МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема: Web-сайт «Сборник текстовых квестов»

**Исполнитель**

студент 2 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Д. Савко

подпись, дата

**Руководитель**

Доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Жиляк

(должность, уч. звание) (подпись, дата)

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Жиляк

подпись дата инициалы и фамилия

Минск 2023

**Содержание**

[Введение 5](#_Toc122376412)

[1 Постановка задач и обзор аналогичных решений 6](#_Toc122376413)

[1.1 Актуальность темы 6](#_Toc122376414)

[1.2 Обзор аналогичных решений 6](#_Toc122376415)

[1.3 Постановка задачи 10](#_Toc122376416)

[1.4 Выводы по разделу 10](#_Toc122376417)

[2 Проектирование веб-сайта 11](#_Toc122376418)

[2.1 Выбор средств реализации 11](#_Toc122376419)

[2.2 Проектирование страниц 14](#_Toc122376422)

[2.3 Выбор стилевого оформления 15](#_Toc122376423)

[2.4 Разработка пользовательских элементов 16](#_Toc122376424)

[2.5 Выводы по разделу 18](#_Toc122376425)

[3 Реализация структуры веб-сайты 19](#_Toc122376426)

[3.1 Структура JSX-документа 19](#_Toc122376427)

[3.2 Структура Redux-Toolkit 24](#_Toc122376428)

[3.3 Реализация серверной части 26](#_Toc122376429)

[3.4 Структура базы данных 28](#_Toc122376430)

[3.6 Выводы по разделу 30](#_Toc122376432)

[4 Тестирование веб-сайта 31](#_Toc122376433)

[4.1 Адаптивный дизайн веб-сайта 31](#_Toc122376434)

[4.2 Кроссбраузерность веб-сайта 32](#_Toc122376435)

[4.3 тестирование 34](#_Toc122376436)

[4.4 Выводы по разделу 36](#_Toc122376437)

[5 Руководство пользователю 37](#_Toc122376438)

[Заключение 38](#_Toc122376439)

# Введение

В настоящее время количество новых сайтов увеличивается с неимоверной скоростью. Именно поэтому, основной задачей, ставящейся при разработке сайта – сделать его максимально привлекательным для потенциальных пользователей и придать ему индивидуальность. На заре развития интернета можно было привлечь достаточно большое количество посетителей на сайт, не прикладывая к его разработке больших усилий и без особых материальных затрат. Сейчас же, в условиях возросшей конкуренции, функциональность сайта, его стиль и запоминаемость играют очень большую роль. Ведь независимо от того создается сайт для представления какой-либо фирмы или просто для заработка на рекламе, основной его функцией является привлечение как можно большей аудитории. Поэтому для достижения успеха важен правильный подход к выбору стратегии и исполнителей.

Профессиональные студии веб – дизайна предлагают создание сайтов различного предназначения, это может быть корпоративный сайт, сайт – визитка, интернет – магазин, сайты для ведения бизнеса как онлайн, так и в реальном мире.

Наше время называют «информационным веком». Это название возникло потому, что самым важным, ценным и необходимым ресурсом является информация. Обладание информацией и умение своевременно, быстро и четко донести необходимую информацию до клиентов – ключ к успеху в реалиях современного бизнеса. В настоящий момент именно Интернет, способен оперативно и массово передавать текст, звук, изображения и даже видеоролики, т.е. практически любую информацию. C точки зрения пользователя, Интернет – это огромный информационный ресурс, в котором можно найти все, что угодно: от прогноза погоды до личных предпочтений голливудских кинозвезд. Аудитория сети Интернет растет с каждым годом. Интернет – самое быстроразвивающееся средство передачи информации за все историю человечества. Пользователи сети интернет являются целью особого интереса для рекламодателей, потому что факт остается фактом – среди пользователей сети куча потенциальных клиентов. Актуальность темы сборника текстовых квестов в современном мире обусловлена тем, что данный жанр компьютерных игр сохраняет свою популярность, так как он позволяет погрузиться в новые миры через текст и вместе с тем делает их интерактивными, кроме того на базе функционала текстовых квестов можно реализовать головоломки и викторины. Кроме того, текстовые квесты являются удобным для взаимодействия форматом на самых разнообразных устройствах благодаря возможности масштабировать текст, а сами текстовые квесты не требует какой-либо скорости.

Цель курсовой работы: разработать сборник текстовых квестов с использованием React JS, а также с применением CSS3/CSS modules.

Задачи курсовой работы:

* проанализировать и описать основные принципы работы React и Redux;
* разработать приложение, используя React и Redux;
* провести тестирование приложения;
* провести анализ и оценку качества приложения.

# 1 Постановка задач и обзор аналогичных решений

## Актуальность темы

Актуальность темы заключается в том, что текстовые квесты являются популярных жанром компьютерных игр. Несмотря на их внешнюю простоту они притягивают игроков своей атмосферой, возможностью насладиться литературным языком, игроки, используя свое воображение погружаются в неизведанные миры. На базе инструментария текстовых можно создать интересные для игроков головоломки.

Сегодня существуют различные сборники текстовых квестов, в том числе и онлайн, которые используют различные технологии и платформы для создания своих приложений или сайтов. Но, для создания сборника текстовых квестов, с целью реализовать удобный и качественный интерфейс оптимальным решением будет React JS для написания пользовательского интерфейса.

## 1.2 Обзор аналогичных решений

Веб-сайт – это набор взаимосвязанных веб-страниц, которые представляют собой информацию, предназначенную для просмотра в браузере. Веб-сайт представляет собой совокупность веб-страниц, которые могут быть связаны между собой посредством гипертекстовых ссылок.

Для создания быстрого и динамического веб-сайта необходимо использовать современные технологии, которые позволяют создавать веб-сайт с минимальными затратами времени и средств. Веб-сайт должен быть удобным для пользователя, а также должен быть адаптирован под мобильные устройства. Тема веб-сайта должна быть востребованной, а также должна быть актуальной. Поэтому было принято решение разработать сборник текстовых квестов, который будет представлять собой веб-сайт, на котором можно будет поиграть в различные текстовые квесты. Прежде чем приступить к разработке веб-сайта, необходимо проанализировать аналоги, чтобы понять какие есть недостатки и преимущества веб-сайтов аналогов и найти в каждом аналоге свои уникальные особенности, которые будут отличать этот сайт от других. Для этого будет рассмотрены 3 аналога.

Аналог №1 – сайт плеер квестов из игры Космические Рейнджеры [1].

Данный сайт является фанатским проектом, он позволяет сыграть в текстовые квесты из игры “Космические рейнджеры”. На рисунке 1.1 представлено оформление сайта. Веб-сайт очень простой и удобный для пользователя, а также имеет красивый дизайн. Шрифт читабельный. Сайт использует два основных цвета (синий и белый), что положительно сказывается на общем восприятии сайта. На этом сайте есть функция поиска квестов и их сортировки по частям игры. Также можно изменить параметры, которые передаются в сам квест. На страницы сайта с запущенным квестом копируют дизайн оригинальной игры.

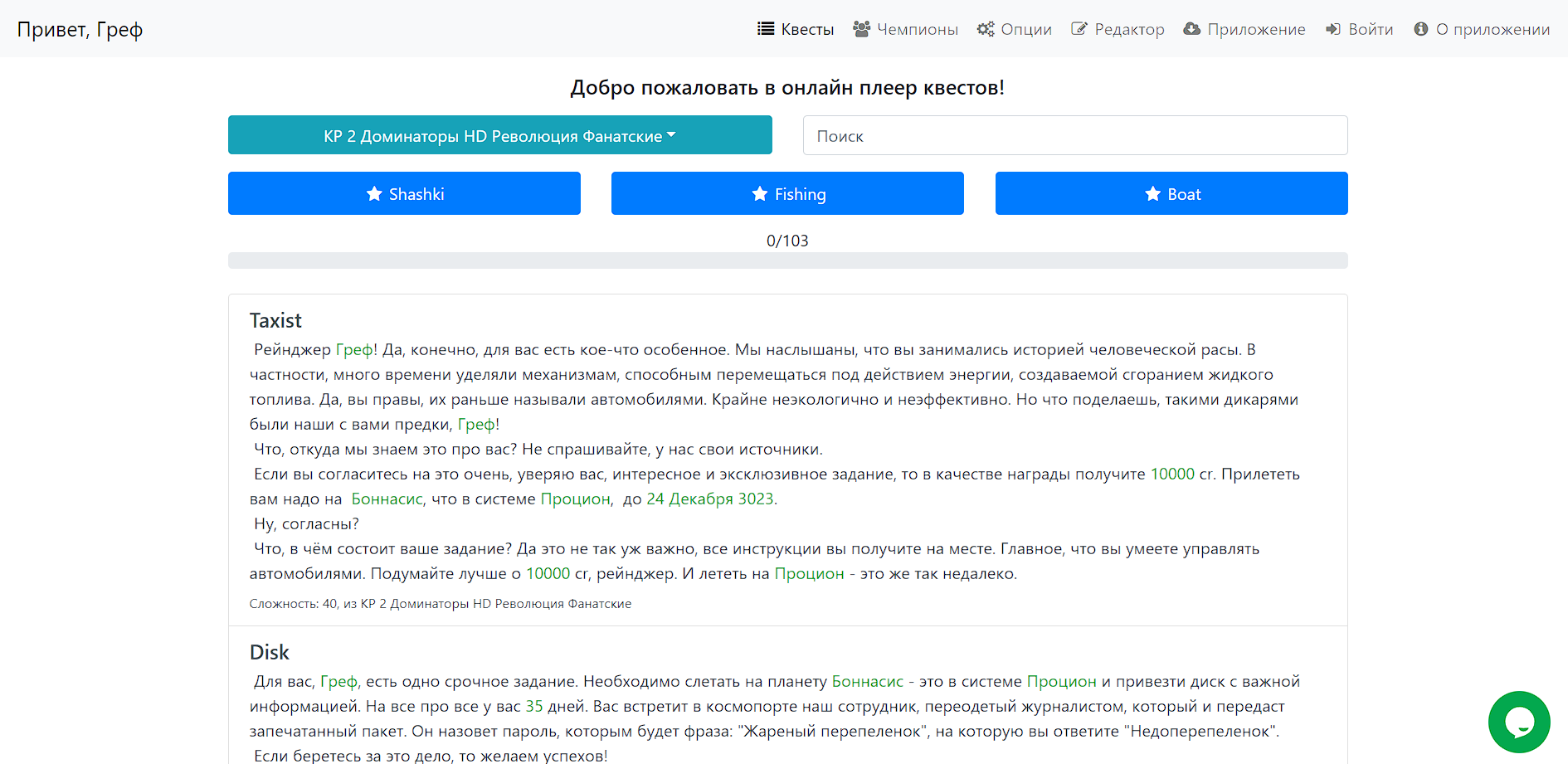


Рисунок 1.1 – Главная страница сайта “плеер квестов из игры Космические Рейнджеры”



Рисунок 1.2 –Страница с квестом “плеер квестов из игры Космические Рейнджеры”

Сайт адаптивен, на главной странице опции сворачиваются в бургер-меню. Страница с запущенным квестом также является адаптивной и масштабируемой, элементы интерфейса выстраиваются вертикально.

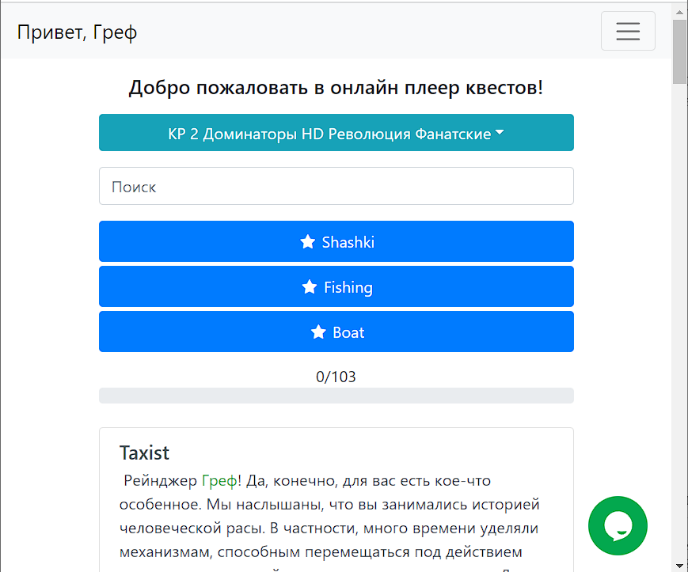


Рисунок 1.3 – Мобильная версия сайта “плеер квестов из игры Космические Рейнджеры”

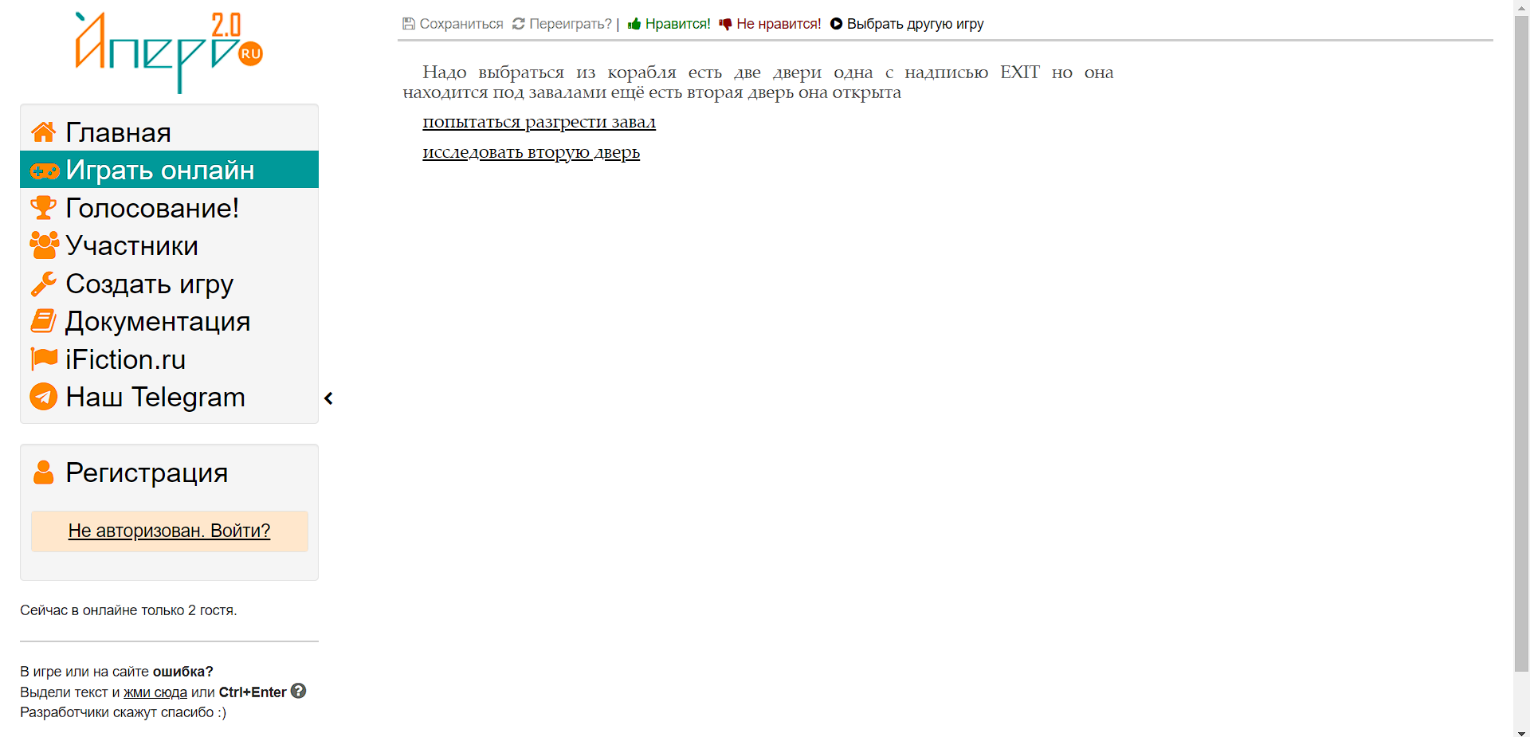


Рисунок 1.4 – Мобильная версия квестовой страницы “плеер квестов из игры Космические Рейнджеры”

Из минусов данного веб-сайта можно выделить то, адаптивность страницы с квестами реализована плохо, из-за чего при использовании на устройствах с вертикальной ориентацией экрана может сильно изменятся интерфейс. Кроме того, шрифт мобильной версии достаточно мал.

Аналог №2 – сайт Apero [2].

Следующим аналогом будет сайт Apero, на нем можно запустить различные текстовые квесты и игры близких жанров. Сайт поддерживает игры с самыми разными методами взаимодействия. Веб-сайт имеет простой дизайн, в тоже время интерфейс сайта неудобен, а дизайн сайт реализован примитивно, верстка реализована плохо.

Рисунок 1.3 – Главная страница сайта

Также сайт имеет адаптивный дизайн, который позволяет просматривать сайт на мобильных устройствах, но мобильная верстка реализована плохо. Из недостатков можно отметить неудобный интерфейс, плохой дизайн, плохая адаптивность

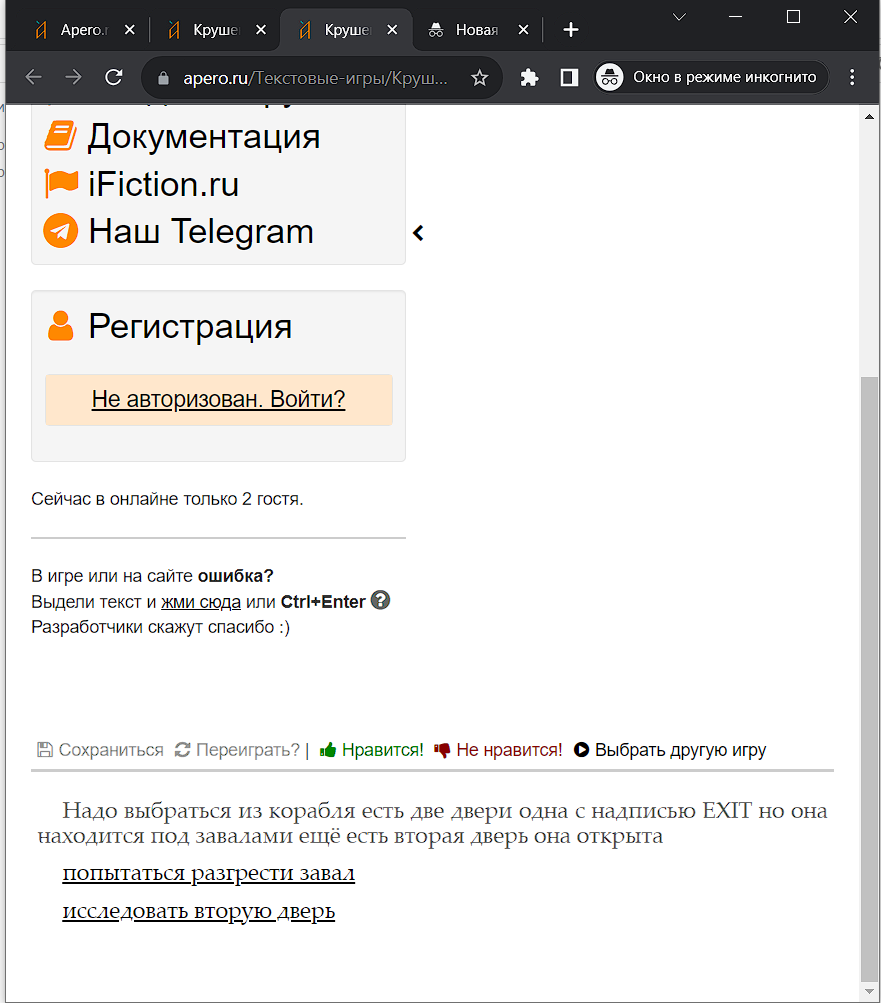


Рисунок 1.4 – Мобильная версия сайта

Аналог №3 – “текстовые квесты онлайн – космические рейнджеры” [3].

Данный сайт, который является фанатским проектом по адаптации квестов из игры “космические рейнджеры”. На сайте можно поиграть в различные текстовые квесты. Сайт реализован просто, цветовая гамма содержит только черный и белый цвета. На рисунке 1.5 представлено оформление сайта.

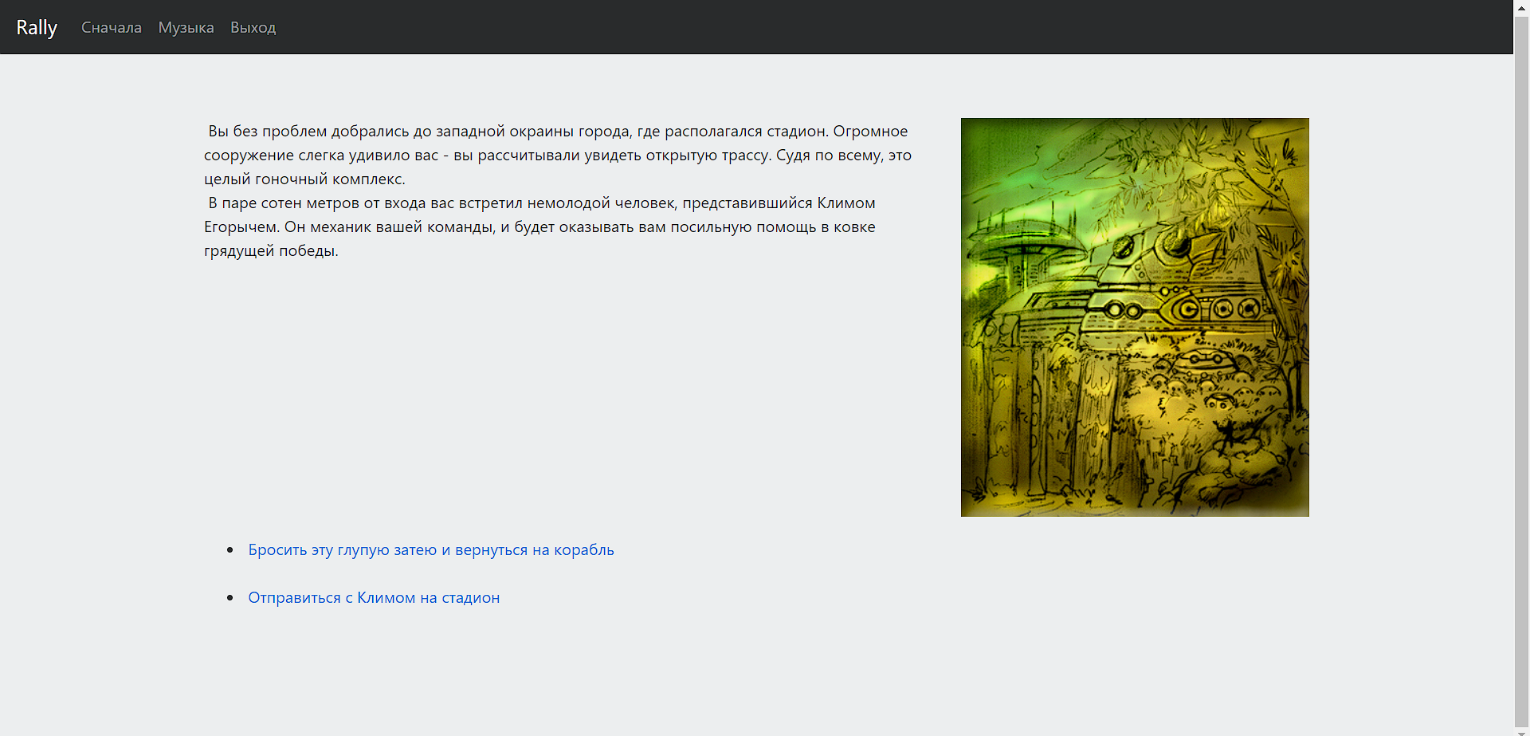


Рисунок 1.5 – страница сайта с квестом

Также данный сайт, адаптирован под мобильные устройства. На рисунке 1.6 представлено оформление на мобильных устройствах. Хорошо реализована адаптивность, но размер картинки сильно скачет при масштабировании.

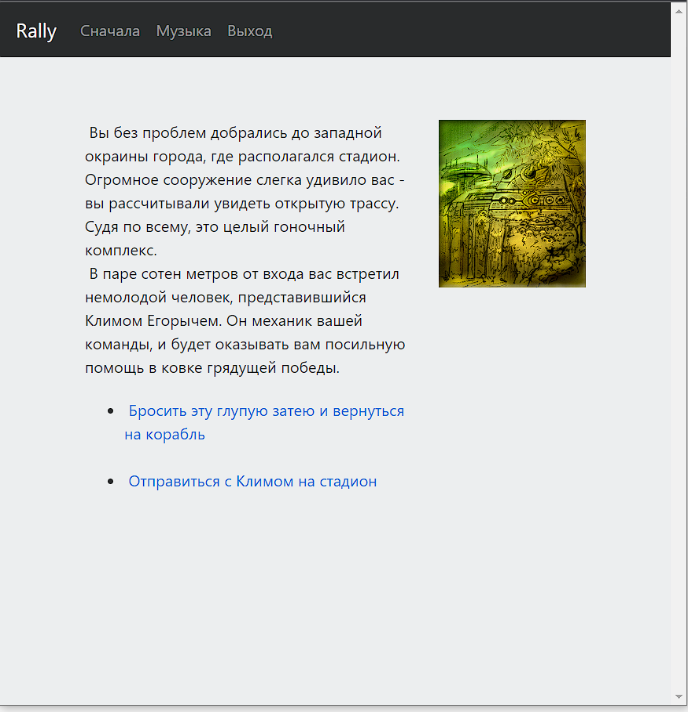


Рисунок 1.6 – Мобильная страницы версия сайта с квестом

И таким образом, было проанализировано 3 сайта, которые являются аналогами нашего будущего сайта. Основные преимущества и недостатки этих сайтов были выделены, а также были выделены технологии, которые используются для создания сайтов. Можно сделать вывод, что сайт, который будет создан в рамках данной работы, будет иметь следующие преимущества:

* простой и удобный дизайн;
* приятные и не раздражающие глаза цвета;
* кроссбраузерность;
* адаптивность под мобильные устройства;
* удобная навигация по сайту.
* Поддержка текстовых квестов с использованием переменных.

## Постановка задачи

Основные задачи для сборника текстовых квестов:

* разработать структуру хранения текстовых квестов;
* разработать интерпретатор текстовых квестов
* разработать структуру веб-сайта;
* разработать сервер для хранения текстовых квестов и передачи их в интерпретатор;
* разработать возможность выбора текстовых квестов;
* разработать обработчики событий ошибок 500 и 404.

## 1.4 Выводы по разделу

В данном разделе была подчеркнута необходимость создания сборника текстовых квестов, а также были выделены основные преимущества и недостатки существующих сайтов. Были выделены технологии, которые используются для создания сайтов. Были выделены основные задачи сборника текстовых квестов.

# 2 Проектирование веб-сайта

## 2.1 Выбор средств реализации

Главной задачей сайта является воспроизведение текстовых квестов. Данный сайт будет предназначен для людей, которые хотят поиграть в различные текстовые квесты. Сайт будет адаптирован под мобильные устройства.

Интернет-магазин будет разработан в текстовом редакторе Visual Studio Code (1.73.1). Страницы сайта могут просматриваться в таких браузерах как Microsoft Edge, Opera, Mozilla Firefox, Google Chrome.

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса и средства для рефакторинга.

Для реализации будущего сайта будет использована следующая архитектура:

* Front-end.
* Back-end.

Front-end — это часть сайта, которая видна пользователю. В данном случае это будет внешний вид сайта, его дизайн, а также навигация по сайту. В данном случае будет использованы следующие технологии: React JS (16.13.1), CSS modules, Redux-Toolkit (1.3.5), React Router (5.1.2).

JavaScript – это язык программирования, который используется для создания динамических веб-страниц. JavaScript позволяет создавать интерактивные веб-страницы, которые могут реагировать на действия пользователя.

React JS — это библиотека JavaScript, предназначенная для разработки пользовательских интерфейсов. В отличие от других библиотек JavaScript React JS использует компонентный подход, который позволяет разрабатывать сложные интерфейсы из простых компонентов. С помощью React разработчики создают веб-приложения, которые изменяют отображение без перезагрузки страницы. Благодаря этому приложения быстро реагируют на действия пользователя, например, подгрузка списков, смена вариантов ответа и так далее. React применяют для отрисовки компонентов пользовательского интерфейса. Также библиотека может полностью управлять фронтендом. В этом случае React используют с библиотеками для управления состоянием и роутинга, например, Redux и React Router.

CSS-модули — это подход к организации CSS-кода, который позволяет вам изолировать стили для каждого компонента. Таким образом, стили одного компонента не влияют на другие компоненты и позволяют избежать конфликтов имен классов.

Redux — это библиотека для управления состоянием приложения.

Redux Toolkit — это набор инструментов для упрощения работы с Redux. Redux Toolkit позволяет создавать более простые и понятные редьюсеры, а также упрощает работу с асинхронными запросами.

React Router — это библиотека для маршрутизации в React-приложениях. React Router позволяет создавать маршруты, которые будут отображаться в зависимости от URL-адреса. Благодаря React Router можно создавать SPA (Single Page Application), которые не перезагружают страницу при переходе по ссылкам.

Back-end - это часть приложения, которая отвечает за обработку данных, которые приходят от клиента и возвращают данные клиенту. Также Back-end отвечает за работу с базой данных, которая хранит данные приложения. Для создания Back-end будет использованы следующие технологии: Node.js (12.16.1), Express (4.17.1).

Node.js — это JavaScript-окружение, которое позволяет запускать JavaScript-код на сервере. Node.js позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

Express — это фреймворк для Node.js, который позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

Для передачи данных между клиентом и сервером, а также в качестве основы для языка написания квестов будет использоваться формат JSON.

JSON — это текстовый формат обмена данными, который основан на JavaScript. JSON позволяет передавать данные между клиентом и сервером. JSON позволяет передавать данные в виде объектов, массивов, строк, чисел, логических значений и null.

## 2.2 Проектирование страниц

В данном проекте будет использована Flexbox-верстка (Flexbox), так как она позволяет создавать адаптивные веб-страницы, которые будут отображаться корректно на всех устройствах. С помощью этой технологии можно очень просто и гибко расставить элементы в контейнере, распределить доступное пространство между ними, и выровнять их тем или иным способом даже если они не имеют конкретных размеров. CSS Flexbox поддерживается всеми используемые на сегодняшний момент современными браузерами (с использованием префиксов: IE10+, Edge12+, Firefox 2+, Chrome 4+, Safari 3.1+, Opera 12.1+, iOS Safari 3.2, Opera mini, Android 2.1+).

Исходя из выбора способа верстки были расположены все основные элементы сайта. На сайте будут представлены следующие страницы:

* + Главная страница.
  + Квестовая страница.
  + О нас.
  + Faq.

На главной странице будут расположены селекторы квестов с различными текстовыми квестами.

На квестовой странице будет воспроизводится текстовый квест, она содержит картинку, текст описания, варианты ответа.

На странице “о нас” будет приведена краткая информация.

На странице “faq” будут расположены самые главные вопросы по использованию, которые могут возникнуть у пользователей.

Также будет фиксированный Navbar с ссылками на главную страницу, страницу “”, страницу “”, квестовую страницу. Navbar будет расположен вверху страницы, страница для перехода переключается кнопкой в правом углу Navbar-а.

## 2.3 Выбор стилевого оформления

Для реализации стилевого оформления сайта было выбрано минималистическое направление с элементами классического веб-дизайна. Такой стиль позволяет привлечь внимание новых пользователей. Сайт не загромождается лишними деталями (боковыми панелями и т.п.), большим количеством спецэффектов. Акцент на сайте ориентируется на основной контент.

Сайт будет состоять из 2-х основных цветов - белого и чёрного. Белый цвет будет использоваться для фона, а чёрный для текста и некоторых элементов дизайна. Эти цвета были подобраны так, потому что они будут приятны для глаз и это цвета, на которое пользователь сразу обратит внимания на сайт.

В данном проекте будут использован один основной шрифт: Latin Modern Mono. Шрифт подключался хранится рядом со стилями в папке fonts.

Шрифт Latin Modern Mono подходит для данного сайта, так как он похож на старинный, кроме того, он моноширинный, что делает его еще более старинным. Он достаточно хорошо читается, имеется поддержка utf-8.

## 2.4 Выводы по разделу

Подводя итоги этого раздела, можно сказать, что была разработана структура сайта. Были выбраны основные цвета и шрифты, которые будут использоваться на сайте. Были определены основные задачи. Были выбраны инструменты, необходимые для написания сайта. Самое главное, чтобы сайт был удобным для пользователя и отвечал его потребностям.

# 3 Реализация структуры веб-сайты

## 3.1 Структура JSX-документа

Для реализации проекты было принято решение создавать разные компоненты. Компонент — это функция, которая возращает JSX формат.  JavaScript XML (JSX) — это расширение синтаксиса JavaScript, которое позволяет использовать HTML-подобный синтаксис для описания структуры интерфейса. Как правило, компоненты написаны с использованием JSX, но также есть возможность использования обычного JavaScript. Код примера JSX-формата показан в листинге 3.1.

|  |
| --- |
| const Footer: FC = () => {     return (        <div className='footer'>           <div className="wrapper">              <div className="logo">                 WebShop              </div>              <ul className="nav">              <div className="contacts">                 <ul>                    {contacts.map((item: contactsTS) => {                       const { img, name, Link } = item;                       return (                          <li key={name}>                             <a href={Link}>                                <img src={img} alt={name} />                                <p>{name}</p>                             </a>                          </li>                       )                    })}                 </ul>              </div>           </div>        </div >     )  }  export default Footer |

Листинг 3.1 – Структура JSX-формата

Самым главным компонентом будет с названием файлом и функции App.tsx. Он будет содержать основные маршруты по сайту. Для этого нужно использовать пакет React-router-dom. Почему лучше использовать его, чем обычный тег <a>? Потому что React это SPA. SPA (single page application) – это веб-приложение, работающее на одной странице. Оно подгружает все необходмые javascript и css файлы при первой загрузке страницы, а затем все общение между клиентом и сервером сводится к минимуму. Т.е. при таком подходе большая часть работы сайта производится на стороне клиента, а если нужно получить данные с сервера, то это обычно делается с помощью JSON. Код примера маршрутов будет показан в листинге 3.2.

|  |
| --- |
| <BrowserRouter>          <Navbar />          <Suspense fallback={<Loading />}>            <Routes>              <Route path="/" element={<Home />} />              <Route path="/basket" element={<Basket />} />              <Route path="/product" element={<Product />} />              <Route path="/productFull/:id" element={<ProductFull />} />              <Route path="\*" element={<Error />} />            </Routes>          </Suspense>   </BrowserRouter> |

Листинг 3.2 – Структура маршрутов

Следующим компонентом будет Product компонент. В нем будет находится карточка товара. Карточка товара будет состоять из:

* Фото товара.
* Названия товара.
* Описание товара.
* Цена товара.
* Кнопки добавления товара в корзину.

Кнопка добавления товара в корзину будет висеть поверх фото товара. При наведении на кнопку добавления товара в корзину, она будет менять цвет. При нажатии на кнопку добавления товара в корзину, она будет менять цвет и появится надпись, что товар добавлен в корзину. Код карточки товара будет показан в листинге 3.3.

|  |
| --- |
| const { id\_product, img } = item;  return (  <div className="product" key={index}>  {item.sale !== 0 ? (  <div className="sale">  <p className='sales'>{item.sale}%</p>  ) :  <div className="sale">  {item.price}$  </div>  }  <div className="choise\_product">  <div className="blocks">  <div className="block">  <p onClick={() => AddToBasket(item)}> 🛒</p>  </div>  <div className="block">  <Link to={`/product/:${id\_product}`}>  <p>🔍</p>  </Link>  </div>  </div>  </div>  <img src={img} alt={item.category}  />  </div> |

Листинг 3.3 – Структура карточки товара

Следующим компонентом будет Basket компонент. В нем будет находится товары, которые были добавлены в корзину. Также будет находится блок с общей суммой товаров в корзине и кнопкой оформления заказа. На товаре будут кнопки добавить в корзину и удалить из корзины. Код корзины будет показан в листинге 1.4.

|  |
| --- |
| <div className="product" key={img}>  <div className="id">  <p>{index + 1}</p>  </div>  <div className="img">  <Link to={`/product/:${id\_product}`}>  <img src={img} alt="" />  </Link>  </div>  <div className="name">  <Link to={`/product/:${id\_product}`}>  <p>{name}</p>  </Link>  </div>  <div className="buttons\_add\_del">  <button className='add' onClick={() => AddToBasket(item)}>Add</button>  <p className='count'><p>{count || 0}</p></p>  <button className='del'  onClick={() => DeleteToBasket(item)}  >Del</button>  </div>  </div>  </div>  <div className="title\_list">  <p>Total: {sum} $</p>  <button onClick={() => handleBuyOrder()}>Buy</button>  </div> |

Листинг 3.4 – Структура кода компонента Basket

После нажатия на кнопку оформления заказа, будет появляться модальное окно с формой для ввода данных для оформления заказа. После заполнения формы и нажатия на кнопку отправить, будет появляться модальное окно с сообщением о том, что заказ оформлен. Код формы будет показан в листинге 3.5.

|  |
| --- |
| {state === 1 && (<div className="form\_pages two">  <h1>Country</h1>  <select value={country} onChange={(e: ChangeEvent<HTMLSelectElement>) => setCountry(e.target.value)}>  <option value="Russia">Russia</option>  <option value="Belarus">Belarus</option>  <option value="Ukraine">Ukraine</option>  </select>  <h1>City</h1>  {  country === "Russia" && (  <select value={addres} onChange={(e: ChangeEvent<HTMLSelectElement>) => setAdress(e.target.value)}>  <option value="Moscow">Moscow</option>  <option value="Saint-Petersburg">Saint-Petersburg</option>  <option value="Novosibirsk">Novosibirsk</option>  </select>  country === "Belarus" && (  <select value={addres} onChange={(e: ChangeEvent<HTMLSelectElement>) => setAdress(e.target.value)}>  <option value="Minsk">Minsk</option>  <option value="Gomel">Gomel</option>  <option value="Brest">Brest</option>  <option value="Vitebsk">Vitebsk</option>  </select>  )  } |

Листинг 3.5 – Структура код формы

Следующим основным компонентом является Profile, в котором можно посмотреть информацию о пользователе. Пользователь сможет редактировать свои данные, просматривать историю заказов, информацию о доставке. Код Profile будет показан в листинге 3.6.

|  |
| --- |
| <div className="profile">  <div className="photo">  <img src={`http://localhost:3001/User/${user?.image}`  } alt="prfile"  onClick={() => PhotoModal(`http://localhost:3001/User/${user?.image}`)}  /> Update Photo  </button>  </div>  <div className="information">  <div className="info\_about\_user">  <div className="name\_info">  <div className="value\_info">  <p>  {user?.name || "No name"}  </p>  <p>  {user?.surname || "No surname"}  </p>  </div>  </div>  <div className="changeProfile">  <button  onClick={() => ChangeModel(true)}  >  Change Profile  </button>  </div>  </div> |

Листинг 3.6 – Структура компоненты Profile

Таким образом, в этом разделе были рассмотрены основные компоненты приложения, которые будут использоваться в дальнейшем. Хотелось бы отметить, что JSX-формат очень удобный для написания кода, так как он позволяет писать код в HTML-формате, что упрощает его чтение и понимание.

## 3.2 Структура Redux-Toolkit

Для того, чтобы удобно работать с данными, был использован Redux-Toolkit. В нем будет хранится информация о товарах, о пользователях, о заказах, о корзине. Благодаря CreateAsyncThunk был реализован асинхронный запрос к серверу для получения данных о товарах, о пользователях, о заказах, о корзине. Асинхронный запрос к серверу реализован с помощью библиотеки Axios. Асинхронный запрос к серверу будет показан в листинге 3.7.

|  |
| --- |
| export const DeleteComment = createAsyncThunk(  "product/deleteComment",  async ({ id\_comment, id\_product, id\_person }: IDeleteComment) => {  try {  const { data } = await axios.delete("/product/deletecomment", {  data: {  id\_comment,  id\_product,  id\_person,  },  });  return data;  } catch (error) {  console.log(error);  }  }  ); |

Листинг 3.7 – Код асинхронной функции

Благодаря CreateSlice был реализован редьюсер для получения данных о товарах, о пользователях, о заказах, о корзине. В CreateSlice будет находиться имя редьюсера, начальное состояние, редьюсеры для изменения состояния, асинхронные запросы к серверу. Для того, чтобы получить данные нужно использовать ExtraReducers, в котором будет находиться асинхронный запрос к серверу. Код CreateSlice будет показан в листинге 3.8.

|  |
| --- |
| export const productSlice = createSlice({  name: "product",  initialState,  reducers: {  Reset: (state) => {  state.product = [];  state.product = [];  state.basket = [];  state.basket\_length = 0;  state.orders = [];  state.top\_persons = [];  },  },  extraReducers: (builder) => {  builder.addCase(GetProducts.pending, (state) => {  state.loading = true;  state.message = "";  });  builder.addCase(GetProducts.fulfilled, (state, action) => {  state.loading = false;  state.product = Array.from(action.payload);  }); |

Листинг 3.8 – Структура кода CreatSlice

Затем все редьюсеры нужно объединить в один редьюсер и поместить его в Store. В Store будет находиться все состояние приложения. Для того, чтобы получить данные из Store нужно использовать useSelector. Для того, чтобы изменить данные в Store нужно использовать useDispatch. Код структуры Store будет показан в листинге 3.9.

|  |
| --- |
| const store = configureStore({  reducer: {  auth: authSlice,  product: productSlice  },  });  export default store;  export type RootState = ReturnType<typeof store.getState>;  export type AppDispatch = typeof store.dispatch;  export const useAppDispatch = () => useDispatch<AppDispatch>();  export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector; |

Листинг 3.9 – Структура кода Store

В этом разделе были рассмотрены осноыные положения по созданию глобального хранилища данных с помощью Redux-Toolkit. Были рассмотрены основные понятия по созданию редьюсеров, асинхронных запросов к серверу, получения данных из Store.

## 3.3 Реализация серверной части

Серверная часть была реализована с помощью Node JS и Express JS. Благодаря Express JS был реализован сервер, который будет обрабатывать запросы от клиента. Главным файлом сервера является express.js, потому что он будет запускать сервер и производить операции над файлами квестов. Код файла express.js будет показан в листинге 3.10.

|  |
| --- |
| const http = require("http");  const filesys = require("fs")  const cors = require("cors")   const express = require("express")   const app = express()   const dir = "./quests"  console.log("server running. let's quest")  var filename = filesys.readdirSync(dir,{withFileTypes: true})  app.use(cors())  app.use("/list",function (request, response) {      response.send(filename);  });  app.use("/preload",function (request, response) {        if(filesys.existsSync(dir+"/"+request.query.q+".json"))      {      let filebuf = filesys.readFileSync(dir+"/"+request.query.q+".json","utf-8")      response.send(JSON.parse(filebuf).description);      }      else{response.send(JSON.stringify({error:"404"}))}    });  app.use("/load",function (request, response) {      if(filesys.existsSync(dir+"/"+request.query.q+".json"))      {      let filebuf = filesys.readFileSync(dir+"/"+request.query.q+".json","utf-8")      response.send(filebuf);      }      else      {          response.send(JSON.stringify({error:"404"}))      }  });  app.use(function (request, response) {      response.send(request.query);  });  app.listen(3300); |

Листинг 3.10 – Структура кода express.js

Для работы сервера нужно расположить рядом с файлом express.js папку quests, в которой необходимо разместить текстовые квесты написанные в формате json. Запрос по адресу list нужен для получения списка квестов, preload для получения описаний квестов, а load для получения выбранного квеста полностью.

Подводя итог, в этом разделе были рассмотрены основные положения по созданию серверной части приложения. Было рассмотрено, как работает сервер, какие файлы нужно создать для того, чтобы сервер работал корректно.

## 3.5 Добавление таблиц стилей Sass

Для стилизации были использованы CSS modules, которые позволяют решить проблему каскадности и обеспечить простоту различных элементов интерфейса. Код использования CSS modules показан в листинге 3.14.

|  |
| --- |
|  |

Листинг 3.14 – Структура кода CSS modules

В этом разделе была рассмотрена структура CSS modules, что такое CSS modules и как они используются в проекте.

## 3.6 Выводы по разделу

Таким образом, были рассмотрены основные моменты, которые будут реализованы в проекте. В проекте будет реализована загрузка текстовых квестов с сервера, выбор квеста из списка, воспроизведение текстового квеста, выбранного пользователем.

# 4 Тестирование веб-сайта

## 4.1 Адаптивный дизайн веб-сайта

Сегодня количество пользователей, которые заходят в Интернет с мобильных устройств очень велико, и с каждым днем их становится все больше. А потому адаптация веб-сайта под маленькие экраны – актуальная задача и для курсового проекта.

Передо мной стояла задача: создать сайт, который будет сам подстраиваться под разные типы экранов. Главными условиями адаптивности является не фиксированный размер элементов страницы.

Для создания адаптивности веб-сайта были использованы следующие инструменты: медиа-запросы (@media), единицы измерения, которые связаны непосредственно с параметрами области просмотра (представляют собой процентные величины от текущей области просмотра браузера).

После написания основной структуры страницы на JSX и внешнего стилевого оформления появился вопрос о тестировании. Веб-сайт был открыт при помощи различных браузеров. После этого было выявлено, что во всех браузерах сайт выглядит одинаково.

При отображении сайта на мобильных устройствах навигационное меню в верхнем колонтитуле меняется на выпадающее меню, открывающееся при нажатии на кнопку меню. На рисунке 4.1 можно увидеть пример адаптивного дизайна для мобильных устройств.



Рисунок 4.1 – Адаптивный главной страницы для телефонов

Также при использовании мобильного устройства блоки с продуктами на странице для отображения располагаются друг под другом. На рисунке 4.2 можно увидеть пример.

****

Рисунок 4.2 – Блоки с продуктами на мобильных устройствах

И таким образом можно сказать, что все блоки располагаются друг за друг, потому что пользователю было все хорошо видно на его мобильном устройстве.

## 4.2 Кроссбраузерность веб-сайта

Очень важно, чтобы наш сайт был кроссбраузерным. Кроссбраузерность – это способность веб-ресурса отображаться одинаково и работать во всех популярных браузерах, без перебоев в функционировании и ошибок в верстке, а также с одинаково корректной читабельностью контента.

У каждого браузера есть свои встроенные, экспериментальные или нестандартные свойства и для того, чтобы они корректно работали было принято решения о внедрении вендорных префиксов. Они были автоматически прописаны после компиляции файла с расширением sass.

При проверке веб-сайта в Google Chrome никаких искажений текста, анимации, переходов не было выявлено. На рисунке 4.3 можно увидеть пример веб-страницы в Google Chrome.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рисунок 4.3 – Вид веб-страницы в Google Chrome а) мобильная версия; б) компьютерная версия

В Opera сайт показал схожий результат: никаких отклонений как для мобильной версии сайта, так и для компьютерной не было выявлено, текст не съехал, не произошло его наложения на картинки. На рисунке 4.4 можно увидеть пример веб-страницы в Opera.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рис 4.4 – Вид веб-страницы в Opera а) мобильная версия; б) компьютерная версия

В Яндекс всё так же работает без неисправностей, весь контент обладает правильным шрифтом, не наблюдается сдвиг текста или отсутствие картинок. На рисунке 4.5 можно увидеть пример веб-страницы в Яндекс.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рис 4.5 – Вид веб-страницы в Яндекс а) мобильная версия; б) компьютерная версия

Также для тестирования сайта было использовано Linter. Linter - это инструмент, который проверяет код на наличие ошибок. В нашем случае он проверяет код на наличие ошибок в HTML и CSS.

При просмотре веб-сайта с помощью таких браузеров как Google Chrome, Яндекс и Opera общий вид веб-страницы не изменился.

Главной целью такой тщательной проверки сайта является грамотная настройка всех показателей, одной или всех страниц сайта, оперативное обнаружение и устранение всех дефектных мест на сайте, а также его адаптация под различные устройства.

Динамика сайта на разных устройствах – наиболее важная часть в написании сайта, поэтому, на данном этапе было проанализировано поведение веб-сайта при использовании с разных устройств, а также веб-ресурс был протестирован в большом количестве браузеров. По итогу выполнения данного этапа для веб-сайта был разработан адаптивный дизайн и кроссбраузерность. Работа, проделанная с сайтом, позволяет пользователям ориентироваться по сайту. Лишних элементов нет, что упрощает пользование сайтом. При пользовании сайтом при помощи мобильного устройства некоторые элементы меняют свое положение, для более удобного пользования. В ходе тестирования были проверены все ссылки (включая модальные окна), проверены формы регистрации пользователя. Была выполнена проверка контента. Также в ходе тестирования была проверена совместимость с браузерами. (Google Chrome, Opera, Яндекс).

## 4.3 Ручное тестирование

Ручное тестирование – это тестирование, которое выполняется вручную. В данном случае ручное тестирование было проведено с целью проверки веб-сайта на соответствие требованиям, а также на наличие ошибок.

Тестирование frontend части включает в себе проверку правильности осуществления перехода между страницами, которые совершают пользователи системы, также проверку правильности отправления запросов, их обработки frontend сервером, а также правильности отображение представлений.

Рассмотрим сценарий регистрации. При заполнении формы неправильными данными должно быть получено оповещение об ошибки. Заполнение формы неправильными данными и уведомление об ошибке изображено на рисунке 4.6.

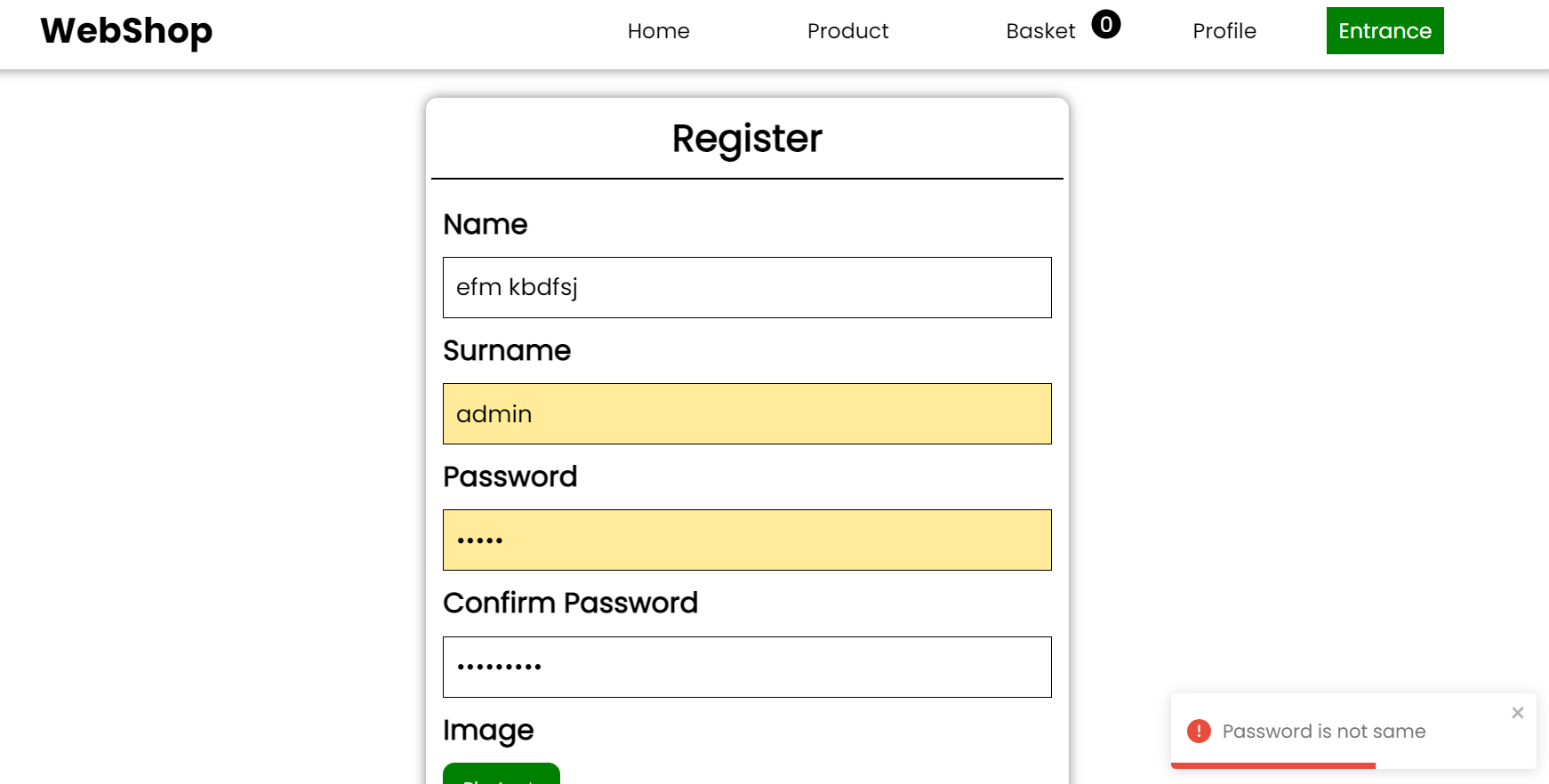


Рисунок 4.6 – Форма заполнения неверными данными

Попробуем зайти под несуществующим пользователем. При неверном введение данным, получим оповещение об ошибке. Оповещение об ошибке изображено на рисунке 4.7.

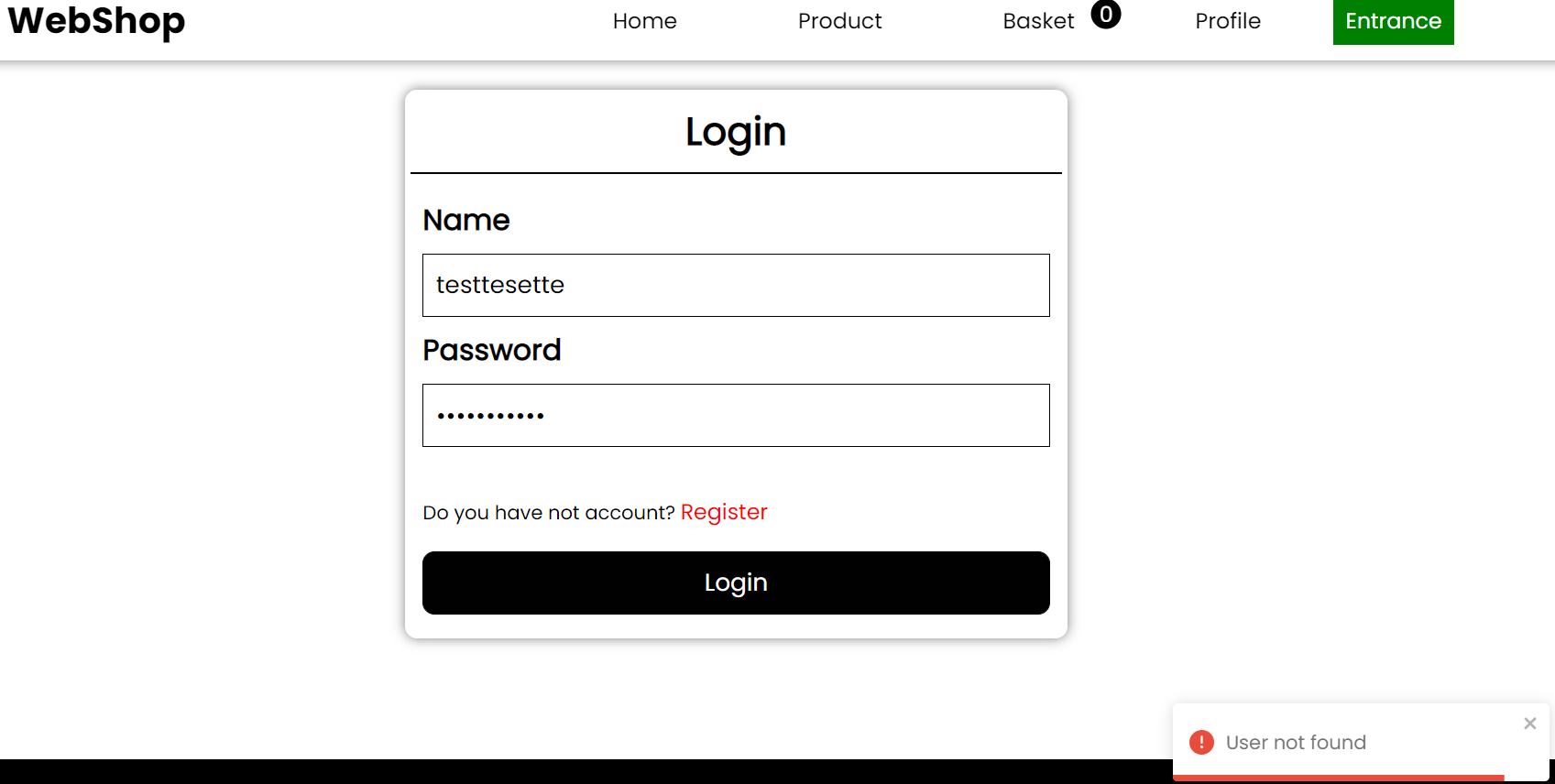


Рисунок 4.7 – Неправильный ввод данных для входа и оповещение об ошибке

Ввод правильных данных изображен на рисунке 4.8

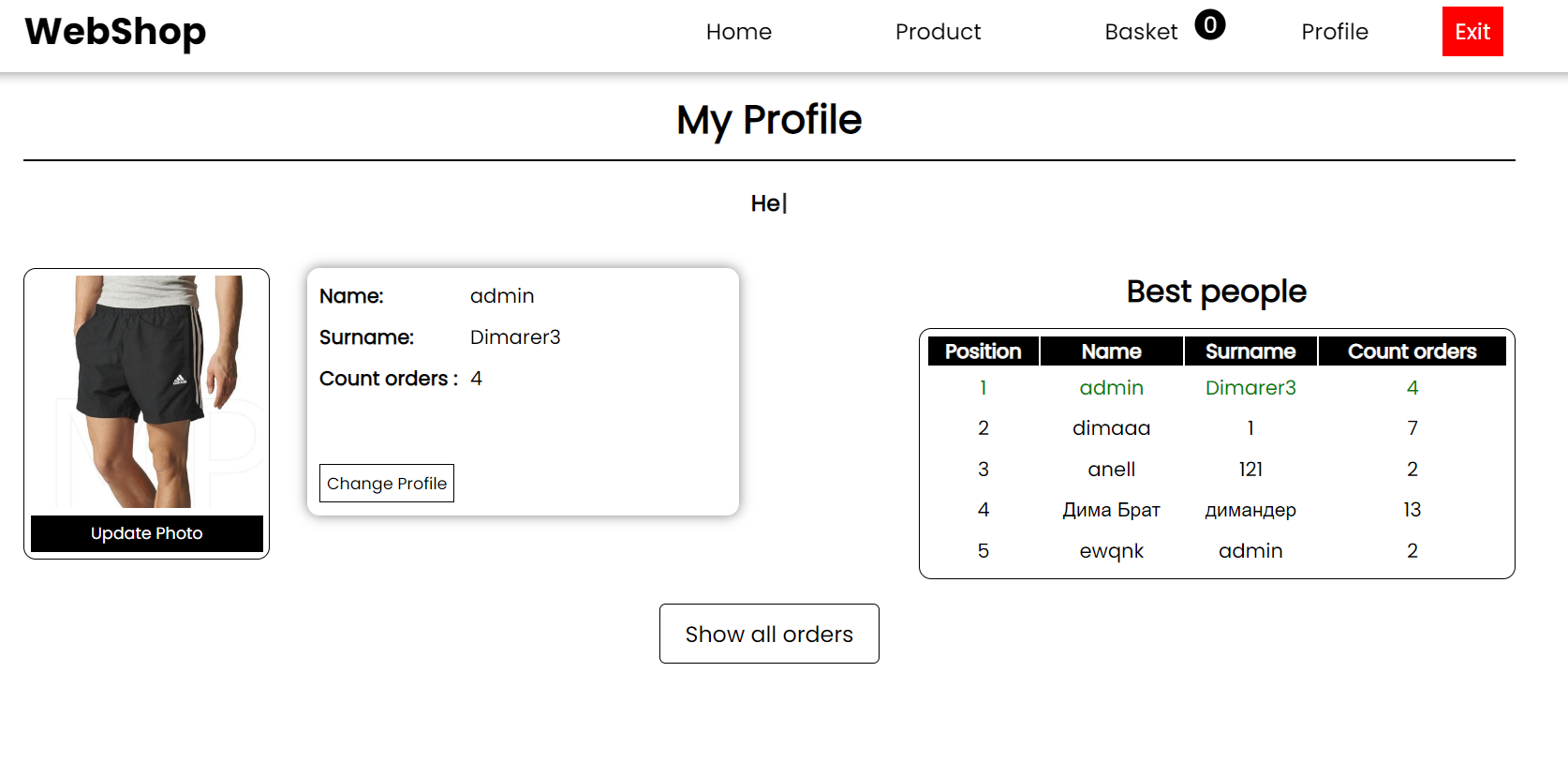


Рисунок 4.8 – Правильный ввод данных при авторизации

В ходе ручного тестирования были проведена валидация всех страниц сайта, проверены все ссылки, проверены все формы, проверен контент, проверена совместимость с браузерами. Также была проведена проверка работы корзины, где товары добавляются и удаляются из корзины. Все комментарии отображаются без ошибок. Во время тестирования не было обнаружено ошибок. Во всех браузерах работает корректно. Тестирование функционала сайта было выполнено вручную с использованием следующие браузеров Google Chrome (42.0.2311.90), Opera (29.0.1795.47) и Яндекс (22.1.0.2510)

Подводя итоги тестирования функционала сайта, можно сказать, что все функции сайта работают корректно. В ходе тестирования были проверены все ссылки (включая модальные окна), проверены формы регистрации пользователя. Была выполнена проверка контента.

## 4.4 Выводы по разделу

В этом разделе было проведено тестирование функционала сайта. Были проверены все функции сайта, которые должны работать корректно, а также были проверены все элементы, которые должны быть доступны для пользователя. Не было обнаружено никаких ошибок, которые могли бы повлиять на работу сайта.

# 5 Руководство пользователю

Данный веб-сайт предназначен воспроизведения текстовых квестов. Для того, чтобы запустить текстовый квест, нужно выбрать его и нажать кнопку “начать” под описанием.

Далее будет загружена страница с описанием, здесь можно отключить воспроизведение музыки, если это необходимо, после будет загружена квестовая страница.

На квестовой странице есть текст описания и картинка, дополняющая его, снизу находятся варианты действий, нужно выбрать наиболее подходящий вариант ответа.

Для заглушения музыки нужно нажать на значок динамика, для остановки музыки нужно нажать на паузу, для перезагрузки нужно нажать на кнопку перезагрузки, для выхода на главную страницу нужно нажать на иконку дома.

# Заключение

Целью на курсовое проектирование было создание интернет-магазина для

продажи различных товаров для пользователей, которые хотят купить товары

в интернете.

В ходе курсовой работы были выполнены следующие задачи:

* Проанализировать и описать основные принципы работы React и Redux.
* Разработать приложение, используя React и Redux.
* Провести тестирование приложения.
* Провести анализ и оценку качества приложения.

А также были выполнены задачи по созданию сборника текстовых квестов:

* Разработать интерфейс селектора квестов.
* Разработать возможность интерпретации текстовых квестов.
* Разработать структуру веб-сайта.
* Разработать сервер для хранения текстовых квестов.

Подводя итоги курсовой работы, можно сделать вывод, что в ходе работы были выполнены все поставленные задачи. Были изучены основные принципы работы React и Redux, что позволило разработать приложение и закрепить полученные знания.

# Список использованных источников

1. Ozon [Электронный ресурс]. – Режим доступа: https://spacerangers.gitlab.io/

. – Дата обращения 19.11.2023.

1. Wildberries [Электронный ресурс]. – Режим доступа: https://www.wildberries.by/. – Дата обращения 21.11.2022.
2. Aliexpress [Электронный ресурс]. – Режим доступа: https://aliexpress.ru/. – Дата обращения 21.11.2022.
3. Документация по React JS [Электронный ресурс]. - Режим доступа: https://reactjs.org/. – Дата обращения 21.11.2022.
4. Документация по Figma [Электронный ресурс]. - Режим доступа: <https://help.figma.com>. – Дата обращения 21.11.2022.
5. Документация по Redux/Redux-tolkit [Электронный ресурс]. - Режим доступа: https://redux-toolkit.js.org/ . – Дата обращения 21.11.2022.
6. Документация по Sass/Scss [Электронный ресурс]. - Режим доступа : <https://sass-scss.ru/guide/> – Дата обращения 21.11.2022.

# Приложение А