Частное образовательное учреждение высшего образования

«Казанский инновационный университет имени В. Г. Тимирясова (ИЭУП)»

Факультет менеджмента и инженерного бизнеса

Кафедра компьютерных технологий и техносферной безопасности

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине «Интернет-программирование»**

**на тему: «Разработка информационного ресурса для изучения теории музыки «Musthe»»**

Направление подготовки: 09.03.03 «Прикладная информатика»

**Выполнил:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

обучающийся гр. 1091

очной формы обучения

Кудряшов П.А.

**Руководитель**: \_\_\_\_\_\_\_\_\_\_\_

к.т.н., доцент каф. КТиТБ  
Фасхутдинов Л.М.

**Результат защиты:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Казань, 2022**

Private Educational Institution of Higher Education

“Kazan Innovative University named after V.G. Timiryasov (IEUP)”

Faculty of Management and Engineering Business

Department of Computer Technologies and Technosphere Safety

**TERM PROJECT**

**on the discipline "Internet-programming"**

**on a theme: "Development of the information resource for studying the theory of music "Musthe""**

Field of study: 09.03.03 "Applied Computer Science"

**Done: \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

gr. 1091

full-time education

Kudryashov P.A.

**Supervisor: \_\_\_\_\_\_\_\_\_\_\_**

Candidate of Technical Sciences, assistant professor of the department. CT&TS

Faskhutdinov L.M

**Result of defense:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Kazan, 2022**

**АННОТАЦИЯ**

на курсовой проект

по дисциплине «Интернет-программирование»

|  |
| --- |
| Ф.И.О. студента: Кудряшов Павел Александрович |
| Student’s Name: Kudryashov Pavel Alexandrovich |
| Направление: 09.03.03 «Прикладная информатика» |
| № группы: 1091 |

|  |
| --- |
| Название курсового проекта: «Разработка информационного ресурса для изучения теории музыки «Musthe»» |
| Paper Title: Development of an information resource for the study of music theory “Musthe” |

Аннотация:

|  |
| --- |
| Целью данной работы является Разработка информационного ресурса для изучения теории музыки «Musthe», который позволит публиковать и читать статьи, связанные с теорией музыки.  Для данной работы поставлены такие задачи как:   * Проектирование ресурса * Реализация Frontend части * Реализация Backend части |

Summary:

|  |  |
| --- | --- |
| The purpose of this work is to develop an information resource for the study of music theory "Musthe", which will allow you to publish and read articles related to music theory.  The objectives for this work are as follows:   * Designing the resource * Frontend part implementation * Backend part implementation | |
| Руководитель: доцент кафедры компьютерных технологий и техносферной безопасности Фасхутдинов Л.М. | Перевод проверил: доцент кафедры иностранных языков и переводов Гутарова А.В |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2022 г. | «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2022 г. |

Содержание

[Введение 5](#_Toc102808745)

[1 Проектирование ресурса 6](#_Toc102808746)

[1.1 Проектирование Frontend части 6](#_Toc102808747)

[1.2 Проектирование Backend части 10](#_Toc102808748)

[2 Реализация Frontend части 12](#_Toc102808749)

[2.1 HTML 12](#_Toc102808750)

[2.2 CSS 13](#_Toc102808751)

[2.3 Java Script 17](#_Toc102808752)

[3 Реализация Backend части 19](#_Toc102808753)

[3.1 Handlebars 20](#_Toc102808754)

[3.2 MongoDB 22](#_Toc102808755)

[3.3 Jsonwebtoken 24](#_Toc102808756)

[Заключение 26](#_Toc102808757)

[Список использованных источников 27](#_Toc102808758)

[Приложение А. Опросный лист 28](#_Toc102808759)

[Приложение Б. Исходный код 31](#_Toc102808760)

# Введение

Музыка сопровождает нас всю жизнь везде и повсюду. Но мало кто задумывается, как она создается. Именно для тех, кто заинтересовался ею и создается данный ресурс. Ведь музыка столь же сложна, сколь и увлекательна. Данный сайт позволит людям разбирающимся, помогать тем, кто только делает свои первые шаги. Реализовано это будет засчет возможности добавление статей любым пользователем.

Перед тем, как разработать портал необходимо продумать, из каких элементов она будет состоять и для чего они нужны. Так как сайт будет состоять из статей, необходимо организовать инструмент для написания статьи. Кроме того, будет выделена отдельная страница для статьи о квинтовом круге, который имеет большое значение в теории музыки.

Целью данной работы является разработка информационного ресурса для изучения теории музыки «Musthe».

Для достижения установленной цели, необходимо выполнение следующих задач:

• Проектирование портала

• Реализация Frontend части

• Реализация Backend части

# 1 Проектирование ресурса

## 1.1 Проектирование Frontend части

Перед тем, как приступить к проектированию frontendа, необходимо понять, что это такое.

Frontend — это публичная часть web-приложений (вебсайтов), с которой пользователь может взаимодействовать и контактировать напрямую.

Для разработки будем использовать следующие «языки» : html, css и js.

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов для просмотра веб-страниц в браузере.

CSS (от англ. Cascading Style Sheets «каскадные таблицы стилей») — формальный язык описания внешнего вида документа (веб-страницы).

JS (JavaScript) — это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах.

Теперь перейдем непосредственно к проектированию. Начнем с того, что продумаем дизайн сайта. Основными цветами будут являться: серый, голубой, розовый. Последние 2 цвета будут использоваться в виде градиента в качестве фона сайта. Серый же цвет будет использоваться для текста и элементов навигации. Черный и белый цвета также будут использоваться, однако в редких случаях. В качестве основного шрифта был выбран Poppins. Изображений на сайте практически не будет.

Теперь необходимо создать макет страниц. Вверху страницы будет находиться так называемая «шапка» сайта, которая в нашем случае является и меню навигации. Необходимо также учитывать то, что она будет отличаться, в зависимости от того, вошел человек в аккаунт или нет. В случае, если авторизация не произведена, шапка будет выглядеть следующим образом (рисунок 1).



Рисунок 1 – Макет панели навигации сайта до авторизации

После того, как человек зайдет в аккаунт, «шапка» изменится и будет иметь вид, представленный на рисунке 2. Добавилась кнопка с ссылкой на страницу создания статьи, а также кнопка «Вход» изменилась на «Выход».



Рисунок 2 – Макет панели навигации сайта после авторизации

Все страницы со статьями будут иметь одну и ту же структуру: панель навигации вверху страницы и ниже сама статья. Примерный вид представлен на рисунке 3.



Рисунок 3 – Макет окна статьи

Однако страницы списка статей и написания статьи будут отличаться. Страница с статьями будет содержать список всех имеющихся статей с указанием их названий, небольшого начала и логина автора. Именно для последнего поля необходима регистрация и авторизация на сайте. При нажатии на название, текст или автора статьи, будет открываться окно, о котором говорилось выше. Высота окна будет изменяться в зависимости от количества статей. То как будет выглядеть окно со статьями можно увидеть на рисунке 4.

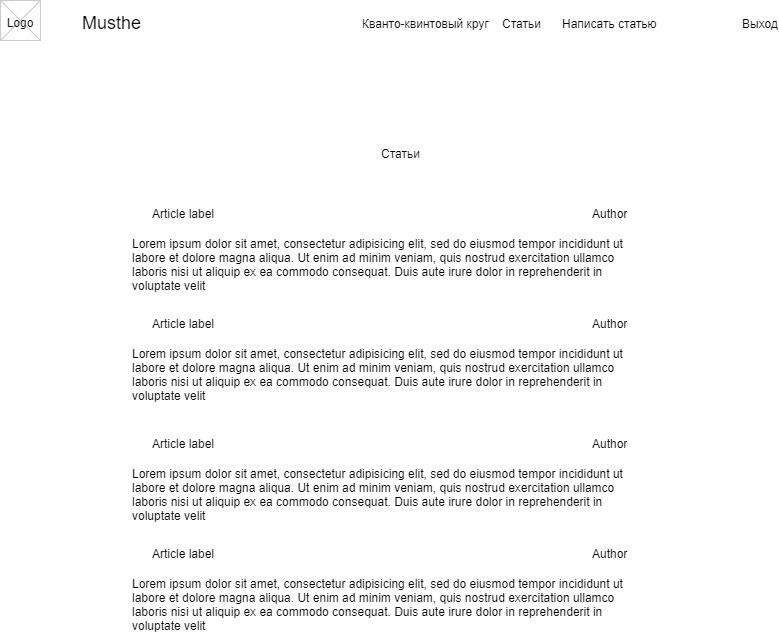


Рисунок 4 – Макет страницы со списком статей

Окно написания статьи будет иметь 2 поля ввода: название статьи и ее содержание, и кнопку сохранения (рисунок 5).

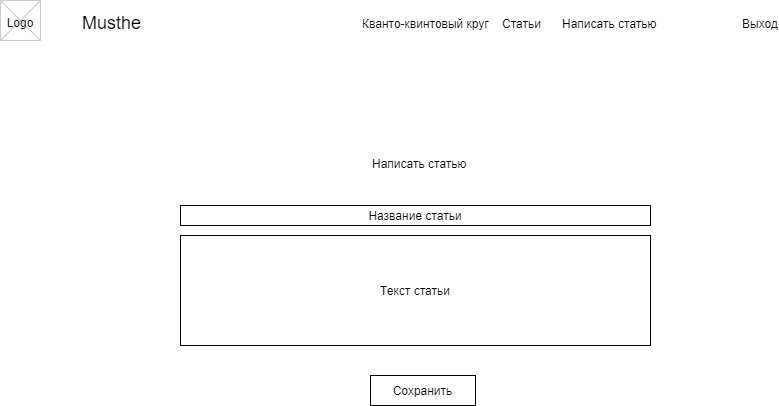


Рисунок 5 – Макет страницы написания статьи

Окно входа будет находится не на отдельной странице, а выводиться поверх текущего окна. Переключение между полями для входа или регистрации будет производиться в этом же окне. Окно в «режиме» входа представлено на рисунке 6.

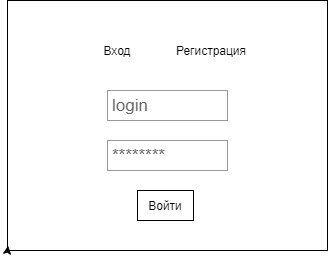


Рисунок 6 – Окно входа

А в «режиме» регистрации – на рисунке 7.



Рисунок 7 – Окно регистрации

## 1.2 Проектирование Backend части

Теперь приступим к планированию backend. И, как и прежде, разберемся с тем, что это такое.

Backend – программно-аппаратная часть сервиса, отвечающая за функционирование его внутренней части. То есть, это «мозги» всего приложения. На стороне backendа также находится и база данных.

Базы данных (БД) – это структурированная информация, хранящаяся на информационных носителях.

Для разработки backend части в качестве серверного языка будем использовать Node JS, а базой данных послужит MongoDB Atlas. Для взаимодействия с БД будет использоваться пакет mongoose.

Так как нам необходимо хранить статьи и информацию о пользователях, нужно создать две соответствующие таблицы. В базе данных MongoDB таблицы называются коллекциями, а записи – документами.

Коллекция «Пользователи» будет состоять из полей:

* Login, с типом string; поле уникальное, обязательное
* Email, с типом string; поле обязательное
* Password, с типом string; поле обязательное

Коллекция «Статьи» будет состоять из полей:

* Label, с типом string; поле обязательное
* Content, с типом string; поле обязательное
* Author, с типом string; ссылается на коллекцию «Пользователи»

В итоге база данных будет иметь следующий вид (рисунок 8).

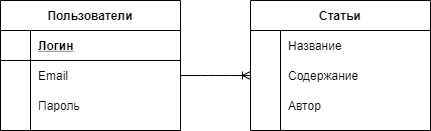


Рисунок 8 – Схема базы данных

Для запуска веб сервера будет использоваться пакет-фреймворк Express.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

На стороне backendа также будет проходить процесс регистрации и авторизации пользователя. Вход будет сопровождаться созданием токена.

Токен – средство идентификации пользователя или отдельного сеанса работы в компьютерных сетях и приложениях.

Для реализации создания и проверки токена будет использоваться пакет jsonwebtoken. Токен передается сервером клиенту после успешной авторизации, а после отправляется с каждым запросом с клиента на сервер, где проверяется на подлинность. Для передачи и хранения токена буду использоваться cookie.

Ку́ки (англ. cookie, букв. — «печенье») — небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя.

# 2 Реализация Frontend части

## 2.1 HTML

Как говорилось выше, для создания клиентской части будет использоваться язык разметки html. Его структурными компонентами являются тег – элемент языка разметки гипертекста. Практически все теги должны закрываться.

В теге <head> хранятся настройки страницы. Здесь указывается основной язык сайта, прикрепляется или вводится стиль, написанный на css, прикрепляются шрифты. Для разрабатываемого сайта тег <head> выглядит так (листинг 1)

Листинг 1 – Тег head

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{{title}}</title>

    <link rel="icon" href="./logo\_white.png" type="image/x-icon">

    <link rel="stylesheet" href="./style.css">

    <link rel="stylesheet" href="./stylePages.css">

    <link href="https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;1,300;1,600&display=swap" rel="stylesheet">

</head>

В теге <body> создается разметка для страницы. Основными тегами являются: div, a (для ссылок), form (для форм), button (для кнопок) и т.д. Внутри тега также указываются настройки для этого конкретного тега: класс, id, стиль, источник и т.д. Ниже приведен листинг 2 фрагмента разметки навигационного меню сайта.

Листинг 2 – Часть html кода навигационного меню

<section class="headder" id="headder">

    <div class="header\_flex">

        <a href="#headder">

            <img class="head\_logo" src="./logo\_white.png" alt="#">

        </a>

        <a href="/">

            <div class="header\_text">

                <h1 class="header\_title">Musthe</h1>

                <h2 class="header\_desc" style="font-family: 'Poppins', sans-serif;">Musica pro omnibus</h2>

            </div>

        </a>

        <a href="/ring">

            <div class="buttons, ring" >Кварто-квинтовый круг</div>

        </a>

        {{#if isArticles}}

        <a href="/articles">

            <div class="currentPage, buttons, articles">Статьи</div>

        </a>

        {{else}}

        <a href="/articles">

            <div class="buttons, articles">Статьи</div>

        </a>

        {{/if}}

        <div class="buttons, signup">Войти</div>

    </div>

</section>

Очень важным является тег <form>, так как с помощью него происходит отправка вводимых пользователем данных на сервер. У этого тега есть атрибуты method и action, которые будут использоваться при создании сайта.

В самом низу, перед закрытием тега <body>, в теге <script> указывается ссылка на JS файл.

## 2.2 CSS

После того, как мы создали разметку для сайта, необходимо ее стилизовать. Это можно сделать как в самом html файле внутри тега <style>, так и отдельно в файле с расширением «.css», а потом прикрепить его.

В CSS настраиваются стили для конкретного тега html на основе его класса или id. Если это класс, то перед названием пишется «.» а после в «{}» указываются настройки стиля. Если же обращение по id, то перед этим ставится «#».

Настроек стиля огромное количество: от изменения цвета объекта (color) до преобразования объекта при наведении на него мыши (hover). Последнее достигается засчет псевдокласса hover. А вот как стилизовалось меню навигации (листинг 3).

Листинг 3 – Стилизация навгационного меню

.headder {

    background-image: linear-gradient(90deg, powderblue, pink);

    text-align: center;

    color: rgb(90, 90, 90);

    margin: 0;

    padding: 1rem 2rem;

    height: 5em;

    position: relative;

    overflow: hidden;

}

.header\_flex {

    display: flex;

    justify-content: space-between;

    z-index: 100;

    position: relative;

}

.head\_logo {

    margin-top: 0.5em;

    height: 2.5rem;

    width: 2.5rem;

    position: fixed;

}

.header\_title {

    margin: 0;

    padding: 0;

    text-transform: uppercase;

    font-weight: 600;

    font-style: italic;

    font-size: 2.5rem;

    color: rgb(90, 90, 90);

}

.info\_button, .ring, .articles, .writeArticle, .signup, .favorites {

    padding: 0.5em;

    margin-right: 1rem;

    cursor: pointer;

    color: rgb(90, 90, 90);

    text-decoration: none;

    font-size: 1.4em;

    height: 2em;

}

.info\_button:hover, .signup:hover, .ring:hover, .articles:hover, .writeArticle:hover, .favorites:hover {

   /\* background-color: powderblue; \*/

   /\* transition-duration: 1s;  \*/

   border-bottom-style: solid;

   border-bottom-color: rgb(90, 90, 90);

}

.header\_desc {

    color: rgb(99, 99, 99);

    margin: 0 auto;

    padding: 0em 3em;

    font-size: 1em;

    text-align: center;

    font-weight: 400;

    font-family: 'Poppins', sans-serif;

}

В итоге, после применения различных возможностей CSS, главная страница стала выглядеть следующим образом (рисунок 9).

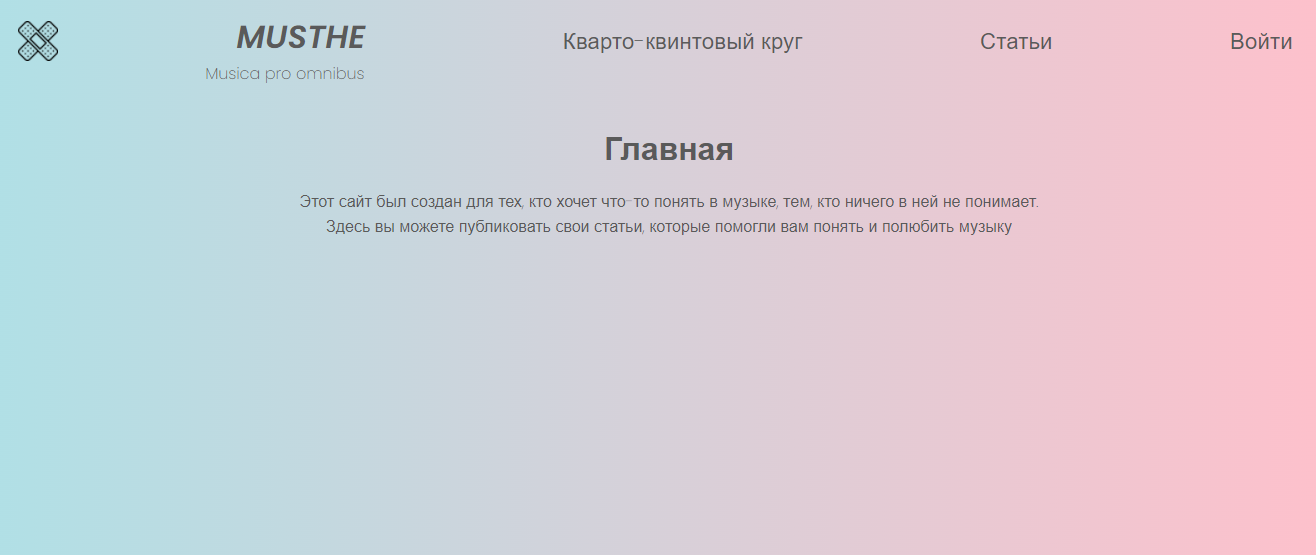


Рисунок 9 – Главная страница

Реализация страницы написания статей представлено на рисунке 10.

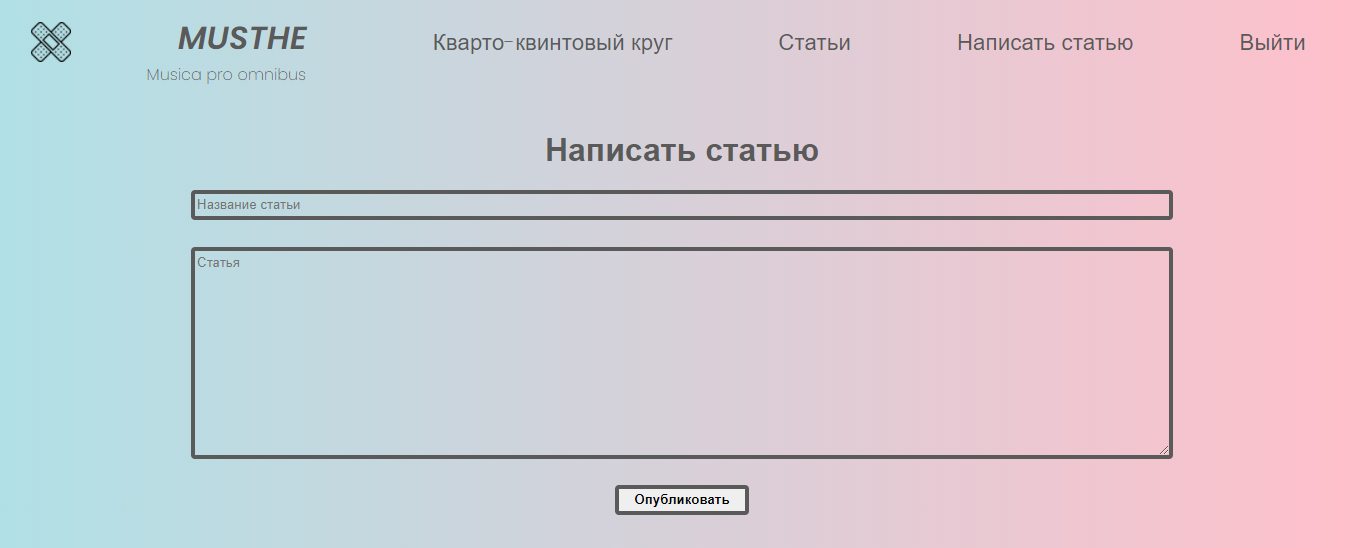


Рисунок 10 -Страница «Написать статью»

А окно входа стало выглядеть так (рисунок 11).

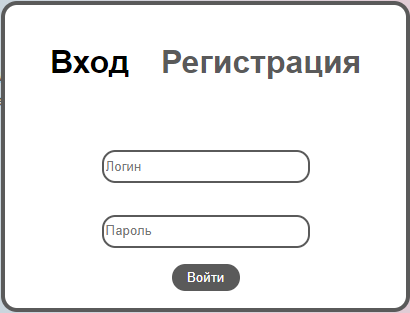


Рисунок 11 – Окно входа

Ну и окно со статьей о кварто-квинтовом круге представлено на рисунке 12.



Рисунок 12 – Кварто-квантовый круг

## 2.3 Java Script

JS необходим для создания каких-либо действующих элементов. С его помощью создаются hamburger меню, слайдеры картинок – все то, что как-то двигается. Для текущего сайта JS нужен для отображения окна «Вход» и его деактивации.

Как и CSS, код можно писать как внутри файла html, так и отдельно, после прикрепив его. В данной работе будет использоваться второй вариант. Для этого самом конце напишем следующее (листинг 4)

Листинг 4 – Подключение Java Script файла

<script src="./script.js"></script>

После этого можно приступать к созданию скрипта. Перед тем, как взаимодействовать с каким-либо объектом html, его нужно выбрать. Для этого, в случае поиска по классу необходимо воспользоваться командой «document.querySelector(‘название класса’)», а в случае id - @document.getElementById(‘название id’).

И уже потом можно с этими объектами взаимодействовать. Так как для сайта требовалось не так уж и много скрипта, код получился небольшим (листинг 5).

Листинг 5 – Java Script код

const initiate\_entrance = document.querySelector('.signup'),

        entrance\_window = document.querySelector('.signup\_window\_hidden'),

        signin\_switch = document.querySelector('.win\_signin'),

        signup\_switch = document.querySelector('.win\_signup'),

        signin\_form = document.querySelector('.signin\_form'),

        signup\_form = document.querySelector('.signup\_form');

initiate\_entrance.addEventListener('click',() => {

    entrance\_window.classList.toggle('signup\_window\_showed')

})

signup\_switch.addEventListener('click', ()=> {

    signin\_form.hidden = true;

    signup\_form.hidden = false;

    signin\_switch.style.color = "rgb(90, 90, 90)";

    signup\_switch.style.color = "black";

})

signin\_switch.addEventListener('click', ()=> {

    signup\_form.hidden = true;

    signin\_form.hidden = false;

    signup\_switch.style.color = "rgb(90, 90, 90)";

    signin\_switch.style.color = "black";

})

Таким образом, после создания JS, frontend часть разработки подходит к концу. Клиентская часть очень важна, так как именно от нее зависит, понравится ли сайт пользователю, или нет. Привлекательность может достигаться засчет использования различных возможностей кастомизации, предоставляемых CSS. Или сайт может завлечь публику своей технологичностью и удобством, которые можно обеспечить благодаря Java Script. Но все это является обычной пустышкой без серверной части. Вся информация, которая имеется на сайтах, хранится в базе данных, откуда она отправляется клиенту с помощью backend.

# 3 Реализация Backend части

Как говорилось ранее, в качестве серверного языка был выбран Node.js. Именно на нем будет создаваться вся бизнес-логика ресурса.

Для начала необходимо запустить веб сервер. На нашем сайте им будет являться Express. Параллельно с запуском сервера подключимся к базе данных используя атрибут пакета mongoose – connect. Листинг всего данного процесса представлен ниже (листинг 6).

Листинг 6 – Запуск сервера и подключение к БД

const express = require('express');

const mongoose = require('mongoose');

const app = express();

const host = '127.0.0.1';

const port = process.env.PORT || 7000;

const start = async () => {

  try {

    await mongoose.connect('mongodb+srv://<login>:<password>@cluster0.gw5os.mongodb.net/coursework'); //подключаемся к бд

    //запускаем сервер

    app.listen(port, host , () => console.log(`Server listens http://${host}:${port}`))

  }catch(er) {

    console.log(er)

  }

}

start()

Связь между клиентом и сервером осуществляется с помощью запросов. В данной работе были использованы всего 2 вида: POST, GET.

Метод запроса POST предназначен для направления запроса, при котором веб-сервер принимает данные, заключённые в тело сообщения, для хранения. Он был использован для отправки с клиента на сервер данных из форм (листинг 7).

Листинг 7 – Отправка данных на сервер с помощью POST-запроса

const {Router} = require('express')

const router = Router()

router.post('/writearticle',cookieJwtAuth, async (req, res) => {

    const payload = jwt.verify(req.cookies.token,secret)

    const article = new Article({

        label: req.body.articleLabel,

        author: payload.login,

        content: req.body.articleBody

    })

    await article.save()

    res.redirect('/articles')

})

А GET предназначен для получения информации от сервера. С его помощью, в данной работе, клиентский браузер «получает» веб страницу (листинг 8).

Листинг 8 – Пример использования GET-запроса

router.get('/', (req, res) => {

    if (req.cookies.token){ //если токен есть в куки(т.е пользователь зашел в аккаунт)

        res.render('main', {

            title: 'Главная',

            layout: 'loggedIn', //отрисовываем страницу с шаблоном, где есть страница "Написать статью"

        })

    } else

    {res.render('main', {

        title: 'Главная', //отрисовываем страницу с шаблоном, где нет страницы "Написать статью"

    })}

});

## 3.1 Handlebars

Практически все современные сайты создают html страницу в зависимости от информации, которую нужно отобразить. Такие сайты называются динамическими. Динамический сайт – сайт, состоящий из динамичных страниц – шаблонов, контента, скриптов и прочего, в большинстве случаев хранящихся на сервере как отдельные ресурсы (файлы, данные в базах данных и на сторонних серверах).

Текущий сайт тоже является динамическим, потому что страница со списком статей формируется на основе имеющихся в базе данных записей. Для того, чтобы реализовать данную особенность, воспользуемся пакетом handlebars. Он позволяет создавать шаблоны html разметки, которые потом заполняются данными. Подключение пакета происходит следующим образом (листинг 9).

Листинг 9 – Подключение шаблонизатора handlebars

const exphbs = require('express-handlebars')

// настраиваем шаблонизатор

const hbs = exphbs.create({

  defaultLayout:'mainArticle',

  extname:'hbs'

})

app.engine('hbs',hbs.engine) //активируем шаблонизатор

app.set('view engine', 'hbs')

app.set('views', 'views')

Для работы шаблонизатора необходимо создать папки layout, views, в которых будут храниться соответствующие объекты. Layout – это своеобразный статичный шаблон, куда подставляется динамический шаблон из views.

Позже, при отрисовки страницы, в один из шаблонов views передаются какие-то данные (листинг 10), на основе которых происходит рендер.

Листинг 10 – Передача данных в шаблон

res.render('main', {

            title: 'Главная',

            layout: 'loggedIn', //отрисовываем страницу с шаблоном, где есть страница "Написать статью"

        })

Пример шаблона для страницы «Статьи» представлен на листинге 11.

Листинг 11 – Использование

{{!-- Шаблон для страницы "Статьи" --}}

{{#if articless.length}}

    {{#each articless}}

        <a href="/article{{\_id}}">

            <div  class="listArticle">

                <div class="flexLabelAuthor">

                    <div class="listArticleLabel">{{label}}</div>

                    <div class="listArticleAuthor">{{author}}</div>

                </div>

                <div class="listArticlePreview">{{content}}</div>

            </div>

        </a>

    {{/each}}

{{else}}

<p>Нет статей :(</p>

{{/if}}

## 3.2 MongoDB

Ранее мы подключились к базе данных. Пора создавать коллекции. В MongoDB перед тем как создать коллекцию(модель), необходимо создать ее Схему. В листинге 12 показано создание схемы для коллекции «Статьи»

Листинг 12 – Создание схемы «Статьи»

const {Schema,model} = require('mongoose')

const Article = new Schema({

    label: {type: String, requires: true},

    author: {type: String, ref: 'user'},

    content: {type: String, requires: true}

})

module.exports = model('Article', Article)

И уже позже, на основе этой схемы создается модель (листинг 13).

Листинг 13 – Создание модели «Статьи»

const article = new Article({

        label: req.body.articleLabel,

        author: payload.login,

        content: req.body.articleBody

    })

    await article.save()

А примером использования информации из базы данных может послужить обработка post-запроса на регистрацию. Также здесь был использован пакет bycrypt для хэширования паролей (листинг 14).

async signup(req,res) { //регистрация

        try {

            const error = validationResult(req)

            if (!error.isEmpty){        //проверяется наличие ошибки

                return res.redirect('/') //если есть, вереводит на главную сайта

                // return res.status(400).json({message:'Ошибка при регистрации'})

            }

            const {login,email,password} = req.body //достаем из тела запроса поля

            const candidate = await User.findOne({login}) //проверка, есть литакой пользователь уже

            if (candidate) {

                return res.redirect('/')

                // return res.status(400).json({message:`Пользователь ${login} уже существует!`})

            }

            const hashPassword = bcrypt.hashSync(password,7) //хэшируем(зашифровываем пароль)

            const user = new User({login,email,password:hashPassword})  //заполняем таблицу введенными данными

            await user.save()  //сохраняем таблицу

            return res.redirect('/')

            // return res.json({message:'Пользователь успешно создан'})

        } catch (er) {

            console.log(er)

            return res.redirect('/')

            // res.status(400).json({message: 'Ошибка регистрации'})

        }

    }

## 3.3 Jsonwebtoken

После того, как мы настроили базу данных, мы можем приступить к осуществлению авторизации. Хранить информацию о том, какой пользователь делает запрос на сервер мы будем с помощью токена. Он создается при аутентификации (листинг 15).

Листинг 15 – Аутентификация и создание токена

const jwt = require('jsonwebtoken')

const {secret} = require('./config')

//создание токена

const generateAccessToken = (id,login) => {

    const payload = {

        id,

        login

    }

    return jwt.sign(payload,secret, {expiresIn: '15min'})

}

async signin(req,res) { //вход

        try {

            const {login,password} = req.body //достаем поля

            // console.dir(req)

            const user = await User.findOne({login}) //ищем пользователя в бд

            if (!user) {

                return res.redirect('/')

                // res.status(400).json({message: `Пользователя ${login} не существует! Зарегистрируйтесь`})

            }

            const validPassword = bcrypt.compareSync(password,user.password) //сравниваем пароли

            if (!validPassword) {

                return res.redirect('/')

                // res.status(400).json({message: `Неверный пароль`})

            }

            const token = generateAccessToken(user.\_id,user.login) //создаем токен

            res.cookie('token',token,{  //сохраняем его в cookie

                httpOnly: true

            })

            return res.redirect('/')

        } catch (er) {

            console.log(er)

            return res.redirect('/')

            res.status(400).json({message: 'Ошибка авторизации'})

        }

    }

Для того, чтобы потом передать этот токен клиенту, используем пакет cookie-parser.

Позже, уже при принятии запроса со стороны клиента, нам необходимо проверить подлинность токена. Для этого был создан middleware (промежуточная функция), которая выполняет эту задачу (листинг 16).

Листинг 16 – Middleware для проверки подлинности токена

const jwt = require('jsonwebtoken')

const {secret} = require('../config')

//проверка токена

exports.cookieJwtAuth = (req,res,next) => {

    const token = req.cookies.token //достаем токен из cookie

    try {

        const user = jwt.verify(token,secret)  //проверяем токен

        req.user = user

        next() //идем дальше

    } catch(er) {

        res.clearCookie('token') //если токен не сошелся, удаляем

        return res.redirect('/')

    }

}

На этом основные моменты на стороне сервера выполнены.

# Заключение

В результате выполнения данной работы была создана база данных «Книжный магазин». Прежде чем приступить непосредственно к созданию, была исследована предметная область и проведены концептуальное и логическое проектирования. После завершения этих этапов была получена реляционная модель, отображающая приведенные к 3НФ отношения. Затем эта модель была реализована в СУБД PostgreSQL и заполнена небольшим количеством информации, достаточным, для того чтобы показать работоспособность система. Также были созданы: 15 запросов, 3 представления, 3 функции и 3 триггера, которые предоставляют новые возможности для пользователя БД.

Появление баз данных несомненно упростило хранение и обработку большого количества информации, так как пользователю больше не приходится разбирать огромные стеллажи с бумажной литературой. А большая скорость ее обработки, которая продолжает расти засчет появления новых алгоритмов и программных средств, позволяет пользователю получать информацию в короткие сроки и в том виде, в котором она ему понадобилась.

# Список использованных источников

1. Лисьев, Г. А. Программное обеспечение компьютерных сетей и web-серверов [Электронный ресурс] : учебное пособие / Г. А. Лисьев, П. Ю. Романов, Ю. И. Аскерко. – М. : ИНФРА-М, 2020. – 145 с. – Режим доступа: https://znanium.com/catalog/product/1068576

2. Тузовский, А. Ф. Проектирование и разработка web-приложений [Электронный ресурс] : учебное пособие для вузов / А. Ф. Тузовский. – М. : Изд-во Юрайт, 2020. – 218 с. – Режим доступа: https://urait.ru/bcode/451207

1. Гуриков, С. Р. Интернет-технологии [Электронный ресурс] : учебное пособие / С. Р. Гуриков. – М. : Форум : ИНФРА-М, 2019. – 184 с. – Режим доступа: https://znanium.com/catalog/product/995496

2. Лавлинский, В. В. WEB-инжиниринг [Электронный ресурс] : учебное пособие / В. В. Лавлинский, О. В. Курипта, Ю. Г. Табаков. – Воронеж : ВГЛТУ им. Г. Ф. Морозова, 2013. – 268 с. – Режим доступа: https://znanium.com/catalog/product/858312

3. Лежебоков, А. А. Программные средства и механизмы разработки информационных систем [Электронный ресурс] : учебное пособие / А. А. Лежебоков. – Таганрог : Южный федеральный университет, 2016. – 86 с. – Режим доступа: https://znanium.com/catalog/product/997088

4. Малышева, Е. Н. Web-технологии [Электронный ресурс] : учебное пособие / Е. Н. Малышева. – Кемерово [Электронный ресурс] : Кемеров. гос. ин-т культуры, 2018. – 116 с. – Режим доступа: https://znanium.com/catalog/product/1041185

5. Сысолетин, Е. Г. Разработка интернет-приложений [Электронный ресурс] : учебное пособие для вузов / Е. Г. Сысолетин, С. Д. Ростунцев. – М. : Изд-во Юрайт ; Екатеринбург : Изд-во Урал. ун-та, 2020. – 90 с. – Режим доступа: https://urait.ru/bcode/453345

# Приложение А. Опросный лист

|  |  |
| --- | --- |
| Общая информация | |
| Отрасль деятельности | Хобби |
| Название сайта | Musthe |
| Язык сайта | Русский |
| **Цели создания** | |
| Привлечение клиентов | нет |
| Реклама товара | нет |
| Информационная поддержка | да |
| Усиление имиджа / бренда | нет |
| Другое (описать) | - |
| **Целевая аудитория** | |
| Возрастная категория | (10-50) |
| Профессиональные категории | Музыканты/любители |
| Географическое расположение | Не важно |
| Социальные категории | Творческие личности |
| Другое (описать) | - |
| **Дизайн сайта** | |
| Имиджевый |  |
| Информационный | (минимум графики, максимум текста) |
| Продажи |  |
| Другое (описать) | - |
| Стилистика | Строгий, с уклоном в минимализм, сопровождение текста изображением/видео |
| Референсы |  |
| Сайты конкурентов |  |
| Особенности продукции или самого сайта |  |
| **Фирменный стиль** | |
| Логотип / товарный знак | Нет |
| Фирменные цвета | Серый, голубой, розовый |
| Фирменные шрифты | Poppins |
| Визитные карточки |  |
| Слоган / девиз | Musica pro omnibus |
| Другое (описать) |  |
| **Тип сайта** | |
| Сайт-визитка | нет |
| Корпоративный сайт | нет |
| Каталог продукции | нет |
| Интернет-магазин | нет |
| Информационный портал | да |
| Блог | да |
| Портфолио | нет |
| Социальная сеть | нет |
| Другое (описать) | - |
| **Функционал сайта** | |
| Карта сайта | Главная(приветствие), основная страница с информацией, кварто-квинтовый круг, статьи, написание статьи, вход |
| Меню сайта | Панель навигации вверху с полями: кварто-квинтовый круг, статьи, написать статью, вход/выход |
| Фильтры / сортировка / поиск |  |
| Публикация статей / новостей | Публикация статей |
| Рассылка сообщений / оповещения |  |
| Фотоальбомы |  |
| Онлайн-консультант |  |
| Каталог товаров |  |
| Корзина |  |
| Личный кабинет | Вход в аккаунт для доступа к публикации статей |
| Онлайн оплата |  |
| Калькулятор цен |  |
| Акции / распродажи |  |
| Интеграция с соцсетями |  |
| Голосования и опросы |  |
| Доска объявлений |  |
| Форма обратной связи |  |
| Хостинг файлов |  |
| Вопрос-ответ |  |
| Форум |  |
| Отзывы |  |
| Другие задачи (описать) |  |
| **Информационное наполнение** | |
| Количество страниц |  |
| Количество наименований товаров/продукции |  |
| Информация в <header> | Название сайта и кнопки навигации |
| Информация в <footer> |  |
| Контактная информация |  |
| Прерогативная информация | Продукция  Цены  Новости  Личный профиль  Контакты  Другое: Click or tap here to enter text. |
| Блоки на главной странице сайта |  |
| **Программное обеспечение** | |
| Frontend-фреймворки | не используется |
| Backend-фреймворки | Express |
| Серверный язык | Node JS |
| База данных | MongoDB Atlas |
| Текстовые форматы | (JSON, XML) |
| Web-сервер |  |
| Хост-система | Windows |
| Система управления контентом | не используется |
| Используемые API |  |
| Интеграция со сторонним ПО |  |
| Другое (указать название и версию) |  |

# Приложение Б. Исходный код

<https://github.com/lyaguh/courssework>