Министерство образования Белгородской области

Областное государственное автономное профессиональное образовательное учреждение

«Белгородский индустриальный колледж»

**«К ЗАЩИТЕ ДОПУЩЕН»**

зам. директора по УР

\_\_\_\_\_\_\_\_\_\_/Выручаева Н.В./

«\_\_» июня 2022 г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к дипломному проекту

Тема: **«Проектирование и разработка интернет-магазина по продаже автозапчастей»**

БИК О. 09.02.07 ДП 111756 ПЗ

|  |  |  |
| --- | --- | --- |
| РАЗРАБОТАЛ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Гергерт И.М./ |
| РУКОВОДИТЕЛЬ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Солдатенко М.Н./ |
| РЕЦЕНЗЕНТ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Рвачева Т.Н./ |
| Н.КОНТРОЛЬ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Третьяк И.Ю./ |
| ЗАВ. ОТДЕЛЕНИЕМ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /Беляева Г.Н./ |

Белгород 2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc105582558)

[1 ОБЩАЯ ЧАСТЬ 5](#_Toc105582559)

[1.1 Обоснование целесообразности и актуальности разработки интернет-магазина по продаже автозапчастей 5](#_Toc105582560)

[1.2 Обзор аналогичных веб-ресурсов 6](#_Toc105582561)

[1.3 Обзор средств разработки интернет-магазинов 8](#_Toc105582562)

[2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 15](#_Toc105582563)

[2.1 Обоснование и выбор программных средств реализации интернет-магазина 15](#_Toc105582564)

[2.2 Проектирование и разработка структуры интернет-магазина 17](#_Toc105582565)

[2.3 Проектирование структуры базы данных 19](#_Toc105582566)

[2.4 Программная реализация модулей приложения 20](#_Toc105582567)

[2.5 Тестирование программного продукта, размещение в глобальной сети 25](#_Toc105582568)

[3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 31](#_Toc105582569)

[4 МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ, ПРОТИВОПОЖАРНОЙ ТЕХНИКЕ И ОХРАНЕ ТРУДА ПРИ РАБОТЕ С ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКОЙ 44](#_Toc105582570)

[ЗАКЛЮЧЕНИЕ 47](#_Toc105582571)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48](#_Toc105582572)

[ПРИЛОЖЕНИЕ А 50](#_Toc105582573)

[ПРИЛОЖЕНИЕ Б 52](#_Toc105582574)

## ВВЕДЕНИЕ

В данном дипломном проекте рассматривается проектирование и разработка интернет-магазина по продаже автозапчастей.

В последнее время невозможно представить мир без современных технологий, возможностями которых пользуются практически все люди.

С каждым днем все больше и больше набирает популярность продажа услуг через интернет. Интернет-магазин является современным путем развития бизнеса, так как сегодня за какой-либо услугой человек в первую очередь будет искать информацию в интернете. Также интернет-магазин позволяет покупателю сэкономить время на поиски требуемого товара, а компании расширить область торговли, вне зависимости от расположения покупателя, отправить товар доставкой. Поэтому тема является достаточно актуальной.

Целью курсового проекта является разработка интернет-магазина по продаже автозапчастей.

Объектом исследования является процесс разработки интернет-магазинов.

Предметом исследования являются способы и методы разработки интернет-магазина по продаже автозапчастей

Для достижения поставленной цели необходимо решить следующие задачи:

* продумать и определить функционал к интернет-магазину;
* выполнить постановку задачи;
* описать предметную область;
* спроектировать структуру интернет-магазина;
* спроектировать базу данных интернет-магазина;
* реализовать интернет-магазин;

Данная работа состоит из:

* введения (в нем обосновывается актуальность выбранной темы и постановка задач);
* общей части (в этой части содержится описание актуальности темы дипломного проекта, анализ рынка на предмет аналогичных продуктов, основные технологии, с помощью которых планируется выполнить дипломный проект);
* технологической части (эта часть содержит описание выполнения всех этапов дипломного проекта. Для каждого этапа следует указать назначение, избранные методы решения, код программы и предоставить результаты выполнения этапа);
* заключения (в заключении подводятся итоги исследования и формулируются окончательные выводы по поставленным во введении цели и задачам дипломного проекта);
* экономической части (перечислены основные затраты на реализацию интернет-магазина, рассчитаны сроки окупаемости программного продукта);
* мероприятия по технике безопасности, противопожарной технике и охране труда при работе с вычислительной техникой;
* списка используемых источников (в этом списке должны быть указаны все информационные источники, использованные для подготовки дипломного проекта).

## ОБЩАЯ ЧАСТЬ

## Обоснование целесообразности и актуальности разработки интернет-магазина по продаже автозапчастей

В настоящее время все больше компаний переходят в online режим работы, то есть продажа своих услуг через интернет. Интернет дает неограниченные возможности предпринимателям, т.к. магазин который осуществлял свою деятельность в определенном районе, области, городе, получает возможность расширить охват торговли на уровень города, страны или даже мира.

Также интернет-магазин позволяет продавать товары в любое время суток, увеличить доход за счет сокращения расходов на аренду помещения, заработную плату сотрудникам.

С каждым днем все больше и больше людей получают водительские удостоверения, соответственно возникает необходимость приобретения собственного транспортного средства. Для того, чтобы эксплуатировать автомобиль, его необходимо обслуживать, начиная от минимальных вложений, таких как замена жидкостей, заканчивая ремонтом неисправностей. Соответственно возникает необходимость в поиске запчастей на транспортное средство.

Разработка интернет-магазина по продаже автозапчастей является целесообразным решением, так как автовладелец получит возможность быстро находить необходимые товары.

Согласно статистике у 50 процентов населения России есть автомобиль, что говорит об актуальности разработки интернет-магазина как для автовладельца, так и для уже существующей или новой компании.

## Обзор аналогичных веб-ресурсов

Существует достаточное количество интернет-магазинов по продаже автозапчастей. Исходя из результатов поиска можно выделить следующие:

1. Exist.Ru. Данный сайт является первым в результатах поиска. Ресурс имеет достаточно приятный и понятный интерфейс. Имеется каталог запчастей по марке машины. В шапке сайта присутствует навигация по приложению, контактные данные магазина, а также каталоги. Каталог достаточно разнообразен, имеется множество категорий и подкатегорий. Данный веб-ресурс имеет большой ассортимент товаров. Главная страница сайта отображена на рисунке 1.

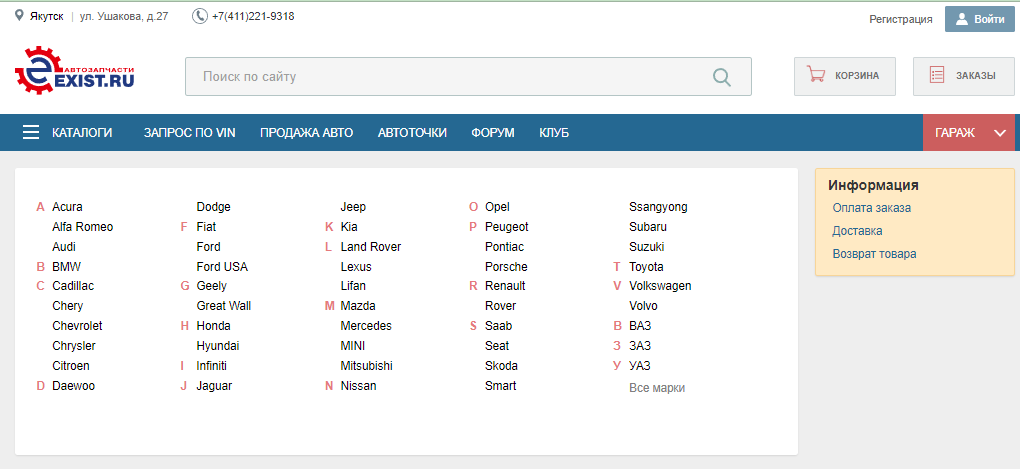


Рисунок 1 – Главная страница сайта Exist.Ru

1. Автомагазин 777. Второй по результатам поиска. На главной странице отображен поиск по товарам, вместе с каталогом. Также имеется информация о магазине. В шапке сайта расположена навигация по информационным страницам магазина, а также вход и регистрация. Каталог содержит только общие категории. Ассортимент товаров достаточно разнообразный, но не такой большой как у Exist.Ru. Главная страница сайта отображена на рисунке 2.

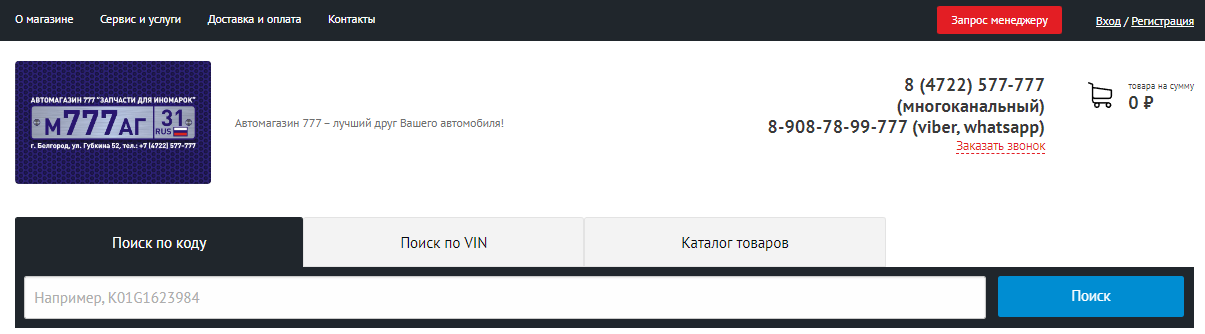


Рисунок 2 – Главная страница сайта Автомагазин 777

1. Auto3n. Данный ресурс имеет самый приятный дизайн из выше перечисленных. В нем сочетаются цвета черного, белого и оранжевого. В шапке сайта расположены ссылки на информационные страницы сайта, каталог, поиск, личный кабинет и корзина. На главной странице сайта расположены популярные категории, поиск запчастей по vin и марке авто. Ассортимент товаров разнообразный, интерфейс очень удобный для навигации и поиска необходимых товаров. Главная страница сайта отображена на рисунке 3.

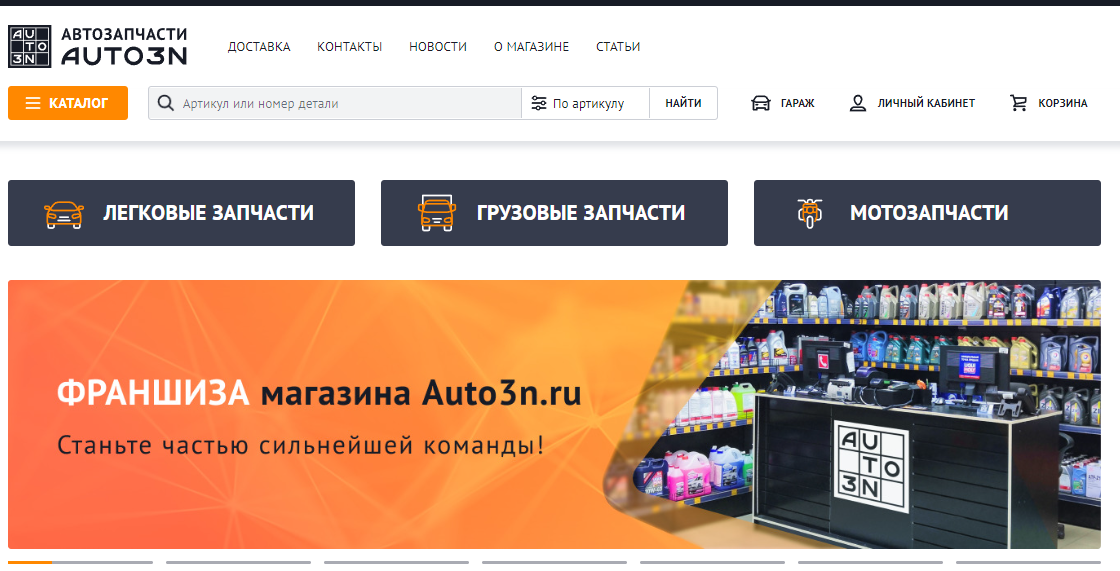


Рисунок 3 – Главная страница сайта Auto3n

Таким образом были проанализированы веб-ресурсы конкурентов, учтены достоинства и недостатки каждого.

## Обзор средств разработки интернет-магазинов

HTML (Hypertext Markup Language) — это код, который используется для структурирования и отображения веб-страницы и её контента. HTML не является языком программирования; это язык разметки, и используется, чтобы сообщать вашему браузеру, как отображать веб-страницы, которую посещает пользователь. HTML состоит из ряда элементов, чтобы вкладывать или оборачивать различные части контента для того, чтобы контент отображался или действовал определенным образом. Язык состоит из тегов — это своеобразные команды, которые преобразовываются в визуальные объекты в браузере пользователя [1]. Простыми словами, HTML — это каркас сайта. Структура HTML-элемента представлена на рисунке 4.

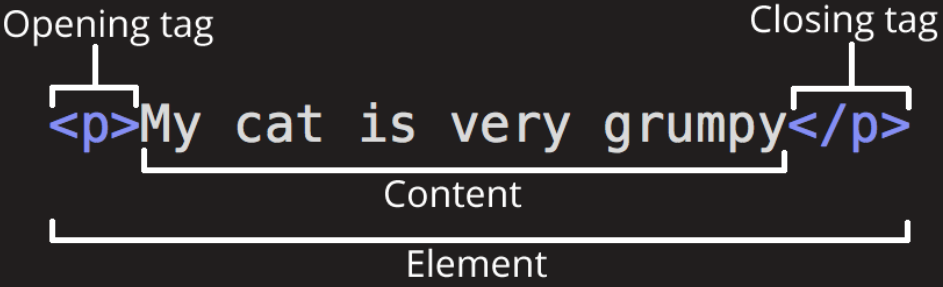


Рисунок 4 – Структура HTML-элемента

CSS (Cascading Style Sheets) — каскадные таблицы стилей, по сути — формальный язык описания внешнего вида HTML-документа. Если HTML структурирует контент на странице, то CSS позволяет отформатировать его, сделать более привлекательным для читателя. Как и HTML, CSS на самом деле не является языком программирования. CSS — это язык таблицы стилей. Это означает, что он позволяет применять стили выборочно к элементам в документах HTML [2]. Синтаксис CSS представлен на рисунке 5.

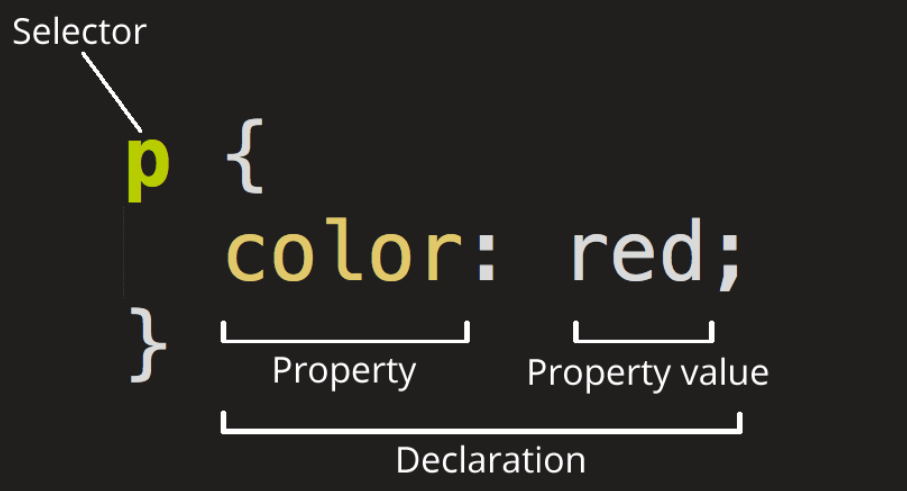


Рисунок 5- Синтаксис CSS

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Изначально JavaScript был создан, чтобы «сделать веб-страницы живыми». Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы. Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска. Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся «движком» JavaScript [3].

В связи с ростом количества веб-приложений, сложность интерфейса постоянно растет, разрабатывать и поддерживать большие проекты сложно и дорого, появились различные фреймворки для повышения скорости, качества и удобства разработки, а также для облегчения дальнейшей поддержки проекта. Фреймворк – это готовая модель, заготовка, шаблон, на основе которого можно дописать собственный код [4].

Список популярных фреймворков:

1. React – лидер в области инфраструктуры Javascript, разработанный кампанией Facebook. На самом деле это не фреймворк, а библиотека, которая ничем не уступает фреймворкам. React имеет компонентно-ориентированный подход, в котором написанные компоненты можно пере использовать неограниченное количество раз. Компоненты — это блоки кода, которые можно классифицировать как классы или функции. Каждый компонент представляет определенную часть страницы, такую как логотип, кнопка или поле ввода. Логотип ReactJS представлен на рисунке 6.

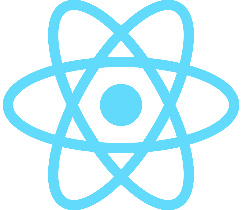


Рисунок 6 – Логотип ReactJS

1. Vue.js – это JavaScript – фреймоврк с открытым исходным кодом для создания интерфейса веб-страницы. Vue довольно прост в использовании, так как в сравнении с React во Vue многие вещи сделаны за разработчика, что требует меньшее количество написанного кода. Также Vue больше подходит новичкам так как не требует больших знаний JavaScript. Логотип Vue.js представлен на рисунке 7.



Рисунок 7 – Логотип Vue.js

1. Angular - фреймворк для создания интерфейса веб-страницы, написанный на языке TypeScript (язык, расширяющий возможности JavaScript, со строгой типизацией), разработанный кампанией Google. Самый сложный из вышеперечисленных, так как написан на TypeScript. Достоинствами данного фреймворка является количество возможностей без отдельных библиотек (все есть “из коробки”), а также определенная структура разработки приложения, облегчает разработку и поддержку приложения [5]. Логотип Angular представлен на рисунке 8.



Рисунок 8 – Логотип Angular

Для разработки серверной части приложения необходимо хранилище данных для сохранения внесенной пользователем информации, а также технология разработки для написания логики серверной части приложения.

Для хранения данных существует множество систем, одна из таких MySQL — это система управления базами данных (СУБД), распространяемая как свободное программное обеспечение (пользователи имеют право на неограниченную установку, запуск, свободное использование).

Преимущества MySQL:

1. Открытый исходный код. Распространяется бесплатно для домашнего применения.
2. Простота. MySQL легко устанавливается, имеет понятный интерфейс, а разнообразие плагинов и дополнительных приложений упрощает работу с БД.
3. Функционал. Включает в себя практически весь необходимый набор инструментов, который может пригодиться при разработке любого проекта.
4. Безопасность. Многие системы безопасности уже встроены и работают по умолчанию.
5. Масштабируемость. Может использоваться в работе как с малым, так и с большим объемом данных.
6. Скорость. Является одной из самых быстрых среди имеющихся на современном рынке [6].

Для разработки серверной части также существует множество технологий, описание которых приведено ниже.

1. PHP – это распространённый язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML. РНР позволяет изменять веб-страницу на сервере непосредственно перед тем, как она будет отправлена браузеру. В ходе исполнения PHP может изменить или динамически создать любой HTML-код, который и является результатом исполнения сценария. Затем сервер отправляет этот код браузеру. Основным преимуществом PHP является возможность внедрить PHP-скрипт в HTML-разметку [7].
2. Node.js — это программная платформа, которая транслирует JavaScript в машинный код, исполняемый на стороне сервера. Таким образом, JavaScript можно использовать для создания серверной части. В отличие от PHP, который работает под управлением "стороннего" веб-сервера, например, Apache или IIS, Node.js сама является веб-сервером. Но часто она работает в связке с Nginx в качестве основного сервера. Использование Nginx совсем не обязательно, но оправдано для кэширования данных запроса, отдачи статических файлов или разделения доменов в пределах одного сервера (reverse proxy). В основе Node.js лежит разработанный Google движок V8, который используется в браузере Chrome и отвечает за компиляцию кода JavaScript во внутренний код машины [8].
3. Java — язык программирования общего назначения. Относится к объектно-ориентированным языкам программирования, к языкам с сильной типизацией. Один из самых популярных языков программирования. Он универсален и используется уже более 20 лет. Универсальность обеспечивается виртуальной машиной Java (Java Virtual Machine, JVM). Во многих языках во время компиляции программа переводится в код, который может работать по-разному на разных устройствах или платформах. В Java этой проблемы нет. JVM играет роль промежуточного уровня — из программы на Java она делает код, который может выполняться на любом компьютере независимо от того, где код был скомпилирован [9].

Существует большое количество инструментов для разработки веб-приложений. Каждый инструмент по-своему уникален, использование которого зависит от предпочтений разработчика. Ниже рассмотрены наиболее востребованные и популярные инструменты.

1. Visual Studio Code — бесплатный и очень популярный редактор кода от Microsoft. Редактор поможет в работе веб-разработчикам и верстальщикам всех уровней. С одной стороны, он подходит новичкам, потому что его интерфейс интуитивно прост и понятен. С другой стороны, в VS Code встроены много возможностей, которые интересны опытным разработчикам. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации [10]. Из минусов разработчики отмечают достаточно большое время запуска приложения. Также, данная программа осуществляет долгий поиск по проектам. Логотип Visual Studio Code представлен на рисунке 9.

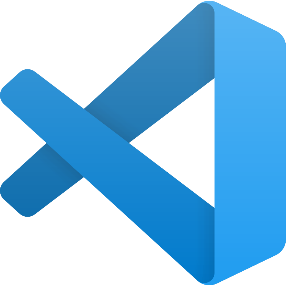


Рисунок 9 – Логотип Visual Studio Code

1. JetBrains WebStorm — интегрированная среда разработки на JavaScript, CSS & HTML от компании JetBrains, разработанная на основе платформы IntelliJ IDEA. WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями. Важным преимуществом интегрированной среды разработки WebStorm является работа с проектами (в том числе, рефакторинг кода JavaScript, находящегося в разных файлах и папках проекта, а также вложенного в HTML). Модификация файлов .css, .html, .js с одновременным просмотром результатов (англ. Live Edit, в некоторых источниках эта функциональность называется «редактирование файлов на лету» или «в реальном времени». Недостатком WebStorm является то, что он не распространяется по свободной лицензии. Также, данный инструмент требует большой объем оперативной памяти. На рисунке 11 представлен логотип JetBrains WebStorm [11].



Рисунок 10 – Логотип JetBrains WebStorm

1. Sublime Text — это кроссплатформенный текстовый редактор, разработанный для пользователей, которые ищут эффективный, но минималистский инструмент для редактирования кода. Редактор, конечно же, прост, в котором отсутствуют панели. Sublime Text это легкий текстовый редактор, который подойдет любому программисту. Программа сделана со скоростью, находящейся в ее основе. Особенность программы в ее скорости и отзывчивости пользовательского интерфейса, полностью настраиваемый — текстовый редактор создан, чтобы позволить конечному пользователю легко «поиграть» с ПО на свой лад. Sublime позволяет настраивать множество функций, включая: привязки клавиш, меню, фрагменты, макросы и многие другие. На рисунке 12 представлен логотип Sublime Text [12].



Рисунок 12 – Логотип Sublime Text

1. PhpStorm — коммерческая кроссплатформенная интегрированная среда разработки для PHP. PhpStorm представляет собой интеллектуальный редактор для PHP, HTML и JavaScript с возможностями анализа кода на лету, предотвращения ошибок в коде и автоматизированными средствами рефакторинга для PHP и JavaScript. Имеется полноценный SQL-редактор с возможностью редактирования полученных результатов запросов.PhpStorm разработан на основе платформы IntelliJ IDEA, написанной на Java. Пользователи могут расширить функциональность среды разработки за счет установки плагинов, разработанных для платформы IntelliJ, или написав собственные плагины. Вся функциональность WebStorm включена в PhpStorm. Недостатки данного инструмента такие же как у WebStorm, а именно повышенная нагрузка на компьютер и данный инструмент платный. На рисунке 13 представлен логотип JetBrains PhpStorm [13].

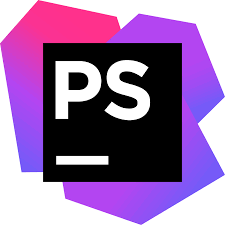


Рисунок 13 – Логотип JetBrains PhpStorm

Таким образом, был проведен анализ средств разработки интернет-магазина, определены достоинства и недостатки каждого из них.

## ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## Обоснование и выбор программных средств реализации интернет-магазина

Для разработки интернет-магазина необходимо выбрать программные средства, возможности которых соответствуют требованиям, поставленным в рамках дипломного проекта.

Для хранения информации решено использовать объектно-реляционную систему управления базами данных PostgreSQL. PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности.

Преимущества PostgreSQL:

* поддержка БД неограниченного размера;
* мощные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования и поддержка загрузки C-совместимых модулей;
* наследование;
* легкая расширяемость [14].

Для разработки веб-интерфейса (клиентской части) решено использовать JavaScript фреймворк React, из-за его популярности, экосистемы и возможностей реализации интерфейса пользователя.

Для разработки серверной части решено использовать фреймворк Express для платформы Node.JS. Node.JS позволяет разрабатывать и клиент и сервер на одном языке JavaScript. Данная платформа хорошо оптимизирована под высокую нагрузку, а также имеет огромное количество библиотек, облегчающих разработку серверной части [15].

Также для разработки вместо JavaScript, был использован язык программирования TypeScript, расширяющий возможности JavaScript и добавляющий строгую типизацию проекту, что дает более простое понимание кода, а также упрощает поддержку ранее написанного проекта. Код на TypeScript компилируется в JS и подходит для разработки любых проектов под любые браузеры — тем более что можно выбрать версию JS, в которую будет компилироваться код [16].

Для разработки интернет-магазина решено использовать среду разработки Visual Studio Code. Visual Studio Code — это сервис, который позиционируется как «легкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Особенности Visual Studio Code:

1. VS Code позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ.
2. В редакторе присутствуют встроенный отладчик, инструменты для работы с Git и средства рефакторинга, навигации по коду, автодополнения типовых конструкций и контекстной подсказки.
3. Продукт поддерживает разработку для платформ ASP.NET и Node.js, и считается легковесным решение, которое позволяет обойтись без полной интегрированной среды разработки.
4. Большим плюсом редактора является поддержка большого количества языков, таких как C++, C#, Python, PHP, JavaScript и других.

Таким образом, были определены средства для разработки инетрнет-магазина.

## Проектирование и разработка структуры интернет-магазина

Проектирование структуры веб-приложения одновременно очень важный и сложный процесс, требующий от разработчика знаний в области веб-разработки, а также немалого опыта для разработки качественного приложения.

Проектирование структуры обладает определенной последовательностью и составом стадий и этапов разработки проекта, совокупностью процедур и привлекаемых технических средств, взаимодействием участников процесса.

Любое веб-приложение представляет собой набор статических и динамических веб-страниц. Статическая веб-страница — это страница, которая всегда отображается перед пользователем в неизменном виде. Веб-сервер отправляет страницу по запросу веб-браузера без каких-либо изменений. В противоположность этому, сервер вносит изменения в динамическую веб-страницу перед отправкой ее браузеру. По причине того, что страница меняется, она называется динамической [1].

Веб-приложение в рамках дипломного проекта построено на архитектуре «клиент-сервер». Архитектура «Клиент-Сервер» (также используются термины «сеть Клиент-Сервер» или «модель Клиент-Сервер») предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других.

В архитектуре «Клиент-Сервер» несколько компьютеров-клиентов (удалённые системы) посылают запросы и получают услуги от централизованной служебной машины – сервера (server – англ. «официант, обслуга»), которая также может называться хост-системой (host system, от host – англ. «хозяин», обычно гостиницы).

Клиентская машина предоставляет пользователю т.н. «дружественный интерфейс» (user-friendly interface), чтобы облегчить его взаимодействие с сервером [17]. Архитектура «клиент-сервер» представлена на рисунке 13.

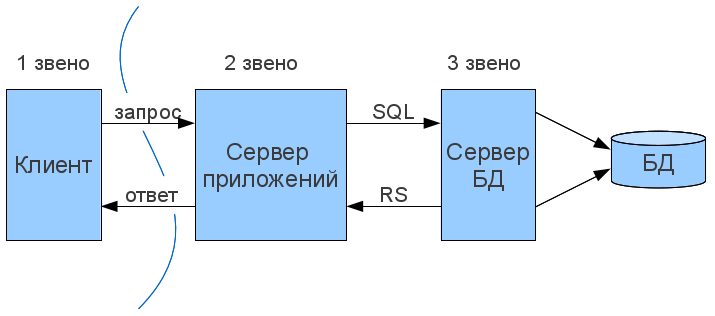


Рисунок 13 – Архитектура «клиент-сервер»

Таким образом, была определена структура интернет-магазина, в рамках дипломного проекта.

## Проектирование структуры базы данных

Правильная структура базы данных является очень важным аспектом для разработки интернет-магазина, так как от нее будет зависеть скорость получения данных, что в свою очередь влияет на скорость работы веб-приложения.

База данных спроектирована с помощью технологии ORM. ORM (англ. Object-Relational Mapping — «объектно-реляционное отображение или преобразование») — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». ORM библиотекой решено использовать Prisma. Prisma представляет собой «открытую ORM нового поколения для Node.js и TypeScript», реализующую «новую парадигму объектно-реляционного отображения».

Поддерживаемые языки программирования:

* JavaScript
* TypeScript
* Go (в разработке) [18].

Работа с базами данных (моделирование данных, изменение схем, формирование запросов и т.п.) — одна из наиболее сложных задач, возникающих при разработке приложений. Prisma предлагает решение, позволяющее сосредоточиться на данных вместо SQL.

База данных состоит из восьми основных таблиц для хранения данных и из трёх вспомогательных таблиц, автоматически сгенерированных Prisma. Структура базы данных отображена на рисунке 14.

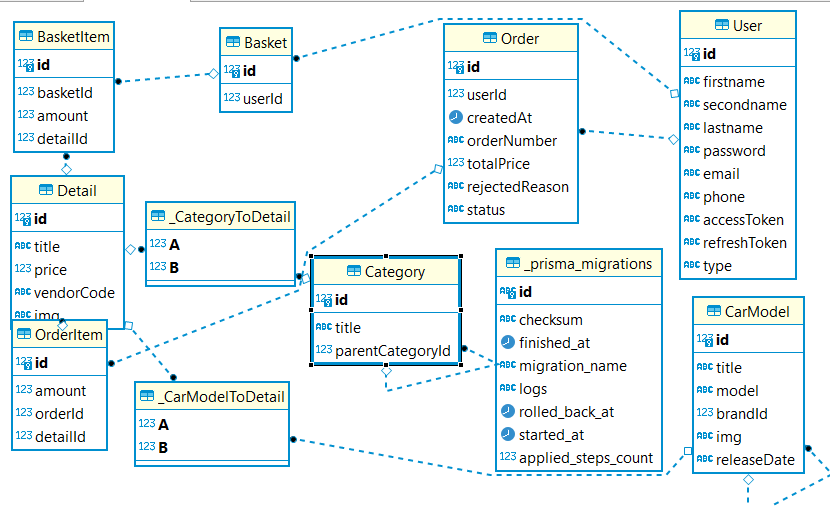


Рисунок 14 – Структура базы данных

Код схемы базы данных представлен в приложении А.

Таким образом, была спроектирована база данных, для дальнейшей разработки серверной части приложения.

## Программная реализация модулей приложения

Серверная часть приложения построена с помощью паттерна MVC. MVC расшифровывается как модель-представление-контроллер (от англ. model-view-controller). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения. Компоненты MVC:

* 1. Модель — этот компонент отвечает за данные, а также определяет структуру приложения. Например, если вы создаете To-Do приложение, код компонента model будет определять список задач и отдельные задачи.
  2. Представление — этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования.
  3. Контроллер — этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя [19].

Первым этапом является запуск сервера express, а также подключение к базе данных. Код инициализации сервера отображен на рисунке 15.



Рисунок 15 – Код инициализации сервера express

Далее необходимо создать обработчик для регистрации и авторизации пользователя. В первую очередь нужно определить маршрутизацию. Маршрутизация определяет, как приложение отвечает на клиентский запрос к конкретному адресу (URI). Для этого в express присутствует элемент, который называется Router. Код маршрутизатора для пользователя представлен на рисунке 16.



Рисунок 16 – Код маршрутизатора для пользователя

Далее в контроллере происходит вызов необходимых функций, для обработки запроса. Код регистрации пользователя представлен на рисунке 17

. 

Рисунок 17 – Код регистрации пользователя

Для авторизации пользователя необходимо проверить являются ли телефон и пароль достоверными. В зависимости от достоверности данных, отправить ответ с корректной или некорректной авторизацией. Код проверки авторизации представлен на рисунке 18.

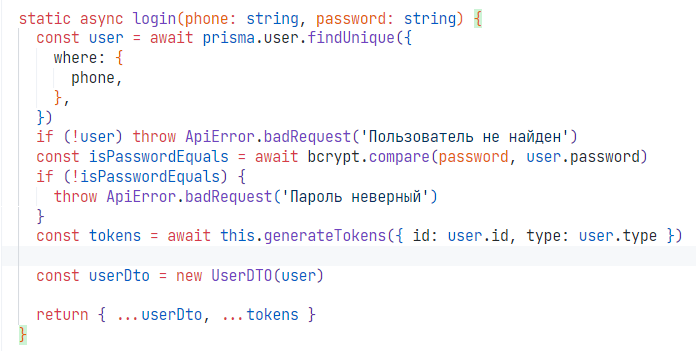


Рисунок 18 – Код проверки авторизации пользователя

Для отображения списка товаров реализован обработчик под названием detail. Код получения данных о товарах представлен на рисунке 19.

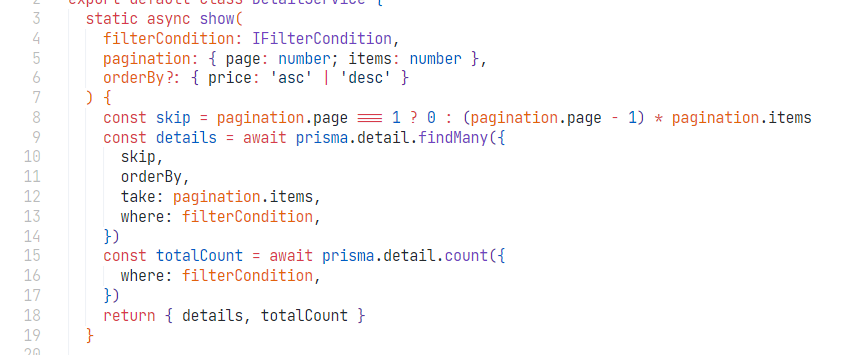


Рисунок 19 – Код получения данных о товарах

Для взаимодействия с корзиной реализован обработчик под названием basket, позволяющий получать, изменять, добавлять товары в корзину. Получение добавленных в корзину товаров представлено на рисунке 20.

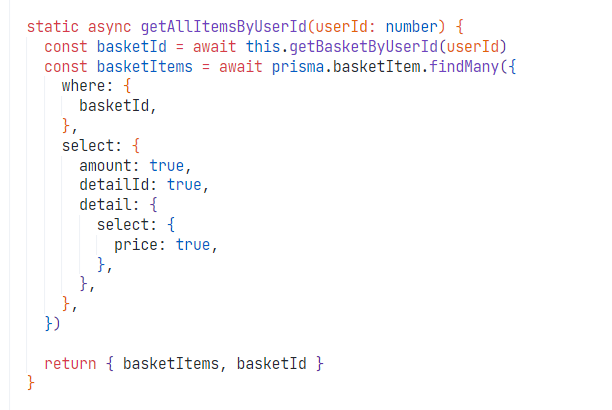


Рисунок 20 – Код получения добавленных в корзину товаров

Для оформления заказов реализован обработчик под названием order. Для того чтобы оформить заказ, необходимо получить товары, добавленные в корзину, создать запись в таблице order с добавленными товарами, а затем очистить корзину. Код оформления заказа представлен на рисунке 21.



Рисунок 21 – Код оформления заказа

Для разработки клиентской части приложения использован JavaSript фреймворк React с TypeScript. Также для глобального хранения данных в приложении использована библиотека Redux Toolkit. Код функции App представлен на рисунке 22.



Рисунок 22 – Код функции App

Для отправки запросов к серверу использовалась функция createAsyncThunk библиотеки Redux Toolkit. Код отправки запроса регистрации представлен на рисунке 23.



Рисунок 23 – Код отправки запроса регистрации.

Для отображения данных использовались функциональные компоненты React. Код отображения товаров представлен на рисунке 24.

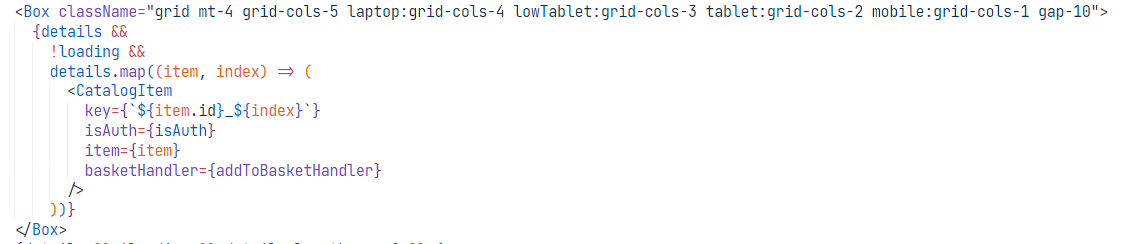


Рисунок 24 – Код отображения товаров

Листинг функций представлен в приложении Б.

Таким образом, реализация веб-приложения является достаточно сложным и трудоемкий процессом разработки интернет-магазина, так как необходимо применить знания изученных технологий. Данный этап дипломного проекта выполнен, можно приступать к тестированию веб-приложения.

## Тестирование программного продукта, размещение в глобальной сети

Тестирование приложения – важный конечный этап разработки приложения. На данной стадии уже имеется готовое веб-приложение, которое как раз-таки и необходимо протестировать на наличие ошибок, которые необходимо исправить при выявлении.

При попытке регистрации пользователя с телефоном, уже существующим в базе данных, будет выдана ошибка, с сообщением что, телефон уже занят. Сообщение представлено на рисунке 25.

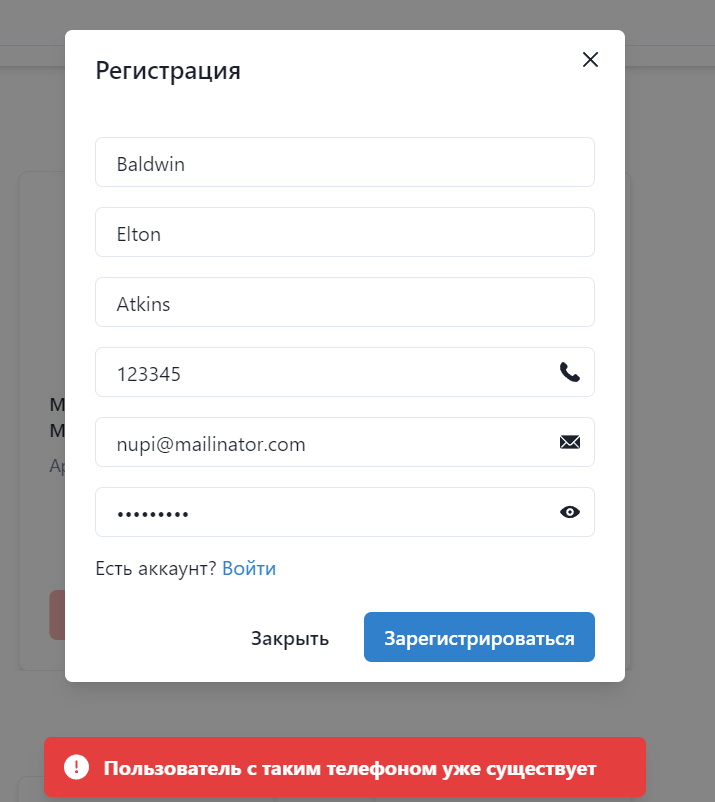


Рисунок 25 – Сообщение о существующем пользователе

При корректном заполнении данных, пользователь регистрируется и автоматически попадает в аккаунт. Об этом указывает выведенные справа вверху фамилия и имя пользователя. Данное поведение отображено на рисунке 26.

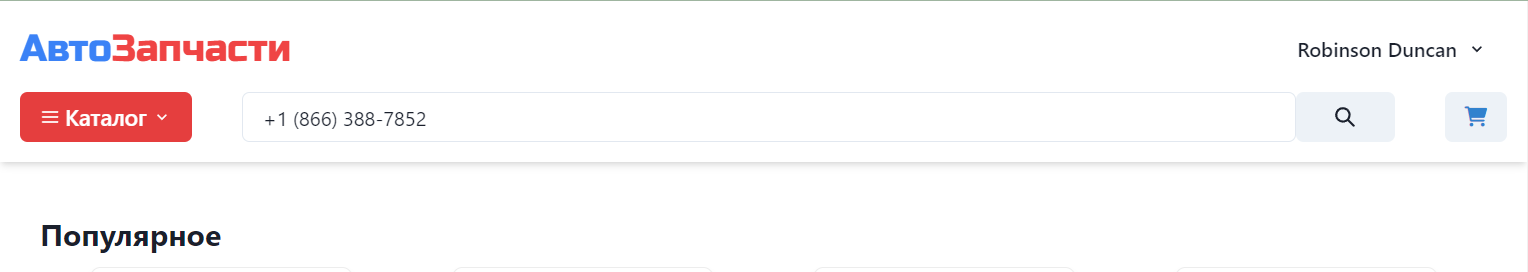


Рисунок 26 – Корректная регистрация пользователя

При авторизации с некорректными данными выводится сообщения о том, какие данные неверны. Сообщение представлено на рисунке 27.

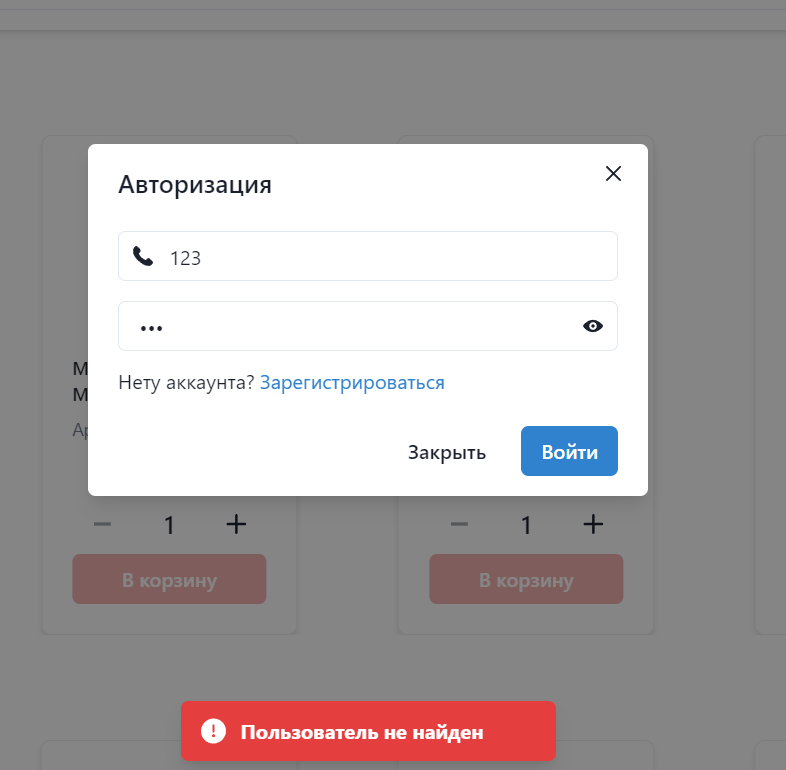


Рисунок 27 – Сообщение о неверных данных при авторизации

Если пользователь не авторизован, у пользователя нет возможности добавить товар в корзину, так как корзина доступна только авторизированным пользователям. Сообщение отображено на рисунке 28.

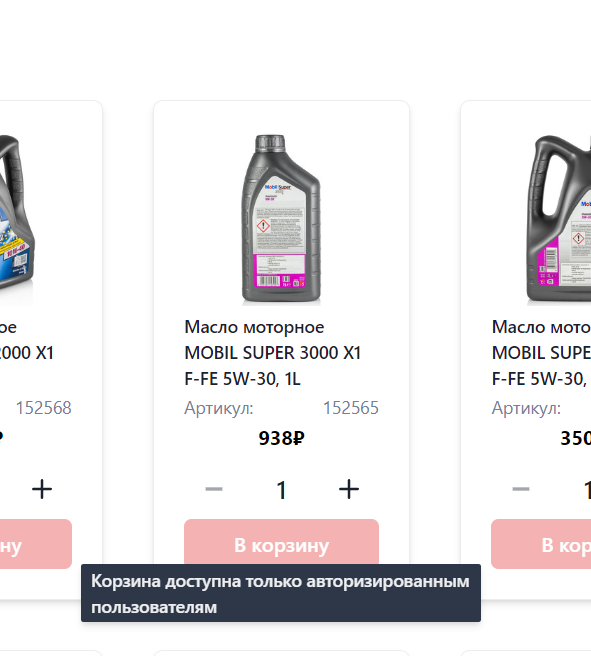


Рисунок 28 – Сообщение о недоступности корзины

При корректном поиске по запросу, по которому есть товары, интерфейс отображается корректно. Корректное поведение отображено на рисунке 29.

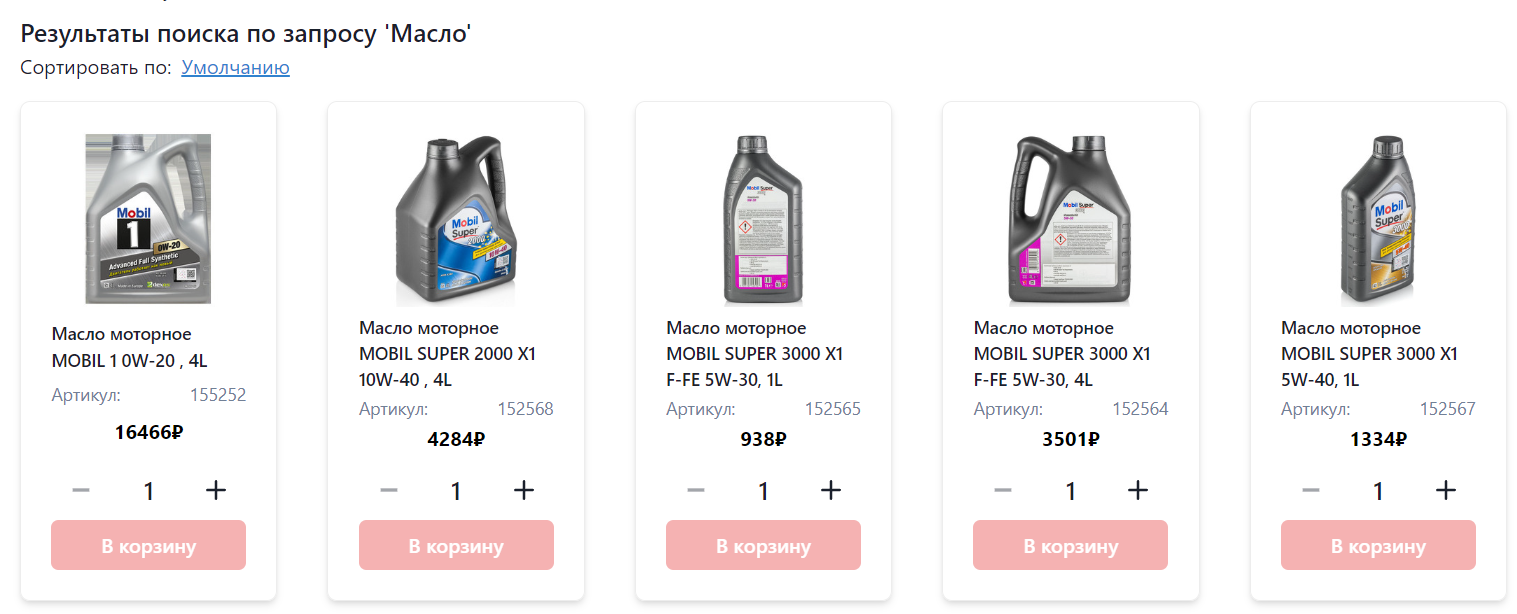


Рисунок 29 – Корректно поведение поиска по товарам

Если ввести в поиске запрос, по которому нету товаров, в интерфейс будет выведено соответствующее сообщение. Сообщение отображено на рисунке 30.

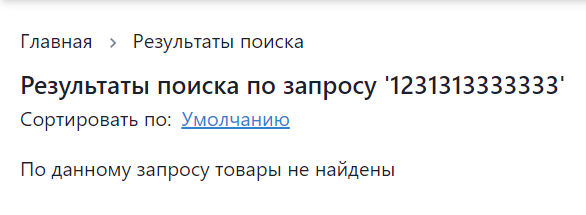


Рисунок 30 – Сообщение о том, что товары не найдены

При поиске товаров по каталогу, товары отображаются корректно. Список товаров отображен на рисунке 31.

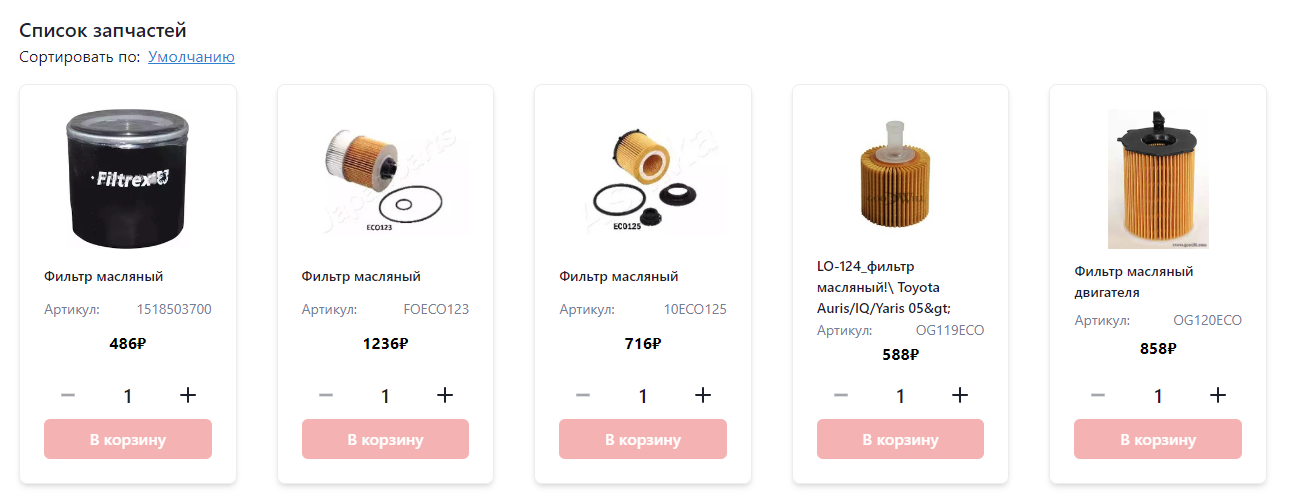


Рисунок 31 – Список товаров

При добавлении товара в корзину отображается сообщение о том, что товар успешно добавлен в корзину. Сообщение отображено на рисунке 32.

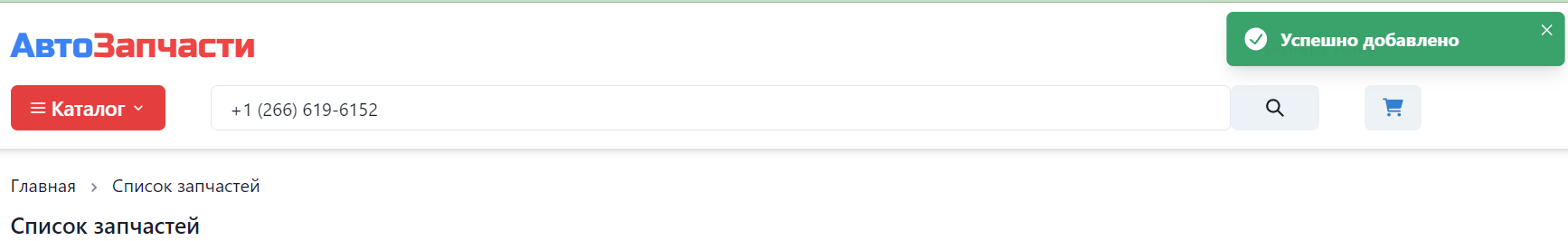


Рисунок 32 – Сообщение об успешном добавлении товара в корзину

Также товар корректно отображается в корзине. Отображение товаров в корзине представлено на рисунке 33.

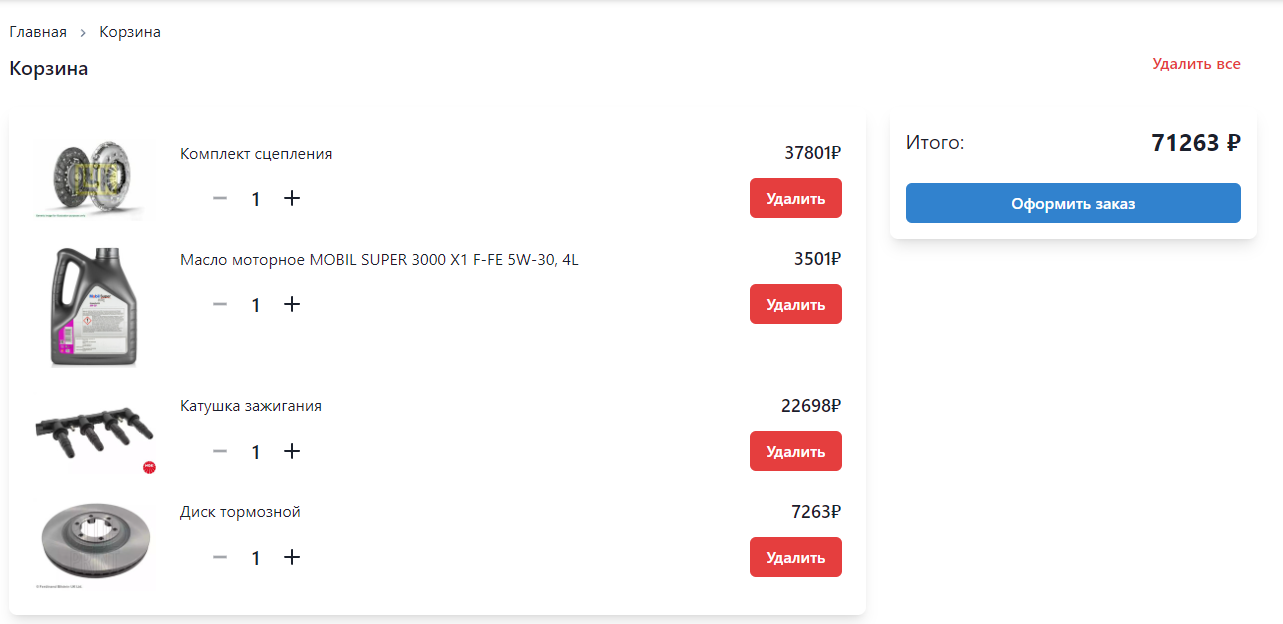


Рисунок 33 – Отображение товаров в корзине

При оформлении заказа, отображается сообщение об успешном оформлении заказа, а также очищается корзина. Создание заказа отображено на рисунке 34.

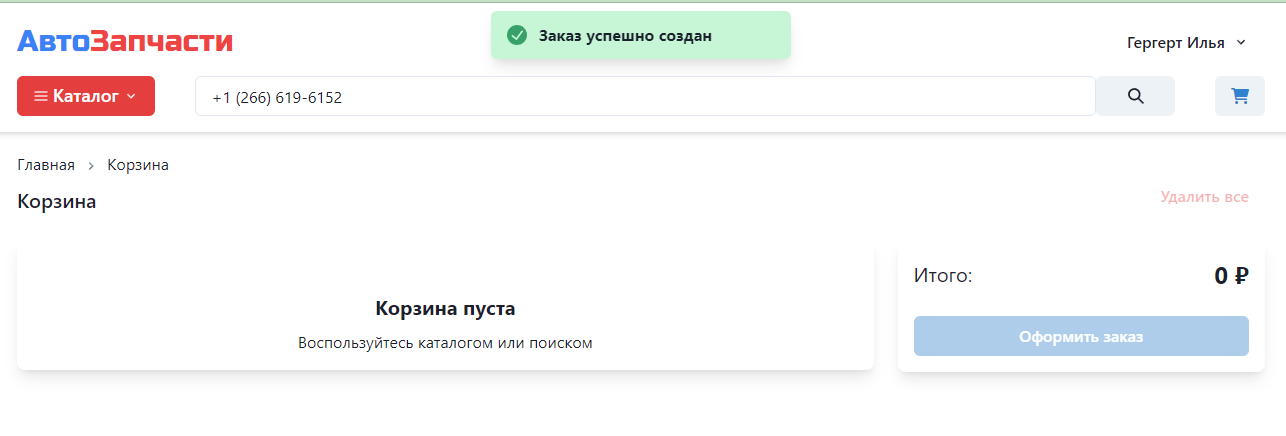


Рисунок 34 – Создание заказа

Страница отображения заказов пользователя отображается корректно, показан список заказов, а также список товаров в заказе. Отображение заказов пользователя отображено на рисунке 35.

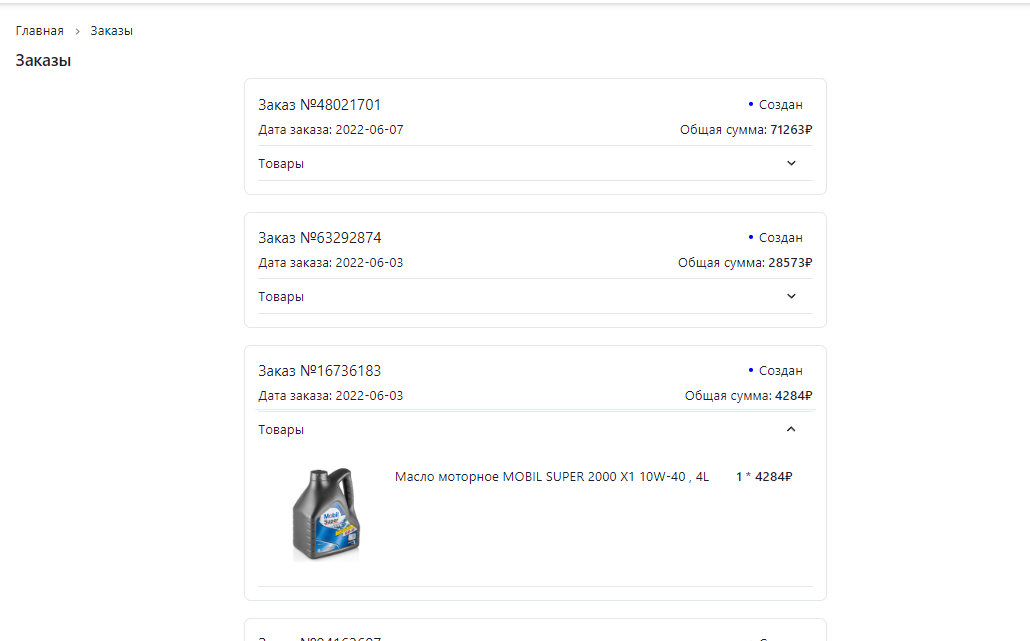


Рисунок 35 – Отображение заказов пользователя

Для размещения продукта в глобальной сети, использован бесплатный ресурс Heroku. Heroku — это облачная платформа, которая позволяет компаниям создавать, доставлять, отслеживать и масштабировать приложения, поддерживающая ряд языков программирования [20].

Таким образом, протестирован программный продукт на наличие ошибок и уязвимостей системы. Также веб-ресурс был размещен в глобальной сети.

1. 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. Расчет себестоимости произведем 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 трудоемкость по стадиям проектирования представлена в таблице 3.1.

Таблица 3.1 – Затраты времени на различных стадиях разработки ПО

|  |  |  |  |
| --- | --- | --- | --- |
| Стадия НИР | Содержание работ | Трудоемкость | |
| 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 | 8,33 |
| Продолжение таблицы 3.1 | | | |
| 1 | 2 | 3 | 4 |
| 2.Эскизный проект | Теоретическая разработка темы. Предварительная разработка структуры входных и выходных данных. Разработка общего описания алгоритма решения задачи. | 2 | 8,33 |
| 3.Технический проект | Проектирование. Определение основных блоков, классов, объектов | 2 | 8,33 |
| 4.Рабочий проект | Написание и отладка программ. Тестирование и сборка системы. | 13 | 54,17 |
| 5. Внедрение | Подготовка инструкций пользователям, написание, оформление и защита отчета (дипломного проекта). | 5 | 20,84 |
| 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 – Расчет стоимости расходных материалов

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Кол-во | Цена, ед., руб. | Всего, руб. |
| 1 | 2 | 3 | 4 |
| Картридж | 1 | 450,00 | 450,00 |
| СD-R диск | 1 | 30,00 | 30,00 |
| Бумага | 1 | 250,00 | 250,00 |
| Ручка шариковая | 1 | 20,00 | 20,00 |
| Дипломная папка | 1 | 240,00 | 240,00 |
| Итого: | 5 | 990,00 | 990,00 |

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.

, (3.1)

где – основная заработная плата разработчика, руб.;

– дополнительная заработная плата разработчика, руб.

Основная заработная плата веб-разработчика за период разработки программы вычисляется по формуле 3.2:

, (3.2)

где – заработная плата веб-разработчика за один час работы, руб.;

– количество рабочих дней за период разработки вес-ресурса, дни;

– продолжительность рабочей смены, час.

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

, (3.3)

где – заработная плата веб-разработчика за месяц на предприятии, руб.;

– количество рабочих дней в месяце, дни (принять 22 дня);

– продолжительность рабочей смены, час. (принять 8 час.).

Заработная плата веб-разработчика за период разработки программы будет равна:

Дополнительная заработная плата включает выплаты, предусмотренные действующим законодательством за неотработанное время. Рассчитывается в процентах от основной заработной платы (15%), формула 3.4:

(3.4)

где – заработная плата веб-разработчика за период разработки веб-ресурса, руб.

Заработная плата исполнителя работ по созданию веб-ресурса вычисляется по формуле 3.5:

(3.5)

где – заработная плата веб-разработчика за период разработки веб-ресурса, руб.;

– дополнительная заработная плата, руб.

Отчисления на социальные нужды (ОСНр) устанавливаются в процентах от расходов на оплату труда (30%), формула 3.6:

(3.6)

где – заработная плата веб-разработчика за период разработки веб-ресурса, руб.;

– дополнительная заработная плата, руб.

()\*30% = 17 660,72 руб.

Сумма амортизации за период разработки веб-ресурса вычисляется линейным методом по формуле 3.7:

, (3.7)

где – годовая норма амортизации, % рассчитывается по формуле 3.8;

– балансовая стоимость оборудования, руб.;

– машинное время, затрачиваемое на создание ПП, час, формула 3.9;

– годовой фонд рабочего времени оборудования, час., определяется по формуле 3.10.

(3.8)

где – нормативный срок службы оборудования, год.

, (3.9)

где – количество рабочих дней за период разработки веб-ресурса, дни;

– продолжительность рабочей смены, час.

*,* (3.10)

где 360 – количество календарных дней в году;

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

– продолжительность рабочей смены, ч;

S – количество смен работы в сутки;

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

Расходы на инструментальные средства за месяц рассчитываются по формуле 3.11:

(3.11)

где – балансовая стоимость инструментальных средств, руб.;

12 – 12 месяцев нормативный срок службы НМА.

Стоимость оборудования (таблица 3.3) включает в себя стоимость технических средств, использованных при разработке проекта.

Таблица 3.3 – Стоимость оборудования

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование оборудования | Количество, шт. | Первоначальная стоимость, руб. | Общая стоимость, руб. |
| Компьютер | 1 | 56 200,00 | 56 200,00 |
| Принтер | 1 | 6 500,00 | 6 500,00 |
| Итого: | | | 62 700,00 |

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. |
| ОС Windows 10 | 1 000,00 |
| Хостинг и домен | 500,00 |
| Microsoft Office 2010 | 2800,00 |
| Итого: | 4 300,00 |

Вычисляем сумму амортизации за период разработки ПП линейным методом равна:

Основой для расчета 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, веб-разработчик, оператор.

Заработная плата обслуживающего персонала рассчитывается по формуле 3.12:

, (3.12)

где – заработная плата обслуживающего персонала по категориям работников, руб./мес.;

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

б) дополнительную заработную плату обслуживающего персонала (ДЗПоп) берется в процентах от основной (15%), формула 3.13:

(3.13)

где – заработная плата обслуживающего персонала, руб.

в) отчисления на социальные нужды (ОСНоп) устанавливаются в процентах от расходов на оплату труда (30%), формула 3.14:

(3.14)

где – заработная плата обслуживающего персонала, руб.;

– дополнительную заработную плату обслуживающего персонала, руб.

г) затраты на электроэнергию складываются из:

* затраты на силовую электроэнергию;
* затраты на электроэнергию, которая идет на освещение.

Затраты на силовую электроэнергию определяются по формуле 3.15:

, (3.15)

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

– годовой фонд рабочего времени оборудования, час.;

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

Затраты на электроэнергию, которая идет на освещение определяется по формуле 3.16:

, (3.16)

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

– годовой фонд рабочего времени оборудования, час.;

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

Общие затраты на электроэнергию определяются по формуле 3.17:

, (3.17)

где – затраты на электроэнергию освещения, руб.;

– затраты на силовую электроэнергию, руб.

д) стоимость ремонта оборудования определяется в процентах от балансовой стоимости оборудования по формуле 3.18:

, (3.18)

где – балансовая стоимость оборудования, руб.

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

Годовые расходы на содержание и эксплуатацию 1 АРМ с оплатой инженера определяются по формуле 3.19:

(3.19)

где – заработная плата обслуживающего персонала, руб.

– дополнительную заработную плату обслуживающего персонала, руб.

– отчисления на социальные нужды, руб.;

Общие затраты на электроэнергию

– затраты на силовую электроэнергию, руб.;

– стоимость ремонта оборудования, руб.

Себестоимость 1-го машино-часа работы оборудования определяется по формуле 3.20:

, (3.20)

где – годовые расходы на содержание и эксплуатацию 1 АРМ с оплатой инженера, руб.;

– годовой фонд рабочего времени оборудования, час.

Расходы на содержание и эксплуатацию оборудования, относящиеся к данному ПП определяется по формуле 3.21:

, (3.21)

где – себестоимость 1-го машино-часа работы ПЭВМ, руб.;

– сумму амортизации за период разработки ПП.

10 486,43 руб.

Прочие расходы (ПР) определяются в процентах от основной заработной платы разработчика (ОЗПр), (принять 30%), формула 3.22:

(3.22)

где – заработная плата веб-разработчика за месяц, руб.

Итого себестоимость ПП составит, формула 3.23:

, (3.23)

где – материальные затраты, руб.;

– заработная плата исполнителя работ, руб.;

– отчисления на социальные нужды, руб.;

– амортизация, руб.;

– на инструментальные средства, руб./мес.;

– расходы на содержание и эксплуатацию ПЭВМ, руб.;

– прочие расходы, руб.

На основании произведенных расчетов анализируется структура себестоимости, таблица 3.5.

Таблица 3.5 – Структура себестоимости программного продукта

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Сумма (руб.) | Структура (%) |
| 1 | 2 | 3 |
| Материальные затраты | 990,00 | 1 |
| Заработная плата программиста | 58 869,08 | 58,6 |
| Отчисления на социальные нужды | 17 660,72 | 17,6 |
| Амортизация ПЭВМ | 1 831,06 | 1,82 |
| На инструментальные средства | 358,33 | 0,35 |
| Расходы на содержание и эксплуатацию ПЭВМ | 10 486,43 | 10,44 |
| Прочие расходы | 10 237,00 | 10,19 |
| Итого: полная себестоимость АИС | 100 432,62 | 100 |

Себестоимость созданного программного продукта составляет 100 432,62 руб.

Определим себестоимость ПП без учета оплаты труда разработчика по формуле 3.24:

(3.24)

где – полная себестоимость АИС, руб.;

– заработная плата исполнителя работ, руб.;

– отчисления на социальные нужды, руб.

Экономия денежных средств определяется по формуле 3.25:

(3.25)

где – заработная плата исполнителя работ, руб.;

– отчисления на социальные нужды, руб.

Цена, определяется себестоимостью и прибылью. Величина прибыли составляет 30% от итога сметы, за минусом экономии денежных средств, формула 3.26:

, (3.26)

где – полная себестоимость АИС, руб.;

– экономия денежных средств, руб.

Цена продукта рассчитывается по формуле 3.27:

(3.27)

где – полная себестоимость АИС, руб.;

– экономия денежных средств, руб.;

– прибыль, 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, формула 3.28.

, (3.28)

где – цена продукта, руб.;

– прибыль, руб.

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.

При эксплуатации компьютерной техники не исключена опасность различного рода возгораний. В современных компьютерах очень высока плотность размещения элементов электронных системы, в непосредственной близости друг от друга располагаются соединительные провода, коммуникационные кабели.

При работе с компьютерной техникой должна отражать и мероприятия по обеспечению пожарной безопасности. Ниже приведены примерные противопожарные мероприятия:

* на рабочем месте запрещается использовать или хранить огнеопасные вещества;
* в качестве профилактических мероприятий для обеспечения пожарной безопасности следует использовать скрытую электросеть, надежные розетки из пожаробезопасных материалов;
* регулярно производить очистку комплектующих компьютера и смежного оборудования от пыли;
* помещения с электрооборудованием должны быть оснащены углекислотными огнетушителями типа ОУ-2 или ОУ-3.

Запрещается:

* включать и выключать компьютерную технику без необходимости (это может привести к его программным повреждениям);
* трогать разъемы соединительных кабелей, проводов, вилки и розетки;
* прикасаться к экрану и к тыльной стороне блоков компьютера;
* работать за компьютером с мокрыми руками;
* работать на компьютерах, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе;

При работе с персональным компьютером очень важную роль играет соблюдение правильного режима труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и кистях.

В соответствии со СанПиН 2.2.2/2.4.2198-07 все виды трудовой деятельности, связанные с использованием компьютера, разделяются на три группы:

‒ группа А: работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;

‒ группа Б: работа по вводу информации;

‒ группа В: творческая работа в режиме диалога с ЭВМ.

Выбранную категорию работы с ЭВМ в соответствии со сведениями о регламентированных перерывах, которые необходимо делать при работе на компьютере, можно отнести ко 2 категории.

То есть суммарное время регламентированных перерывов при смене длительностью в 8 часов должно быть 50 минут.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта был получен полнофункциональный интернет-магазин по продаже автозапчастей. Данный программный продукт ориентирован на любого автовладельца, которому необходимы какие-либо детали на автомобиль. С помощью ресурса, пользователи получают возможность получать информацию о товарах, а также заказывать необходимые.

При разработке программного продукта были изучены современные технологии, позволяющие создавать многофункциональные информационные ресурсы. Были применены знания таких средств разработки как, TypeScript, React, Redux, Node.JS, Express, Prisma, PostgreSQL.

Разработанный программный продукт удовлетворяет всем требованиям, поставленным на этапе постановки задачи:

* Изучены основные определения, термины и понятия структуры Web-сайтов;
* Рассмотрено современное положение в сфере Web-дизайна;
* Изучены ключевые особенности современных Web-редакторов;
* Обоснован выбранный способ Web-конструирования;
* Реализован программный продукт;
* Размещен в сети интернет и оптимизирован.

В дальнейшем планируется совершенствование продукта, например добавление поиска товаров по марке автомобиля.

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

1. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ
   1. MDN Web docs [Электронный ресурс] // Основы HTML – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics (дата обращения 20.03.2022)
   2. MDN Web docs [Электронный ресурс] // Основы CSS – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics (дата обращения 21.03.2022)
   3. Javascript.ru [Электронный ресурс] // Введение в JavaScript – Режим доступа: https://learn.javascript.ru/intro (дата обращения 21.03.2022)
   4. Secretmag.ru [Электронный ресурс] // Что такое фреймворк – Режим доступа: https://secretmag.ru/enciklopediya/chto-takoe-freimvork-obyasnyaem-prostymi-slovami.htm (дата обращения 21.03.2022)
   5. Vc.ru [Электронный ресурс] // Популярные фреймворки JavaScript – Режим доступа: https://vc.ru/dev/147263-populyarnye-freymvorki-javascript (дата обращения 22.03.2022)
   6. Макхост [Электронный ресурс] Что такое MySQL – Режим доступа: https://mchost.ru/articles/chto-takoe-mysql/ (дата обращения 22.03.2022)
   7. Html academy [Электронный ресурс] // Определение и особенности языка PHP – Режим доступа: https://htmlacademy.ru/tutorial/php/basics (дата обращения 22.03.2022)
   8. Node js [Электронный ресурс] // Документация Node.js – Режим доступа: https://nodejsdev.ru/doc/ (дата обращения 23.03.2022)
   9. Хекслет [Электронный ресурс] // Язык программирования Java: особенности, популярность, ситуация на рынке труда– Режим доступа: https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda (дата обращения 23.03.2022)
   10. Код журнал Яндекс Практикума [Электронный ресурс] // Visual Studio Code: кому и зачем он нужен – Режим доступа: https://thecode.media/visual-studio-code/ (дата обращения 23.03.2022)
   11. Википедия [Электронный ресурс] // WebStorm - Режим доступа: https://ru.wikipedia.org/wiki/WebStorm (дата обращения 23.03.2022)
   12. CoderLessons [Электронный ресурс] // Sublime Text — Краткое руководство - Режим доступа: https://coderlessons.com/tutorials/raznoe/vyuchit-vozvyshennyi-tekst/sublime-text-kratkoe-rukovodstvo (дата обращения 23.03.2022)
   13. Википедия [Электронный ресурс] // PhpStorm – Режим доступа https://ru.wikipedia.org/wiki/PhpStorm (дата обращения 23.03.2022)
   14. Web Creator [Электронный ресурс] // PostgreSQL — объектно-реляционная система управления базами данных– Режим доступа https://web-creator.ru/articles/postgresql (дата обращения 23.03.2022)
   15. CoderNet [Электронный ресурс] // Плюсы и минусы разработки веб-приложений на Node.js— объектно-реляционная система управления базами данных – Режим доступа https://codernet.ru/articles/web/plyusyi\_i\_minusyi\_razrabotki\_veb-prilozhenij\_na\_nodejs/ (дата обращения 23.03.2022)
   16. Skillbox [Электронный ресурс] // TypeScript: как с ним работать и чем он отличается от JavaScript – Режим доступа https://skillbox.ru/media/code/typescript\_kak\_s\_nim\_rabotat\_i\_chem\_on\_otlichaetsya\_ot\_javascript/ (дата обращения 23.03.2022)
   17. Itelon [Электронный ресурс] // Архитектура «Клиент-Сервер» – Режим доступа https://itelon.ru/blog/arkhitektura-klient-server/ (дата обращения 23.03.2022)
   18. Habr [Электронный ресурс] // Обзор Prisma ORM: как забыть об SQL и сосредоточиться на данных – Режим доступа https://habr.com/ru/company/macloud/blog/554952/ (дата обращения 23.03.2022)
   19. Хекслет [Электронный ресурс] // Что такое MVC: рассказываем простыми словами – Режим доступа https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami (дата обращения 23.03.2022)
   20. Heroku [Электронный ресурс] // What is Heroku – Режим доступа https://www.heroku.com/what (дата обращения 23.03.2022)

# ПРИЛОЖЕНИЕ А

Код схемы базы данных

generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "postgresql"

url = env("DATABASE\_URL")

}

model User {

id Int @id @default(autoincrement())

firstname String

secondname String

lastname String

password String

email String @unique

phone String @unique

type Role @default(user)

accessToken String?

refreshToken String?

orders Order[]

basket Basket?

}

enum Role {

user

admin

}

model Category {

id Int @id @default(autoincrement())

title String?

parentCategoryId Int?

parentCategory Category? @relation("CategoryType", fields: [parentCategoryId], references: [id], onDelete: Cascade)

childCategories Category[] @relation("CategoryType")

details Detail[]

}

model Detail {

id Int @id @default(autoincrement())

title String

price Int

img String?

vendorCode String?

models CarModel[]

categories Category[]

basketItems BasketItem[]

orderItems OrderItem[]

}

enum OrderStatus {

created

confirmed

rejected

}

model Order {

id Int @id @default(autoincrement())

totalPrice Int

orderNumber String

status OrderStatus @default(created)

rejectedReason String?

createdAt DateTime @default(now())

userId Int

user User @relation(fields: [userId], references: [id], onDelete: Cascade)

orderItems OrderItem[]

}

model OrderItem {

id Int @id @default(autoincrement())

amount Int @default(1)

orderId Int

detailId Int

order Order @relation(fields: [orderId], references: [id], onDelete: Cascade)

detail Detail @relation(fields: [detailId], references: [id], onDelete: Cascade)

}

model Basket {

id Int @id @default(autoincrement())

userId Int @unique

user User @relation(fields: [userId], references: [id], onDelete: Cascade)

basketItems BasketItem[]

}

model BasketItem {

id Int @id @default(autoincrement())

basketId Int

basket Basket @relation(fields: [basketId], references: [id], onDelete: Cascade)

amount Int @default(1)

detailId Int

detail Detail @relation(fields: [detailId], references: [id], onDelete: Cascade)

}

model CarModel {

id Int @id @default(autoincrement())

title String

model String?

releaseDate String?

brandId Int?

img String?

brand CarModel? @relation("Brand", fields: [brandId], references: [id], onDelete: Cascade)

brandModels CarModel[] @relation("Brand")

details Detail[]

}

ПРИЛОЖЕНИЕ Б

Листинг кода

Код класса user.controller.ts

import { NextFunction, Request, Response } from 'express'

import { ExpressJwtRequest } from 'express-jwt'

import { header, validationResult } from 'express-validator'

import ApiError from '../../exceptions/api-error'

import UserDTO from './user.dto'

import UserService from './user.service'

import { IUserEdit } from './user.types'

const userSelectedFieldsFromPrisma = {

  firstname: true,

  secondname: true,

  lastname: true,

  phone: true,

  email: true,

  type: true,

  accessToken: true,

}

export default class UserController {

  static async show(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      const { id } = req.auth.payload

      const user = await UserService.show(+id)

      return res.json(user)

    } catch (error) {

      next(error)

    }

  }

  static async registration(req: Request, res: Response, next: NextFunction) {

    try {

      const errors = validationResult(req)

      if (!errors.isEmpty()) {

        return next(ApiError.badRequest('Ошибка при валидации', errors.array()))

      }

      const userData = await UserService.signup(req.body)

      res.cookie('refreshToken', userData.refreshToken, {

        maxAge: 30 \* 24 \* 3600 \* 1000,

        httpOnly: true,

      })

      const { refreshToken, ...data } = userData

      res.json(data)

    } catch (error) {

      next(error)

    }

  }

  static async login(req: Request, res: Response, next: NextFunction) {

    try {

      const { phone, password } = req.body

      const userData = await UserService.login(phone, password)

      res.cookie('refreshToken', userData.refreshToken, {

        maxAge: 30 \* 24 \* 3600 \* 1000,

        httpOnly: true,

      })

      const { refreshToken, ...data } = userData

      return res.json(data)

    } catch (error) {

      next(error)

    }

  }

  static async logout(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      await UserService.logout(req.auth.payload.id)

      res.clearCookie('refreshToken')

      return res.status(200).json({ message: 'Успешно' })

    } catch (error) {

      next(error)

    }

  }

  static async refresh(req: Request, res: Response, next: NextFunction) {

    try {

      const { refreshToken } = req.cookies

      const userData = await UserService.refreshTokens(refreshToken)

      res.cookie('refreshToken', userData.refreshToken, {

        maxAge: 30 \* 24 \* 60 \* 60 \* 1000,

        httpOnly: true,

      })

      return res.json({ accessToken: userData.accessToken })

    } catch (error) {

      next(error)

    }

  }

  static async edit(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      const body: IUserEdit = req.body

      const userData = await UserService.editUser(req.auth.payload.id, body)

      const userDto = new UserDTO(userData)

      return res.json(userDto)

    } catch (error) {

      next(error)

    }

  }

}

Код файла user.service.ts

import { IUser, IUserEdit } from './user.types'

import bcrypt from 'bcrypt'

import jwt, { JwtPayload } from 'jsonwebtoken'

import UserDTO from './user.dto'

import prisma from '../../prisma'

import ApiError from '../../exceptions/api-error'

import { TokenData } from '../../types/types'

import { Prisma } from '@prisma/client'

export default class UserService {

  static async show(id: number) {

    const user = await prisma.user.findUnique({

      where: {

        id,

      },

      select: {

        firstname: true,

        secondname: true,

        lastname: true,

        email: true,

        phone: true,

        type: true,

      },

    })

    return user

  }

  static async signup(data: IUser) {

    const { phone, email, password, firstname, lastname, secondname } = data

    const replacedPhone = phone.replace(/[^0-9]/g, '')

    const isUserExists = await prisma.user.findFirst({

      where: {

        phone: replacedPhone,

      },

    })

    if (isUserExists) throw ApiError.badRequest(`Пользователь с таким телефоном уже существует`)

    const hashPassword = await bcrypt.hash(password, 3)

    const user = await prisma.user.create({

      data: {

        firstname,

        secondname,

        lastname,

        phone: replacedPhone,

        email,

        password: hashPassword,

        basket: { create: {} },

      },

    })

    const tokens = await this.generateTokens({ id: user.id, type: user.type })

    const userDto = new UserDTO(user)

    return { ...userDto, ...tokens }

  }

  static async generateTokens(

    payload: TokenData

  ): Promise<{ accessToken: string; refreshToken: string }> {

    const accessToken = jwt.sign({ payload }, process.env.JWT\_ACCESS\_SECRET, {

      expiresIn: '30m',

    })

    const refreshToken = jwt.sign({ payload }, process.env.JWT\_ACCESS\_REFRESH, { expiresIn: '30d' })

    await prisma.user.update({

      where: {

        id: payload.id,

      },

      data: {

        accessToken: accessToken,

        refreshToken: refreshToken,

      },

    })

    return { accessToken, refreshToken }

  }

  static async refreshTokens(refreshToken: string) {

    try {

      if (!refreshToken) throw ApiError.unauthError()

      const userData = jwt.verify(refreshToken, process.env.JWT\_ACCESS\_REFRESH) as JwtPayload

      if (!userData) throw ApiError.unauthError()

      const refreshTokenFromDB = await prisma.user.findUnique({

        where: {

          id: userData.payload.id,

        },

        select: {

          refreshToken: true,

        },

      })

      if (refreshToken !== refreshTokenFromDB.refreshToken) throw ApiError.unauthError()

      const newTokens = await this.generateTokens(userData.payload)

      return newTokens

    } catch (error) {

      throw ApiError.unauthError()

    }

  }

  static async login(phone: string, password: string) {

    const user = await prisma.user.findUnique({

      where: {

        phone,

      },

    })

    if (!user) throw ApiError.badRequest('Пользователь не найден')

    const isPasswordEquals = await bcrypt.compare(password, user.password)

    if (!isPasswordEquals) {

      throw ApiError.badRequest('Пароль неверный')

    }

    const tokens = await this.generateTokens({ id: user.id, type: user.type })

    const userDto = new UserDTO(user)

    return { ...userDto, ...tokens }

  }

  static async logout(userId: number) {

    await prisma.user.update({

      where: {

        id: userId,

      },

      data: {

        accessToken: null,

        refreshToken: null,

      },

    })

    return true

  }

  static async editUser(userId: number, body: IUserEdit) {

    const { email, firstname, lastname, phone, secondname, oldPassword, password } = body

    let newUser: Prisma.UserUpdateInput = { email, firstname, lastname, phone, secondname }

    if (oldPassword || password) {

      const passFromDb = await prisma.user.findUnique({

        where: {

          id: userId,

        },

        select: {

          password: true,

        },

      })

      const isPassEquals = await bcrypt.compare(oldPassword, passFromDb.password)

      if (!isPassEquals) throw ApiError.badRequest('Предыдущий пароль неверный')

      const newPassword = await bcrypt.hash(password, 3)

      newUser.password = newPassword

    }

    const user = await prisma.user.update({

      where: {

        id: userId,

      },

      data: newUser,

    })

    return user

  }

}

Код файла basket.controller.ts

import { NextFunction, Request, Response } from 'express'

import { ExpressJwtRequest } from 'express-jwt'

import BasketService from './basket.service'

import { IBasketItem } from './basket.types'

export default class BasketController {

  static async getBasket(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      const { id: userId } = req.auth.payload

      const basket = await BasketService.getBasket(+userId)

      return res.json(basket)

    } catch (error) {

      next(error)

    }

  }

  static async addBasketItem(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      const { id: userId } = req.auth.payload

      const basketItem = req.body as IBasketItem

      const userBasketId = await BasketService.getBasketByUserId(userId)

      const newBasketItem = await BasketService.addBasketItem(basketItem, userBasketId)

      return res.json(newBasketItem)

    } catch (error) {

      next(error)

    }

  }

  static async editBasketItem(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      const basketItem = req.body as IBasketItem

      const newBasketItem = await BasketService.editBasketItem(basketItem.id, basketItem)

      return res.json(newBasketItem)

    } catch (error) {

      next(error)

    }

  }

  static async deleteBasketItem(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      const basketItemId = req.params.itemId

      const deletedBasketItem = await BasketService.deleteBasketItem(+basketItemId)

      return res.json('Успешно')

    } catch (error) {

      next(error)

    }

  }

  static async deleteBasketItemsAll(req: ExpressJwtRequest, res: Response, next: NextFunction) {

    try {

      const { id: userId } = req.auth.payload

      const basketId = await BasketService.getBasketByUserId(userId)

      await BasketService.deleteBasketItemsByBasketId(basketId)

      return res.json('Успешно')

    } catch (error) {

      next(error)

    }

  }

}

Код файла basket.service.ts

import prisma from '../../prisma'

import { IBasket, IBasketItem } from './basket.types'

export default class BasketService {

  static async createBasket(userId: number) {

    const newBasket = await prisma.basket.create({

      data: {

        userId,

      },

    })

    return newBasket

  }

  static async getBasketByUserId(userId: number) {

    const basket = await prisma.basket.findUnique({

      where: {

        userId,

      },

      select: {

        id: true,

      },

    })

    return basket.id

  }

  static async addBasketItem(item: IBasketItem, basketId: number) {

    const isItemExists = await prisma.basket.findUnique({

      where: {

        id: basketId,

      },

      select: {

        basketItems: {

          where: {

            detailId: item.detailId,

          },

        },

      },

    })

    if (isItemExists.basketItems.length !== 0) {

      const newItem = await prisma.basketItem.update({

        where: {

          id: isItemExists.basketItems[0].id,

        },

        data: {

          amount: {

            increment: item.amount,

          },

        },

      })

      return newItem

    }

    const newItem = await prisma.basketItem.create({

      data: {

        ...item,

        basketId,

      },

    })

    return newItem

  }

  static async getBasket(userId: number) {

    const basket = await prisma.basket.findUnique({

      where: {

        userId,

      },

      select: {

        basketItems: {

          orderBy: {

            id: 'asc',

          },

          select: {

            amount: true,

            id: true,

            detail: {

              select: {

                id: true,

                img: true,

                price: true,

                title: true,

              },

            },

          },

        },

      },

    })

    return basket

  }

  static async editBasketItem(basketItemId: number, basket: IBasketItem) {

    const newItem = await prisma.basketItem.update({

      where: {

        id: basketItemId,

      },

      data: basket,

    })

    return newItem

  }

  static async deleteBasketItem(basketItemId: number) {

    const newItem = await prisma.basketItem.delete({

      where: {

        id: basketItemId,

      },

    })

    return newItem

  }

  static async getAllItemsByUserId(userId: number) {

    const basketId = await this.getBasketByUserId(userId)

    const basketItems = await prisma.basketItem.findMany({

      where: {

        basketId,

      },

      select: {

        amount: true,

        detailId: true,

        detail: {

          select: {

            price: true,

          },

        },

      },

    })

    return { basketItems, basketId }

  }

  static async deleteBasketItemsByBasketId(basketId: number) {

    const items = await prisma.basketItem.deleteMany({

      where: {

        basketId,

      },

    })

    return items

  }

}

Код файла Router.tsx

import { Box, Spinner } from '@chakra-ui/react'

import React, { lazy, ReactElement, Suspense } from 'react'

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

import { useAppSelector } from 'store/hooks'

import { selectUserAuth, selectUserType } from 'store/user/user.selector'

const Admin = lazy(() => import('pages/Admin/Admin'))

const Home = lazy(() => import('pages/Home/Home'))

const Basket = lazy(() => import('pages/Basket/Basket'))

const CarModel = lazy(() => import('pages/CarModel/CarModel'))

const Orders = lazy(() => import('pages/Orders/Orders'))

const Profile = lazy(() => import('pages/Profile/Profile'))

const SearchResult = lazy(() => import('pages/SearchResult/SearchResult'))

const SelectCategory = lazy(() => import('pages/SelectCategory/SelectCategory'))

const ShowCatalog = lazy(() => import('pages/ShowCatalog/ShowCatalog'))

const ShowCatalogCategory = lazy(() => import('pages/ShowCatalogCategory/ShowCatalogCategory'))

interface IRoutes {

  routes: string[]

  element: ReactElement

}

const authRoutes: IRoutes[] = [

  {

    routes: ['/profile'],

    element: <Profile />,

  },

  {

    routes: ['orders'],

    element: <Orders />,

  },

  {

    routes: ['/basket'],

    element: <Basket />,

  },

]

const publicRoutes: IRoutes[] = [

  {

    routes: ['/car/:carId/model/:modelId', '/category'],

    element: <SelectCategory />,

  },

  {

    routes: ['/'],

    element: <Home />,

  },

  {

    routes: ['/car/:carId'],

    element: <CarModel />,

  },

  {

    routes: ['/search/:searchStr'],

    element: <SearchResult />,

  },

  {

    routes: ['/car/:carId/model/:modelId/category/:categoryId'],

    element: <ShowCatalog />,

  },

  {

    routes: ['/category/:categoryId'],

    element: <ShowCatalogCategory />,

  },

  {

    routes: ['\*'],

    element: <Navigate to="/" />,

  },

]

const Router = () => {

  const { type, isAuth } = useAppSelector((state) => ({

    type: selectUserType(state),

    isAuth: selectUserAuth(state),

  }))

  return (

    <Box className="mt-5">

      <Suspense

        fallback={

          <Box className="text-center mt-10">

            <Spinner color="blue.500" width={'150px'} height="150px" speed="0.8s" />

          </Box>

        }

      >

        <Routes>

          {type === 'admin' && <Route path="/admin" element={<Admin />} />}

          {isAuth &&

            authRoutes.map((parentItem) =>

              parentItem.routes.map((item) => <Route path={item} element={parentItem.element} />)

            )}

          {publicRoutes.map((parentItem) =>

            parentItem.routes.map((item) => <Route path={item} element={parentItem.element} />)

          )}

        </Routes>

      </Suspense>

    </Box>

  )

}

export default Router