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 11](#_Toc122376416)

[1.4 Выводы по разделу 11](#_Toc122376417)

[2 Проектирование веб-сайта 12](#_Toc122376418)

[2.1 Выбор средств реализации 12](#_Toc122376419)

[2.2 Проектирование страниц 13](#_Toc122376422)

[2.3 Выбор стилевого оформления 14](#_Toc122376423)

[2.4 Выводы по разделу 14](#_Toc122376425)

[3 Реализация структуры веб-сайты 15](#_Toc122376426)

[3.1 Структура JSX-документа 15](#_Toc122376427)

[3.2 Структура Redux-Toolkit 19](#_Toc122376428)

[3.3 Реализация серверной части 21](#_Toc122376429)

[3.4 Выводы по разделу 23](#_Toc122376432)

[4 Тестирование веб-сайта 24](#_Toc122376433)

[4.1 Адаптивный дизайн веб-сайта 25](#_Toc122376434)

[4.2 Кроссбраузерность веб-сайта 25](#_Toc122376435)

[4.3 тестирование 27](#_Toc122376436)

[4.4 тестирование компонентов 27](#_Toc122376436)

[4.5 Выводы по разделу 27](#_Toc122376437)

[5 Руководство пользователя 28](#_Toc122376438)

[Заключение 30](#_Toc122376439)

[Список использованных источников 31](#_Toc122376439)

[Приложение А 32](#_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.

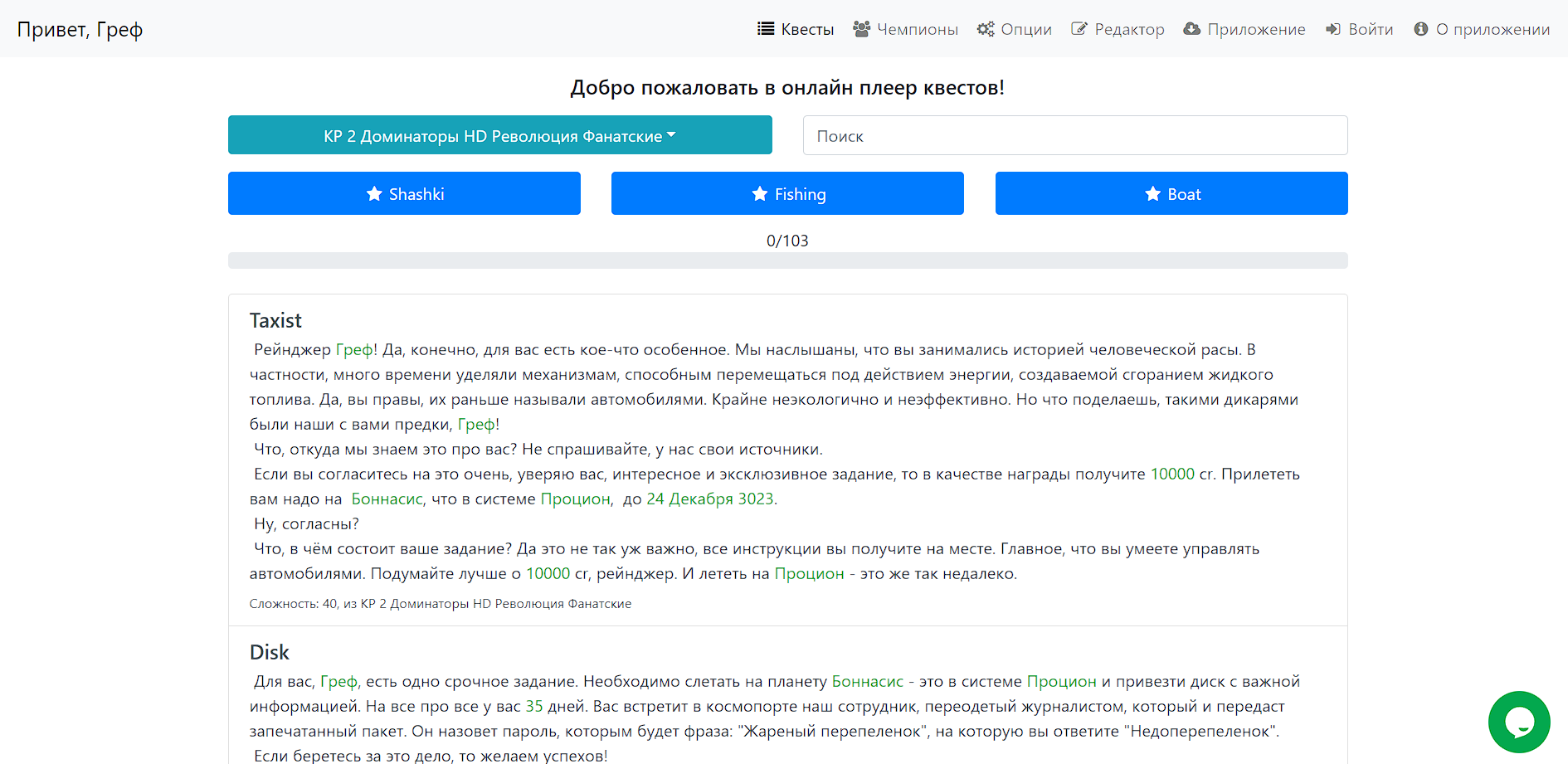


Рисунок 1.1 – Главная страница сайта “плеер квестов из игры Космические Рейнджеры”



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.

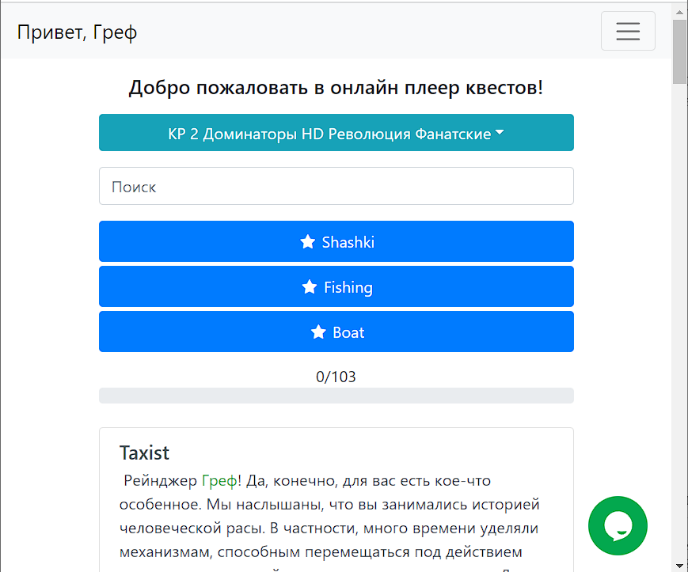


Рисунок 1.3 – Мобильная версия сайта “плеер квестов из игры Космические Рейнджеры”

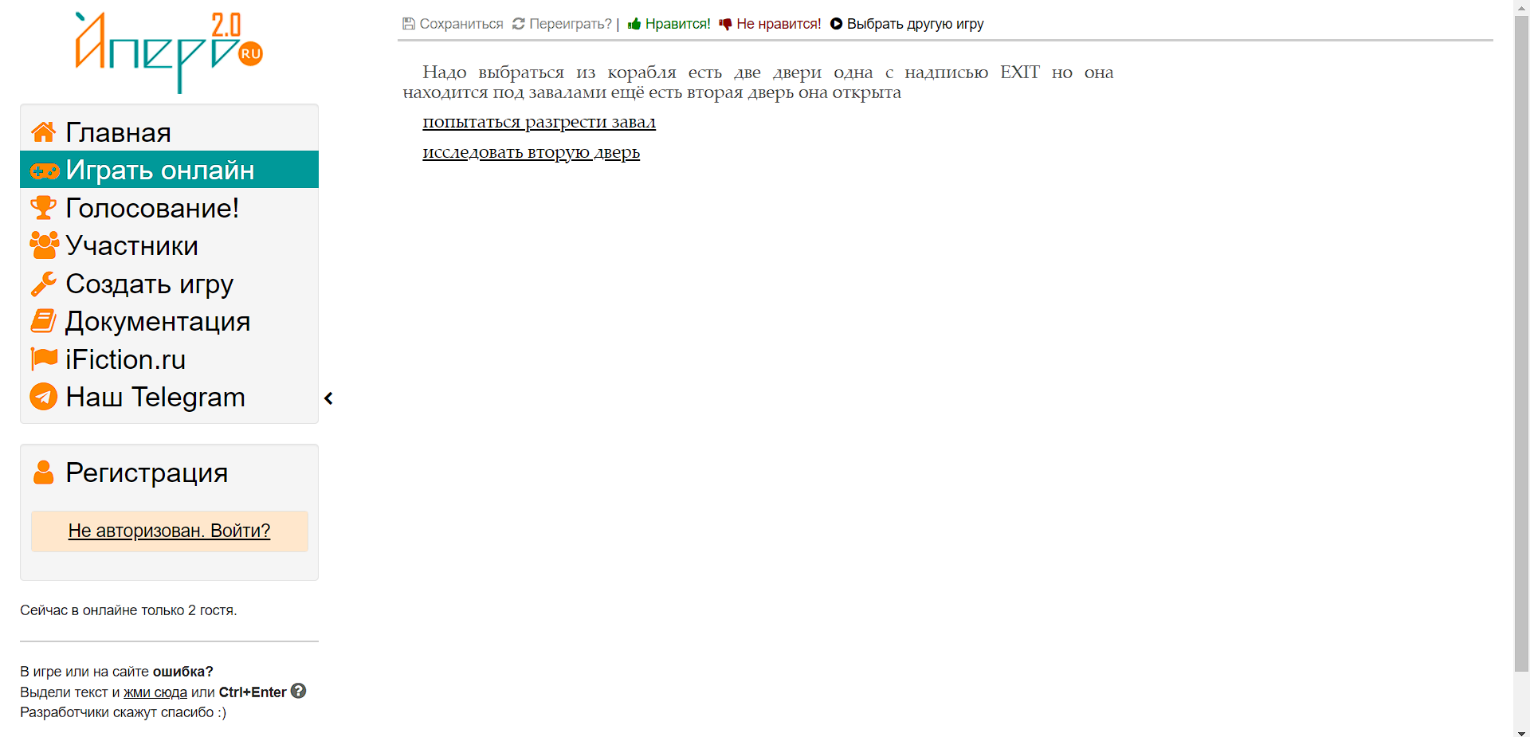


Рисунок 1.4 – Мобильная версия квестовой страницы “плеер квестов из игры Космические Рейнджеры”

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. Его квестовая страница показана на рисунке 1.5.

Рисунок 1.5 – 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. показана на рисунке 1.6 квестовая страница сайта.

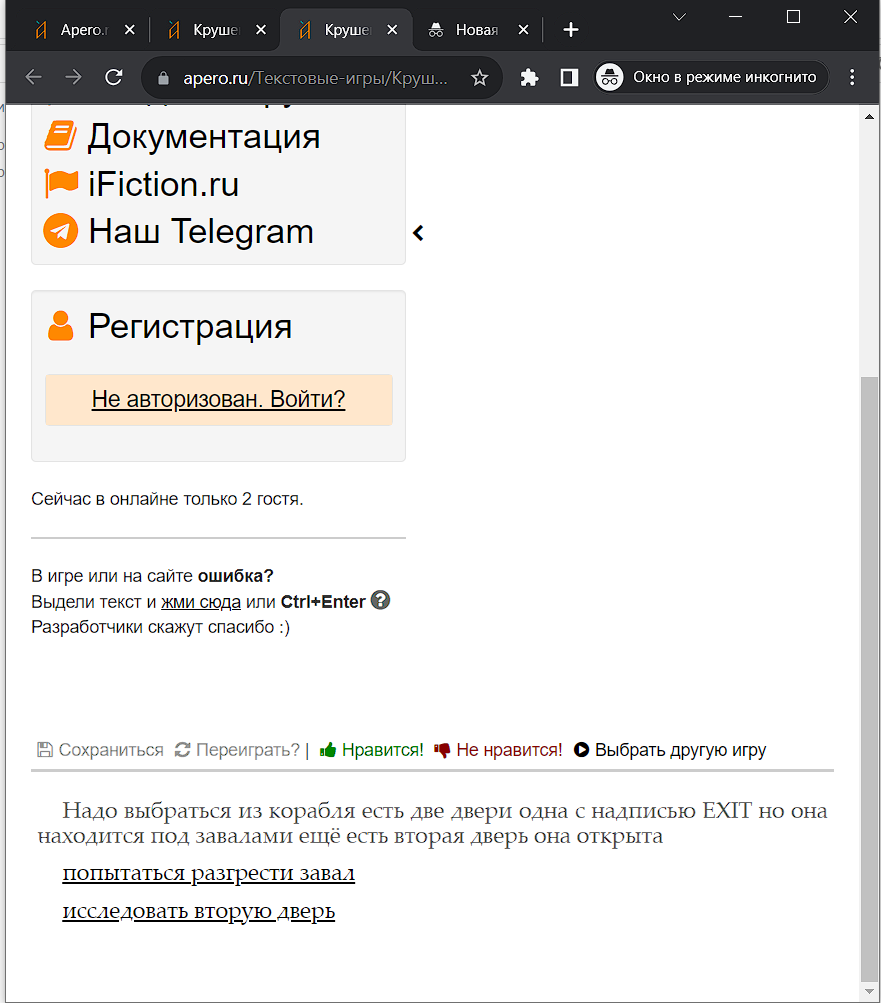


Рисунок 1.6 – Мобильная версия сайта

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 1.7 представлено оформление сайта.

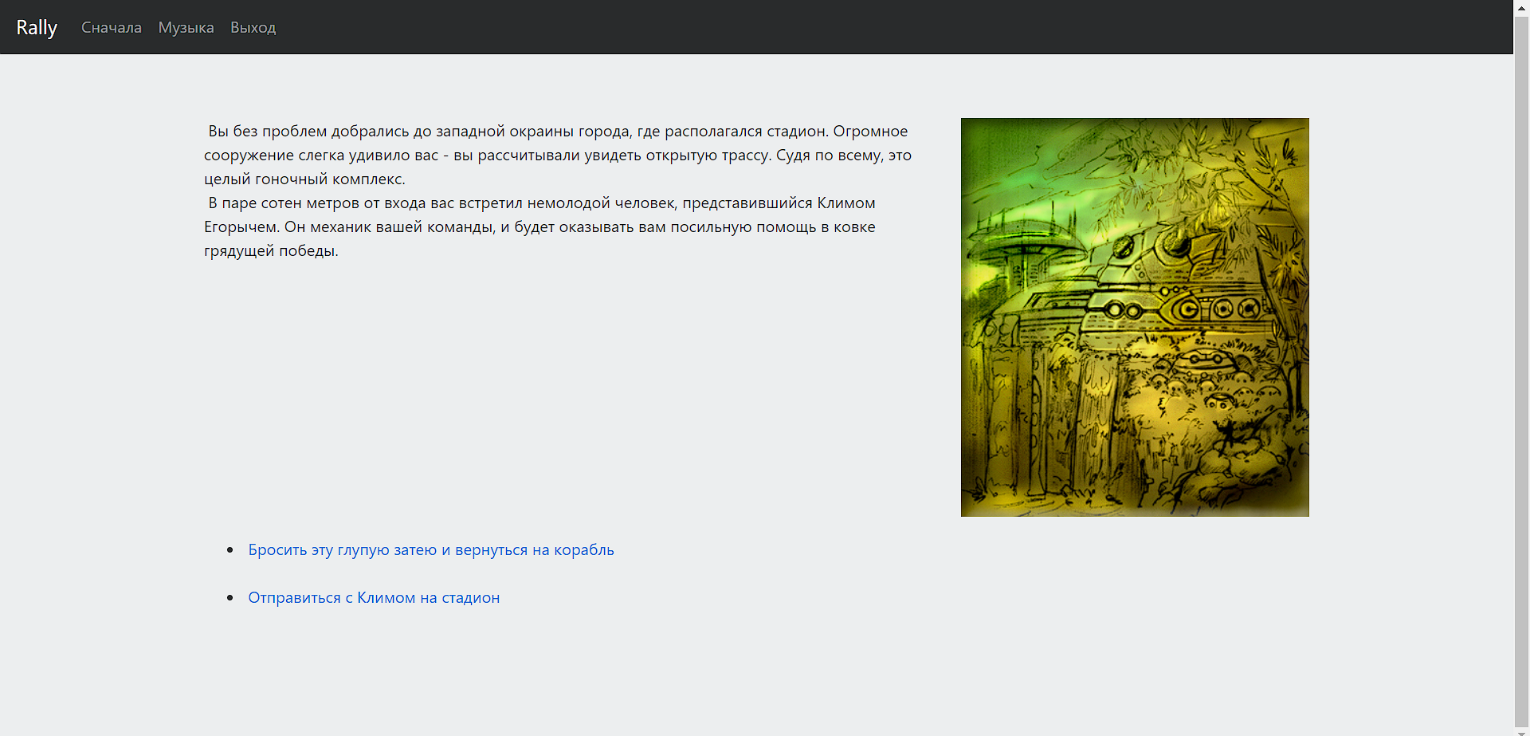


Рисунок 1.7 – страница сайта с квестом

Также данный сайт, адаптирован под мобильные устройства. На рисунке 1.8 представлено оформление на мобильных устройствах. Хорошо реализована адаптивность, но размер картинки сильно скачет при масштабировании.

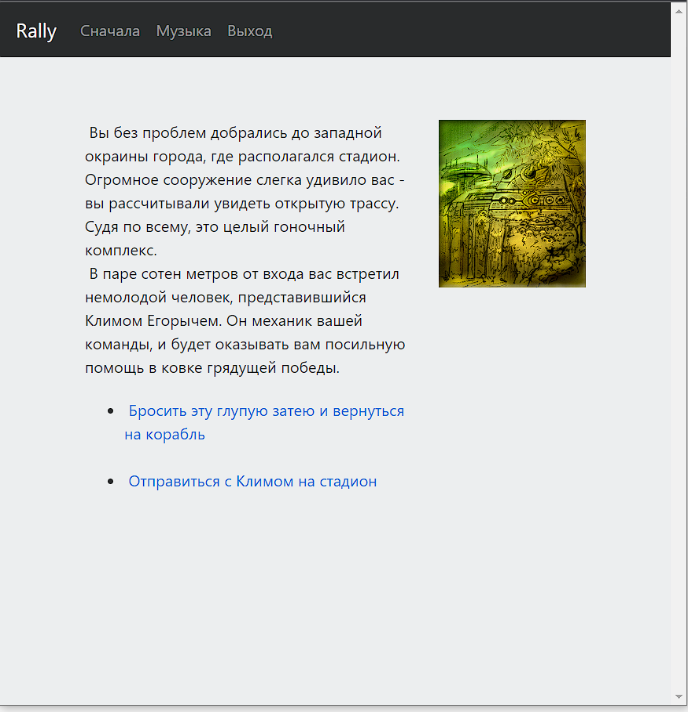


Рисунок 1.8 – Мобильная страницы версия сайта с квестом

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 – бежевого и чёрного. Бежевый цвет будет использоваться для фона, а чёрный и белый для текста и некоторых элементов дизайна. Эти цвета были подобраны так, потому что они будут приятны для глаз и это цвета, благодаря этому пользователям сразу понравится 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. Ниже на Листинге 3.1 показана Структура компонента интерпретатора Quest\_table внутри него мы совмещаем рамку изображения, в которую нужно подать ссылку на картинку или видеофайл, div с текстом-описанием и варианты действий, также в данном компоненте находятся методы для выполнения различного функционала(изменение переменной, переход из “комнаты: в “комнату”, условные операторы, математические и строковые операторы. В приложении A указаны методы всех операций текстового квеста

|  |
| --- |
| class Quest\_Table extends React.Component  {    constructor(props)    {      super(props)      this.props ={}      this.state = {color: "red",current\_room:myquest.start\_from};      this.transit=this.transit.bind(this)      this.post\_transit=this.post\_transit.bind(this)      this.ifswitch=this.ifswitch.bind(this)      this.qswitch\_f=this.qswitch\_f.bind(this)    }  render()  {    const textbuffer=this.instex(myquest.rooms[this.state.current\_room]["text"])      return (<div id={main.quest\_app}>      <PicFrame source={myquest.rooms[this.state.current\_room].image} />      <div id={main.storybox}>{textbuffer} </div>      <div id={main.option\_bar}>      <Quest\_Options  inserter={this.instex} options={myquest.rooms[this.state.current\_room]["options"]}      room={this.state.current\_room}      trans\_f={this.transit}></Quest\_Options>      </div>      <Audio\_player paused={!this.props.unmuted}/>      </div>    )  }  /\*методы \*/  } |

Листинг 3.1 –Структура компонента интерпретатора

Самым главным компонентом будет с названием файла и функции App.js. 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.

|  |
| --- |
| function App()  {      return (    <BrowserRouter>    <Routes>      <Route path='' element={<Selector/>}></Route>      <Route path='\*' element={<Selector/>}></Route>      <Route path='faq' element={<Faq/>}></Route>      <Route path='about' element={<AboutUs/>}/>      <Route path='error' element={<ErrorPage/>}/>      <Route path='game' element={<Routed\_App/>}></Route>    </Routes>    </BrowserRouter>    )  } |

Листинг 3.2 – Структура маршрутов

Следующим компонентом будет компонент Selector, его код указан в листинге 3.3. Он содержит в себе карточки квестов, его задача – загрузить список квестов создать карточки, и заполнить их и вставить их в свою структуру.

|  |
| --- |
| function Selector()  {      const [dirset,getdir]=useState("")      const [dirTrig,trigSet]=useState(true)      const [serv\_ok,setStatus]=useState(true)      if(dirTrig){      let respon =  fetch("http://localhost:3300/list")      .then((response) => response.json())      .then((response) => {              getdir(response);              setStatus(true)              trigSet(false)//resp\_dat =response//this.setState({isLoaded: true});      }).catch((err)=> setStatus(false) )      trigSet(false)      }      let select\_buff=[]      let keynum=0      for(let file of dirset)      {          console.log(file)          select\_buff.push(<Selection\_section key={keynum++} desc\_text={test} header={file.name.substring(0,file.name.lastIndexOf(".json"))}/>)      }      console.log(serv\_ok)      if(serv\_ok){      return (( <div >          {<PageBar/>}          {select\_buff}      </div>))}      else{return(<ErrorPage errorCode={500}/>)}    } |

Листинг 3.3 – Структура селектора

Следующим компонентом будет Selection\_section компонент. Он представляет собой карточку квеста, каждая карточка содержит в себе название квеста, его описание и кнопку “начать”. После нажатия на кнопку начать пользователь переводится на квестовую страницу. Код компонента указан в листинге 3.4.

|  |
| --- |
| function Selection\_section( props )  { const [resp\_dat,getResp]=useState("")      const navi =useNavigate()      let respon =  fetch("http://localhost:3300/preload?q="+props.header)      .then((response) => response.text())      .then((response) => {              console.log(response)              getResp(response)    })      return(<div className={style.Selection\_section}>      <div><h1>{props.header + " "}      </h1>{resp\_dat}{}</div> <button onClick={()=>{navi("game"+"?"+props.header)}}>{butt\_text}</button>      </div>)  } |

Листинг 3.4 – Структура кода компонента Selection\_section

Для создания складных текстовых рамок для faq был создан компонент Foldable\_Tframe, его код указан в листинге 3.5.Для уменьшения объема кода был предварительно создан компонент Textframe, в котором был реализован необходимый функционал для двух видов рамок, а Foldable\_Tframe оборачивает Textframe.

|  |
| --- |
| class Foldable\_Tframe extends Component  {       constructor(props)       {            super(props)            this.state={                 folded:true            }            this.toggleSW=this.toggleSW.bind(this)       }       render()       {            return <Textframe header={this.props.header} text={this.props.text} swFunc={this.toggleSW} folded={this.state.folded}/>         }       toggleSW(event)       {            let buff=this.state.folded            this.setState({folded:!buff})       }  } |

Листинг 3.5 – код компонента Foldable\_Tframe

Последний компонент - Textframe. Он предназначен для создания несворачиваемых рамок для текста, а также для реализации компонента Foldable\_Tframe, внутри него был добавлен функционал дописывания информации об авторе в конце отдельным блоком. Код компонента указан в листинге 3.6.

|  |
| --- |
| export class Textframe extends Component  {       constructor(props)       {            super(props)       }       render()       {              let funcbuffer            if(typeof this.props.swFunc == "function")            {                funcbuffer=this.props.swFunc            }            else{funcbuffer=()=>{}}            if(this.props.folded)            {                 return (                      <div className={style.main\_part}>                      <div className={style.header} onClick={ funcbuffer}>                      {this.props.header}                      </div>                      </div>                 )            }            else{                 let copymark=""                 if(this.props.copy){copymark="Создано Савко А.Д ИСиТ 2 БГТУ 2023"}                   return (                      <div className={style.main\_part}>                      <div className={style.header} onClick={funcbuffer}>                      {this.props.header}                      </div>                      <div className={style.text\_frame}>                      {this.props.text}                      <p>{copymark}</p>                      </div>                      </div>                 )            }       }  } |

Листинг 3.6 – Структура компонента Textframe

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

Управление кнопкой переключения ссылок в Navbar-е было реализовано через redux. Ниже, в листинге 3.7 описан его код.

|  |
| --- |
| function PageBar(props)  {  let count =useSelector((state) => state.value)  const dispatch = useDispatch()  const move = useNavigate()  const [opt\_counter,setOpt]=useState(0)  const move\_map =[{name:"qu\_side",path:"/"},{name:"о нас",path:"/about"},{name:"faq",path:"/faq"}]  return(  <div className={style.pagebar}>  <span  onClick={()=>{move(move\_map[count].path)}}>{move\_map[count].name}</span>  <span className={""}  onClick={              ()=>{dispatch(increment())}          }      >{">"}</span>  </div>)  } |

Листинг 3.7 – Код Navbar-а

Благодаря CreateSlice был реализован редьюсер для получения данных о нажатии переключателя. В CreateSlice будет находиться имя редьюсера, начальное состояние, редьюсеры для изменения состояния. 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 counterSlice = createSlice({      name: 'counter',      initialState,      reducers: {        increment: (state) => {          state.value = (state.value + 1) % 3;        },        }    }); |

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 initialState = {    value: 0,  };    export const Queststore = configureStore  ({      reducer: counterSlice.reducer    }); |

Листинг 3.9 – Структура кода Store

В этом разделе были рассмотрены основные 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.

## 3.4 Добавление таблиц стилей CSS modules

Для стилизации были использованы CSS modules, которые позволяют решить проблему каскадности и обеспечить простоту различных элементов интерфейса. Код использования CSS modules показан в листинге 3.11.

|  |
| --- |
| .main\_part{width: 100%; background-color: wheat;margin-top: 2em;padding-top: 1em;padding-bottom: 1em;}  .body\_part{width: 100%; background-color: rgb(246, 234, 212);}  .header{width: 90%;margin: auto; background-color: rgba(0, 0, 0, 0.679);color: white;      box-sizing: border-box;font-size: 32px;      min-height: 1.5rem;padding: 1rem;}  .text\_frame{width: 90%;margin: auto; background-color: rgba(25, 25, 25, 0.595);      color: white;min-height: 5em;      padding: 1rem;font-size: 28px;      box-sizing: border-box;} |

Листинг 3.11 – 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 квестовая страница занимает всю ширину экрана, а не квадратный сектор, как в компьютерной версии. На рисунке 4.1 можно увидеть пример адаптивного дизайна для мобильных устройств.

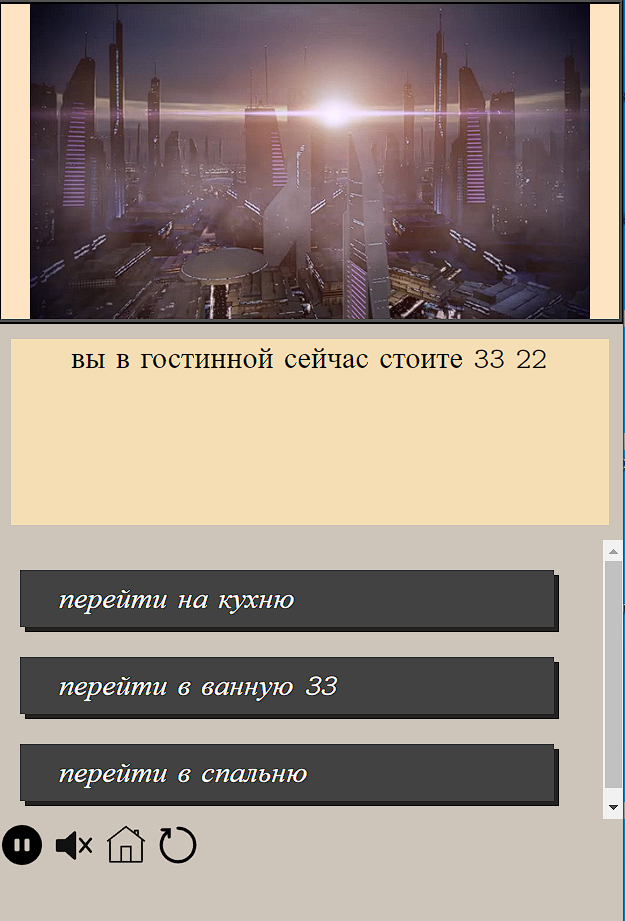


Рисунок 4.1 – Адаптивный квестовой страницы для телефонов

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

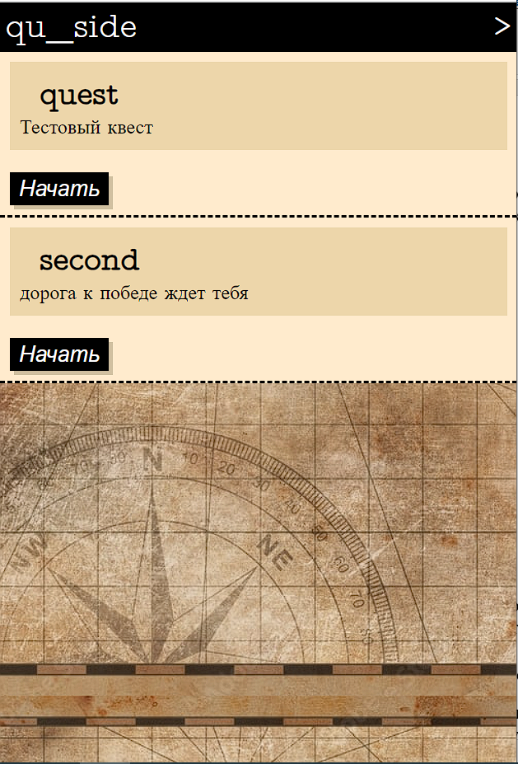
****

Рисунок 4.2 – Блоки с квестами 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. Поскольку все современные браузеры используют как основу движки таких браузеров как Google Chrome и Mozilla Firefox для проверки кроссбраузерности были выбраны именно вышеупомянутые два браузера.

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

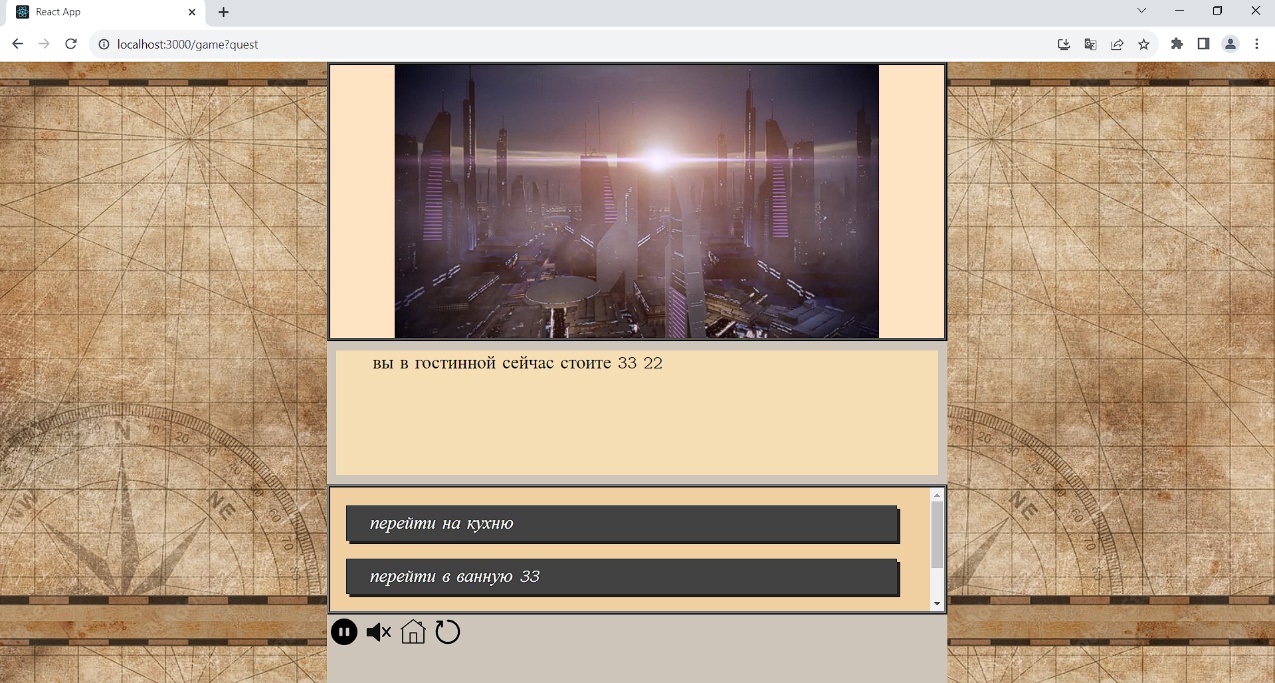


Рисунок 4.3 – Вид веб-страницы в Google Chrome

В Mozilla Firefox сайт показал схожий результат: никаких отклонений не было выявлено, текст не съехал, не произошло его наложения на картинки. На рисунке 4.4 можно увидеть пример веб-страницы в Mozilla Firefox.

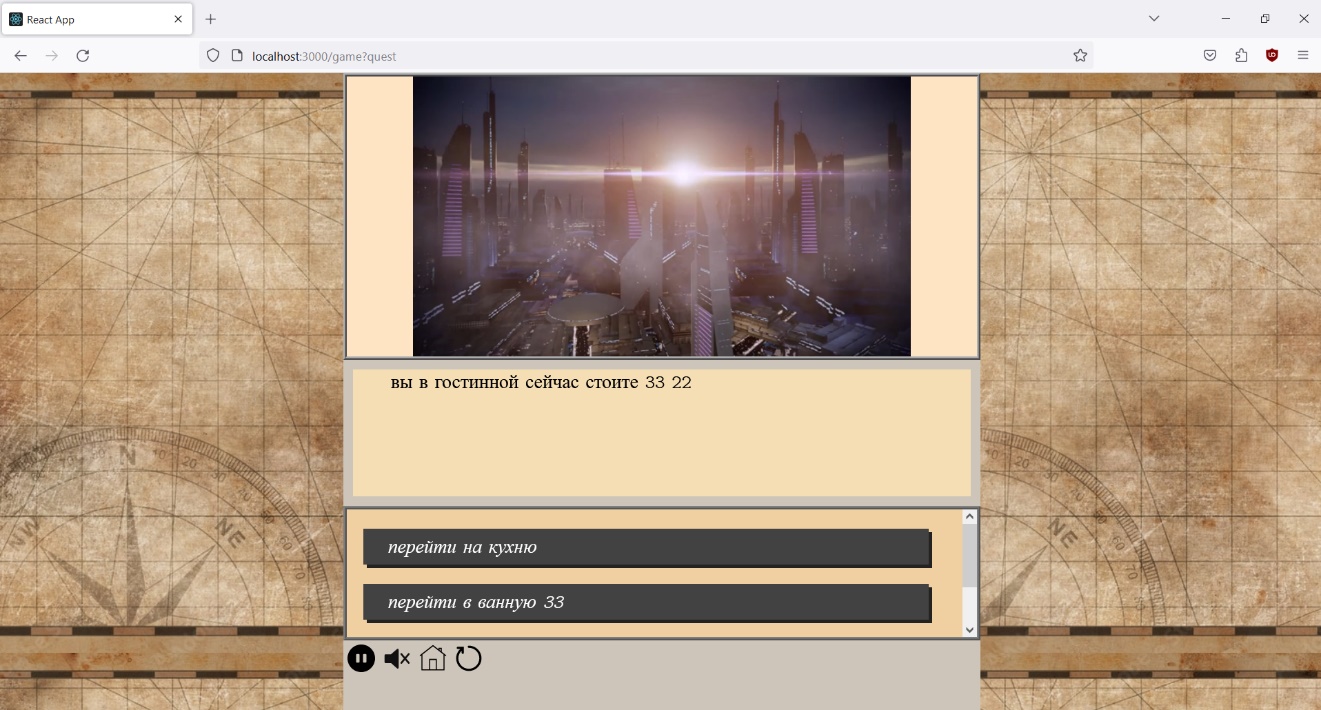


Рис 4.4 – Вид веб-страницы в Mozilla Firefox

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 react js.

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 компьютера появляется поле для обеспечения необходимых размеров интерфейса. В ходе тестирования были проверены все ссылки. Также в ходе тестирования была проверена совместимость с браузерами. (Google Chrome, Mozilla Firefox).

## 4.4 Тестирование компонентов

Для тестирования компонента Pagebar была использована библиотека автоматического тестирования Jest, данная библиотека позволяет осуществить тестирование js кода по определенному сценарию, ниже на листинге 4.5 показан код сценария тестирования компонента Pagebar. В данном сценарии проверяется корректность рендера и наличие текста “qu\_side”.

import PageBar from "../src/pagebar";

test.todo("ffff")

describe("ComponentToTest", () => {

    test("renders component correctly", () => {

      render(<PageBar />);

      expect(screen.getByText("qu\_side")).toBeInTheDocument();

    });

    test("handles click event correctly", () => {

        render(<PageBar />);  });

})

Листинг 4.5 – Сценарий тестирования компонента Pagebar

## 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.

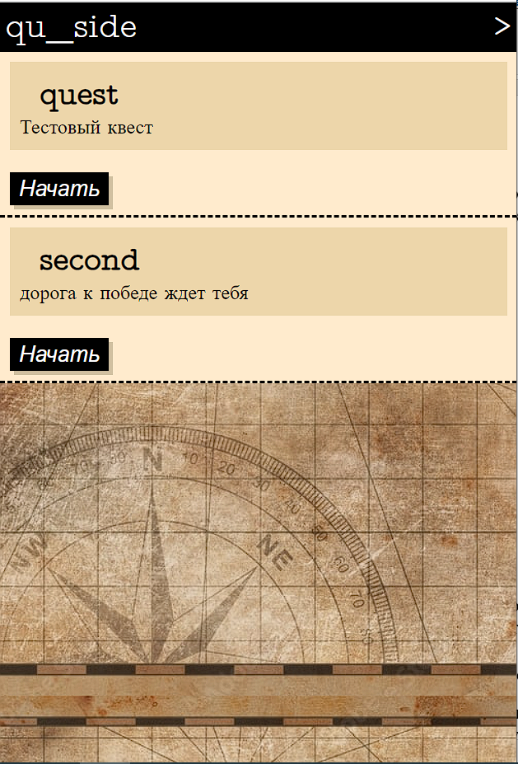
****

Рисунок 5.1 – выбор квеста

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

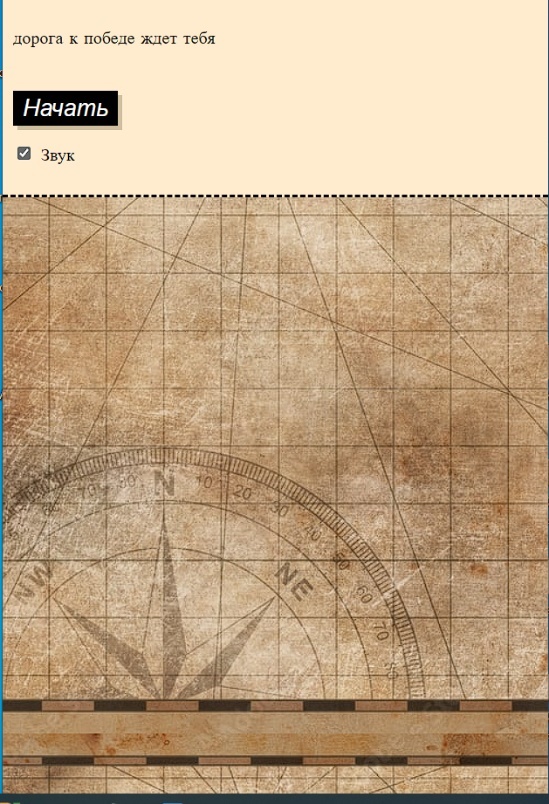


Рисунок 5.2 – Сценарий тестирования компонента Pagebar

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

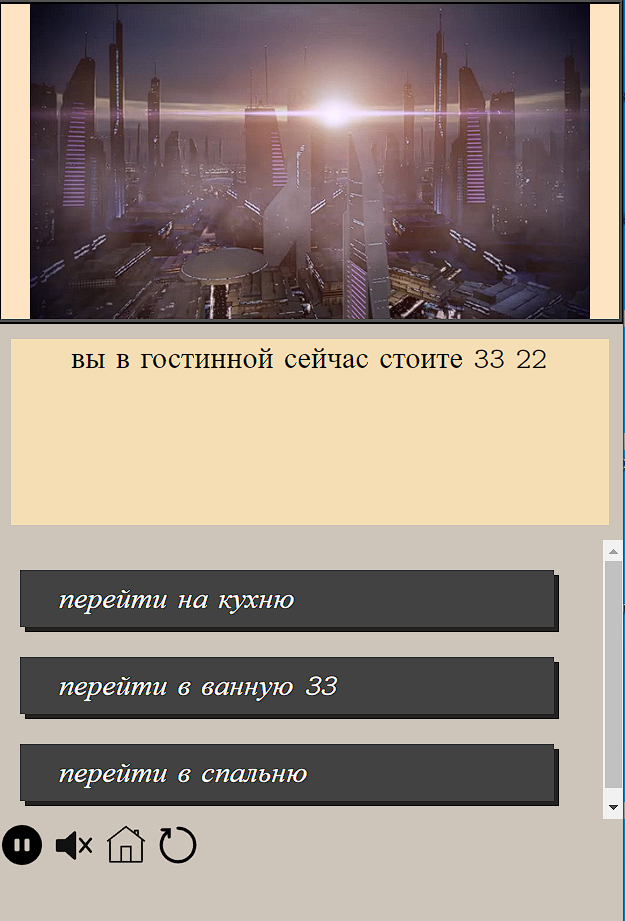


Рисунок 5.3 – Воспроизведение квеста

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. “Плеер квестов из игры Космические Рейнджеры” [Электронный ресурс]. – Режим доступа: https://spacerangers.gitlab.io/

. – Дата обращения 19.11.2023.

1. Apero [Электронный ресурс]. – Режим доступа: https://apero.ru/. – Дата обращения 21.11.2023.
2. “Текстовые квесты онлайн – космические рейнджеры” [Электронный ресурс]. – Режим доступа: https://vasiliy0.gitlab.io/. – Дата обращения 21.11.2022.
3. Документация по React JS [Электронный ресурс]. - Режим доступа: https://reactjs.org/. – Дата обращения 21.11.2023.
4. Документация по Redux/Redux-tolkit [Электронный ресурс]. - Режим доступа: https://redux-toolkit.js.org/ . – Дата обращения 21.11.2023.
5. Практическое руководство по использованию CSS Modules [Электронный ресурс]. - Режим доступа: https://habr.com/ru/articles/335244/ – Дата обращения 21.11.2023.

# Приложение А

instex(replaceble,max\_recursor=3,trim=1)

{

  const capture\_regex=/<\$ins>(.\*?)<\$ins>/gi

  let recursor = 0

    while(replaceble.match(capture\_regex) && recursor<max\_recursor){

    const matchbuff=[...replaceble.matchAll(capture\_regex)]

    for(const key of matchbuff){

    const ins\_regex=new RegExp(`<\\$ins>${key[1]}<\\$ins>`,"gi");

    replaceble= replaceble.replace(ins\_regex,myquest.variables.visible[key[1]])}

    recursor++

  }

  if(trim)

  {

    replaceble=replaceble.replace(capture\_regex,"")

  }

  return replaceble

}

qswitch\_f(switch\_obj)

{

  for(let sw\_obj in switch\_obj){

  let choise =switch\_obj[sw\_obj].cases[myquest.variables.visible[switch\_obj[sw\_obj].cell]]

  console.log(switch\_obj[sw\_obj])

  if(!choise)

  {

    this.post\_transit(switch\_obj[sw\_obj].qdefault)

  }

  else

  {

    this.post\_transit(switch\_obj[sw\_obj].cases[myquest.variables.visible[switch\_obj[sw\_obj].cell]])

  }

  }

}

math\_func(oper\_obj)

{

let arr\_iter=oper\_obj.length

if(Array.isArray(oper\_obj)){}

//else{}

for(let it\_counter=0; it\_counter<arr\_iter;it\_counter++){

let varname\_buff\_a=null

if(typeof oper\_obj[it\_counter].va =="string")

{varname\_buff\_a=(oper\_obj[it\_counter].va.replace("<$ins>","")) }

console.log(oper\_obj[it\_counter],"im hungry",varname\_buff\_a)

oper\_obj[it\_counter].va=Number(this.get\_var(oper\_obj[it\_counter].va))

oper\_obj[it\_counter].vb=Number(this.get\_var(oper\_obj[it\_counter].vb))

let buffer=0

switch(oper\_obj[it\_counter].operator)

{

case "+":{buffer=oper\_obj[it\_counter].va+oper\_obj[it\_counter].vb;break}

case "-":{buffer=oper\_obj[it\_counter].va-oper\_obj[it\_counter].vb;break}

case "/":{buffer=oper\_obj[it\_counter].va/oper\_obj[it\_counter].vb;break}

case "\*":{buffer=oper\_obj[it\_counter].va\*oper\_obj[it\_counter].vb;break}

case "^":{buffer=oper\_obj[it\_counter].va\*\*oper\_obj[it\_counter].vb;break}

case "ln":{ buffer=Math.log(oper\_obj[it\_counter].va);break}

case "log":{ buffer=Math.log(oper\_obj[it\_counter].va)/Math.log(oper\_obj[it\_counter].vb);break}

}

if(oper\_obj[it\_counter].write\_to){

myquest.variables.visible[oper\_obj[it\_counter].write\_to]=buffer

}

else

{

if(varname\_buff\_a){myquest.variables.visible[varname\_buff\_a]=buffer}

}

}

}

get\_var(forcheck)

{

  if(typeof forcheck==="string"&&forcheck.includes("<$ins>"))

  {

    return(myquest.variables.visible[forcheck.replace("<$ins>","")])

  }

  else{return forcheck}

}

ifswitch(switch\_obj)

{

  for(const if\_obj in switch\_obj){

  let choise =false

  console.log(switch\_obj[if\_obj].a\_val,switch\_obj[if\_obj].b\_val)

  switch\_obj[if\_obj].a\_val=this.get\_var(switch\_obj[if\_obj].a\_val)

  switch\_obj[if\_obj].b\_val=this.get\_var(switch\_obj[if\_obj].b\_val)

  switch(switch\_obj[if\_obj].operator)

  {

    case "=" :{choise=switch\_obj[if\_obj].a\_val==switch\_obj[if\_obj].b\_val;break}

    case "<" :{choise=switch\_obj[if\_obj].a\_val<switch\_obj[if\_obj].b\_val;break}

    case ">" :{choise=switch\_obj[if\_obj].a\_val>switch\_obj[if\_obj].b\_val;break}

    case "!=" :{choise=switch\_obj[if\_obj].a\_val==switch\_obj[if\_obj].b\_val;break}

  }

  console.log(switch\_obj[if\_obj].a\_val,switch\_obj[if\_obj].b\_val)

  if(choise)

  {

    this.post\_transit(switch\_obj[if\_obj].then)

  }

  else{this.post\_transit(switch\_obj[if\_obj].qelse)}

}

}

post\_transit(trans\_map)

{

  console.log(trans\_map)

  if(trans\_map.change)

  {

  let ch\_items=trans\_map.change.length

  for(let iter\_ch=0;iter\_ch<ch\_items;iter\_ch++)

  {

  myquest.variables.visible[trans\_map.change[iter\_ch].cell]=trans\_map.change[iter\_ch].value

  }

  }

  if(trans\_map.mathf)

  {

    console.log(333)

    this.math\_func(trans\_map.mathf)

  }

  if(trans\_map.stringf)

  {

    console.log(333)

    this.string\_work(trans\_map.stringf)

  }

  if(trans\_map.move)

  {

  this.setState({current\_room:trans\_map.move})

  }

  else{ if(trans\_map.ifsw)

  {

    this.ifswitch(trans\_map.ifsw)

  }

  else if(trans\_map.qswictch)

  {

    this.qswitch\_f(trans\_map.qswictch)

  }

}

  console.log(myquest.variables.visible)

}

transit\_loop(trans\_obj)

{

if(trans\_obj.qswitch)

{

  this.qswitch\_f(trans\_obj.qswitch)

}

if(trans\_obj.ifsw)

{

  this.ifswitch(trans\_obj.ifsw)

}

else

{

this.post\_transit(trans\_obj)

}

}

transit(handler,index,room)

{

let commandbuff=myquest.rooms[room].options[index]

if(commandbuff.qswitch)

{

  this.qswitch\_f(commandbuff.qswitch)

}

if(commandbuff.ifsw)

{

  this.ifswitch(commandbuff.ifsw)

}

else

{

this.post\_transit(commandbuff)

}

}

string\_work(string\_obj)

{

  let swork\_arr=string\_obj.length

console.log(string\_obj)

  for(let st\_counter=0;st\_counter<swork\_arr;st\_counter++){ let buffer

  let varname\_buff\_a=null

  if(typeof string\_obj[st\_counter].va =="string")

  {varname\_buff\_a=(string\_obj[st\_counter].va.replace("<$ins>","")) }

  string\_obj[st\_counter].va=(this.get\_var(string\_obj[st\_counter].va))

  string\_obj[st\_counter].vb=(this.get\_var(string\_obj[st\_counter].vb))

  console.log(string\_obj[st\_counter].va)

  switch(string\_obj[st\_counter].operator)

{

  case "+":{buffer=string\_obj[st\_counter].va.concat(string\_obj[st\_counter].vb);break}

  case "\*":{buffer=string\_obj[st\_counter].va.repeat(string\_obj[st\_counter].vb);break}

  case "includes":{buffer=string\_obj[st\_counter].va.includes(string\_obj[st\_counter].vb);break}

  case "replace":{buffer=string\_obj[st\_counter].va.replace(string\_obj[st\_counter].vb,string\_obj[st\_counter].vc);break}

} console.log(buffer)

if(string\_obj[st\_counter].write\_to){

  myquest.variables.visible[string\_obj[st\_counter].write\_to]=buffer

  }

  else

  {

  if(varname\_buff\_a){myquest.variables.visible[varname\_buff\_a]=buffer;console.log(buffer)}

  }

Рисунок 5.3 – Методы компонента Quest\_Table