**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»**

**ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите**  Заместитель директора по  учебно-методической работе  \_\_\_\_\_\_\_\_\_\_\_ \_\_Е. Г. Конакина\_\_\_\_\_  (Подпись) (И.О.Ф.)  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема Разработка функциональных модулей для веб-маркетплейса

специальность 09.02.07 группа 42919/1

Студент (ка) Карабовский М.Н.

(подпись) (ФИО)

Руководитель Черемонцева И.В.

(подпись) (ФИО)

Санкт-Петербург

2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc105175426)

[1 Общая часть 5](#_Toc105175427)

[1.1. Анализ предметной области 5](#_Toc105175428)

[1.2. Анализ рынка существующих решений 5](#_Toc105175429)

[1.3. Постановка задачи 9](#_Toc105175430)

[1.4. Обоснование и выбор методики, технологии и инструментальных средств проектирования и разработки 9](#_Toc105175431)

[2 Специальная часть 24](#_Toc105175432)

[2.1. Описание структуры приложения 24](#_Toc105175433)

[2.2. Объектно-ориентированное проектирование системы 33](#_Toc105175434)

[2.3. Разработка пользовательского интерфейса приложения 35](#_Toc105175435)

[3 Экономическая часть 43](#_Toc105175436)

[3.1. Область применения программного продукта и его преимущества перед аналогичным программным продуктом 43](#_Toc105175437)

[3.2. Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад 43](#_Toc105175438)

[3.3. Расчет затрат на разработку 45](#_Toc105175439)

[3.4. Расчет цены и прибыли 50](#_Toc105175440)

[3.5. Вывод по разделу 52](#_Toc105175441)

[4 Техника безопасности и охрана труда 53](#_Toc105175442)

[4.1. Анализ условий труда программиста 53](#_Toc105175443)

[4.2. Расчет искусственного освещения рабочего места 55](#_Toc105175444)

[4.3. Электробезопасность на предприятии 57](#_Toc105175445)

[4.4. Пожарная безопасность на предприятии 59](#_Toc105175446)

[ЗАКЛЮЧЕНИЕ 62](#_Toc105175447)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 63](#_Toc105175448)

[Приложение А (cправочное) Системные требования к разрабатываемому продукту 65](#_Toc105175449)

[Приложение Б (справочное) Исходный код 66](#_Toc105175450)

## **ВВЕДЕНИЕ**

С развитием информационных технологий многие сферы производства начали свой переход в информационную сферу общества, а развитие глобальной паутины обеспечило возможность для любого пользователя совершать покупки, продавать или обменивать товары. Очень быстро стали появляться рекламные объявления, все больше информации выкладывалось на корпоративных ресурсах.

Современный интернет-маркетинг в корне изменил традиционное представление о продвижении бизнеса. Он стал уникальным инструментом для предпринимателей, которым есть что предложить пользователям сети, но они не знают, как это сделать с минимальными затратами времени, денег. Подобный подход стал основанием для современного ведения бизнеса, который позволяет любому предпринимателю, у которого имеется оригинальная идея, товар, обратиться к максимальному количеству заинтересованных пользователей.

Изменение в развитии интернет-маркетинга подтверждают, что стратегии прямого отклика, используемые в глобальной сети, реально эффективны. Благодаря их использованию можно постоянно поддерживать контакт с лояльными клиентами, оперативно следить за статистикой, охватить широкую аудиторию потенциальных потребителей.

Целью разработки выпускной квалификационной работы является разработка функциональных модулей для веб-маркетплейса – веб-приложения, предоставляющее пользователям площадку для продажи и покупки предметов одежды в сети Интернет.

Основными задачами разработки являются:

1. Анализ предметной области. На этом этапе необходимо провести анализ сферы, для внедрения в которую разрабатываются программные модули;
2. Анализ рынка существующих решений. На этом этапе необходимо провести анализ продуктов конкурентов во внедряемой сфере деятельности;
3. Постановка задачи. Необходимо спланировать разрабатываемый функционал программного продукта;
4. Анализ средств разработки. Разработчику необходимо из большого количества предоставляемых технологий разработки выбрать наиболее подходящую и удовлетворяющую требованиям проекта.

## **Общая часть**

## **Анализ предметной области**

В рамках выпускной квалификационной работы велась разработка клиентской части для веб-маркетплейса Neu. В разработку входили разработка пользовательского интерфейса, налаживание маршрутизации для разрабатываемых страниц веб-сайта, отображение созданных на этом веб-маркетплейсе объявлений о продаже тех или иных предметов одежды.

Целевой аудиторией разрабатываемого веб-сайта являются люди, заинтересованные в объявлении о продаже тех или иных предметов одежды в Интернете.

Разрабатываемый проект предназначен как для отдельных пользователей, так и коллективов людей, регистрирующих свой магазин на сайте.

## **Анализ рынка существующих решений**

Практически 2,05 миллиарда человек совершают покупки в Интернете почти каждый день. Таким образом, вывод вашего бизнеса на онлайн-рынок теперь является необходимостью, а не выбором. Люди уже ищут товары и услуги в Интернете и за последние годы потратили более 2,8 триллионов долларов. Пришло время подумать об установлении и укреплении своего присутствия в Интернете. Самый большой плюс онлайн-продаж — существующая клиентская база.

Наряду с этим установленное доверие к рынку и хорошо продуманный интерфейс делают его разумным выбором для продажи продуктов. Интернет-площадки связывают вас с потенциальными покупателями, позволяя вам демонстрировать, демонстрировать и побуждать ваших клиентов покупать продукты. Как онлайн-продавец, успех вашего бизнеса во многом зависит от торговой площадки, которую вы выбираете в зависимости от своего продукта. От Amazon до Flipkart в Интернете существует более 100 торговых площадок, и каждая из них имеет разную аудиторию, которая требует разных стратегий для продажи.

Категории продуктов, целевая аудитория, простота продажи, сборы за листинг, способы оплаты и т. д. При выборе торговой площадки для продажи ваших продуктов необходимо учитывать несколько моментов. Более того, выбор торговой площадки, которая оправдывает вашу нишу, может показаться нереально сложной задачей.

В рамках выпускной квалификационной работы было проанализировано 5 самых обширных маркетплейсов в разных сферах: начиная от предметов одежды и до музыкальных инструментов.

* + 1. **Amazon Marketplace**

Amazon является мировым лидером и самой популярной онлайн-площадкой для бизнеса, которую в прошлом году посещало около 197 миллионов человек в месяц. На торговой площадке Amazon насчитывается более 5 миллионов онлайн-продавцов, а выручка от продаж продукции в 2020 году составила 118,57 миллиарда долларов. Годовой объем продаж более 140 000 сторонних продавцов превысил 100 000 долларов, что делает его очень прибыльным при правильном использовании и в то же время увеличивает конкуренцию. Начавшись как книжный магазин, сегодня Amazon продает все, от иглы до электроники, до самых малоизвестных вещей на планете. Итак, что бы вы ни продавали на Amazon, на это есть всегда есть рынок и спрос.

Чтобы продавать на Amazon, вам необходимо создать учетную запись продавца, выбрать план и загрузить списки своих продуктов на AWS Marketplace. За создание учетной записи профессионального продавца взимается ежемесячная абонентская плата в размере 39,99 долларов США и другие сборы.

* + 1. **Etsy**

Etsy — это онлайн-маркетплейс для инновационных предпринимателей с 33-миллионной базой пользователей. Etsy обычно продает винтажные товары, товары ручной работы и товары для рукоделия. Etsy предлагает независимый вариант продажи, сохраняя уникальность вашего бренда. Всякий раз, когда покупатель ищет продукт на страницах продукта на Etsy и нажимает на название вашего бренда, он будет перенаправлен на вашу витрину Etsy Storefront. Итак, независимо от того, есть у вас веб-сайт или нет, у вас есть независимая витрина в Интернете. Это делает Etsy лучшим выбором для мелкого и среднего бизнеса, художников, а также антикварных или винтажных магазинов, желающих продавать их через Интернет. Регистрация и создание магазина на платформе бесплатны, но включают три основных торговых сбора: листинговый сбор, комиссию за транзакцию и комиссию за обработку платежа.

* + 1. **Reverb**

Reverb.com (приобретен Etsy в 2019 году) — это онлайн-маркетплейс, объединяющий продавцов и покупателей музыкальных инструментов и аппаратуры по всему миру. Reverb ежемесячно посещает до 10 миллионов пользователей, что составляет до 92,5 миллионов посетителей в год, а клиентская база стабильно составляет более 1,05 миллиона человек. Платформа представляет собой активное сообщество людей, любящих музыку, и идеально подходит для любого продавца, занимающегося музыкальным оборудованием. Чтобы продавать на Reverb, поставщики должны заполнить подробную форму, содержащую всю необходимую информацию о продуктах, которые они хотят продавать. Комиссия платформы включает в себя: комиссию за продажу и комиссию за обработку платежа.

* + 1. **ASOS Marketplace**

ASOS — это торговая площадка для молодых людей, объединяющая независимые марки и винтажные товары из разных стран, таких как США, Италия, Испания, Австралия, Франция, Германия и т. д. Если вы занимаетесь продажей аналогичных товаров, торговая площадка ASOS может стать в выгодном выборе для многоканальной стратегии. Для того, чтобы продавать на ASOS, необходимо создать свой бутик (персональный магазин). Взимаемые сборы включают абонентскую плату, комиссию за продажу и рекламные сборы (необязательно). ASOS Marketplace — это отличная возможность для модельеров и брендов одежды, занимающихся винтажной одеждой, необычными модными товарами и аксессуарами.

* + 1. **Houzz**

Houzz была основана как онлайн-платформа в феврале 2009 года Ади Татарко и ее мужем Алоном Коэном в ответ на трудности, с которыми они столкнулись при реализации собственного проекта реконструкции дома. Им было трудно сообщить о своем видении дома и найти подходящих специалистов для своего проекта. Коэн сам закодировал первоначальный веб-сайт, и они попросили нескольких архитекторов из района залива загрузить свои портфолио, чтобы дать реставраторам дома идеи для своих проектов. Сайт распространился из уст в уста, и они начали получать электронные письма от домовладельцев и специалистов по дому за пределами области залива с просьбами открыть больше категорий на Houzz и расшириться на другие области. Houzz стала компанией осенью 2010 года.

Houzz — это лучший выбор для дизайн дома и интерьера, скандинавский онлайн-маркетплейс, на котором представлен широкий ассортимент из 10 миллионов различных продуктов, перечисленных в 900 различных категориях и подкатегориях. Сайт Houzz посещают до 40 миллионов человек в месяц, и более 20 000 различных продавцов продают товары на Houzz. Плюс продаж на Houzz заключается в том, что он использует дополненную 3D-реальность, чтобы помочь потребителям визуализировать мебель и декор в своем доме, и набирает популярность и успех среди покупателей. Houzz взимает комиссию в размере 15% за продажу товаров через свою торговую площадку.

## **Постановка задачи**

Разрабатываемый веб-сайт маркетплейса «Neu.» должен предоставлять пользователям возможность размещения объявлений о продаже элементов одежды и аксессуаров, возможность связи потенциальных покупателя с продавцами.

Интерфейс разрабатываемого продукта должен предоставлять покупателям и продавцам возможности регистрации учетной записи в системе, поиска товаров, сортировки товаров на странице, разрабатываемым интерфейсом должна предоставляться контактная информация продавца конкретного предмета, а именно номер телефона и электронная почта.

Разрабатываемый продукт предусматривает:

1. регистрацию в системе на отдельной странице «Registration»
2. просмотр всех товаров с помощью страницы «Catalog»;
3. просмотр отдельных выборок товаров – популярные «Popular» и новые «Newest»;
4. просмотр детальной информации о товаре на отдельной странице по нажатии на его в списке «Catalog» или «Popular» и «Newest»;

## **Обоснование и выбор методики, технологии и инструментальных средств проектирования и разработки**

Для разработки выпускной квалификационной работы рассматривались следующие аспекты: языки программирования, возможные библиотеки и фреймворки, среды разработки.

* + 1. **Сравнительная характеристика языков программирования**

Основными языками разработки были выбраны:

* 1. HTML (язык гипертекстовой разметки, отвечат за отображение контента на пользовательской стороне);
  2. CSS (каскадные таблицы стилей, отвечает за стилизацию отображаемых элементов);
  3. TypeScript, так как он прекрасно сочетается в разработке с различными front-end фреймворками, целью которых является построение различных пользовательских интерфейсов.
     + 1. **HTML**

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора. Элементы HTML являются строительными блоками HTML страниц. С помощью HTML разные конструкции, изображения и другие объекты, такие как интерактивная веб-форма, могут быть встроены в отображаемую страницу.

HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы HTML выделяются тегами, записанными с использованием угловых скобок. Такие теги, как <img /> и <input />, напрямую вводят контент на страницу. Другие теги, такие как <p>, окружают и оформляют текст внутри себя и могут включать другие теги в качестве подэлементов. Браузеры не отображают HTML-теги, но используют их для интерпретации содержимого страницы.

Язык XHTML является более строгим вариантом HTML, он следует синтаксису XML и является приложением языка XML в области разметки гипертекста. В HTML можно встроить программый код на языке программирования JavaScript, для управления поведением и содержанием веб-страниц. Также включение CSS в HTML описывает внешний вид и макет страницы.

* + - 1. **CSS**

CSS (англ. Cascading Style Sheets «каскадные таблицы стилей») — формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL. CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц.

Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана) или при выводе устройствами, использующими шрифт Брайля.

* + - 1. **TypeScript**

TypeScript — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js. Код экспериментального компилятора, транслирующего TypeScript в JavaScript, распространяется под лицензией Apache. Его разработка ведётся в публичном репозитории через сервис GitHub.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

На момент релиза представлены файлы для восприятия расширенного синтаксиса TypeScript для Vim и Emacs, а также плагин для Microsoft Visual Studio.

TypeScript возник из-за предполагаемых недостатков JavaScript в крупномасштабных приложениях как в компании Microsoft, так и у прочих пользователей JavaScript. Проблемы с разработкой сложных программ на JavaScript привели к необходимости облегчения разработки компонентов языка. Разработчики TypeScript искали решение, которое не будет нарушать совместимость со стандартом и его кроссплатформенной поддержкой. Зная, что только стандарт ECMAScript предлагает поддержку в будущем для программирования на базе классов (Class-based programming), TypeScript был основан на этом предположении. Это привело к созданию компилятора JavaScript с набором синтаксических языковых расширений, увеличенным на основе предложения, которое трансформирует расширения в JavaScript. В этом смысле TypeScript является представлением того, что ожидать от ECMAScript 6. Уникальный аспект не в предложении, а в добавлении в TypeScript статической типизации, что позволяет статически анализировать язык, облегчая оснастки и IDE поддержку.

* + 1. **Сравнительная характеристика библиотек и фреймворков**

На данный момент не существует объективных критериев, которые однозначно показали применимость той или иной технологии, но составим список параметров исходя из основ разработки.

К основным критериям по выбору фреймворка, должны быть отнесены следующие параметры:

1. время разработки;
2. процесс разработки;
3. технологические возможности фреймворка.

Время разработки — основной ресурс разработчика. Никому не нужен проект, который будет сделан более чем за “год/два/три”. Чем больше время разработки, тем выше стоимость, тем меньше привлекательность для инвесторов, тем меньше денег можно заработать. Это палка о двух концах. Но всегда нужно учитывать, если проект нужно делать в сжатые сроки (либо есть какие-либо сроки), то скорость разработки очень важна, и нужно понимать, что скорость разработки обратно пропорционально качеству разработки.

Процесс разработки говорит о том, как зависит разработка от количества задач и разработчиков, на результат. Есть ошибочное утверждение, что если разработчик может сделать задачу за 2 дня, то если добавить ещё одного такого же разработчика (который не является его братом близнецом), то задачу можно выполнить за 1 день, но это чаще всего не так, если разработчики, конечно, не братья близнецы.

Технологические возможности фреймворка стоят на 3 месте из-за того, что только после понимания сроков и процесса разработки, можно рассматривать преимущества фреймворка. Нет смысла выбирать фреймворк с быстрой разработкой для качественной разработки, так как это одна из утопических идей, погубившей не один Uber.

Все это можно свести к следующему банальному выводу:

1. проект определяет потребности;
2. исходя из потребностей проекта и навыков команды необходимо выбирать то, что лучше подходит;
3. определившись с выбором необходимо понять, что из этого можно сделать.
   * + 1. **Angular**

Angular — открытая и свободная платформа для разработки веб-приложений, написанная на языке TypeScript, разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний. Angular — полностью переписанный фреймворк от той же команды, которая написала AngularJS.

Преимущества Angular:

1. Angular используется вместе с Typescript. Он имеет исключительную поддержку для этого;
2. Angular-language-service — обеспечивает интеллектуальные возможности и автозаполнение шаблона HTML-компонента;
3. Новые функции, такие как generation Angular, использующие библиотеки npm из CLI, generation, и разработка компонентов, использующая Angular;
4. Подробная документация, позволяющая разработчику получить всю необходимую информацию, не прибегая к помощи его коллег. Однако это требует больше времени для обучения;
5. Односторонняя привязка данных, которая обеспечивает исключительное поведение приложения, что сводит к минимуму риск возможных ошибок;
6. MVVM (Model-View-ViewModel), которая позволяет разработчикам работать отдельно над одним и тем же разделом приложения, используя один и тот же набор данных;
7. Внедрение зависимостей от компонентов, связанных с модулями и модульностью в целом;
8. Структура и архитектура, специально созданные для большой масштабируемости проекта.

Минусы Angular:

1. Разнообразие различных структур (Injectables, Components, Pipes, Modules и т. д.) усложняет изучение по сравнению с React и Vue.js, у которых есть только «Component».
2. Относительно медленная производительность, учитывая различные показатели. С другой стороны, это можно легко решить, используя так называемый «ChangeDetectionStrategy», который помогает вручную контролировать процесс рендеринга компонентов.
   * + 1. **React**

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

Преимущества React:

1. Легко изучить, благодаря простому дизайну, использованию JSX (HTML-подобный синтаксис) для шаблонов и очень подробной документации. Разработчики тратят больше времени на написание современного JavaScript и меньше беспокоятся о коде, специфичном для фреймворка;
2. Очень быстрая, благодаря реализации React Virtual DOM и различным оптимизациям рендеринга;
3. Отличная поддержка рендеринга на стороне сервера, что делает его мощной платформой для контент-ориентированных приложений;
4. Первоклассная поддержка Progressive Web App (PWA) благодаря генератору приложений `create-react-app`;
5. Привязка данных является односторонней, что означает меньше нежелательных побочных эффектов;
6. Redux, самая популярная платформа для управления состоянием приложений в React, ее легко учить и использовать;
7. React реализует концепции функционального программирования (FP), создавая простой в тестировании и многократно используемый код;
8. Переход между версиями, как правило, очень прост: Facebook предоставляет «кодовые модули» для автоматизации большей части процесса;
9. Навыки, полученные в React, могут быть применены к разработке на React Native.

Минусы React:

1. React не однозначен и оставляет разработчикам возможность выбирать лучший способ развития. Это может быть решено сильным лидерством проекта и хорошими процессами;
2. Сообщество делится по способам написания CSS в React, которые разделяются на традиционные таблицы стилей (CSS Modules) и CSS-in-JS (т.е. Emotion и Styled Components);
3. React отходит от компонентов на основе классов, что может стать препятствием для разработчиков, которым более комфортно работать с объектно-ориентированным программированием (ООП);
4. Смешивание шаблонов с логикой (JSX) может сбить с толку некоторых разработчиков при первых знакомствах с React.
   * + 1. **Vue**

Vue.js — JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

Плюсы Vue:

1. Усиленный HTML – это означает, что Vue.js имеет много характеристик схожих с Angular, а это, благодаря использованию различных компонентов, помогает оптимизации HTML- блоков.
2. Подробная документация – Vue.js имеет очень подробную документацию, которая может ускорить процесс обучения для разработчиков и сэкономить много времени на разработку приложения, используя только базовые знания HTML и JavaScript.
3. Благодаря предусматриваемой Vue адаптивности, может быть осуществлен быстрый переход от других фреймворков к Vue.js из-за сходства с Angular и React с точки зрения дизайна и архитектуры.
4. Потрясающая интеграция фреймворка позволяет использовать Vue.js как для создания одностраничных приложений, так и для более сложных веб-интерфейсов приложений. Важно, что небольшие интерактивные элементы можно легко интегрировать в существующую инфраструктуру без негативных последствий.
5. Масштабирование Vue.js может помочь в разработке довольно больших шаблонов многократного использования, которые могут быть сделаны почти за тоже время, что и более простые.
6. Vue.js весит около 20 КБ, сохраняя при этом свою скорость и гибкость, что позволяет достичь гораздо лучшей производительности по сравнению с другими платформами.

Минусы Vue:

1. Недостаток ресурсов. Vue.js по-прежнему занимает довольно небольшую долю рынка по сравнению с React или Angular, что означает, что обмен знаниями в этой среде все еще находится на начальной стадии.
2. Риск чрезмерной гибкости. Иногда у Vue.js могут возникнуть проблемы при интеграции в огромные проекты, и пока еще нет опыта возможных решений, но они обязательно появятся в ближайшее время.
   * 1. **Сравнительная характеристика сред разработки**
        1. **Visual Studio Code**

В качестве более простого решения имеет смысл рассмотреть бесплатный, но очень мощный и популярный редактор Visual Studio Code — он предлагает не так много возможностей, как IDE, зато позволяет писать код более чем на 72 языках и включает функции отладки. VS Code поддерживается не только на Windows и macOS, но и на Linux. В редакторе есть умное автодополнение IntelliSense, встроенная интеграция с Git, а также огромная библиотека расширений.

Преимущества Visual Studio Code:

1. Сниппеты – это серьёзный способ повышения продуктивности программиста. Дело в том, что они помогают автоматизировать ручной труд. В результате программист может тратить больше времени на размышления о том, как некие конструкции сочетаются и как они работают вместе, а не на то, чтобы писать код этих конструкций. Использование сниппетов, кроме того, может помочь программисту быстрее освоить некий фреймворк или некую библиотеку. Дело тут в том, что человеку приходится тратить меньше времени на то, чтобы понять причину некоей ошибки. То, что ему нужно, чаще оказывается работоспособным с первого раза;
2. IntelliSense — это система автозавершения кода. Она весьма интеллектуальна, что выгодно отличает её от других подобных систем. Об IntelliSense говорят, как об одном из основных средств VS Code, обеспечивающих высокую производительность труда. При этом многие, выбирающие VS Code, пользуются этой возможностью далеко не на полную мощность. Если навести мышь на некий участок кода, IntelliSense покажет дополнительные сведения о типах и даст возможность добраться до сведений об источнике их описания. В то время как эта возможность может оказаться полезной при работе с незнакомыми сущностями, созданными другими людьми, основная ценность IntelliSense заключается в том, что эта система способна предсказывать действия программиста;
3. Клавиатурные сокращения – они позволяют решать большинство задач только с помощью клавиатуры. Панель клавиатурных сокращений можно посмотреть, воспользовавшись комбинацией клавиш CTRL + K + S. Если кому-то не хочется изучать совершенно новый набор «горячих клавиш», он может создать такой список сам или импортировать клавиатурные сокращения из других популярных редакторов. Возможности гибкой настройки клавиатурных сокращений позволяют тем, кто уже к чему-то привык, быстрее достичь высокой продуктивности работы в VS Code;
4. Работа с Git – прямо из VS Code можно делать коммиты в Git-репозитории. Тут вам понадобится панель Source Control, открываемая соответствующей кнопкой, расположенной в левой части экрана. Например, для подготовки изменённого файла к коммиту можно воспользоваться кнопкой со знаком +, находящейся в верхней части этой панели. После этого можно вызвать контекстное меню этого файла и найти в нём дополнительные команды;
   * + 1. **IntelliJ IDEA**

IntelliJ IDEA — Java-ориентированная платформа для разработки от JetBrains. Несмотря на это, она позволяет работать со всеми языками, которые мы упоминали выше (HTML, CSS, JavaScript, PHP и Python). Из коробки вам будут доступны инструменты для написания кода на HTML, CSS и JavaScript (в версии Ultimate). Поддержку PHP и Python можно добавить с помощью плагинов. IntelliJ IDEA доступна для систем Windows, macOS и Linux.

Ключевые функции:

1. умное автодополнение, которое предлагает элементы кода исходя из текущего контекста;
2. встроенная отладка;
3. встроенная интеграция с системами контроля версий;
4. интеграция с инструментами сборки, такими как Apache Maven, Gradle и Webpack.

IntelliJ IDEA поставляется в трёх ценовых вариантах. Community-версия доступна бесплатно, однако она не включает себя поддержку JavaScript и работу с инструментами базами данных, что может быть критично для веб-разработки. Стоимость индивидуальной лицензии IntelliJ IDEA Ultimate — 149 $ в год, для организаций же цена составит 499 $ на пользователя в год. Также можно попробовать версию Ultimate бесплатно в течение 90 дней.

* + - 1. **WebStorm**

Для разработчиков на JavaScript JetBrains предлагает платформу WebStorm. Она поддерживает популярные фреймворки для фронтенда (Angular, React, Vue.js) и бэкенда (Node.js, Meteor).

Среди преимуществ IDE можно выделить:

1. умное автодополнение кода;
2. встроенный отладчик;
3. интеграция с популярными системами контролями версий.

WebStorm не имеет бесплатной лицензии, цена для индивидуального использования — 59 $ в год, для компаний — 129 $ в год на пользователя, но в то же время предоставляет бесплатную 30-дневную версию для ознакомления с данной средой разработки.

* + 1. **Технология разработки**

Разрабатываемое в рамках выпускной квалификационной работы веб-приложение представляет из себя одностраничное приложение (англ. – single page application, SPA). Главной проблемой Single Page приложений является то, что сервер отдает клиенту пустую HTML страницу. Её формирование происходит только после того, как весь JS будет скачан (это весь ваш код, библиотеки, фреймворк). Это в большинстве случаев более 2-х мегабайт размера + задержки на обработку кода. Даже если Google-бот умеет выполнять JS, он получает контент только спустя некоторое время, критичное для ранжирования сайта. Получается, что Google-бот попросту видит пустую страницу несколько секунд, что является серьезной проблемой.

Также есть менее продвинутые поисковые системы, которые попросту не умеют работать с JS. В них Single Page Application не будут индексироваться.

Существует несколько путей как решить проблему пустой странички при загрузке, рассмотрим несколько из них:

1. Static Site Generation (SSG). Сделать пререндер сайта перед тем, как его загрузить на сервер.
2. Server-Side Rendering (SSR). Рендерить контент в рантайме на сервере. При таком подходе мы сможем делать запросы backend API и отдавать HTML вместе с необходимым содержимым.

Архитектура SSR должна быть устроена следующим образом:

1. должен быть спроектирован сервер, который выполняет наше приложение точно так же, как делал бы это пользователь в браузере. Делая запросы на необходимые ресурсы, отображая весь необходимый HTML, наполняя состояние.
2. сервер отдает клиенту наполненный HTML, наполненное состояние, а также отдает все необходимые JS, CSS и прочие ресурсы.
3. клиент, получая HTML и ресурсы, синхронизирует состояние и работает с приложением как с обычным Single Page Application. При этом важным моментом является то, что состояние должно синхронизироваться.

Схематично устройство SSR приложения изображено на рисунке 1:

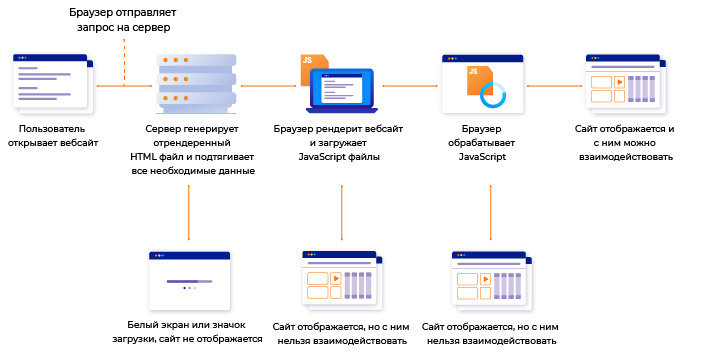


Рисунок 1 – SSR архитектура

* + 1. **Вывод по разделу**

Проведя программных решений для разработки, был выбран следующий пакет инструментов для разработки:

1. языками программирования были выбраны HTML, CSS, TypeScript;
2. в разработке использовался фреймворк React, библиотеки Redux, Redux-Saga и Immer для работы с состоянием приложения, React-Router для обеспечения маршрутизации между страницами;
3. средой разработки был выбран Visual Studio Code;
4. технологией разработки было выбрано SSR (Server-Side Rendering).

## **Специальная часть**

## **Описание структуры приложения**

Клиентская часть разрабатываемого приложения построена с помощью фреймворка React и языка TypeScript. Со стороны клиента отправляется запрос (например, на перенаправление на другую страницу) на сервер; на сервере, в свою очередь, запрос обрабатывается и в зависимости от него выполняется определённая логика, как, например, перенаправление пользователя на другую страницу, асинхронная загрузка данных и так далее.

* + 1. **Клиентская сторона приложения**

Структура клиентского приложения состоит из файла энд-поинта клиентской части (client.tsx), визуальной составляющей, а именно:

1. спроектированных React-компонентов (директория components);
2. страниц браузера (директория pages).

В каждой из подкатегорий той или иной части визуальной составляющей определены поддиректории конкретных компонентов и страниц, которые рендерятся приложением (с расширением .tsx) и файлы .css, стилизирующие их.

Компоненты:

1. App – layout-компонент, общий для каждой страницы, в котором так же обозначена логика для редиректа с использование роутера;
2. Button – кнопка, присутствующая в большинстве страниц;
3. Footer – «подвал» сайта;
4. Header – «шапка» сайта. Этот компонент содержит в себе ссылки на другие страницы приложения, логотип приложения и форму поиска;
5. Page – внешняя обёртка для основного контента страницы;
6. PageContainer – контейнер, размещающий блоки со всей полезной информацией;
7. PageMeta – компонент, который рендерит метаданные документа
8. Rect – компонент, отвечающий за отображение заглушек в период времени, когда страница браузер отрисовал саму страницу, но не отрисовал данные (First Contentful Paint);
9. Sneakers – компонент, выполняющий отрисовку карточки товара на странице сайта;
10. SneakersList – контейнер компонентов Sneakers.

Структура компонентов приложения представлена на рисунке 2.

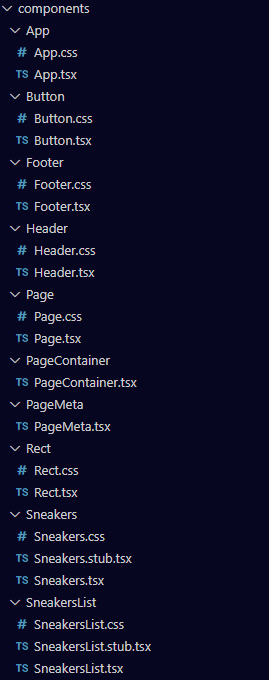


Рисунок 2 – Структура компонентов

Страницы:

1. 404 – страница, которая отрендерится тогда, когда сервер не вернёт указанный URL (попросту не существующий в приложении). В случае отображения это страницы, пользователь увидит сообщение «Page Not Found»;
2. Auth – страница авторизации в системе;
3. Catalog – страница, где отображаются все данные;
4. Home – домашняя страница сайта;
5. Registration – страница регистрации;
6. Sneakers – страница, отображающая полную информацию по конкретному предмету одежды (кроссовку).

Структура страниц приложения изображена на рисунке 3.

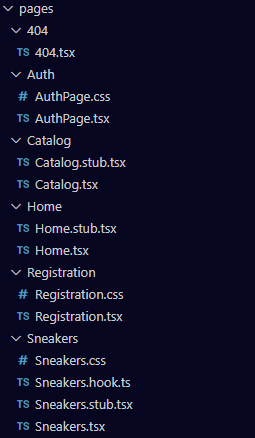


Рисунок 3 – Структура страниц приложения

Так же для разрабатываемой визуальной части была определена логика работы компонентов и страниц: перенаправление на другую страницу (далее – редирект), асинхронная загрузка данных, логика управления состоянием приложения.

Информация, необходимая для маршрутизации находится в файле routes.ts (рисунок 4):



Рисунок 4 – Устройство роутов

У элементов массива есть поле path – шаблон адреса, с которым будет сопоставляться запрошенный адрес URL, component – компонент, который отвечает за обработку запроса по этому маршруту, exact – компонент, который отключает частичное соответствие для маршрута и гарантирует, что он возвращает маршрут только в том случае, если путь является точным совпадением с текущим URL-адресом, метод fetchData() – метод, который запрашивает определённые данные в зависимости от компонента, на который происходит редирект.

* + 1. **Управление состоянием приложения и маршрутизацией с помощью библиотек Redux и Router**
       1. **Redux**

Redux — это инструмент для управления состоянием данных и пользовательским интерфейсом в приложениях JavaScript с большим количеством сущностей. Redux используется в связке с фреймворками для JavaScript: React, TypeScript, Vue, Angular и другими. Реже он бывает нужен для написания кода на чистом JS. Имеет открытый исходный код и доступен бесплатно. Со всеми зависимостями весит всего около 2 Кб.

В веб-разработке библиотека используется для:

1. для управления состоянием приложения, работающего с большим количеством данных;
2. удобной замены встроенных средств работы с состоянием в React;
3. более легкого масштабирования приложения, его преобразования под разные задачи;
4. избавления от ошибок, связанных с беспорядком в объекте состояния;
5. предсказуемости и понятности работы приложения; более простой отладки и доработки;
6. повышения производительности и работоспособности программы.

Состояние — это концепт, который обычно используют в сложных программах. Большое приложение содержит много данных, которые в любой момент могут потребоваться тому или иному компоненту. Компоненты могут общаться напрямую, но в сложной программе реализовывать такой подход неудобно. Необходимо дописывать каждую функцию. Это загромождает код. Поэтому требуется состояние — объект, который можно сравнить с диспетчерской. Он получает, хранит и при необходимости передает одним компонентам данные других. К нему можно обратиться, чтобы узнать, нажата ли кнопка, какое значение сейчас у переменной, выбрано ли условие. Это удобнее и проще, чем получать сведения непосредственно от компонента.

Redux базируется на трёх принципах, из которых следует характер работы с ним:

1. Единый источник состояния. Все данные о состоянии приложения хранятся в одном месте, без копий. Глобальное состояние организовано как дерево объектов. Также используются термины «источник состояния» и «хранилище». Оба означают местонахождение глобального состояния. Единый источник нужен для централизации и отладки приложения. Работа становится проще, если данные находятся в одном месте.
2. Доступ к состоянию — только для чтения. Глобальное состояние заблокировано для записи. Компоненты приложения могут читать из него, но не переписывать по своему желанию. Это предотвращает непредсказуемые изменения. Ситуаций, когда объект изменился, а функции об этом не узнали, не бывает. При изменениях в состояние нужно отправить действие (action). Специальный объект сообщит о произошедшем.
3. Изменения — только через редукторы. Когда в состояние поступает действие, его обрабатывают редукторы, или редьюсеры (reducers). Это чистые функции — результат их выполнения зависит только от входных данных. Редукторы берут объект состояния компонента, который изменился, и действие. На их основе они генерируют новый объект состояния. Функции всегда создают новый объект. Если действие сложное, а приложение большое, для него могут запуститься несколько редукторов. Результат работы редуктора — новый объект состояния с актуальными данными, рассчитанными на основе информации из объекта-действия. Он попадает в дерево состояний вместо старого.

В разрабатываемом приложении Redux используется вкупе с предоставляемым библиотекой промежуточным ПО (англ. – middleware) redux-saga. Это ПО используется в основном для обработки побочных эффектов состояния приложения, с которыми не умеет справляться Redux. Одними из таких эффектов являются асинхронные запросы на сервер, выгрузка данных и помещение их в начальное состояние. На стороне клиента приложение оборачивается в компонент <ReduxProvider>, в который передается store – хранилище дерева состояний приложения.

* + - 1. **React Router**

В React имеется своя система маршрутизации, которая позволяет сопоставлять запросы к приложению с определенными компонентами. Ключевым звеном в работе маршрутизации является модуль React Router, который содержит основной функционал по работе с маршрутизацией. Однако если мы собираемся работать в браузере, то нам также надо использовать модуль react-router-dom, а также history.

Router определяет набор маршрутов и, когда к приложению, приходит запрос, то Router выполняет сопоставление запроса с маршрутами. И если какой-то маршрут совпадает с URL запроса, то этот маршрут выбирается для обработки запроса. И также для выбора маршрута определен объект Switch. Он содержит набор маршрутов и позволяет выбрать первый попавшийся маршрут и его использовать для обработки. Каждый маршрут представляет объект Route. Он имеет ряд атрибутов. В частности, здесь для маршрута устанавливаются следующие атрибуты: path: шаблон адреса, с которым будет сопоставляться запрошенный адрес URL component - тот компонент, который отвечает за обработку запроса по этому маршруту, exact – компонент, который отключает частичное соответствие для маршрута и гарантирует, что он возвращает маршрут только в том случае, если путь является точным совпадением с текущим URL-адресом.

Для совмещения работы Redux и маршрутизации используется <ConnectedRouter> из библиотеки connected-react-router, который может работать с объектами history (рисунок 5).

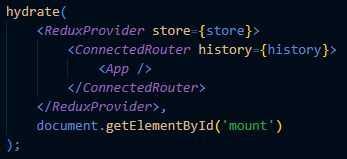


Рисунок 5 – Работа Redux и маршрутизации

В свою очередь, компонент App использует Router для генерирования маршрутов. В компоненте импортируется файл routes.ts, где описаны все маршруты и генерирует каждый из них (рисунок 6):

Изображение выглядит как текст, экран

Автоматически созданное описание

Рисунок 6 – Генерация маршрутов компонентом App

## **Объектно-ориентированное проектирование системы**

Под объектно-ориентированным проектированием понимают методологию проектирования, соединяющую в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы. Именно объектно-ориентированная декомпозиция отличает объектно-ориентированное проектирование от структурного, то есть логическая структура системы при объектно-ориентированном подходе отражается абстракциями в виде классов и объектов, а при структурном - алгоритмами.

Базовым средством документирования результатов проектирования систем посредством этих методологий является Унифицированный язык моделирования – UML, создателями которого являются Гради Буч и Джеймс Рамбо, преследовавшие при его разработке следующие цели:

1. предоставить пользователям готовый к использованию выразительный язык визуального моделирования, позволяющий разрабатывать осмысленные модели и обмениваться ими;
2. предусмотреть механизмы расширяемости и специализации для расширения базовых концепций;
3. обеспечить независимость от конкретных языков программирования и процессов разработки;
4. язык должен быть одновременно точным и доступным для понимания.

UML был создан для определения, визуализации, проектирования и документирования программных систем, при этом не является языком программирования, однако на основании UML-моделей возможна генерация кода. UML содержит стандартный набор диаграмм для моделирования:

1. диаграммы вариантов использования – для моделирования бизнес-процессов организации (требований к системе);
2. диаграммы классов – для моделирования статической структуры классов системы и связей между ними; - диаграммы поведения системы;
3. диаграммы взаимодействия – для моделирования процесса обмена сообщениями между объектами (диаграммы последовательности; кооперативные диаграммы;
4. диаграммы состояний – для моделирования поведения объектов системы при переходе из одного состояния в другое;
5. диаграммы деятельностей – для моделирования поведения системы в рамках различных вариантов использования или моделирования деятельностей;
6. диаграммы реализации: диаграммы компонентов – для моделирования иерархии компонентов системы;
7. диаграммы размещения – для моделирования физической архитектуры системы.

В качестве планирования системы была спроектирована диаграмма прецедентов использования (Use-Case) (рисунок 7).

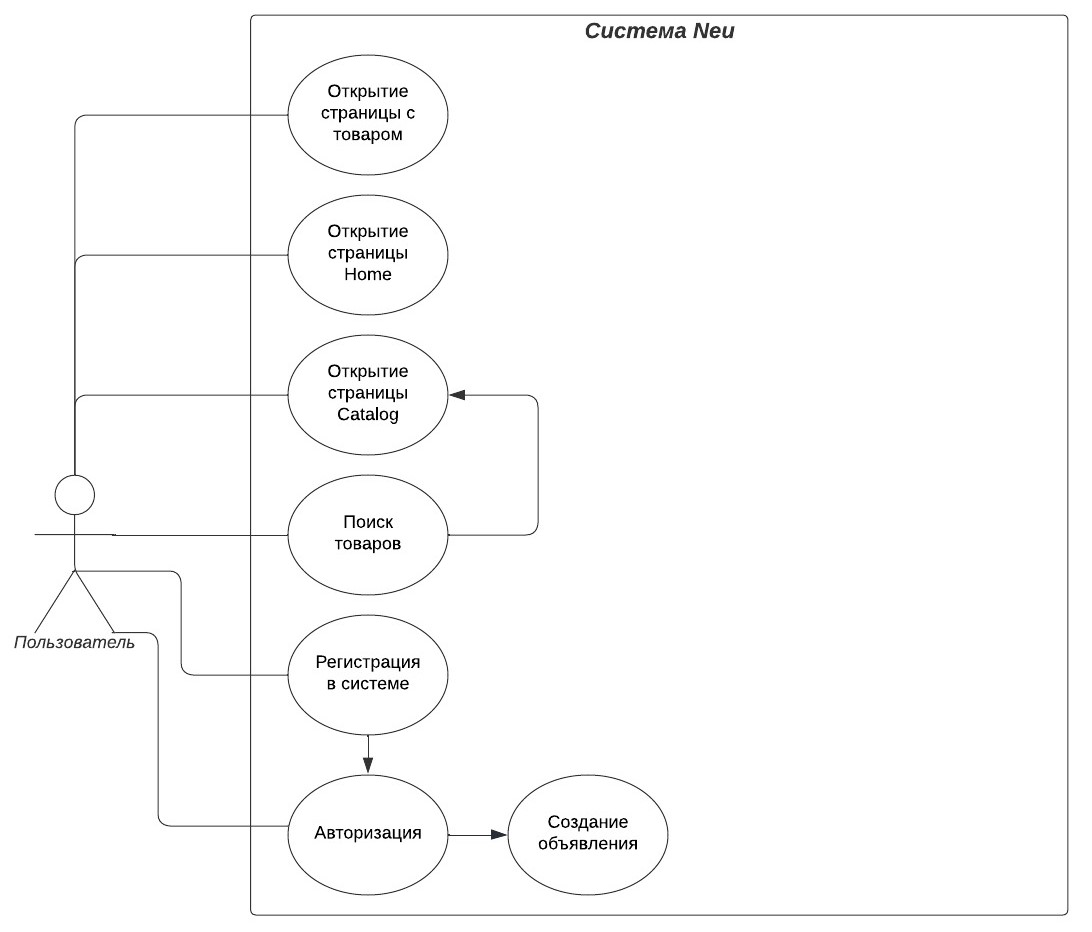


Рисунок 7 – Диаграмма прецедентов использования в нотации UML

## **Разработка пользовательского интерфейса приложения**

Домашняя страница является компонентом, открывающимся по умолчанию при запуске приложения. На ней отображены две выборки из заранее подготовленных данных, а именно «Популярное» и «Последние». Макет страницы представлен на рисунке 10.

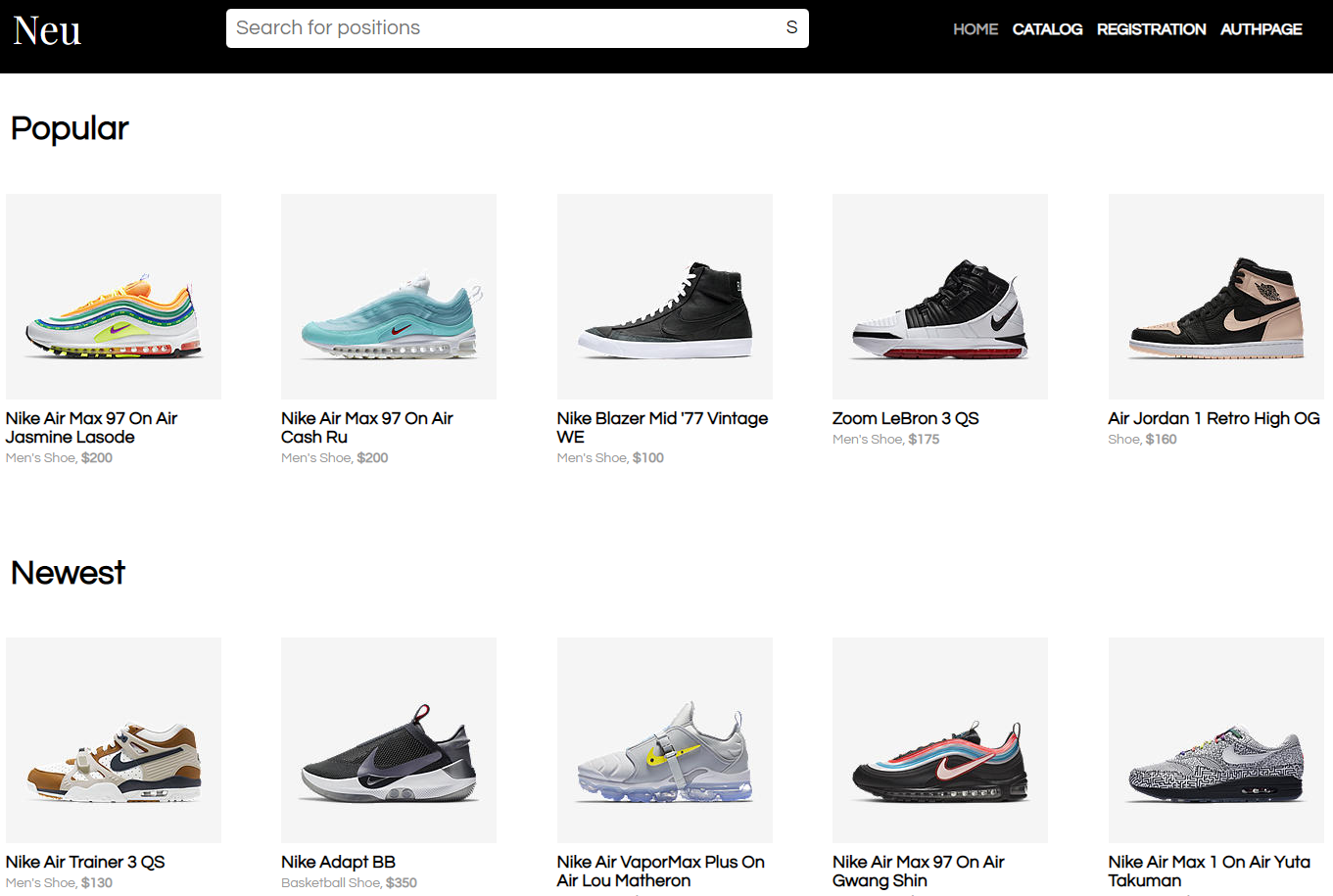


Рисунок 8 – Макет домашней страницы

Приложение не связано с реальными API и базой данных. Поэтому все данные заранее подготовлены, а для эмуляции реальной задержки ответа от сервера в методах-выборках каждой из страниц используется таймаут длинной в две секунды. Установленный таймаут позволяет продемонстрировать поведение страницы при задержке получения данных, а именно отображение страницы с «заглушками» в местах, где должен отображаться реальный контент. В случае с домашней страницей, таким компонентом является Homepage.stub. Отображение Homepage.stub представлено на рисунке 11.

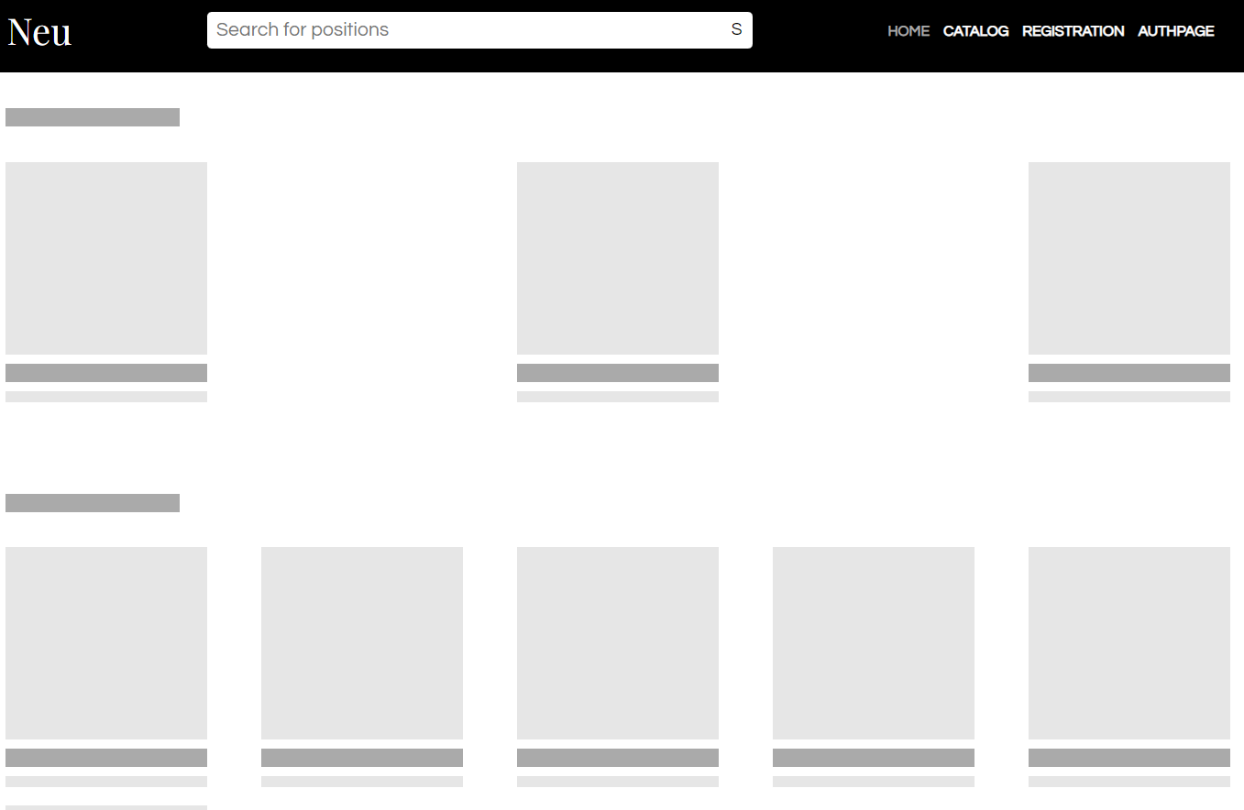


Рисунок 9 – отображение Homepage.stub

На странице каталога товаров представлена выборка всех предметов, находящихся в подготовленных данных. Макет страницы каталога товаров изображен на рисунке 10. Внешний вид страницы во время обработки загрузки данных представлен на рисунке 11.

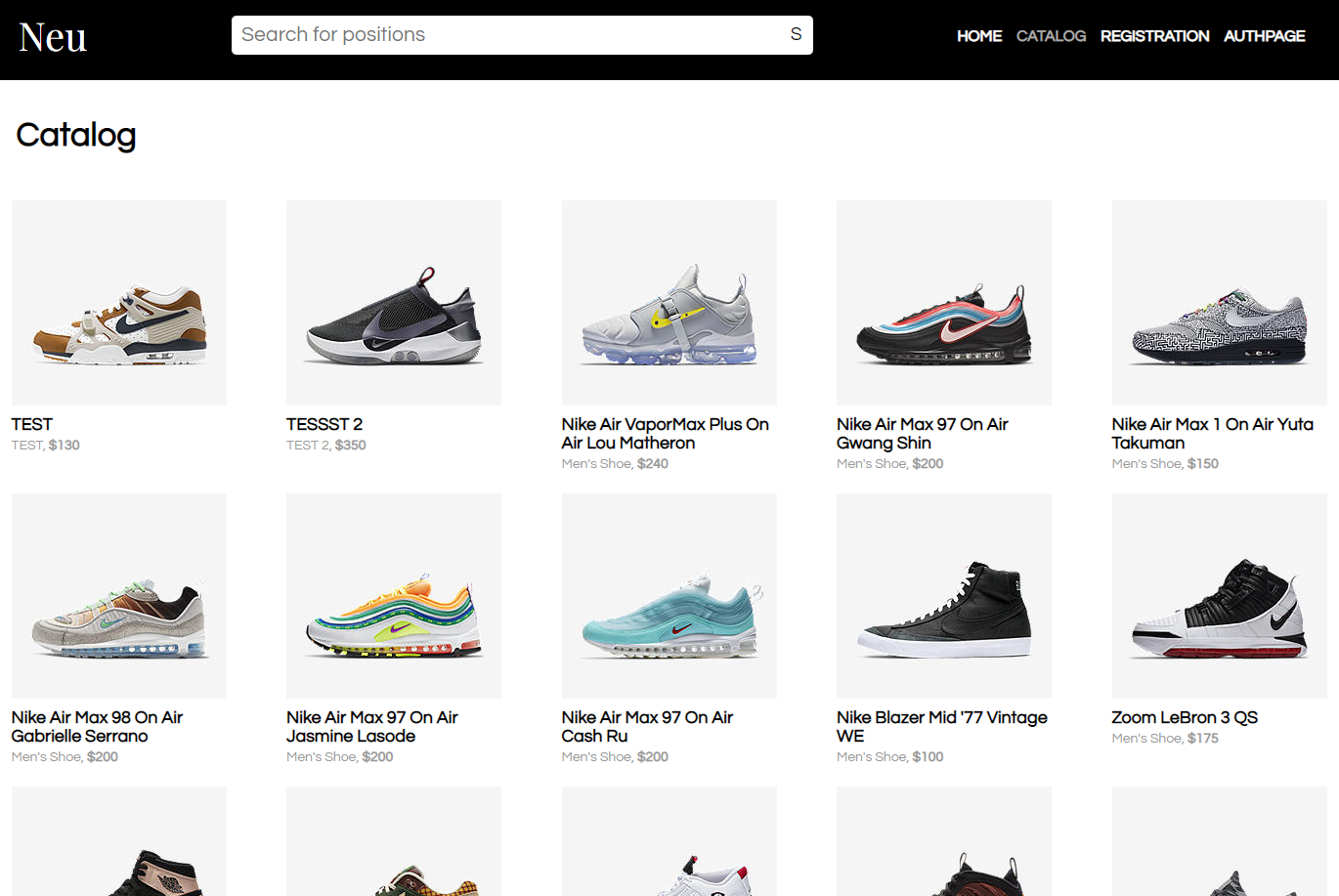


Рисунок 10 – Страница Catalog

Страница с информацией о конкретном кроссовке имеет URL-адрес типа ‘sneakers/:slug’, где slug – это понятная человеку строка-идентификатор конкретного элемента (эквивалент id). Отображение формируемого адреса представлено на рисунке 11. Из-за того, что не существует реальной базы данных, при каждой реадрессации с домашней страницы и каталога на страницу с подробной информацией о товаре будет отображаться slug конкретного товара; данные для отображения подгружаются из заранее приготовленного файла, который должен являться прототипом ответа от сервера на такого рода запрос (на получение данных по товару в отдельной странице).

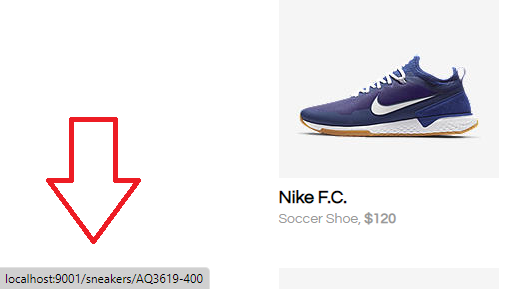


Рисунок 11 – Формирование адреса страницы с информацией товара

Макет страницы с подробной информацией о товаре представлен на рисунке 12.

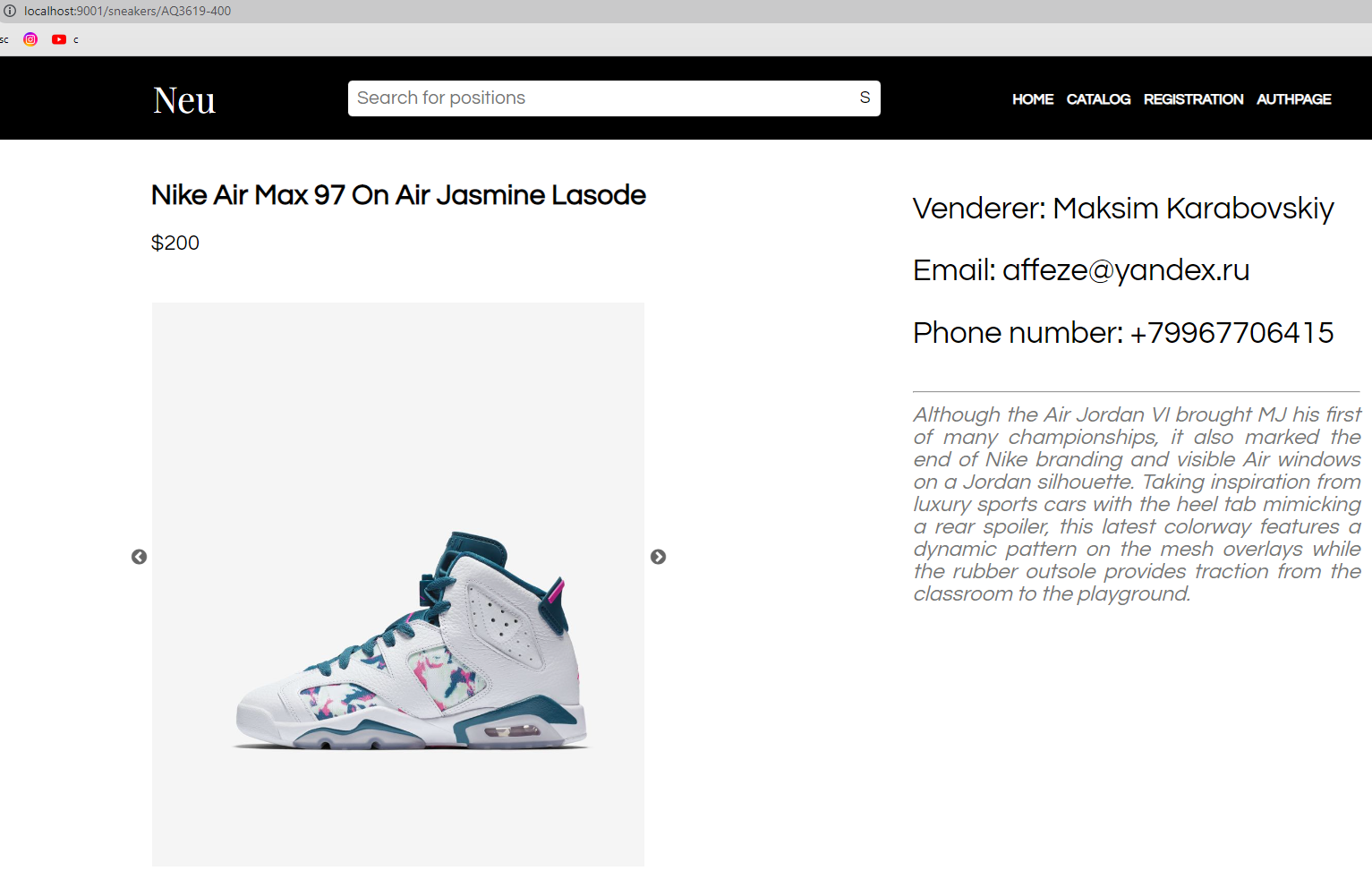


Рисунок 12 – Страница с информацией товара

Макет страницы Registration (регистрации) изображен на рисунке 16.

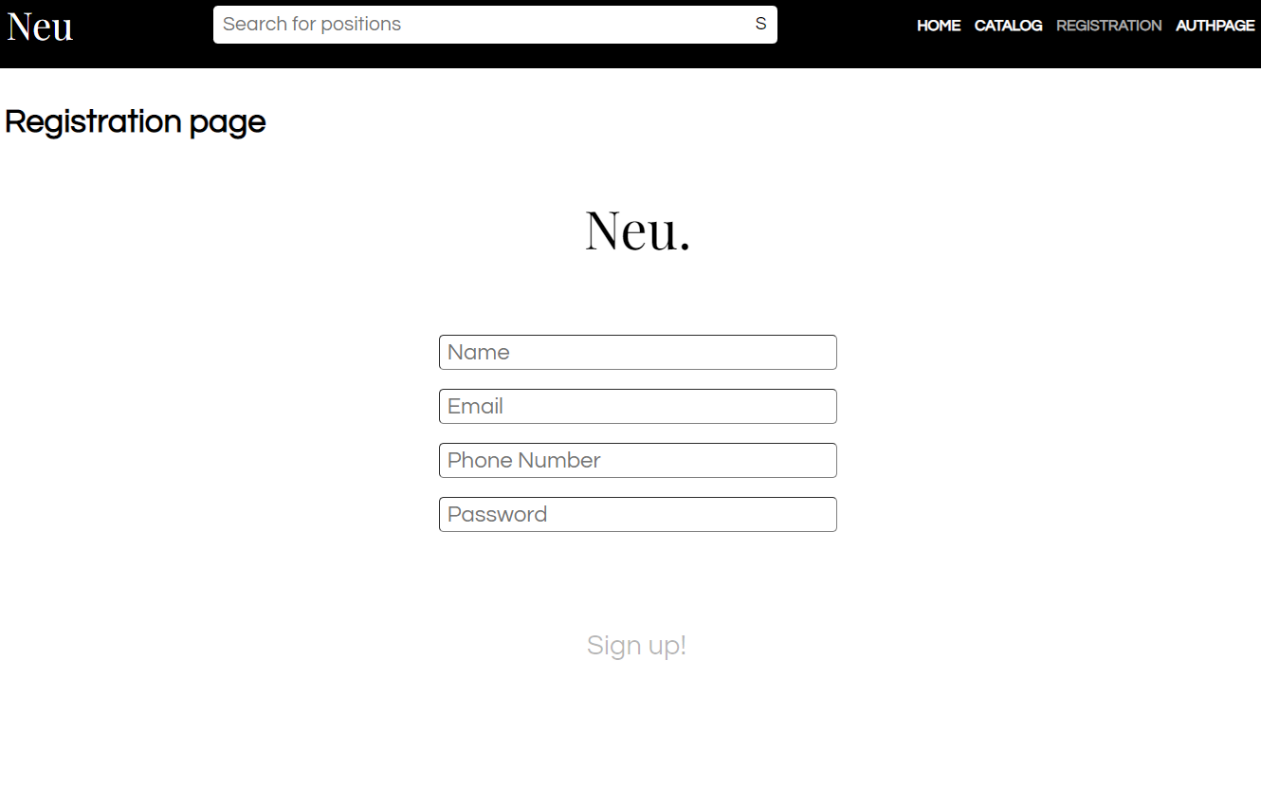


Рисунок 13 – Страница регистрации

Для регистрации в системе пользователю необходимо указать всю контактную информацию: его имя, адрес электронной почты, контактный номер телефона и пароль.

Валидация формы регистрации устроена таким образом, что когда пользователь теряет фокус с поля для ввода данных, интерфейс отображает ему на экране сообщение об ошибке с текстом «This field cannot be empty». Так же для таких полей ввода, как поле ввода электронной почты и пароля установлена дополнительная валидация – поле ввода электронной почты сравнивает вводимые символы с заранее подготовленным регулярным выражением (проверка на такие символы, как «@», символ точки), а поле ввода пароля сообщает пользователю о том, что пароль должен быть длинной не менее 8 символов.

В случае, если обработчики событий выявляют хотя бы одну ошибку, кнопка «Sign up!» становится недоступной. В случае, если ошибок нет и данные прошли валидацию, отправка формы становится доступной для пользователя.

Отображение ошибок ввода пользователю представлено на рисунке 14.

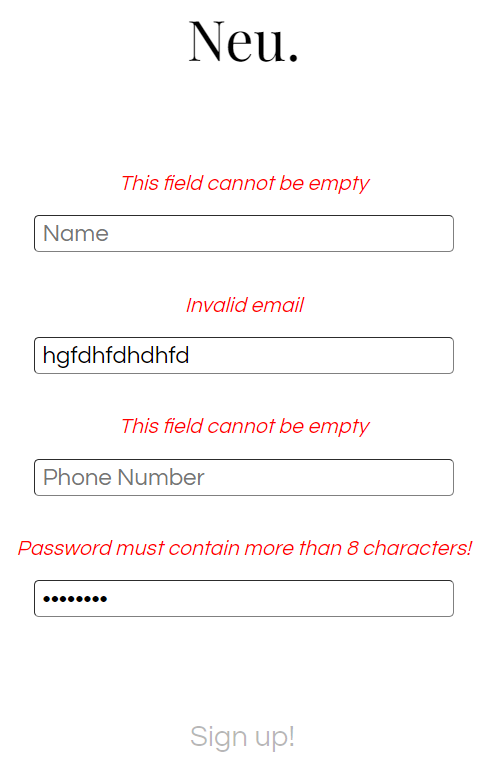


Рисунок 14 – Валидация вводимых данных формой

Прошедшая валидацию форма делает доступной для клика кнопку регистрации.



Рисунок 15 – Валидированная форма

Для авторизации в системе пользователю необходимо ввести адрес электронной почты и пароль на странице авторизации. Макет страницы авторизации представлен на рисунке 16.

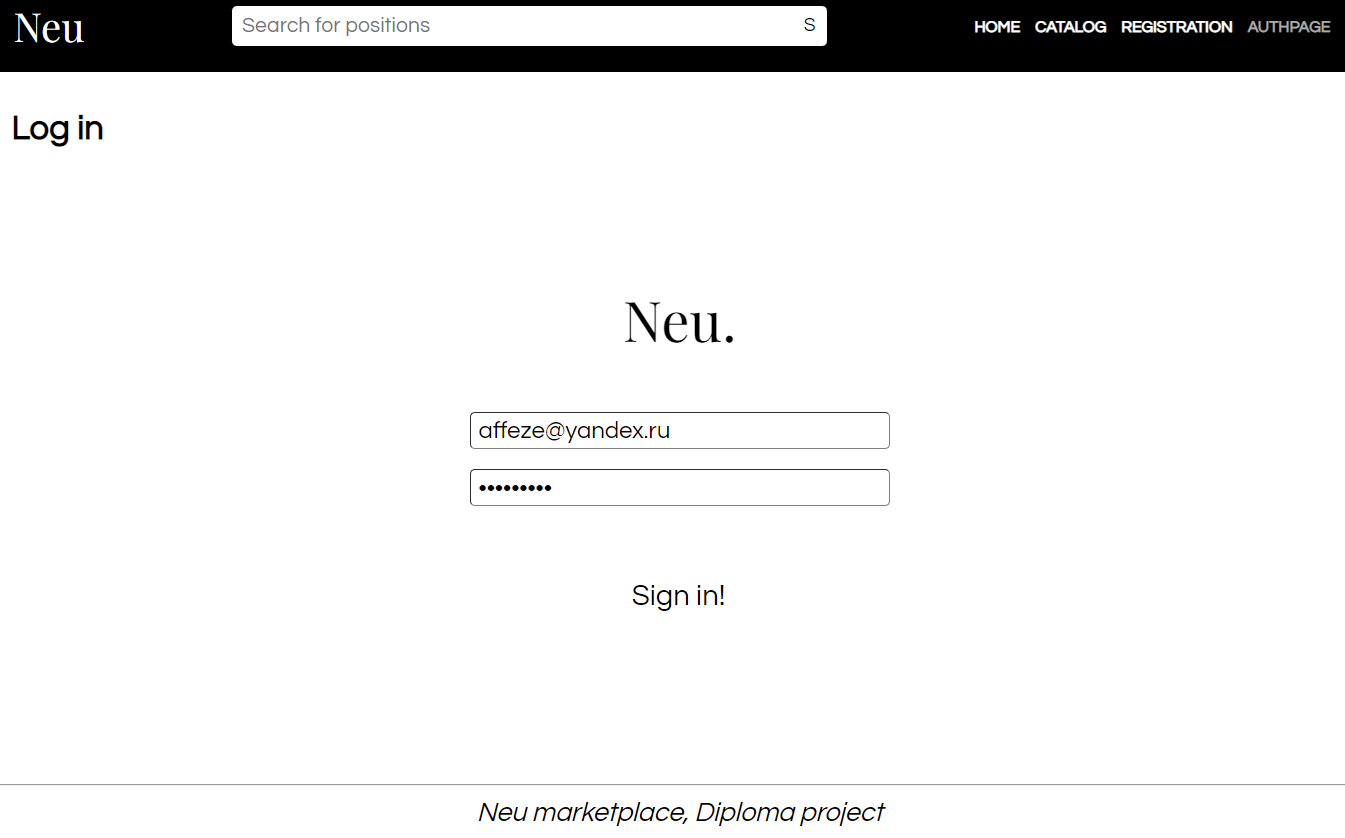


Рисунок 19 – Страница авторизации

## **Экономическая часть**

## **Область применения программного продукта и его преимущества перед аналогичным программным продуктом**

Разрабатываемый веб-сервис предназначен для предоставления возможности пользователями размещать объявления о продаже предметов одежды и аксессуаров, а также их поиска и связи с продавцами.

Достоинствами разрабатываемого продукта является понятный интерфейс, отображение всей контактной информации о продавцах, что подразумевает возможность быстрой связи с ними, быстрое размещение объявления о продаже на сайте, продуманная система поисковой оптимизации продукта и распространения в социальных сетях.

## **Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад**

Трудоёмкость разработки определяется в таблице 1. Строка «Всего» отображает общую трудоемкость разработки.

Таблица 1 – Трудоемкость разработки программного продукта

| **Наименование этапа** | **Условное обозначение** | **Трудоемкость выполнения этапа, час** |
| --- | --- | --- |
| Описание задания | То | 12 |
| Разработка UML - диаграмм | Тд | 15 |
| Проектирование программы | Тпп | 352 |
| Отладка программы | Топ | 15 |
| Оценка качества программы | Тоц | 20 |
| Оформление документации | Тд | 20 |
| Всего | Тобщ | 434 |

Разработчики программного продукта представлены в таблице 2

Таблица 2 – Разработчики программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| **Исполнители** | **Оклад, руб.** | **Часовая тарифная ставка, руб/час** | **Количество сотрудников** |
| Разработчик-программист | 50500 | 286,93 | 1 |
| Руководитель проекта | 60500 | 343,75 | 1 |

Часовая тарифная ставка, ЧТС, руб./час., определяется исходя из месячного оклада, количества рабочих дней в месяце и продолжительности рабочего процесса по формуле:

(1)

где Ом – оклад исполнителя в месяц, руб. /мес.;

Д – количество рабочих дней в месяце (для расчета Д = 22 раб. дня);

Тс – продолжительность рабочего дня (для расчета Тс = 8час.).

- ЧТС разработчик программист

- ЧТС разработчик программист

В таблице 3 представлена стоимость технических средств разработки ИСУ.

Таблица 3 – стоимость технических средств разработки ИСУ

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование компонента** | **Цена, руб.** | **Количество, шт.** | **Стоимость, руб.** |
| Ноутбук Asus X540M | 24990 | 1 | 24990 |
| МФУ лазерное HP LaserJet Pro 400 M428dw | 26499 | 1 | 26499 |
| Итого: | | | 51489 |

В таблице 4 предоставлены затраты на расходные материалы.

Таблица 4 – Планируемые затраты на расходные материалы

| **Затраты** | **Стоимость** | **Количество** | **Сумма, руб.** |
| --- | --- | --- | --- |
| Интернет | 800 руб./месяц | 2 месяца | 1600 |
| Электричество | 4,82 руб./КВт\*ч | 256 КВт\*ч | 1233,92 |
| Бумага | 250 руб. | 5 упаковок | 1250 |
| Ручка | 50 руб. | 5 штук | 250 |
| Заправка МФУ | 3 000 руб. | 2 раза | 6000 |
| Итого: | | | 10333,92 |

## **Расчет затрат на разработку**

Исходные данные, связанные с разработкой программного продукта представлены в таблице 5.

Таблица 5 – Исходные данные

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оклад разработчика | Ор | 111000 руб. |
| Время разработки | Трп | 2 месяца |
| Машинное время разработки | Тмч | 2 месяца |
| Коэффициент дополнительной заработной платы | Кд | 0,15 |
| Коэффициент страховых взносов | Кст | 0,3 |
| Количество единиц техники | Q | 1 шт. |
| Себестоимость содержания техники | См/ч | 15 руб./час |
| Коэффициент готовности техники | Кгт | 0,95 |
| Число рабочих дней в месяце | ЧРД | 22 дня |
| Продолжительность смены | Тсм | 8 часов |
| Коэффициент сменности | Ксм | 1 |
| Коэффициент транспортных расходов | Кт | 0,15 |
| Коэффициент накладных расходов | Кнр | 0,54 |

Произведя вычисления было выявлено, что полные затраты на разработку составляют 538141,00 рублей. Промежуточные результаты вычислений представлены в таблице 6.

Таблица 6 – Результаты вычислений

| **Наименование** | **Обозначение** | **Значение, руб.** |
| --- | --- | --- |
| Оплата труда | Зот | 255300 |
| Страховые взносы | Зст | 76590 |
| Содержание ЭВМ | Зэвм | 5016 |
| Затраты на специальные программы | Зсп | 51489 |
| Расходные материалы | Зрм | 11884 |
| Накладные расходы | Знр | 137862 |
| Итого затрат на разработку | Зрп | 538141,00 |

Расчет полных затрат на разработку проектного решения (программного продукта) осуществляется по формуле:

Зрп = Зот + Зст + Зэвм + Зсп + Зрм + Знр, руб. (2)

где Зот – затраты на оплату труда разработчика (разработчиков), руб.;

Зст – страховые взносы по оплате труда во внебюджетные фонды, руб.;

Зэвм – затраты, связанные с содержанием вычислительной техники, руб.;

Зсп – затраты на специальное программное обеспечение, руб.;

Зрм – затраты на расходные материалы, необходимые при разработке программного продукта, руб.;

Знр – затраты по накладным расходам, приходящиеся на разработку программного продукта, руб.

Зрп = 255300 + 76590 + 5016 + 51489 + 11884 + 137862 = 538141,00 руб.

* + 1. **Затраты на оплату труда разработчиков (Зот)**

Размер фонда оплаты труда разработчиков (Зот) рассчитывается по формуле:

Зот = Ор \* Трп (1+Кд), руб (3)

где Ор - месячный оклад разработчика проектного решения, руб./мес.;

Трп- время разработки проектного решения разработчиком, мес.(час.) включает в себя машинное время работы над проектом (Тмрп);

Кд–коэффициент дополнительной заработной платы разработчика.

Зот=111000 \* 2 \* (1 + 0,15) = 255300 руб.

Если время разработки программного продукта менее месяца, т. е. определено в часах, то в формулу подставляется часовая тарифная ставка, определяемая по формуле 4.

* + 1. **Затраты по страховым взносам (Зст)**

Сумма страховых взносов определяется по формуле:

Зст = Кст \* Зот, руб (4)

где Кст – коэффициент страховых взносов для расчета отчислений во внебюджетные фонды.

Зст = 0,3 \* 255300 = 76590 руб.

* + 1. **Затраты по содеражнию ЭВМ (Зэвм)**

Затраты, связанные с эксплуатацией и содержанием ЭВМ, определяются по формуле:

Зэвм = Тмрп \* Кгт \* Q\* Cм/ч, руб. (5)

где Тмрп - машинное время на разработку проектного решения, час;

Кгт - коэффициент готовности техники;

Q - количество условных единиц, используемой техники;

Cм/ч – стоимость машино-часа, эксплуатации оборудования, руб. в час.

Зэвм=352 \* 0,95 \* 1 \* 15 = 5016 руб.

Так как машинное время может измеряться в месяцах, а себестоимость машино-часа за один час, то машинное время необходимо перевести в часы.

Перевод рабочего времени в часы осуществляется по формуле:

Тмрп = Тмч \* Чрд \* Тсм \* Ксм, час (6)

где Тмч – рабочее время в месяцах;

Чрд – число рабочих дней в месяце;

Тсм – продолжительность рабочей смены;

Ксм – количество рабочих смен.

Тмрп = 2 \* 22 \* 8 \* 1 = 352 час.

* + 1. **Затраты на расходные материалы (Зрм)**

Затраты на расходные материалы необходимые для разработки проектного решения определяются по формуле:

Зрм = , руб (7)

где -количество материала, шт.;

цена вида материала, руб.;

n – количество видов материалов;

Кт- коэффициент транспортных расходов.

Зрм1 = 2 \* 800 \* (1 + 0,15) = 1840 руб.

Зрм2 = 256 \* 4,82 \* (1 + 0,15) = 1419,008 руб.

Зрм3 = 5 \* 250 \* (1 + 0,15) = 1437,5 руб.

Зрм4 = 5 \* 50 \* (1 + 0,15) = 287,5 руб.

Зрм5 = 2 \* 3000 \* (1 + 0,15) = 6900 руб.

Зрм = 1840 + 1419,008 + 1437,5 + 287,5 + 6900 = 11884 руб.

* + 1. **Затраты по накладным расходам (Знр)**

Затраты по накладным расходам определяются по формуле:

Знр = Кнр \* Зот, руб. (8)

где Кнр- коэффициент накладных расходов (или процент), принимается для расчета по данным предприятия;

Зот- затраты по оплате труда, руб.

Знр = 0,54 \* 255300 = 137862 руб.

## **Расчет цены и прибыли**

Исходные данные, связанные с разработкой программного продукта приведены в таблице 7.

Таблица 7 – Исходные данные

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Коэффициент рентабельности | Кр | 0,3 |
| Коэффициент налога на добавочную стоимость | Кндс | 0,2 |
| Ставка налога | Кнп | 0,2 |
| Стоимость затрат на разработку | Квпр | 538141,00 |

Произведя вычисления было выявлено, что чистая прибыль будет составлять 129133,84 рубля. Промежуточные результаты вычислений представлены в таблице 8.

Таблица 8 – Результаты вычислений

| **Наименование** | **Обозначение** | **Значение, руб.** |
| --- | --- | --- |
| Плановая прибыль | Ппл | 161442,3 |
| Налог на добавочную стоимость | НДС | 139916,66 |
| Цена программного продукта | Цпп | 839499,96 |
| Налог на прибыль | НП | 32288,46 |
| Чистая прибыль | ЧП | 129133,84 |
| Поступление в бюджет | ПБ | 172205,12 |

Цена программного продукта, который разработан одной организаций по заказу другой и не предназначен для тиражирования, определяется по формуле:

Цпп = Квпр + Ппл + НДС, руб. (9)

где Ппл - планируемая прибыль рассчитывается по формуле:

Ппл = Квпр × Кр, руб. исходя из нормативной рентабельности (коэффициента) Кр = 0,3;

Ппл = 538141 \* 0,3 = 161442,3 руб.

НДС- налог на добавленную стоимость определяется исходя из Кндс = 0.2 (ставка налога 20%) по формуле:

НДС = (Квпр + Ппл) × Кндс, руб. (10)

НДС = (538141 + 161442,3) \* 0,2 = 139916,66

Цпп = 538141 + 161442,3 + 139916,66 = 839499,96 руб.

Каждое предприятие с полученной прибыли перечисляет государству налог на прибыль. На сегодня ставка налога 20% (Кнп = 0.2) от полученной прибыли, и определяется по формуле:

НП = Ппл × Кнп, руб. (11)

НП = 161442,3 \* 0,2 = 32288,46 руб.

Чистая прибыль, остающаяся в распоряжении предприятия

ЧП = Ппл – НП, руб. (12)

ЧП = 161422,3 – 32288,46 = 129133,84 руб.

Поступление в бюджет складываются из налога на прибыль и НДС

ПБ = НП + НДС, руб. (13)

ПБ=32288,46 + 139916,66 = 172205,12 руб.

## **Вывод по разделу**

Исходя из расчетов экономической части были получены следующие данные. Трудоемкость разработки составила 434 часа. Часовая тарифная ставка для разработчика-программиста составила 286,93 рубля, для руководителя проекта – 343,75. Затраты на разработку составили 538141,00 рубль. Цена программного обеспечения составила 839499,96 рублей, и чистая прибыль равна 129133,84 рубля.

## **Техника безопасности и охрана труда**

## **Анализ условий труда программиста**

Рабочее место программиста — это его рабочий кабинет, либо часть помещения, в котором он проводит большую часть работы. Совершенно очевидно, что чем лучше приспособлено рабочее место для выполнения функций программиста, тем производительнее и эффективнее будет его трудовая деятельность.

Программист имеет собственное рабочее место, оборудованное ПЭВМ. Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также – расстоянию спинки до переднего края сиденья.

Рабочая поверхность стола сделана из дерева. На столе размещены ЭВМ, монитор и принтер. Площадь поверхности стола предполагает размещение монитора, устройств ввода/вывода и рабочей зоны с местом расположения оперативной документации.

Кабинет, в котором расположено рабочее место, представляет собой помещение площадью 15,28 м2, длиной 4,48 м, шириной 3,41 м и высотой 2,4 м.

Для создания благоприятных условий для зрительного восприятия кабинет оформлен в светло-бежевом цвете.

Для создания и поддержания независимо от наружных условий оптимальных значений температуры, влажности, чистоты воздуха, в холодное время используется водяное отопление, в теплое время года применяется кондиционирование воздуха.

Режим работы программиста - односменный с пятидневной рабочей неделей и нормированным рабочим днем. Продолжительность ежегодного отпуска - 28 календарных дней.

При идентификации вредных производственных факторов было установлено, что опасными могут являться:

- шум и вибрация, источником которого является оргтехника;

- электромагнитные поля и излучения от дисплея ПК;

- статическое электричество, накапливаемое на клавиатуре, дисплее, корпусе системного блока;

- электрический ток в электрических сетях;

- запыленность рабочей зоны, так влажная уборка рабочей поверхности проводиться не ежедневно, а только по мере загрязнения;

- статические перегрузки, так как работа программиста связана с неизменной статической позой, иногда неудобной позе (работа за ПК), что может привести к искривлению позвоночника, остеохондрозу, застою в органах малого таза;

- перенапряжение анализаторов, в большей степени зрительного, что может привести к различным нарушениям зрения (снижение остроты, спазм и уменьшение запаса аккомодации), помимо этого повышение зрительной нагрузки может приводить к общему утомлению, возникновению головной боли, ухудшению самочувствия.

- умственное перенапряжение, так как в процессе деятельности программист анализирует и перерабатывает огромное количество административных сетей, разрабатывает программные продукты, поддерживает технику в рабочем состоянии.

В целях выявления вредных и (или) опасных производственных факторов и осуществления мероприятий по приведению условий труда в соответствие государственными нормативными требованиями охраны труда ежегодно проводится аттестация рабочих мест по условиям труда - оценка условий труда на рабочих местах. Аттестации подлежат все имеющиеся в организации рабочие места.

При аттестации рабочих мест проводят оценку условий труда, гигиеническую оценку и оценку травмоопасности рабочих мест. При этом учитывают наличие средств коллективной защиты, обеспеченность работников средствами индивидуальной защиты и определяют эффективность этих средств.

Согласно санитарным правилам СП 2.2.3670-20 "Санитарно-эпидемиологические требования к условиям труда", дисплей должен располагаться на расстоянии 60-70 см, но не менее 50 см от глаз. Между боковыми поверхностями мониторов должно быть не менее 1,2 м. При использовании жидкокристаллических дисплеев на каждый компьютер должно приходиться не менее 4,5 м2 площади. На дисплей ПК не должен попадать прямой солнечный свет во избежание бликов и повышенной нагрузки на зрение.

Поэтому в кабинете, где работает программист, стоит только один компьютер, а комплектующая оргтехника расположена достаточно далеко непосредственно от самого специалиста. Это обусловлено тем, что все копировальные аппараты, принтеры и факс в периоды своей работы выделяют не только вредные токсичные газы, но и шумы и вибрацию. Данные факторы оказывают гораздо меньшее негативное влияние, нежели излучение от компьютера, но также являются вредными и могут нанести вред здоровью сотрудника.

Для снижения напряжения с глаз, а также для снятия мышечного напряжения во время рабочего дня проводится проветривание в течение 10–15 минут каждые 1,5 часа.

## **Расчет искусственного освещения рабочего места**

Через зрительный анализатор человек получает около 80% из общего объема информации. Качество поступающей информации во многом зависит от освещения: неудовлетворительное количественно или качественно, оно не только утомляет зрение, но и вызывает утомление организма в целом. Кроме того, нерациональное освещение может явиться причиной травматизма: плохо освещенные опасные зоны, слепящие источники света и блики от них, резкие тени ухудшают видимость настолько, что вызывают полную потерю ориентировки работающих, снижают производительность труда и увеличивают брак продукции. Поэтому необходимо достаточное искусственное освещение для рабочего места.

Условия работы для расчета показателя освещенности рабочего места программиста являются:

* Помещение размером 38,97 квадратных метров;
* Высота помещения 2,44 метра, длина 7,48 метра, ширина 5,21 метра;
* Лампы люминесцентные (дневного света) в количестве 6 штук, новые;
* Крупногабаритная мебель отсутствует.

Расчет общего равномерного искусственного освещения горизонтальной рабочей поверхности выполняется методом коэффициента использования светового потока. Световой поток (лм) одной лампы рассчитывается по следующей формуле:

 (14)

где Ен - нормируемая минимальная допустимая освещенность по СП 1.3330.2016, для помещения 2000 лк;

*S -* площадь освещаемого помещения, м2;

z - коэффициент неравномерности освещения, *z* = 1,1;

K - коэффициент запаса, учитывающий снижение освещенности в процессе эксплуатации вследствие загрязнения и старения ламп и светильников, а также снижения отражающих свойств поверхностей помещения для кабинета будет равен 1,3;

Nc - число светильников в помещении;

*-* коэффициент затенения, обычно 1;

=i - коэффициент использования светового потока.

Коэффициент использования светового потока, давший название методу расчета, определяется по индексу помещения i в зависимости от типа светильника и коэффициентов отражения света от потолка, стен и пола:

 (15)

где А, В - длина и ширина помещения в плане, м;

Нс - высота подвеса светильников над рабочей поверхностью, м.

Согласно СП 15.13330.2016 величина полученного светового потока достаточна для данного помещения.

## **Электробезопасность на предприятии**

При поступлении на работу сотрудник должен пройти инструктаж по технике безопасности и электробезопасности. Работника знакомят с основными правилами по технике безопасности, предлагают внимательно прочитать действующие на предприятии инструкции, поясняя при этом отдельные правила и требования.

Инструктаж по технике безопасности при выполнении конкретной работы проводит руководитель соответствующего производственного участка, показывая инструктируемому правильные безопасные приемы работы.

Повторный инструктаж проводится для рабочих независимо от их квалификации, стажа и опыта работы, не реже 1 раза в 6 месяцев по программе инструктажа на рабочем месте.

При нарушении работающими правил и инструкций по технике безопасности, технологической и производственной дисциплины, а также в случаях изменения технологического процесса или вида работы проводят дополнительные инструктажи.

Все инструктажи оформляются записями в специальном журнале с указанием номеров или шифров инструкций. Журнал о проведении инструктажа хранится у руководителя подразделения.

Работник на рабочем месте должен соблюдать общие меры электробезопасности. При этом запрещается:

* прикасаться к задней панели системного блока при включенном питании;
* допускать попадание влаги на поверхность системного блока, монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств;
* производить самостоятельно вскрытие и ремонт оборудования;
* пользоваться неисправными электроприборами и электропроводкой;
* ремонтировать электроприборы самостоятельно;
* подвешивать электропровода на гвоздях, металлических и деревянных предметах, перекручивать провод, закладывать провод и шнуры на водопроводные трубы и батареи отопления, вешать что-либо на провода, вытягивать за шнур вилку из розетки;
* прикасаться одновременно к персональному компьютеру и к устройствам, имеющим соединение с землей (радиаторы отопления, водопроводные краны, трубы и т.п.);
* пользоваться самодельными электронагревательными приборами и электроприборами с открытой спиралью;
* наступать на переносимые электрические провода, лежащие на полу.

При работе с электроприборами и оргтехникой (персональные компьютеры, принтеры, сканеры, копировальные аппараты, факсы, бытовые электроприборы, приборы освещения):

* автоматические выключатели и электрические предохранители должны быть всегда исправны;
* изоляция электропроводки, электроприборов, выключателей, штепсельных розеток, ламповых патронов и светильников, а также шнуров, с помощью которых включаются в электросеть электроприборы, должны быть в исправном состоянии;
* электроприборы необходимо хранить в сухом месте, избегать резких колебаний температуры, вибрации, сотрясений;
* для подогрева воды пользоваться сертифицированными электроприборами с закрытой спиралью и устройством автоматического отключения, с применением несгораемых подставок.

## **Пожарная безопасность на предприятии**

Инструктаж по пожарной безопасности проводится по программе, разработанной инженером по охране труда ООО «Омега», с учетом требований стандартов, правил, норм и инструкций о мерах пожарной безопасности. Продолжительность инструктажа устанавливается в соответствии с утвержденной программой. Инструктаж по пожарной безопасности, как правило, проводится совместно с инструктажем по технике безопасности и в те же сроки.

Инструктаж по пожарной безопасности проходят все работники организации, независимо от их образования, стажа работы по данной профессии или должности, временные работники, командированные, обучающиеся и студенты, прибывшие на практику.

О проведении инструктажа по пожарной безопасности работник, проводивший инструктаж, делает запись в журнале учета проведения инструктажей по пожарной безопасности, с обязательной подписью инструктируемого и инструктирующего. При регистрации внепланового инструктажа по пожарной безопасности указывают причину его проведения.

Все работники организаций должны допускаться к работе только после прохождения противопожарного инструктажа, а при изменении специфики работы проходить дополнительное обучение предупреждению и тушению возможных пожаров в порядке, установленном руководителем.

На проведение первичного противопожарного инструктажа необходимо отводить не менее 1 ч. Инструктируемые должны ознакомиться:

* с действующими на объекте правилами пожарной безопасности и инструкциями;
* с производственными участками, наиболее опасными в пожарном отношении, где запрещается курить, применять открытый огонь;
* с возможными причинами возникновения пожара и мерами его предупреждения;
* с практическими действиями в случае возникновения пожара - вызов пожарной помощи, использование первичных средств пожаротушения, место расположения ближайшего телефона и ознакомление с правилами поведения в случае возникновения пожара, эвакуации людей и материальных ценностей.

При первичном инструктаже инструктирующий обязан рассказать о производственных установках с повышенной пожарной опасностью, мерах предотвращения пожаров и загораний, указать место курения, ознакомить вновь поступившего с имеющимися на объекте средствами пожаротушения, показать ближайший телефон и объяснить правила поведения в случае возникновения пожара.

Проведение противопожарного инструктажа в обязательном порядке должно сопровождаться практическим показом способов использования имеющихся на объекте средств пожаротушения (противогазы, респираторы, огнетушители и так далее).

Соблюдение рассмотренных в данном разделе мероприятий по охране труда и технике безопасности на предприятии позволяет снизить утомляемость и травматизм, повысить производительность труда, обеспечить комфортные условия трудовой деятельности специалиста, работающего в должности программиста.

## **ЗАКЛЮЧЕНИЕ**

В рамках выпускной квалификационной работы были разработаны функциональные модули для веб-маркетплейса. Разработанные модули налаживают взаимодействия компонентов веб-приложения, спроектированного на Server-Side Rendering архитектуре. Благодаря взаимной работе менеджеров управления маршрутизацией и состояния, пользователю доступен полный функционал разработанного веб-приложения.

Тем не менее, работа над проектом полностью не завершена, что говорит о том, что разворачивать приложение на хостинге не имеет смысла. Для завершения работы над продуктом до сих пор необходимо:

1. спроектировать базу данных, полностью удовлетворяющую предусмотренному функционалу приложения;
2. привязать приложение к реальному API, чтобы развернуть полноценное взаимодействие приложения с базой данных;
3. добавить возможности фильтрации данных на страницах;
4. реализовать чат для поддержания контакта между потенциальными покупателем и продавцом соответственно.

Перед внедрением продукта в Интернет необходимо провести различные тестирования продукта на предмет таких проблем, как XSS-уязвимости и тому подобное.

Выпускная квалификационная работа удовлетворяет требованиям, поставленным на начальном этапе разработки программного продукта, веб-приложение успешно функционирует.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Сухов, К. К. HTML 5. Путеводитель по технологии HTML5 / К. К. Сухов, - М. : ДМК Пресс, 2012. – 312 с.
2. Панфилов, К. С. По ту сторону веб-страницы / К. С. Панфилов, - М. : ДМК Пресс, 2012. – 440 с.
3. Крокфорд, Д. Как устроен JavaScript / Д. Крокфорд, - СПб. : Питер, 2019. – 304 с.
4. Прасти, Н. Введение в ECMAScript6 / Н. Прасти, - М. : ДМК Пресс, 2015. – 176 с.
5. Браун, Э. Java Script. Изучаем JavaScript. Руководство по созданию современных веб-сайтов / Э. Браун, - М. : Символ-Плюс, 2017. – 368 с.
6. React или Angular или Vue.js – что выбрать? [Электронный ресурс] / URL: https://habr.com/ru/post/476312/ (дата обращения: 18.04.2022)
7. Immer: новый подход к иммутабельности в JavaScript [Электронный ресурс] / URL: https://habr.com/ru/company/ruvds/blog/349492/ (дата обращения: 06.05.2022)
8. Горячая перезагрузка компонентов в React [Электронный ресурс] / URL: https://habr.com/ru/post/346142/ (дата обращения: 06.05.2022)
9. A comparison of Marketplaces for Online Sellers [Элекронный ресурс] / URL: https://mindstorm.in/blog/a-comparison-of-marketplaces-for-online-sellers (дата обращения: 01.06.2022)
10. Getting Started with Redux [Электронный ресурс] / URL: https://redux.js.org/introduction/getting-started (дата обращения: 15.05.2022)
11. Redux-Saga Getting Started [Электронный ресурс] / URL: https://redux-saga.js.org/docs/introduction/GettingStarted/ (дата обращения: 17.05.2022)
12. Bem React [Электронный ресурс] / URL: https://ru.bem.info/technologies/bem-react/ (дата обращения: 10.05.2022)
13. TypeScript Documentation [Электронный ресурс] / URL: https://www.typescriptlang.org/docs/ (дата обращения: 20.04.2022)
14. Документация по React Hooks [Электронный ресурс] / URL: https://reactjs.org/docs/hooks-intro.html (дата обращения: 21.04.2022)
15. Документация по React Context [Электронный ресурс] / URL: https://reactjs.org/docs/context.html (дата обращения: 21.04.2022)
16. Документация по React Code Splitting [Электронный ресурс] / URL: https://ru.reactjs.org/docs/context.html (дата обращения: 21.05.2022)
17. Scaling your Redux app with ducks [Электронный ресурс] / URL: https://www.freecodecamp.org/news/scaling-your-redux-app-with-ducks-6115955638be (дата обращения: 25.04.2022)
18. Что такое XSS-уязвимость и как тестировщику не пропустить её [Электронный ресурс] / URL: https://habr.com/ru/post/511318/ (дата обращения: 01.06.2022)
19. The Meaning of SEO in Marketing Your Business [Электронный ресурс] / URL: https://www.constantcontact.com/blog/increase-website-traffic-website-seo/ (дата обращения: 20.05.2022)
20. Client-side vs. Server-side vs Pre-rendering for Web Apps [Электронный ресурс] / URL: https://www.toptal.com/front-end/client-side-vs-server-side-pre-rendering (дата обращения 25.04.2022)
21. Евкова А. Применение объектно-ориентированного подхода при проектировании информационной системы [Электронный ресурс] / URL: https://www.evkova.org/kursovye-raboty/primenenie-obektno-orientirovannogo-podhoda-pri-proektirovanii-informatsionnoj-sistemyi (дата обращения: 01.06.2022)
22. История развития интернет-маркетинга [Электронный ресурс] / URL: https://neuron.group/blog/istoriya-razvitiya-internet-marketinga (дата обращения: 03.06.2022)

## Приложение А

(справочное)

**Системные требования к разрабатываемому продукту**

Минимальные системные требования, необходимые для исполнения приложения представлены в таблице A.1.

Таблица А.1. – Минимальные системные требования

| **Наименование требования** | **Значение требования** |
| --- | --- |
| Версия ОС | Локализированная x86 (32-bit) или x64 (64-bit) Windows 7 и выше |
| Процессор | 1,6 ГГц и выше |
| ОЗУ | 1 ГБ ОЗУ (1,5 ГБ для виртуальной машины |
| Место на жёстком диске | Не менее 20 МБ свободного дискового пространства |
| Видеоадаптер | DirectX 9 или более поздняя версия с драйвером WDDM 1.0 |
| Экран | 1024 х 768 точек и выше |

## Приложение Б

(справочное)

**Исходный код приложения**

Б.1. client.tsx – клиентский энд-поинт;

import \* as React from 'react';

import 'babel-polyfill';

import { hydrate } from 'react-dom';

import { ConnectedRouter } from 'connected-react-router';

import { Provider as ReduxProvider } from 'react-redux';

import { loadableReady } from '@loadable/component';

import { App } from 'components';

import { State } from 'types';

import { configureStore } from './store/rootStore';

const { store, history } = configureStore(window.\_\_INITIAL\_STATE\_\_);

// global redeclared types

declare global {

    interface Window {

        \_\_INITIAL\_STATE\_\_: State;

        \_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_: Function;

    }

}

loadableReady(() => {

    hydrate(

        <ReduxProvider store={store}>

            <ConnectedRouter history={history}>

                <App />

            </ConnectedRouter>

        </ReduxProvider>,

        document.getElementById('mount')

    );

});

Б.2. routes.ts – коллекция маршрутов;

import loadable from '@loadable/component';

import { fetchCatalog } from 'store/ducks/catalog/actions';

import { fetchHomepage } from 'store/ducks/homepage/actions';

import { fetchShoes } from 'store/ducks/shoes/actions';

import { RouterFetchDataArgs } from 'types';

const CatalogPage = loadable(() => import('./pages/Catalog/Catalog'));

const RegistrationPage = loadable(() => import('./pages/Registration/Registration'));

const SneakersPage = loadable(() => import('./pages/Sneakers/Sneakers'));

const HomePage = loadable(() => import('./pages/Home/Home'));

const NotFoundPage = loadable(() => import('./pages/404/404'));

const AuthPage = loadable(() => import('./pages/Auth/AuthPage'))

export default [

    {

        path: '/',

        component: HomePage,

        exact: true,

        fetchData({ dispatch }: RouterFetchDataArgs) {

            dispatch(fetchHomepage());

        },

    },

    {

        path: '/catalog',

        component: CatalogPage,

        exact: true,

        fetchData({ dispatch }: RouterFetchDataArgs) {

            dispatch(fetchCatalog());

        },

    },

    {

        path: '/sneakers/:slug',

        component: SneakersPage,

        exact: true,

        fetchData({ dispatch, match }: RouterFetchDataArgs) {

            dispatch(fetchShoes(match.params.slug));

            dispatch(fetchHomepage());

        },

    },

    {

        path: '/registration',

        component: RegistrationPage,

        exact: true,

    },

    {

        path: '/auth',

        component: AuthPage,

        exact: true,

    },

    {

        path: '\*',

        component: NotFoundPage,

        exact: true,

    },

];

Б.3. App.tsx – компонент-layout;

import \* as React from 'react';

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

import { hot } from 'react-hot-loader/root';

import { Header, Footer } from 'components';

import routes from 'routes';

import './App.css';

function App() {

    return (

        <div className="app">

            <Header />

            <Switch>

                {routes.map(({ fetchData, ...routeProps }) => (

                    <Route key={routeProps.path} {...routeProps} />

                ))}

            </Switch>

            <Footer />

        </div>

    );

}

const Component = hot(App);

export { Component as App };

Б.4. Header.tsx – шапка сайта;

import \* as React from 'react';

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

import loadable from '@loadable/component';

import \* as bem from 'b\_';

import './Header.css';

enum PageName {

    Home = 'Home',

    Catalog = 'Catalog',

    Registration = 'Registration',

    AuthPage = 'AuthPage'

}

const menu = [

    { to: '/', exact: true, page: PageName.Home },

    { to: '/catalog', exact: true, page: PageName.Catalog },

    { to: '/registration', exact: true, page: PageName.Registration },

    { to: '/auth', exact: true, page: PageName.AuthPage },

];

const preloadPage = (pageName: string) =>

    loadable(() => import(`../../pages/${pageName}/${pageName}`));

const b = bem.with('header');

export function Header() {

    return (

        <div className={b()}>

            <div className={b('logo')}><a>Neu</a></div>

            <form className="headerBlockSearchBar-form">

                <input type="search" placeholder="Search for positions" className="headerBlockSearchBar-search" />

                <input type="submit" value="S" className="headerBlockSearchBar-submit" />

            </form>

            <nav className={b('nav')}>

                {menu.map(data => (

                    <NavLink key={data.to} exact={data.exact} to={data.to}

                        activeClassName="header\_\_nav-item\_active" className={b('nav-item')}

                        onMouseMove={() => preloadPage(data.page).preload()}>

                        {data.page}

                    </NavLink>

                ))}

            </nav>

        </div>

    );

}

Б.5. Footer.tsx – подвал сайта;

import \* as React from 'react';

import \* as bem from 'b\_';

import './Footer.css';

const b = bem.with('footer');

export function Footer() {

    return (

        <div className={b()}>

            <hr/>

            Neu marketplace, Diploma project

        </div>

    );

}

Б.6. – Page.tsx - внешняя обёртка для основного контента страницы;

import \* as React from 'react';

import \* as bem from 'b\_';

import './Page.css';

type Props = {

    children: React.ReactNode[] | React.ReactNode;

    align: string;

    mix?: string;

};

const b = bem.with('page');

function Page(props: Props) {

    const { align, children, mix } = props;

    const cls = `${b({ align })} ${mix}`;

    return <div className={cls}>{children}</div>;

}

Page.defaultProps = {

    align: 'center',

    mix: '',

};

export { Page };

Б.7 – PageContainer.tsx – контейнер для контента веб-страницы;

import \* as React from 'react';

import \* as bem from 'b\_';

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

import { Button, Page } from '..';

type Props = {

    children: React.ReactNode[] | React.ReactNode;

    btn?: {

        to: string;

        text: string;

    };

    mix?: string;

};

const b = bem.with('page');

function PageContainer(props: Props) {

    const { children, btn, mix } = props;

    const cls = `${b('container')} ${mix}`;

    return (

        <div className={cls}>

            {children}

            {btn && (

                <div className={b('container-more')}>

                    <Link to={btn.to}>

                        <Button>{btn.text}</Button>

                    </Link>

                </div>

            )}

        </div>

    );

}

PageContainer.defaultProps = {

    mix: '',

};

export { PageContainer };

Б.8. – PageMeta.tsx – мета-теги страницы;

import \* as React from 'react';

import Helmet from 'react-helmet';

type Props = {

    title?: string;

    description?: string;

    image?: string;

};

const cutTags = (text: string = '') => {

    return text.replace(/<\/?.+?>/gi, '');

};

const prepareData = ({ title, description, image }: Props) => {

    return {

        title: cutTags(title),

        description: cutTags(description).substr(0, 250),

        image,

    };

};

function PageMeta(props: Props) {

    const { title, description, image } = prepareData(props);

    return (

        <Helmet>

            <title>{title}</title>

            <meta property="og:title" content={title} />

            <meta property="twitter:title" content={title} />

            {Boolean(description) && (

                <meta name="description" content={description} />

            )}

            {Boolean(description) && (

                <meta property="og:description" content={description} />

            )}

            {Boolean(description) && (

                <meta property="twitter:description" content={description} />

            )}

            {Boolean(image) && <meta property="og:image" content={image} />}

        </Helmet>

    );

}

PageMeta.defaultProps = {

    title: 'Site',

    description: null,

    image: null,

};

export { PageMeta };

Б.9. – Sneakers.tsx – обёртка для карточки товара;

import \* as React from 'react';

import \* as bem from 'b\_';

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

import './Sneakers.css';

const b = bem.with('sneakers');

type Props = {

    title: string;

    image: string;

    subtitle: string;

    price: string;

    url: string;

};

export function Sneakers(props: Props) {

    const { title, image, subtitle, price, url } = props;

    return (

        <div className={b()}>

            <Link to={url}>

                <div className={b('image')} style={{ backgroundImage: `url(${image})` }}/>

            </Link>

            <div className={b('title')}>

                <Link to={url}>{title}</Link>

            </div>

            <div className={b('category')}>

                {subtitle},&nbsp;<strong>{price}</strong>

            </div>

        </div>

    );

}

Б.10. – SneakersList.tsx – контейнер карточек товара;

import \* as React from 'react';

import \* as bem from 'b\_';

import { Sneakers } from '../Sneakers/Sneakers';

import { Sneakers as SneakersType } from '../../types';

import './SneakersList.css';

const b = bem.with('sneakers-list');

type Props = {

    items: SneakersType[];

};

export function SneakersList(props: Props) {

    const { items } = props;

    return (

        <div className={b()}>

            {items.map(sneakers => (

                <Sneakers key={sneakers.slug} {...sneakers} />

            ))}

        </div>

    );

}

Б.11. – Sneakers.stub.tsx – заглушка компонента;

import \* as React from 'react';

import { Rect } from 'components';

import './Sneakers.css';

export function SneakersStub() {

    return (

        <div className="sneakers-stub">

            <Rect height="210px" width="220px" />

            <Rect type="black" height="20px" width="220px" />

            <Rect height="12px" width="220px" />

        </div>

    );

}

Б.12. – Rect.tsx – прямоугольник, используемый для заглушки на стадии First Contentful paint;

import \* as React from 'react';

import \* as bem from 'b\_';

import './Rect.css';

type Props = {

    type?: string;

    width?: string;

    height?: string;

    className?: string;

};

const b = bem.with('rect');

function Rect({ width, height, type, className }: Props) {

    const classes = className ? `${b({ type })} ${className}` : b({ type });

    return <div style={{ width, height }} className={classes} />;

}

Rect.defaultProps = {

    type: 'default',

    width: null,

    height: null,

    className: null,

};

export { Rect };

Б.13. – Button.tsx – макет кнопки;

import \* as React from 'react';

import \* as bem from 'b\_';

import './Button.css';

export enum ButtonSizes {

    L = 'l',

    M = 'm',

    S = 's',

}

type Props = {

    size: ButtonSizes;

    type: 'submit' | 'button' | 'reset';

    children: React.ReactNode;

};

const b = bem.with('button');

const Button = (props: Props) => {

    const { size, type, children } = props;

    return (

        <button type={type} className={b({ size })}>

            {children}

        </button>

    );

};

Button.defaultProps = {

    size: ButtonSizes.M,

    type: 'button',

    children: null,

};

export { Button };

Б.14. – 404.tsx – страница «Page not found»;

import \* as React from 'react';

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

import { Page, PageContainer } from 'components';

type Props = {

    code: number;

    children: React.ReactNode;

};

// Component is used for passing http status for server side rendering;

// For example, if page isn't found, client give page with 404 status code

const Status = ({ code, children }: Props) => {

    const render = ({ staticContext }: RouteComponentProps) => {

        if (staticContext) {

            staticContext.statusCode = code;

        }

        return children;

    };

    return <Route render={render} />;

};

export default function NotFoundPage() {

    return (

        <Status code={404}>

            <Page>

                <PageContainer>

                    <h1>Page not found</h1>

                </PageContainer>

            </Page>

        </Status>

    );

}

Б.15. – Auth.tsx – страница авторизации;

import \* as React from 'react';

import { Page, PageContainer, PageMeta } from 'components';

import './AuthPage.css';

export default function AuthPage() {

    const [email, setEmail] = React.useState(null);

    const [password, setPassword] = React.useState(null);

    const authorization = () => {

        try {

            let data = [{ email }, { password }]

            alert(JSON.stringify(data));

        } catch (err) {

            throw err;

        }

    }

    return (

        <Page>

            <PageMeta title="Log in" description="" />

            <PageContainer>

                <h2>Log in</h2>

                <div className="auth-form">

                    <p>Neu.</p>

                    <form>

                        <div className="auth-form-info">

                            <input onChange={(e: { target: { value: any; }; }) => { setEmail(e.target.value) }}

                                name="email"

                                type="text"

                                placeholder="Email" />

                            <input onChange={(e: { target: { value: any; }; }) => { setPassword(e.target.value) }}

                                name="password"

                                type="password"

                                placeholder="Password" />

                            <input onClick={e => authorization()}

                                type="submit" id="signup-submit" value="Sign in!" />

                        </div>

                    </form>

                </div>

            </PageContainer>

        </Page>

    );

}

Б.16. – Registration.tsx – страница регистрации;

import \* as React from 'react';

import { Page, PageContainer, PageMeta } from 'components';

import './Registration.css';

export default function Registration() {

    const [name, setName] = React.useState(null);

    const [nameDirty, setNameDirty] = React.useState(false);

    const [nameError, setNameError] = React.useState("This field cannot be empty");

    const [email, setEmail] = React.useState(null);

    const [emailDirty, setEmailDirty] = React.useState(false);

    const [emailError, setEmailError] = React.useState("This field cannot be empty");

    const [phoneNumber, setPhoneNumber] = React.useState(null);

    const [phoneNumberDirty, setPhoneNumberDirty] = React.useState(false);

    const [phoneNumberError, setPhoneNumberError] = React.useState("This field cannot be empty");

    const [password, setPassword] = React.useState(null);

    const [passwordDirty, setPasswordDirty] = React.useState(false);

    const [passwordError, setPasswordError] = React.useState("This field cannot be empty");

    const [formValid, setFormValid] = React.useState(false);

    React.useEffect(() => {

        if (nameError || emailError || phoneNumberError|| passwordError) {

            setFormValid(false);

        } else {

            setFormValid(true);

        }

    }, [nameError, emailError, phoneNumberError, passwordError])

    const blurHandler = (e: React.FocusEvent<HTMLInputElement>) => {

        switch (e.target.name) {

            case 'name':

                setNameDirty(true);

                break;

            case 'email':

                setEmailDirty(true);

                break;

            case 'phoneNumber':

                setPhoneNumberDirty(true);

                break;

            case 'password':

                setPasswordDirty(true);

                break;

        }

    }

    const nameHandler = (e: React.ChangeEvent<HTMLInputElement>) => {

        setName(e.target.value)

        if (!e.target.value) {

            setNameError("Name field cannot be empty")

        } else {

            setNameError("")

        }

    }

    const emailHandler = (e: React.ChangeEvent<HTMLInputElement>) => {

        setEmail(e.target.value);

        const re = /^(([^<>()[\]\.,;:\s@\"]+(\.[^<>()[\]\.,;:\s@\"]+)\*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i

        if (!re.test(String(e.target.value).toLowerCase())) {

            setEmailError('Invalid email')

        } else {

            setEmailError("")

        }

    }

    const phoneNumberHandler = (e: React.ChangeEvent<HTMLInputElement>) => {

        setPhoneNumber(e.target.value)

        if (!e.target.value) {

            setPhoneNumberError("Name field cannot be empty")

        } else {

            setPhoneNumberError("")

        }

    }

    const passwordHandler = (e: React.ChangeEvent<HTMLInputElement>) => {

        setPassword(e.target.value)

        if (e.target.value.length <= 8) {

            setPasswordError("Password must contain more than 8 characters!")

            if (!e.target.value) {

                setPasswordError("Password field cannot be empty")

            }

        } else {

            setPasswordError("")

        }

    }

    const registration =  () => {

        let data = [

            {name}, {email}, {phoneNumber}, {password}

        ]

        alert(JSON.stringify(data));

    }

    return (

        <Page>

            <PageMeta title="Registration" description="" />

            <PageContainer>

                <h2>Registration page</h2>

                <div className="reg-page-main">

                    <p>Neu.</p>

                    <form>

                        <div className="reg-form-info">

                            <div className="reg-form-left">

                                {(nameDirty && nameError) &&

                                    <div className="reg-error"><p>{nameError}</p></div>}

                                <input onChange={e => nameHandler(e)}

                                    onBlur={e => blurHandler(e)} value={name}

                                    name="name" type="text" placeholder="Name" />

                                {(emailDirty && emailError) &&

                                    <div className="reg-error"><p>{emailError}</p></div>}

                                <input onChange={e => emailHandler(e)}

                                    onBlur={e => blurHandler(e)} value={email}

                                    name="email" type="text" placeholder="Email" />

                                {(phoneNumberDirty && phoneNumberError) &&

                                    <div className="reg-error"><p>{phoneNumberError}</p></div>}

                                <input onChange={e => phoneNumberHandler(e)}

                                    onBlur={e => blurHandler(e)} value={phoneNumber}

                                    name="phoneNumber" type="text" placeholder="Phone Number" />

                                {(passwordDirty && passwordError) &&

                                    <div className="reg-error"><p>{passwordError}</p></div>}

                                <input onChange={e => passwordHandler(e)}

                                    onBlur={e => blurHandler(e)} value={password}

                                    name="password" type="password" placeholder="Password" />

                            </div>

                            <div className="reg-form-right">

                                <input type="file" />

                            </div>

                        </div>

                        <div className="reg-form-submit">

                            <input disabled={!formValid} onClick={e => registration()}

                                type="submit" id="signup-submit" value="Sign up!" />

                        </div>

                    </form>

                </div>

            </PageContainer>

        </Page>

    );

}

Б.17. – Catalog.tsx – страница «каталог товаров»;

import \* as React from 'react';

import { connect } from 'react-redux';

import { State, Sneakers } from 'types';

import { fetchCatalog } from 'store/ducks/catalog/actions';

import { getCatalog, isLoading } from 'store/ducks/catalog/selectors';

import { SneakersList, PageMeta, PageContainer, Page } from 'components';

import { CatalogStub } from './Catalog.stub';

type Props = {

    data: Sneakers[];

    fetchCatalog: () => void;

    isLoading: boolean;

};

function Catalog(props: Props) {

    const { isLoading, data, fetchCatalog } = props;

    React.useEffect(() => {

        if (!data.length) {

            fetchCatalog();

        }

    }, []);

    if (isLoading || !data.length) {

        console.log(data.length);

        return <CatalogStub />;

    }

    return (

        <Page>

            <PageMeta

                title="Catalog page"

                description="See awesome collection of snickers"

            />

            <PageContainer>

                <h2>Catalog</h2>

                <SneakersList items={data} />

            </PageContainer>

        </Page>

    );

}

const mapStateToProps = (state: State) => ({

    data: getCatalog(state),

    isLoading: isLoading(state),

});

const mapDispatchToProps = { fetchCatalog };

export default connect(

    mapStateToProps,

    mapDispatchToProps

)(Catalog) as React.ComponentType;

Б.18. – Home.tsx – домашняя страница сайта;

import \* as React from 'react';

import { connect } from 'react-redux';

import \* as bem from 'b\_';

import { State, Sneakers as SneakersType } from 'types';

import { PageMeta, SneakersList, Page, PageContainer } from 'components';

import { fetchHomepage } from 'store/ducks/homepage/actions';

import { getHomepage, isLoading } from 'store/ducks/homepage/selectors';

import { HomeStub } from './Home.stub';

type Props = {

    data: {

        popular: SneakersType[];

        newest: SneakersType[];

    };

    fetchHomepage: () => void;

    isLoading: boolean;

};

const b = bem.with('page');

function Home(props: Props) {

    const { isLoading, data, fetchHomepage } = props;

    React.useEffect(() => {

        if (!data.popular.length) {

            fetchHomepage();

        }

    }, []);

    if (isLoading) {

        return <HomeStub />;

    }

    return (

        <Page>

            <PageMeta title="Neu." description="Buy awesome snickers" />

            <PageContainer>

                <h2>Popular</h2>

                <SneakersList items={data.popular} />

            </PageContainer>

            <PageContainer btn={{ to: '/catalog', text: 'See more' }}>

                <h2>Newest</h2>

                <SneakersList items={data.newest} />

            </PageContainer>

        </Page>

    );

}

const mapStateToProps = (state: State) => ({

    data: getHomepage(state),

    isLoading: isLoading(state),

});

const mapDispatchToProps = { fetchHomepage };

export default connect(

    mapStateToProps,

    mapDispatchToProps

)(Home) as React.ComponentType<Props>;

Б.19. – Sneakers.tsx – страница информации о товаре.

import \* as React from 'react';

import Slider from 'react-slick';

import \* as bem from 'b\_';

import { Page, Button, ButtonSizes, PageMeta, SneakersList, PageContainer, } from 'components';

import { SneakersStub } from './Sneakers.stub';

import { useSneakers } from './Sneakers.hook';

import 'slick-carousel/slick/slick.css';

import 'slick-carousel/slick/slick-theme.css';

import './Sneakers.css';

const b = bem.with('sneakers-page');

const settings = {

    dots: false,

    infinite: true,

    speed: 500,

    slidesToShow: 1,

    slidesToScroll: 1,

    arrows: true,

    className: 'sneakers-page\_\_slider',

    responsive: [

        {

            breakpoint: 480,

            settings: {

                slidesToShow: 1,

                slidesToScroll: 1,

                dots: true,

                arrows: false,

            },

        },

    ],

};

function SneakersPage() {

    const { match, popular, isLoading, data, fetchShoes, fetchHomepage, } = useSneakers();

    React.useEffect(() => {

        if (!data || data.slug !== match.params.slug) {

            fetchShoes(match.params.slug);

        }

    }, [match]);

    React.useEffect(() => {

        if (!popular.length) {

            fetchHomepage();

        }

    }, []);

    if (isLoading || !data) {

        return <SneakersStub />;

    }

    const { title, price, images, description, vender, email, phoneNumber } = data;

    return (

        <Page mix={b()}>

            <PageMeta

                title={title}

                description={description}

                image={images[0]} />

            <PageContainer>

                <div className="sneakers-content">

                    <div className="sneakers-content-left">

                        <h1>{title}</h1>

                        <p>{price}</p>

                        <Slider {...settings}>

                            {images.map(url => (

                                <div key={url}>

                                    <img alt="" className={b('image')} src={url} />

                                </div>

                            ))}

                        </Slider>

                    </div>

                    <div className="sneakers-content-right">

                        <div className="sneakers-vender">

                            <p>Venderer: {vender}</p>

                            <p>Email: {email}</p>

                            <p>Phone number: {phoneNumber}</p>

                        </div>

                        <div className={b('description')}><hr />{description}</div>

                    </div>

                </div>

            </PageContainer>

            {Boolean(popular.length) && (

                <PageContainer>

                    <h2>See also</h2>

                    <SneakersList items={popular} />

                </PageContainer>

            )}

        </Page>

    );

}

export default SneakersPage;

Б.20. – getInitialState.ts – формирование начального состояния роутера;

import { RouterState } from 'connected-react-router';

import { State } from 'types';

import { initialState as homepage } from './ducks/homepage/reducer';

import { initialState as catalog } from './ducks/catalog/reducer';

import { initialState as shoes } from './ducks/shoes/reducer';

export const getInitialState = (pathname: string = '/'): State => {

    return {

        homepage,

        catalog,

        shoes,

        router: {

            location: { pathname, search: '', hash: '', key: '' },

            action: 'POP',

        } as RouterState,

    };

};

Б.21. – rootReducer.ts – сборщик редьюсеров каждой из страниц;

import { combineReducers } from 'redux';

import { connectRouter } from 'connected-react-router';

import { History } from 'history';

import homepage from './ducks/homepage/reducer';

import catalog from './ducks/catalog/reducer';

import shoes from './ducks/shoes/reducer';

import { State } from 'types';

export default (history: History) =>

    combineReducers<State>({

        homepage,

        catalog,

        shoes,

        router: connectRouter(history),

    });

Б.22. – rootSage.ts – запуск saga-генераторов;

import { fork, all } from 'redux-saga/effects';

import { homepageSaga } from './ducks/homepage/saga';

import { routerSaga } from './ducks/router/saga';

import { catalogSaga } from './ducks/catalog/saga';

import { shoesSaga } from './ducks/shoes/saga';

export default function\* rootSaga() {

    yield all([

        fork(homepageSaga),

        fork(routerSaga),

        fork(catalogSaga),

        fork(shoesSaga),

    ]);

}

Б.23. – rootStore.ts – конфигурация корневого хранилища дерева состояния;

import { createStore, compose, applyMiddleware, Store } from 'redux';

import createSagaMiddleware, { END, SagaMiddleware } from 'redux-saga';

import { routerMiddleware } from 'connected-react-router';

import { createBrowserHistory, createMemoryHistory } from 'history';

import { AppStore, State } from 'types';

import createRootReducer from './rootReducer';

import rootSaga from './rootSaga';

function getComposeEnhancers() {

    if (process.env.NODE\_ENV !== 'production' && !isServer) {

        return window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_COMPOSE\_\_ || compose;

    }

    return compose;

}

export const isServer = !(

    typeof window !== 'undefined' &&

    window.document &&

    window.document.createElement

);

export function configureStore(initialState: State, url = '/') {

    const history = isServer

        ? createMemoryHistory({ initialEntries: [url] })

        : createBrowserHistory();

    const sagaMiddleware = createSagaMiddleware();

    const composeEnhancers = getComposeEnhancers();

    const middlewares = [routerMiddleware(history), sagaMiddleware];

    const store = createStore(

        createRootReducer(history), initialState, composeEnhancers(applyMiddleware(...middlewares))

    ) as AppStore;

    // используются на сервере

    store.runSaga = sagaMiddleware.run;

    store.close = () => store.dispatch(END);

    if (!isServer) {

        sagaMiddleware.run(rootSaga);

    }

    return { store, history };

}

Б.24. – models.ts – модель товара;

// Product scheme

export type Sneakers = {

    id: string;

    slug: string;

    title: string;

    category: string;

    description: string;

    image: string;

    images: string[];

    price: string;

    subtitle: string;

    url: string;

    text: string;

    date?: string;

    email: string;

    phoneNumber: string;

    vender: string;

};

Б.25. – redux.ts – прототипирование саги, редакс-экшена и стейта

import { Action, Store } from 'redux';

import { SagaMiddleware } from '@redux-saga/core';

import { RouterState } from 'connected-react-router';

import { HomepageState } from 'store/ducks/homepage/reducer';

import { CatalogState } from 'store/ducks/catalog/reducer';

import { ShoesState } from 'store/ducks/shoes/reducer';

// Redux types

export interface ReduxAction<T = any, P = any> extends Action {

    type: T;

    payload?: P;

}

export type AppStore = Store & {

    runSaga: SagaMiddleware['run']; // запуск саги

    close: () => void;  // остановка саги

};

export interface State {

    readonly homepage: HomepageState;

    readonly catalog: CatalogState;

    readonly shoes: ShoesState;

    readonly router: RouterState;

}

Б.26. – index.ts – сбор модели товара, состояния редакса, саги и экшена вместе для импорта в компонентах.

import { Dispatch } from 'react';

import { match } from 'react-router';

import { State, AppStore, ReduxAction } from './redux';

import { Sneakers } from './models';

export type RouterFetchDataArgs = {

    dispatch: Dispatch<ReduxAction>;

    match: match<{ slug: string }>;

};

export { Sneakers, State, AppStore, ReduxAction };