 = 14 556,00 руб.

б) Дополнительную заработную плату обслуживающего персонала (ДЗПоп) берётся в процентах от основной (13%);

ДЗПоп = 14 556,00 × 13/100 = 1 892,28 руб.

в) Отчисления на социальные нужды (ОСНоп) устанавливаются в процентах от расходов на оплату труда (30%);

ОСНоп = (12 130 + 1 892,28) \* 0,3 = 4 206,68

г) Затраты на электроэнергию складываются из:

* Затраты на силовую электроэнергию;
* Затраты на электроэнергию, которая идёт на освещение.

Затраты на силовую электроэнергию определяются по формуле 3.9:

Зс.эл = Мпр **×** Фд **×** Сквт/ч, (3.9)

где Мпр – электроэнергия, потребляемая вычислительной машиной, квт/час (принять = 1,21 квт/час);

Сквт/ч – стоимость 1 квт/час (тарифы на электроэнергию для юридических лиц по Белгородской области на текущий период).

Зс.эл = 1,21 × 1871 × 3,92 = 8 874,52 руб.

Затраты на электроэнергию, которая идет на освещение определяется по формуле 3.10:

Зосв = Фд **×** Мосв **×** Сквт/ч (3.10)

Мосв – суммарная мощность, которая идет на освещение, квт/час (принять = 0,15 квт/час).

Зосв = 1871 × 0,15 × 3,92 = 1 100,14 руб.

Общие затраты на электроэнергию определяются по формуле 3.11:

Зэл = Зс.эл + Зосв (3.11)

Зэл = 8 874,52 + 1 100,14 = 9 974,66 руб.

д) Стоимость ремонта оборудования определяется в процентах от балансовой стоимости ПЭВМ по формуле 3.12:

Зрем = Соб **×** Нр / 100%, (3.12)

Нр – величина отпускаемых средств на ремонт вычислительной техники относительно стоимости этой техники, % (принять 2-4%);

Соб – балансовая стоимость ПЭВМ, руб.

Зрем = 36 000,00 × 2% = 720 руб.

Годовые расходы на содержание и эксплуатацию 1 ПЭВМ определяются по формуле 3.13:

Рс.э = ОЗПоп + ДЗПоп + ОСНоп + Зэл + Зрем (3.13)

Рс.э = 14 556 + 1 892,28 + 4 206,68 + 9 974,66 + 720 = 31 349.62 руб.

Себестоимость 1-го машино-часа работы ПЭВМ определяются по формуле 3.14:

Смч = Рс.э / Фд (3.14)

Смч = 31 349.62 / 1871 = 16,75 руб.

Расходы на содержание и эксплуатацию ПЭВМ, относящиеся к данному ПП определяется по формуле 3.15:

Рс.э.п = Смч **×** Тм (3.15)

Рс.э.п = 16,75 \* 176 = 2 948 руб.‬

Прочие расходы (ПР) определяются в процентах от основной заработной платы разработчика (ОЗПр), руб. (принять 30%).

ПР = 13 232,64 × 30 / 100 = 3 969,79 руб.

Итого себестоимость ПП составит (формула 3.16):

С = МЗ + ЗПобщ.р + ОСНр + Ао + На ис + Рс.э.п + ПР (3.16)

С = 780 + 14952,88 + 4485,86 + 846,61 + 233,33 + 2948 + 3969,79 = 28216,47 руб

На основании произведённых расчётов анализируется структура себестоимости (таблица 3.5).

Таблица 3.5 - Структура себестоимости программного продукта

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Сумма (руб.) | Структура (%) |
| 1 | 2 | 3 |
| Материальные затраты | 780 | 3 |
| Заработная плата программиста | 14 952,88 | 53 |
| Отчисления на социальные нужды | 4 485,86 | 16 |
| Амортизация ПЭВМ, инструментальных средств | 846,61 | 3 |
| Расходы на содержание и эксплуатацию ПЭВМ | 2 948 | 11 |
| Прочие расходы | 3 969,79 | 14 |
| Итого: полная себестоимость АИС | 28 216,47 | 100 |

Себестоимость созданного программного продукта составляет 28 216,47 руб. Так как, оплата труда не производилась, то реальная себестоимость определяется по формуле 3.17.

Ср = С - (ЗПобщ.р + ОСНр) (3.17)

Ср = 28 216,47 - (14 952,88 + 4 485,86) = 8 777,73 руб.

Следовательно, экономия денежных средств определяется по формуле 3.18.

Э = ЗПобщ.р + ОСНр (3.18)

Э = 14 952,88 + 4 485,86 = 19 438,74 руб.

Экономия денежных средств при разработке программного продукта составила 19 438,74 руб. (68, 89%).

Цена, определяется себестоимостью и прибылью. Величина прибыли составляет 30% от итога сметы, за минусом экономии денежных средств.

П = (С - Э) **×** 0,3 (3.19)

П = (28 216,47 - 19 438,74) \* 0,3 = 2 633,31 руб.

Цпр = (С - Э) + П (3.20)

Цпр = (28 216,47 - 19 438,74) + 2 633,31 = 11 411,04 руб.

Экономическая эффективность – оценочный показатель результативности деятельности фирмы, представляющий собой сопоставление результатов этой деятельности.

Экономическая эффективность исчисляется как отношение результатов к затратам, а экономический эффект – разница между результатами и затратами.

Ср. ок. = Цпр / П (3.21)

Ср. ок. = 11 411,04 / 2 633,31 = 4,3 мес.

Таким образом, период окупаемости программного продукта составит 4,3 месяцев.

# 4 МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ, ПРОТИВОПОЖАРНОЙ ТЕХНИКЕ И ОХРАНЕ ТРУДА ПРИ РАБОТЕ С ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКОЙ

Соблюдение правил безопасной работы является необходимым условием предупреждения производственного травматизма.

Данный раздел содержит общие указания по безопасному применению электрооборудования в учреждении, а также требования по обеспечению пожарной безопасности.

Требования к персоналу, эксплуатирующему средства вычислительной техники и периферийное оборудование:

* К самостоятельной эксплуатации электроаппаратуры допускается только специально обученный персонал не моложе 18 лет, пригодный по состоянию здоровья и квалификации к выполнению указанных работ;
* Перед допуском к работе персонал должен пройти вводный и первичный инструктаж по технике безопасности с показом безопасных и рациональных примеров работы. Затем не реже одного раза в 6 месяцев проводится повторный инструктаж, возможно, с группой сотрудников одинаковой профессии в составе не более 20 человек;
* Внеплановый инструктаж проводится при изменении правил по охране труда, при обнаружении нарушений персоналом инструкции по технике безопасности, изменении характера работы персонала.

Руководители структурных подразделений несут ответственность за организацию правильной и безопасной эксплуатации средств вычислительной техники и периферийного оборудования, эффективность их использования; осуществляют контроль за выполнением персоналом требований настоящей инструкции по технике безопасности.

Эксплуатирующий средства вычислительной техники и периферийное оборудование персонал может подвергаться опасным и вредным воздействия, которые по природе действия подразделяются на следующие группы:

* Поражение электрическим током;
* Механические повреждения;
* Электромагнитное излучение;
* Инфракрасное излучение;
* Опасность пожара;
* Повышенный уровень шума и вибрации.

Для снижения или предотвращения влияния опасных и вредных факторов необходимо соблюдать санитарные правила и нормы.

Гигиенические требования к видео дисплейным терминалам, персональным электронно-вычислительным машинам и организации работы.

При пользовании средствами вычислительной техники и периферийным оборудованием каждый работник должен внимательно и осторожно обращаться с электропроводкой, приборами и аппаратами и всегда помнить, что пренебрежение правилами безопасности угрожает и здоровью, и жизни человека.

Во избежание поражения электрическим током необходимо твердо знать и выполнять следующие правила безопасного пользования электроэнергией:

* Необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, и заземления. При обнаружении неисправности немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности.

Во избежание повреждения изоляции проводов и возникновения коротких замыканий не разрешается:

* Вешать что-либо на провода;
* Закрашивать и белить шнуры и провода
* Закладывать провода и шнуры за газовые и водопроводные трубы, за батареи отопительной системы;
* Выдёргивать штепсельную вилку из розетки за шнур, усилие должно быть приложено к корпусу вилки.
* Для исключения поражения электрическим током запрещается:
* Часто включать и выключать компьютер без необходимости;
* Прикасаться к экрану и к тыльной стороне блоков компьютера;
* Работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;
* Работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе;
* Класть на средства вычислительной техники и периферийном оборудовании посторонние предметы.

Запрещается под напряжением очищать от пыли и загрязнения электрооборудование.

Запрещается проверять работоспособность электрооборудования в неприспособленных для эксплуатации помещениях с токопроводящими полами, сырых, не позволяющих заземлить доступные металлические части.

Ремонт электроаппаратуры производится только специалистами-техниками с соблюдением необходимых технических требований.

Недопустимо под напряжением проводить ремонт средств вычислительной техники и периферийного оборудования.

Во избежание поражения электрическим током, при пользовании электроприборами нельзя касаться одновременно каких-либо трубопроводов, батарей отопления, металлических конструкций, соединённых с замлей.

При пользовании электроэнергией в сырых помещениях соблюдать особую осторожность.

При обнаружении оборвавшегося провода необходимо немедленно сообщить об этом администрации, принять меры по исключению контакта с ним людей. Прикосновение к проводу опасно для жизни.

Спасение пострадавшего при поражении электрическим током главным образом зависит от быстроты освобождения его от действия током.

Во всех случаях поражения человека электрическим током немедленно вызывают врача. До прибытия врача нужно, не теряя времени, приступить к оказанию первой помощи пострадавшему.

Необходимо немедленно начать производить искусственное дыхание, наиболее эффективным из которых является метод рот в рот или рот в нос, а также наружный массаж сердца.

Искусственное дыхание поражённому электрическим током производится вплоть до прибытия врача.

Требования по обеспечению пожарной безопасности.

На рабочем месте запрещается иметь огнеопасные вещества. В помещениях запрещается:

* зажигать огонь;
* включать электрооборудование, если в помещении пахнет газом;
* курить;
* сушить что-либо на отопительных приборах;
* закрывать вентиляционные отверстия в электроаппаратуре.
* Источниками воспламенения являются:
* искра при разряде статического электричества;
* искры от электрооборудования;
* искры от удара и трения;
* открытое пламя.

При возникновении пожароопасной ситуации или пожара персонал должен немедленно принять необходимые меры для его ликвидации, одновременно оповестить о пожаре администрацию.

# ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта было разработано веб-приложение, позволяющие удалённо приобретать мясные продукты.

Данный проект состоит из четырёх основных частей: общая часть, технологическая часть, экономическая часть и мероприятия по технике безопасности.

В общей части дипломного проекта велась постановка задачи и функциональные способности, которыми будет обладать продукт. Далее был проведён обзор рынка на предмет поиска программного продукта аналогичной тематики и обзор возможных средств для разработки программного продукта. Так же были рассмотрены тонкости организации собственной эффективности путём планирования.

В технической части представлено описание и все этапы проектирования и разработки программного продукта.

В экономической части рассматривалось технико-экономическое обоснование разработки программного обеспечения и расчёт затрат на разработку программного обеспечения. Так же была составлена смета затрат на разработку ПО, которая составила 28 216,47 руб., и расчёт срок окупаемости которого составил 4,3 месяцев.

Таким образом, можно сделать вывод, что поставленные цели и задачи успешно выполнены, программный продукт полностью готов к использованию.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Современный учебник JavaScript [Электронный ресурс] /. - Электрон. журн. - Режим доступа: https://learn.javascript.ru, свободный
2. React JavaScript-библиотека для создания пользовательских интерфейсов [Электронный ресурс] /. - Электрон. журн. - Режим доступа: https://ru.reactjs.org/, свободный
3. Redux A Predictable State Container for JS Apps [Электронный ресурс] /. - Электрон. журн. - Режим доступа: https://redux.js.org/, свободный
4. Алгоритмы: разработка и применение. Классика Computers Science / Пер. с англ. Е. Матвеева. - СПб.: Питер, 2016. - 800 с. ил
5. React и Redux: функциональная веб-разработка. - СПб.: Питер, 2018. - 336 с. ил
6. Миньков, С.Л. Технико-экономическое обоснование выполнения проекта: методическое пособие / С.Л. Миньков. – Томск: ТГУ, 2011. – 30 с. ил
7. ГОСТ 12.1.003-83 ССБТ. Шум. Общие требования безопасности, 1983. – 19с.
8. ГОСТ 12.1.004-85 ССБТ. Пожарная безопасность. Общие требования, 1985. – 13с.
9. ГОСТ 12.1.005-88 ССБТ. Воздух рабочей зоны. Общие санитарно-гигиенические требования, 1988. – 15с.
10. ГОСТ 12.1.019-79 ССБТ. Электробезопасность. Общие требования, 1979. – 15с.
11. ГОСТ 12.1.033-81 ССБТ. Пожарная безопасность объектов с электрическими сетями, 1981. – 23с.
12. ГОСТ 25861-83. Машины вычислительные и системы обработки данных. Требования электрической и механической безопасности и методы испытаний, 1983. – 17с.
13. СНиП 2.01.02-85. Противопожарные нормы и правила, 1985. – 37с.
14. СНиП-4-79 Естественное и искусственное освещение, 1979. – 14с