Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт управления бизнес-процессами

Кафедра бизнес-информатики и модерирования бизнес-процессов

**КУРСОВАЯ РАБОТА**

Создание web-приложения с применением современных web-технологий по тематике «Туристические маршруты»

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Никитин

Студент УБ22-09Б, 432218135 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. О. Егорова

Красноярск 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc165047957)

[1 Теоретическая часть 5](#_Toc165047958)

[1.1 HTML и CSS 5](#_Toc165047959)

[1.2 JavaScript 6](#_Toc165047960)

[1.3 PHP 7](#_Toc165047961)

[1.4 MySQL 9](#_Toc165047962)

[2 Практическая часть 11](#_Toc165047963)

[2.1 Требования и описание реализации 11](#_Toc165047964)

[2.2 Общая структура и создание базы данных 11](#_Toc165047965)

[2.3 Главная страница 13](#_Toc165047966)

[2.4 Регистрация и авторизация в веб-приложении 17](#_Toc165047967)

[2.5 Личный кабинет 22](#_Toc165047968)

[2.6 Функционал Аdmin 25](#_Toc165047969)

[ЗАКЛЮЧЕНИЕ 45](#_Toc165047970)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 46](#_Toc165047971)

# **ВВЕДЕНИЕ**

В современном обществе, со стремительным ростом развития информационных технологий, все больше набирает популярность разработка веб-приложений.

Веб-приложения позволяют компаниям и брендам быть видимыми в онлайн-пространстве и устанавливать контакт с клиентами и пользователем. Они обеспечивают удобную платформу для предоставления продуктов, услуг и информации. Веб-приложения могут решать различные бизнес-задачи, автоматизировать процессы и повышать эффективность работы компании. Они могут быть использованы для управления складом, обработки заказов, учета финансовых транзакций и других операций.

Эта область разработки имеет огромное количество возможностей и перспектив. Веб-приложения используются в различных сферах, включая электронную коммерцию, социальные сети, онлайн-банкинг, образование, здравоохранение и многое другое.

Разработка веб-приложений требует знания нескольких ключевых технологий и языков программирования, таких как HTML, CSS, JavaScript, PHP и других. Веб-разработчики также должны быть знакомы с базами данных и серверными технологиями, такими как Apache, Nginx и другие.

HTML – это язык разметки, используемый для создания веб-страниц. HTML позволяет определить структуру и содержимое страницы с помощью различных элементов и тегов, которые задаются в текстовом формате.

Основная задача HTML заключается в структурировании контента веб-страницы и предоставлении доступа к другим ресурсам, таким как изображения, видео и файлы стилей.

MySQL — это свободная система управления базами данных (СУБД), которая является одной из самых популярных и широко используемых среди разработчиков.

MySQL предоставляет мощный набор инструментов для создания, модификации и управления базами данных. Она поддерживает широкий спектр функциональных возможностей, включая структурированные таблицы, индексы, хранимые процедуры, триггеры и многое другое. MySQL позволяет работать с большими базами данных, обеспечивая надежное хранение данных различных типов и позволяя эффективно управлять ими. MySQL обладает высокой скоростью работы и отличной производительностью при обработке больших объемов данных.

Таким образом, применение навыков в работе с HTML, CSS, MySQL, JavaScript, PHP и позволяет разработчикам создавать современные и удобные веб-приложения, уделяя приоритет функциональности и визуальному оформлению в равных частях.

Целью курсовой работы является разработка сайта, используя технологии HTML, CSS, Javascript, PHP, MySQL, согласно теме «Бронирование столиков в ресторане», а также разработка базы данных для управления пользователями, используя СУБД MySQL.

Объектом исследования является web-приложение с применением различных web-технологий.

Предметом исследования является процесс создания web-приложения, базы данных.

# **1 Теоретическая часть**

## **1.1 HTML и CSS**

Одними из основных инструментов для разработки веб-приложений являются HTML и CSS.

HTML (HyperText Markup Language) — это язык разметки, используемый для создания структуры и представления контента на веб-страницах. HTML состоит из тегов, которые определяют различные элементы страницы, такие как заголовки, абзацы, списки, таблицы, изображения и многие другие.

HTML позволяет организовать информацию на веб-странице, определяя ее структуру и семантику. Разработчики могут определить структуру документа и представление его содержимого. HTML позволяет создавать гиперссылки, которые позволяют пользователям перемещаться по различным страницам, файлам или даже другим местам в пределах одной страницы.

Также, данный инструмент предоставляет возможность создавать интерактивные формы, которые позволяют пользователям взаимодействовать с веб-приложением.

HTML является основой для создания веб-приложений и работает в совокупности с другими технологиями, такими как CSS и JavaScript, чтобы предоставлять интерактивное и привлекательное пользовательское веб-приложение.

CSS (Cascading Style Sheets) - это язык, используемый для описания внешнего вида веб-страниц. Он позволяет разработчикам контролировать внешний вид элементов на странице, таких как цвет текста, размер шрифта, расположение и многое другое.

С помощью CSS разработчики могут определить стили для различных элементов на веб-странице. CSS позволяет разработчикам задавать макет страницы, определяя размеры, положение и отступы элементов.

CSS также позволяет создавать адаптивные веб-приложения, которые корректно отображаются на разных устройствах и экранах.

С помощью CSS можно создавать анимацию и переходы на веб-страницах. Разработчики могут задавать свойства анимации, такие как продолжительность, тип и задержку переходов, чтобы создавать интерактивность на странице.

В целом, HTML и CSS являются неотъемлемой частью разработки веб-приложений, которая позволяет разработчикам создавать привлекательный и современный дизайн для пользовательского интерфейса. Он обеспечивает гибкость и контроль над внешним видом, а также облегчает поддержку и сопровождение проектов.

## **1.2 JavaScript**

JavaScript — это интерпретируемый язык программирования, который применяется для создания интерактивных веб-страниц и веб-приложений. Он предоставляет возможность добавлять динамическое содержимое на веб-страницы, управлять событиями и интерактивностью пользовательского интерфейса.

Главной особенностью JavaScript является его способность работать непосредственно в браузере без необходимости установки дополнительного ПО. Он является одним из трех основных технологических компонентов веба, вместе с HTML и CSS.

Одним из главных преимуществ JavaScript является его широкий функционал. Он позволяет разработчикам создавать сложные и интерактивные веб-приложения, обрабатывать события, взаимодействовать с сервером без перезагрузки страницы и многое другое. JavaScript также поддерживает объектно-ориентированное программирование, что делает его мощным инструментом для разработки приложений любого масштаба.

JavaScript позволяет программно изменять содержимое, структуру и стиль элементов HTML и CSS на странице. Это позволяет создавать интерактивные пользовательские интерфейсы и реагировать на действия пользователей.

JavaScript предоставляет механизмы для обработки различных событий, таких как клики, наведения курсора, отправка формы, загрузка страницы и другие. Это позволяет реагировать на пользовательские действия и выполнить определенные действия в ответ.

Также, JavaScript может использоваться для создания анимированных эффектов и переходов на веб-страницах. Он предоставляет механизмы для управления временем, интервалами и таймингами анимаций.

JavaScript позволяет создавать и управлять куки - кусочков информации, которые хранятся на компьютере пользователя. Он также предоставляет возможности для работы с локальным хранилищем, где данные могут быть сохранены и использованы веб-приложением.

Еще одной интересной особенностью JavaScript является создание и управление всплывающими окнами: JavaScript предоставляет возможность создания всплывающих окон, которые могут быть использованы для отображения дополнительной информации или выполнения действий в отдельном окне или вкладке браузера.

JavaScript используется не только на веб-страницах, но и в других средах, таких как серверы (Node.js) и мобильные приложения (React Native). Его гибкость, широкая поддержка и огромное сообщество разработчиков делают его основным языком программирования для создания интерактивных и динамичных веб-приложений.

## **1.3 PHP**

PHP (Hypertext Preprocessor) – это сценарный язык программирования, который часто используется для разработки веб-приложений и динамических веб-сайтов. Он является одним из наиболее популярных языков программирования для веб-разработки.

PHP имеет множество функций и возможностей, которые делают его очень полезным и удобным для разработчиков. Вот некоторые из основных функциональных возможностей PHP:

1. Обработка данных форм: PHP позволяет получить данные, введенные пользователем в веб-формы, и обрабатывать их на стороне сервера. Это позволяет создавать интерактивные веб-сайты с функциональностью обратной связи.

2. Работа с базами данных: PHP позволяет подключаться к различным базам данных, таким как MySQL, PostgreSQL и другим, и выполнять операции добавления, удаления и изменения данных.

3. Генерация динамического контента: PHP позволяет создавать динамический контент на веб-страницах. Это означает, что содержимое страницы может быть изменено в зависимости от условий и параметров, переданных скрипту.

4. Работа с файлами: PHP предлагает функции для работы с файлами, такие как чтение, запись и удаление файлов на сервере.

5. Отправка почты: PHP позволяет отправлять электронные письма с помощью функций для работы с почтовыми протоколами.

6. Взаимодействие с другими сервисами: PHP предоставляет возможности для взаимодействия с другими веб-сервисами и API, такими как Facebook, Twitter и другие.

7. Создание сессий и управление пользовательскими данными: PHP позволяет создавать сессии для отслеживания пользователей и сохранения данных между страницами.

8. Работа с XML: PHP позволяет создавать и обрабатывать XML-документы, что полезно для работы с данными в формате XML.

PHP также имеет большое сообщество разработчиков, что означает, что вы можете найти множество готовых библиотек и решений для большинства задач.

В целом, PHP предоставляет широкий функционал для разработки веб-приложений и позволяет разработчикам создавать мощные и интерактивные веб-сайты.

## **1.4 MySQL**

MySQL широко используется для хранения и обработки данных веб-приложений, таких как интернет-магазины, социальные сети и блоги.

MySQL — это система управления базами данных (СУБД), которая позволяет хранить, управлять и обрабатывать большие объемы данных. Она является одной из самых популярных СУБД, широко используется в веб-приложениях, а также в других областях разработки программного обеспечения.

Система MySQL имеет широкий функционал, который позволяет разработчикам создавать сложные базы данных и производить множество операций с данными. Вот некоторые из главных возможностей и функций MySQL:

1. Создание и управление базами данных: MySQL позволяет создавать новые базы данных, а также управлять существующими базами данных, включая добавление, изменение и удаление таблиц и полей.

2. Хранение и обработка данных: MySQL позволяет хранить и обрабатывать различные типы данных, такие как числа, текст, даты и другие. Он также поддерживает операции с данными, такие как вставка, обновление и выборка данных из таблицы.

3. Обеспечение безопасности данных: MySQL предоставляет функции шифрования и аутентификации данных для обеспечения безопасности хранимых данных. Он также поддерживает ролевую модель доступа, что позволяет контролировать доступ пользователей к базе данных и ее объектам.

4. Высокая производительность: MySQL разработан с учетом высокой производительности и масштабируемости. Он способен обрабатывать большое количество запросов одновременно и обеспечивать быстрый доступ к данным.

5. Репликация данных: MySQL позволяет создавать репликации данных, что обеспечивает отказоустойчивость и увеличивает доступность данных. Репликация позволяет иметь несколько копий базы данных для обработки запросов и распределения нагрузки.

Хранение данных в MySQL осуществляется в таблицах, которые состоят из строк и столбцов. Данные хранятся в определенном формате, в зависимости от типа данных. MySQL также поддерживает возможность создания связей между таблицами, что позволяет объединять и анализировать данные из нескольких источников.

В целом, MySQL предоставляет надежное, масштабируемое и эффективное решение для управления данными, подходящее для широкого спектра приложений и задач.

# **2 Практическая часть**

## **2.1 Требования и описание реализации**

Как уточнялось ранее, курсовой проект предполагает создание web-приложение на тему «Туристические маршруты», используя технологии HTML, CSS, Javascript, PHP, MySQL. Требуется реализовать добавление, редактирование, просмотр и удаление пользователей. Также предусмотреть функции авторизации и аутентификации, с учётом ролей пользователей.

**Требования к проекту:**

1. Должна быть предусмотрена авторизация и регистрация пользователей.
2. Зарегистрированный пользователь имеет право редактировать свои данные и просматривать свои услуги.
3. Администраторы имеют право редактировать пользователей и добавлять услуги.
4. Неавторизированный пользователи имеют право просматривать список услуг
5. Сайт должен быть выдержан в определенном стиле и корректно открываться на любом разрешении.
6. Общий функционал может быть продуман самостоятельно.
7. Важно продемонстрировать не большой объем функционала, а минимальные требования к проекту. Проект может максимально простым, но при этом содержать все необходимые функции.
8. Отдельные бонусы предусмотрены за доп. функционал и красивый дизайн.

## **2.2 Общая структура и создание базы данных**

Установка MAMP позволяет воспроизводить web-приложение локально. С помощью встроенного функционала создалась база данных с кодировкой utf8\_general\_ci.

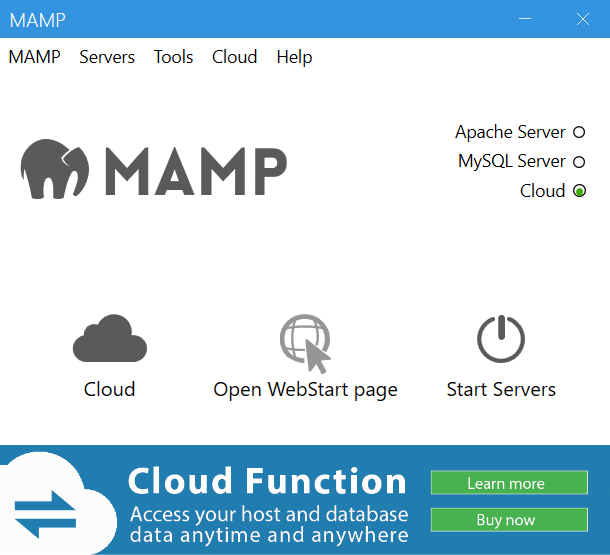


Рисунок 1 – Установка MAMP

Создание базы данных:

**Изображение выглядит как текст, число, Шрифт, программное обеспечение

Автоматически созданное описание**

Рисунок 2 – Создание базы данных

Листинг:

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

login VARCHAR(100) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

role ENUM('admin', 'regular') NOT NULL,

last\_name VARCHAR(255) NOT NULL,

first\_name VARCHAR(255) NOT NULL,

middle\_name VARCHAR(255),

phone VARCHAR(20) NOT NULL

);

CREATE TABLE routes (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL

);

CREATE TABLE user\_routes (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

route\_id INT NOT NULL,

purchase\_date DATE NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (route\_id) REFERENCES routes(id)

);

Название базы данных *tr* с кодировкой *utf8\_general\_ci*. Создаётся таблица пользователей, которая содержит в себе id, login, password, role, last\_name, first\_name, middle\_name, phone. Далее создаётся таблица с доступными маршрутами, которая содержит в себе id, name, description и price. Последней создается таблицы с маршрутами пользователей, которая содержит id, user\_id, route\_id, purchase\_date, start\_date, end\_date.

## **2.3 Главная страница**

При переходе на сайт перед пользователем открывается главная страница сайта, на которой он может сразу же посмотреть список доступных маршрутов или авторизоваться (зарегистрироваться) на сайте.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3 – Главная страницы веб-сайта

Для создания главной страницы создается исходный php файл index.php.

Листинг:

<?php

session\_start();

if (!isset($\_SESSION['user\_logged\_in'])) {

    // Если пользователь не авторизован, устанавливаем его как гостя

    $\_SESSION['username'] = 'guest';

    $\_SESSION['password'] = '';

    $\_SESSION['role'] = 'guest';

} else {

    // Если пользователь авторизован, но роль не установлена, устанавливаем по умолчанию

    if (!isset($\_SESSION['role'])) {

        $\_SESSION['role'] = 'guest';

    }

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Туристические маршруты</title>

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

</head>

<body>

<div class="sidebar">

    <h2>Меню</h2>

    <button class="navButton" onclick="loadContent('/views/view\_user\_data.html')" <?php if($\_SESSION['role'] === 'guest') echo 'disabled'; ?>>Мои данные</button>

    <button class="navButton" onclick="loadContent('/views/view\_user\_route.html')" <?php if($\_SESSION['role'] === 'guest') echo 'disabled'; ?>>Мои маршруты</button>

    <button class="navButton" onclick="loadContent('/views/view\_route.html')">Список маршрутов</button>

    <button class="navButton" id="logoutButton" onclick="window.location.href='<?php echo ($\_SESSION['role'] === 'guest') ? 'login.php' : 'logout.php'; ?>'"><?php echo ($\_SESSION['role'] === 'guest') ? 'Вход' : 'Выйти'; ?></button>

</div>

<div class="content" id="content">

    <div class="sticky-header">

        <h1>Добро пожаловать!</h1>

        <p>Выберите категорию из меню слева для просмотра информации.</p>

    </div>

</div>

<script>

let currentContentUrl = '';

function loadContent(url) {

    fetch(url + '?\_=' + new Date().getTime())

        .then(response => response.text())

        .then(html => {

            const content = document.getElementById('content');

            content.innerHTML = html;

            if (url === '/views/view\_user\_data.html'){

                fillFormData();

                setEditForm();

            }

            executeScripts(content);

            bindCloseButton();

            highlightActiveButton(url);

        })

        .catch(error => console.error('Ошибка при загрузке страницы:', error));

}

function executeScripts(content) {

    const scripts = Array.from(content.querySelectorAll('script'));

    scripts.forEach(script => {

        eval(script.innerText);

    });

}

function setEditForm() {

    document.getElementById('editUserDataForm').addEventListener('submit', function(e) {

        e.preventDefault();

        const formData = new FormData(this);

        formData.append('action', 'updateUser');

        fetch('../DAO/user\_DAO.php', {

            method: 'POST',

            body: formData

        })

        .then(response => response.json())

        .then(data => {

            if (data.success) {

                alert('Данные пользователя успешно обновлены.');

            } else {

                alert('Произошла ошибка при обновлении данных пользователя.');

            }

        })

        .catch(error => console.error('Ошибка:', error));

    });

}

function fillFormData() {

    fetch(`../DAO/user\_DAO.php?action=getUserByIdSession&id`)

    .then(response => response.json())

    .then(data => {

        if (!data.error) {

            document.getElementById('login').value = data.login;

            document.getElementById('password').value = data.password;

            document.getElementById('last\_name').value = data.last\_name;

            document.getElementById('first\_name').value = data.first\_name;

            document.getElementById('middle\_name').value = data.middle\_name;

            document.getElementById('phone').value = data.phone;

        } else {

            console.error('Пользователь не найден:', data.error);

        }

    })

    .catch(error => console.error('Ошибка при получении данных пользователя:', error));

}

function highlightActiveButton(activeUrl) {

    document.querySelectorAll('.navButton').forEach(button => {

        const onclickAttribute = button.getAttribute('onclick');

        if (onclickAttribute && onclickAttribute.includes(activeUrl)) {

            button.classList.add('activeButton');

        } else {

            button.classList.remove('activeButton');

        }

    });

}

document.addEventListener("DOMContentLoaded", function() {

    const sidebar = document.querySelector('.sidebar');

    const content = document.querySelector('.content');

    sidebar.addEventListener('mouseenter', function() {

        this.style.left = '0';

        content.style.marginLeft = '220px';

    });

    sidebar.addEventListener('mouseleave', function() {

        this.style.left = '-180px';

        content.style.marginLeft = '20px';

    });

});

document.getElementById('logoutButton').addEventListener('click', function() {

    window.location.href = 'logout.php';

});

</script>

</body>

</html>

## **2.4 Регистрация и авторизация в веб-приложении**

Для создания страницы регистрации создается файл register.php. Страница регистрации представляет собой окно, в которое включены основные функции регистрации и авторизации.

Форма регистрации содержит в себе поля для ввода логина, пароля, фамилии, имени, отчества и номера телефона, а также кнопки «зарегистрироваться».

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

Рисунок 4 – Панель Регистрации

Листинг:

<?php

require\_once 'register\_user.php';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    $login = $\_POST['login'];

    $password = $\_POST['password'];

    $role = 'regular';

    $lastName = $\_POST['last\_name'];

    $firstName = $\_POST['first\_name'];

    $middleName = $\_POST['middle\_name'] ?? '';

    $phone = $\_POST['phone'];

    $result = registerUser($login, $password, $role, $lastName, $firstName, $middleName, $phone);

    if ($result) {

        $\_SESSION['registration\_success'] = "Регистрация успешно завершена!";

        header("Location: login.php");

        exit;

    } else {

        $\_SESSION['registration\_error'] = "Ошибка при регистрации.";

        header("Location: register.php");

        exit;

    }

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Регистрация</title>

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

</head>

<body>

<div class="form-wrapper">

    <div id="registerForm">

        <h2>Регистрация</h2>

        <?php

        if (isset($\_SESSION['registration\_error'])) {

            echo '<p style="color: red;">' . $\_SESSION['registration\_error'] . '</p>';

            unset($\_SESSION['registration\_error']);

        }

        ?>

        <form action="register.php" method="post">

            <input type="text" name="login" placeholder="Логин" required>

            <input type="password" name="password" placeholder="Пароль" required>

            <input type="text" name="last\_name" placeholder="Фамилия" required>

            <input type="text" name="first\_name" placeholder="Имя" required>

            <input type="text" name="middle\_name" placeholder="Отчество">

            <input type="text" name="phone" placeholder="Телефон" required>

            <button type="submit">Зарегистрироваться</button>

        </form>

        <a href="login.php">Уже зарегистрированы? Войти</a>

    </div>

</div>

</body>

</html>

        <div class="input-group">

          <label>Подтвердите пароль</label>

          <input type="password" name="password\_2">

        </div>

        <div class="input-group">

          <button type="submit" class="btn" name="reg\_user">Регистрация</button>

        </div>

<p>

                Уже есть аккаунт? <a href="login.php">Войти</a>

        </p>

  </form>

</body>

</html>

При нажатии на кнопку «зарегистрироваться» данные пользователя попадают в базу данных. После регистрации пользователь переходит сразу же на главную страницу сайта.

Также, используется файл register\_user для регистрации пользователя с правами администратора, где с помощью php кода на соответствие проверяется роль администратора данного пользователя.

Листинг:

<?php

require\_once 'database\_config.php';

function registerUser($login, $password, $role, $lastName, $firstName, $middleName, $phone) {

    try {

        // Подключение к базе данных с правами администратора

        $conn = DatabaseConfig::connect('admin', 'admin');

        // Вставка записи о пользователе в таблицу users

        $stmt = $conn->prepare("INSERT INTO users (login, password, role, last\_name, first\_name, middle\_name, phone) VALUES (?, ?, ?, ?, ?, ?, ?)");

        $stmt->execute([$login, $password, $role, $lastName, $firstName, $middleName, $phone]);

        // Создание пользователя MySQL и назначение прав

        $conn->exec("CREATE USER '$login'@'localhost' IDENTIFIED BY '$password'");

        $conn->exec("GRANT SELECT, UPDATE ON tr.\* TO '$login'@'localhost'");

        $conn->exec("FLUSH PRIVILEGES");

        return true;

    } catch (PDOException $e) {

        echo "Ошибка при регистрации: " . $e->getMessage();

        return false;

    }

}

Блок авторизации включает в себя поля логина и пароля. Если пользователь не авторизован, он может перейти по ссылке «Регистрация», которая перебрасывает пользователя на страницу регистрации, рассмотренную выше.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5 – Панель авторизации

Для создания авторизации применяется файл login.php.

Листинг:

<?php session\_start(); ?>

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Авторизация</title>

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

</head>

<body>

<div class="form-wrapper">

    <div id="loginForm">

        <h2>Авторизация</h2>

        <form action="../database\_config.php" method="post">

            <input type="text" id="loginUsername" name="username" placeholder="Имя пользователя" required>

            <input type="password" id="loginPassword" name="password" placeholder="Пароль" required>

            <button type="submit">Войти</button>

            <a href="register.php" class="register-btn">Регистрация</a>

        </form>

        <?php

        if (isset($\_SESSION['login\_error'])) {

            echo '<script>alert("' . $\_SESSION['login\_error'] . '");</script>';

            unset($\_SESSION['login\_error']);

        }

        ?>

    </div>

</div>

</body>

</html>

Подключение к базе данных при регистрации и авторизации внесено в отдельный файл database\_config.php.

Листинг:

<?php

session\_start();

class DatabaseConfig {

    private static $url = 'mysql:host=localhost;dbname=tr';

    public static function connect($login = '', $password = '') {

        $username = empty($login) ? ($\_SESSION['username'] ?? '') : $login;

        $password = empty($password) ? ($\_SESSION['password'] ?? '') : $password;

        try {

            $conn = new PDO(self::$url, $username, $password);

            $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

            return $conn;

        } catch (PDOException $e) {

            $\_SESSION['login\_error'] = 'Ошибка подключения к базе данных. Пожалуйста, проверьте ваши учетные данные.';

            return null;

        }

    }

}

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['username']) && isset($\_POST['password'])) {

    $\_SESSION['username'] = $\_POST['username'];

    $\_SESSION['password'] = $\_POST['password'];

    $conn = DatabaseConfig::connect();

    if ($conn) {

        $stmt = $conn->prepare("SELECT id, role FROM users WHERE login = ? AND password = ?");

        $stmt->execute([$\_POST['username'], $\_POST['password']]);

        $user = $stmt->fetch(PDO::FETCH\_ASSOC);

        if ($user) {

            $\_SESSION['user\_logged\_in'] = true;

            $\_SESSION['role'] = $user['role'];

            $\_SESSION['id'] = $user['id'];

            // В зависимости от роли пользователя перенаправляем на разные страницы

            if ($user['role'] === 'admin') {

                header("Location: admin\_panel.php");

            } else {

                header("Location: index.php");

            }

            exit;

        } else {

            $\_SESSION['login\_error'] = 'Неверный логин или пароль.';

            header("Location: login.php");

            exit;

        }

    } else {

        header("Location: login.php");

        exit;

    }

}

?>

## **2.5 Личный кабинет**

В меню сайта пользователь может попасть в свой личный кабинет и редактировать свои данные. В меню слева на главной странице располагается строка «Мои данные», которая перенаправляет пользователя на страницу личного кабинета. На главной панели личного кабинета располагаются несколько функциональных строк для изменение своего логина и пароля, а также кнопка сохранения изменений.

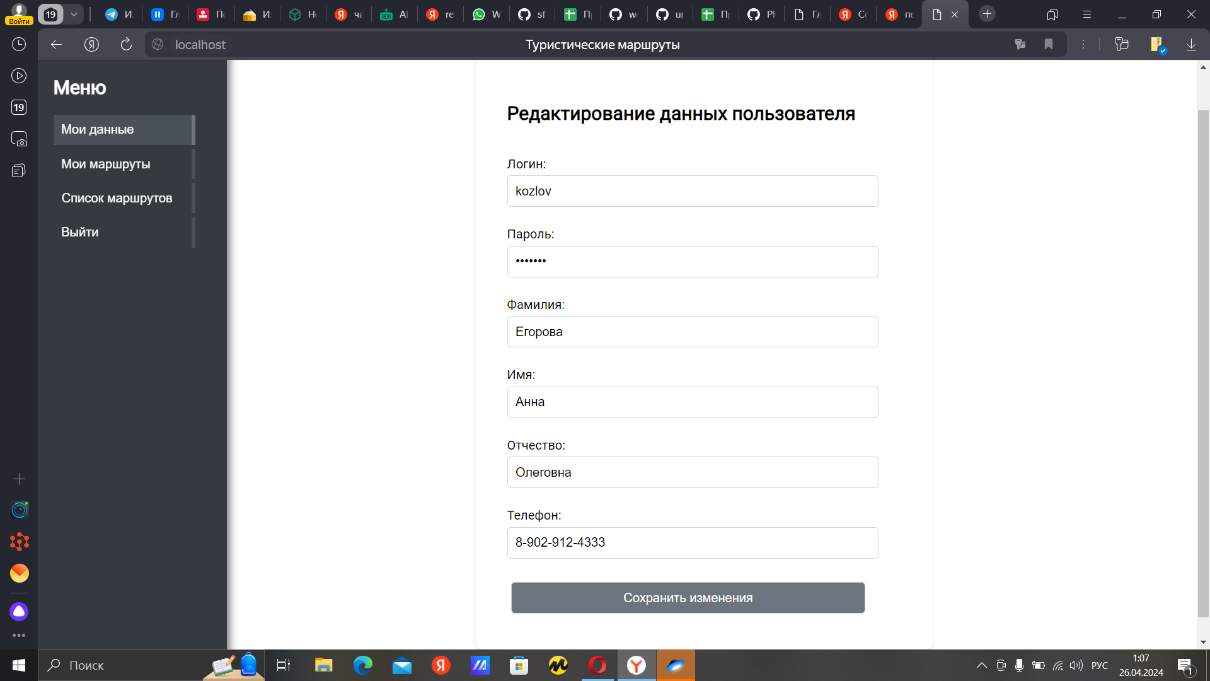


Рисунок 6 – Личный кабинет

Для редактирования личных данных пользователя был создан файл view\_user\_data.html.

Листинг:

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Редактирование данных пользователя</title>

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

</head>

<body>

<div class="form-container">

    <div id="sessionData"></div>

    <h2>Редактирование данных пользователя</h2>

    <form id="editUserDataForm">

        <input type="hidden" id="id" name="id">

        <div class="form-group">

            <label for="login">Логин:</label>

            <input type="text" id="login" name="login" required readonly>

        </div>

        <div class="form-group">

            <label for="password">Пароль:</label>

            <input type="password" id="password" name="password" required readonly>

        </div>

        <div class="form-group">

            <label for="last\_name">Фамилия:</label>

            <input type="text" id="last\_name" name="last\_name" required>

        </div>

        <div class="form-group">

            <label for="first\_name">Имя:</label>

            <input type="text" id="first\_name" name="first\_name" required>

        </div>

        <div class="form-group">

            <label for="middle\_name">Отчество:</label>

            <input type="text" id="middle\_name" name="middle\_name">

        </div>

        <div class="form-group">

            <label for="phone">Телефон:</label>

            <input type="text" id="phone" name="phone" required>

        </div>

        <button type="submit">Сохранить изменения</button>

    </form>

</div>

<script>

    function fillFormData() {

        fetch(`../DAO/user\_DAO.php?action=getUserByIdSession&id`)

        .then(response => response.json())

        .then(data => {

            if (!data.error) {

                document.getElementById('id').value = data.id;

                document.getElementById('login').value = data.login;

                document.getElementById('password').value = data.password;

                document.getElementById('last\_name').value = data.last\_name;

                document.getElementById('first\_name').value = data.first\_name;

                document.getElementById('middle\_name').value = data.middle\_name;

                document.getElementById('phone').value = data.phone;

            } else {

                console.error('Пользователь не найден:', data.error);

            }

        })

        .catch(error => console.error('Ошибка при получении данных пользователя:', error));

    }

    document.addEventListener('DOMContentLoaded', function() {

        fillFormData();

        document.getElementById('editUserDataForm').addEventListener('submit', function(e) {

            e.preventDefault();

            const formData = new FormData(this);

            formData.append('action', 'updateUser');

            fetch('../DAO/user\_DAO.php', {

                method: 'POST',

                body: formData

            })

            .then(response => response.json())

            .then(data => {

                if (data.success) {

                    alert('Данные пользователя успешно обновлены.');

                } else {

                    alert('Произошла ошибка при обновлении данных пользователя.');

                }

            })

            .catch(error => console.error('Ошибка:', error));

        });

    });

</script>

</body>

</html>

При изменении данных новые данные сохраняются в базу данных и появляется осведомляющее сообщение.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 7 – Сообщение об успешном обновлении данных

## **2.6 Функционал Аdmin**

После того, как пользователь зарегистрировался или авторизовался и его роль администратора была проверена и подтверждена с помощью php кода, рассмотренного в пункте 2.3, он попадает на главную страницу администратора.

**Изображение выглядит как текст, снимок экрана

Автоматически созданное описание**

Рисунок 8 – Главная страница администратора

Панель управления администратора представляет собой контекстное меню слева. Здесь администратор может выбрать данные какого раздела он хочет редактировать.

Листинг:

<?php

session\_start();

if (!isset($\_SESSION['user\_logged\_in'])) {

    header('Location: login.php');

    exit();

}

// Если пользователь авторизован, но его роль не "admin", перенаправляем на index.php

if ($\_SESSION['role'] !== 'admin') {

    header('Location: index.php');

    exit();

}

?>

?>

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Панель администратора</title>

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

</head>

<body>

<div class="sidebar">

    <h2>Список таблиц</h2>

    <button class="navButton" onclick="loadContent('/views/view\_users\_table.html')">Пользователи</button>

    <button class="navButton" onclick="loadContent('/views/view\_route\_table.html')">Маршруты</button>

    <button class="navButton" onclick="loadContent('/views/view\_route\_users\_table.html')">Маршруты пользователей</button>

    <button class="navButton" id="logoutButton">Выйти</button>

</div>

<div class="content" id="content">

    <div class="sticky-header">

        <h1>Добро пожаловать!</h1>

        <p>Выберите категорию из меню слева для просмотра информации.</p>

    </div>

</div>

<script>

let currentContentUrl = '';

function loadContent(url) {

    fetch(url + '?\_=' + new Date().getTime())

        .then(response => response.text())

        .then(html => {

            const content = document.getElementById('content');

            content.innerHTML = html;

            executeScripts(content);

            bindCloseButton();

            highlightActiveButton(url);

        })

        .catch(error => console.error('Ошибка при загрузке страницы:', error));

}

function executeScripts(content) {

    const scripts = Array.from(content.querySelectorAll('script'));

    scripts.forEach(script => {

        eval(script.innerText);

    });

}

function bindCloseButton() {

    const closeButton = document.querySelector('.close-button');

    if (closeButton) {

        closeButton.onclick = function() {

            closeModal();

        };

    }

}

function highlightActiveButton(activeUrl) {

    document.querySelectorAll('.navButton').forEach(button => {

        if (button.getAttribute('onclick').includes(activeUrl)) {

            button.classList.add('activeButton');

        } else {

            button.classList.remove('activeButton');

        }

    });

}

function closeModal() {

    document.getElementById('modal').style.display = 'none';

}

document.addEventListener("DOMContentLoaded", function() {

    const sidebar = document.querySelector('.sidebar');

    const content = document.querySelector('.content');

    sidebar.addEventListener('mouseenter', function() {

        this.style.left = '0';

        content.style.marginLeft = '220px';

    });

    sidebar.addEventListener('mouseleave', function() {

        this.style.left = '-180px';

        content.style.marginLeft = '20px';

    });

});

document.getElementById('logoutButton').addEventListener('click', function() {

    window.location.href = 'logout.php';

});

</script>

</body>

</html>

Для каждого раздела на панели администратора использован отдельных html файл.

В разделе «Пользователи» администратор может редактировать данные пользователей, добавлять новых пользователей или удалять.

Также в программе предусмотрено выделение нескольких или сразу всех строк таблицы для добавления или удаления:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 9 – Таблица пользователей

Для создания таблицы применяется файл view\_users\_table.html

Листинг кода для таблицы пользователей:

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Список пользователей</title>

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

</head>

<body>

<div class="sticky-header">

    <h2>Список пользователей</h2>

    <div class="buttons-container">

        <button class="addButton">Добавить</button>

        <button class="editButton" id="editButton" disabled>Редактировать</button>

        <button class="deleteButton" id="deleteButton" disabled>Удалить</button>

    </div>

</div>

<table id="userTable">

    <thead>

        <tr>

            <th>ID</th>

            <th>Логин</th>

            <th>Пароль</th>

            <th>Роль</th>

            <th>Фамилия</th>

            <th>Имя</th>

            <th>Отчество</th>

            <th>Телефон</th>

        </tr>

    </thead>

    <tbody>

    </tbody>

</table>

<!-- Модальное окно для добавления/редактирования записей -->

<div id="modal" class="modal" style="display:none;">

    <div class="modal-content">

        <span class="close-button" onclick="closeModal()">&times;</span>

        <h2 id="formTitle">Добавить пользователя</h2>

        <form id="addEditForm">

            <input type="hidden" id="id" name="id">

            <label for="login">Логин:</label>

            <input type="text" id="login" name="login" required><br><br>

            <label for="password">Пароль:</label>

            <input type="password" id="password" name="password" required><br><br>

            <label for="role">Роль:</label>

            <select id="role" name="role">

                <option value="admin">Администратор</option>

                <option value="regular">Пользователь</option>

            </select><br><br>

            <label for="last\_name">Фамилия:</label>

            <input type="text" id="last\_name" name="last\_name" required><br><br>

            <label for="first\_name">Имя:</label>

            <input type="text" id="first\_name" name="first\_name" required><br><br>

            <label for="middle\_name">Отчество:</label>

            <input type="text" id="middle\_name" name="middle\_name"><br><br>

            <label for="phone">Телефон:</label>

            <input type="text" id="phone" name="phone"><br><br>

            <button type="submit">Сохранить</button>

        </form>

    </div>

</div>

<script>

let selectedRows = [];

document.getElementById('userTable').addEventListener('click', function(e) {

    if (e.target.tagName === 'TD') {

        const currentRow = e.target.parentNode;

        const rowIndex = selectedRows.indexOf(currentRow);

        if (rowIndex > -1) {

            selectedRows.splice(rowIndex, 1);

            currentRow.classList.remove('selectedRow');

        } else {

            selectedRows.push(currentRow);

            currentRow.classList.add('selectedRow');

        }

    }

    // Обновляем состояние кнопок

    document.getElementById('editButton').disabled = selectedRows.length !== 1;

    document.getElementById('deleteButton').disabled = selectedRows.length === 0;

});

// Добавляем обработчик двойного нажатия по заголовку таблицы

document.querySelector('thead').addEventListener('dblclick', function() {

    const allRows = document.querySelectorAll('tbody tr');

    if (selectedRows.length === 0) {

        // Выделяем все строки, если не одна не была выделена

        allRows.forEach(row => {

            if (!selectedRows.includes(row)) {

                selectedRows.push(row);

                row.classList.add('selectedRow');

            }

        });

    } else {

        // Снимаем выделение со всех строк, если уже есть выделенные

        allRows.forEach(row => {

            const index = selectedRows.indexOf(row);

            if (index > -1) {

                selectedRows.splice(index, 1);

                row.classList.remove('selectedRow');

            }

        });

    }

    // Обновляем состояние кнопок

    document.getElementById('editButton').disabled = selectedRows.length !== 1;

    document.getElementById('deleteButton').disabled = selectedRows.length === 0;

});

document.getElementById('deleteButton').addEventListener('click', function() {

    if (selectedRows.length > 0 && confirm('Вы уверены, что хотите удалить выбранные записи?')) {

        const ids = selectedRows.map(row => row.children[0].textContent); // Собираем ID выделенных записей

        deleteUsers(ids.join(',')); // Преобразуем массив в строку, разделенную запятыми

    }

});

function deleteUsers(ids) {

    const formData = new FormData();

    formData.append('ids', ids); // Отправляем строку с ID

    formData.append('action', 'deleteMultipleUsers'); // Действие для множественного удаления

    fetch('./DAO/user\_DAO.php', {

        method: 'POST',

        body: formData

    })

    .then(response => response.json())

    .then(data => {

        if (data.success) {

            alert('Выбранные записи успешно удалены.');

            loadUserList();

        }

        if (data.errors) {

            data.errors.forEach(error => {

                alert(`Ошибка при удалении записи с ID ${error.id}: ${error.message}`);

            });

            loadUserList();

        }

    })

    .catch(error => console.error('Ошибка при удалении:', error));

}

document.getElementById('editButton').addEventListener('click', function() {

    if (selectedRows.length > 1) {

        alert('Редактировать можно только одну запись.');

        return;

    }

    if (selectedRows.length === 1) {

        const id = selectedRows[0].children[0].textContent;

        openModal();

        fetch(`./DAO/user\_DAO.php?action=getUserById&id=${id}`)

            .then(response => response.json())

            .then(data => {

                document.getElementById('id').value = data.id;

                document.getElementById('login').value = data.login;

                document.getElementById('password').value = data.password;

                document.getElementById('role').value = data.role;

                document.getElementById('last\_name').value = data.last\_name;

                document.getElementById('first\_name').value = data.first\_name;

                document.getElementById('middle\_name').value = data.middle\_name;

                document.getElementById('phone').value = data.phone;

                document.getElementById('formTitle').innerText = 'Редактировать пользователя';

            })

            .catch(error => console.error('Ошибка:', error));

    }

});

// Обработка нажатия на кнопку "Добавить"

document.querySelector('.addButton').addEventListener('click', function() {

    // Очистить для новой записи

    document.getElementById('id').value = '';

    document.getElementById('login').value = '';

    document.getElementById('password').value = '';

    document.getElementById('role').value = 'regular';

    document.getElementById('last\_name').value = '';

    document.getElementById('first\_name').value = '';

    document.getElementById('middle\_name').value = '';

    document.getElementById('phone').value = '';

    document.getElementById('formTitle').innerText = 'Добавить пользователя';

    openModal(); // Открыть модальное окно

});

function loadUserList() {

    // Сброс выделенных строк

    selectedRows = []; // Очищаем массив выделенных строк

    document.querySelectorAll('#userTable .selectedRow').forEach(row => {

        row.classList.remove('selectedRow'); // Убираем класс выделения со всех строк

    });

    // Обновляем состояние кнопок

    document.getElementById('editButton').disabled = true;

    document.getElementById('deleteButton').disabled = true;

    fetch('./DAO/user\_DAO.php?action=getAllUsersJson')

    .then(response => response.json())

    .then(data => {

        const tableBody = document.getElementById('userTable').getElementsByTagName('tbody')[0];

        tableBody.innerHTML = ''; // Очищаем текущее содержимое tbody

        data.forEach(row => {

            const tr = document.createElement('tr');

            Object.values(row).forEach(text => {

                const td = document.createElement('td');

                td.appendChild(document.createTextNode(text));

                tr.appendChild(td);

            });

            tableBody.appendChild(tr);

        });

    })

    .catch(error => console.error('Ошибка:', error));

}

function openModal() {

    document.getElementById('modal').style.display = 'block';

}

document.addEventListener('DOMContentLoaded', function() {

    // Добавляем обработчик событий для кнопки закрытия модального окна

    const closeButton = document.querySelector('.close-button');

    if (closeButton) {

        closeButton.addEventListener('click', function() {

            closeModal();

        });

    }

});

function closeModal() {

    document.getElementById('modal').style.display = 'none';

}

document.getElementById('addEditForm').addEventListener('submit', function(e) {

    e.preventDefault();

    const formData = new FormData(this);

    formData.append('action', formData.get('id') ? 'updateUser' : 'addUser');

    fetch('./DAO/user\_DAO.php', {

        method: 'POST',

        body: formData

    })

    .then(response => response.json())

    .then(data => {

        alert(data.message);

        if (data.success) {

            closeModal(); // Закрыть модальное окно

            loadUserList(); // Перезагрузить список пользователей

        }

    })

    .catch(error => {

        console.error('Ошибка:', error);

        closeModal(); // Закрыть модальное окно в случае ошибки

    });

});

loadUserList();

</script>

</body>

</html>

Кнопки добавления, редактирования или удаления пользователей вызывают соответствующую панель управления, содержащую функциональные строки редактирования. При изменении таблицы пользователей, сохраненные новые данные попадают в базу данных.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 10 – Редактирование пользователей веб-сайта

Для редактирования данных пользователей, а также добавления новых пользователей или их удаления применяется файл view\_user\_data.html.

Листинг:

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Редактирование данных пользователя</title>

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

</head>

<body>

<div class="form-container">

    <div id="sessionData"></div>

    <h2>Редактирование данных пользователя</h2>

    <form id="editUserDataForm">

        <input type="hidden" id="id" name="id">

        <div class="form-group">

            <label for="login">Логин:</label>

            <input type="text" id="login" name="login" required readonly>

        </div>

        <div class="form-group">

            <label for="password">Пароль:</label>

            <input type="password" id="password" name="password" required readonly>

        </div>

        <div class="form-group">

            <label for="last\_name">Фамилия:</label>

            <input type="text" id="last\_name" name="last\_name" required>

        </div>

        <div class="form-group">

            <label for="first\_name">Имя:</label>

            <input type="text" id="first\_name" name="first\_name" required>

        </div>

        <div class="form-group">

            <label for="middle\_name">Отчество:</label>

            <input type="text" id="middle\_name" name="middle\_name">

        </div>

        <div class="form-group">

            <label for="phone">Телефон:</label>

            <input type="text" id="phone" name="phone" required>

        </div>

        <button type="submit">Сохранить изменения</button>

    </form>

</div>

<script>

    function fillFormData() {

        fetch(`../DAO/user\_DAO.php?action=getUserByIdSession&id`)

        .then(response => response.json())

        .then(data => {

            if (!data.error) {

                document.getElementById('id').value = data.id;

                document.getElementById('login').value = data.login;

                document.getElementById('password').value = data.password;

                document.getElementById('last\_name').value = data.last\_name;

                document.getElementById('first\_name').value = data.first\_name;

                document.getElementById('middle\_name').value = data.middle\_name;

                document.getElementById('phone').value = data.phone;

            } else {

                console.error('Пользователь не найден:', data.error);

            }

        })

        .catch(error => console.error('Ошибка при получении данных пользователя:', error));

    }

    document.addEventListener('DOMContentLoaded', function() {

        fillFormData();

        document.getElementById('editUserDataForm').addEventListener('submit', function(e) {

            e.preventDefault();

            const formData = new FormData(this);

            formData.append('action', 'updateUser');

            fetch('../DAO/user\_DAO.php', {

                method: 'POST',

                body: formData

            })

            .then(response => response.json())

            .then(data => {

                if (data.success) {

                    alert('Данные пользователя успешно обновлены.');

                } else {

                    alert('Произошла ошибка при обновлении данных пользователя.');

                }

            })

            .catch(error => console.error('Ошибка:', error));

        });

    });

</script>

</body>

</html>

Аналогично с разделами «Маршруты» и «Маршруты пользователей».

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 11 – Таблица маршрутов

Листинг для таблицы маршрутов:

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Список маршрутов</title>

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

</head>

<body>

<div class="sticky-header">

    <h2>Список маршрутов</h2>

    <div class="buttons-container">

        <button class="addButton">Добавить</button>

        <button class="editButton" id="editButton" disabled>Редактировать</button>

        <button class="deleteButton" id="deleteButton" disabled>Удалить</button>

    </div>

</div>

<table id="routeTable">

    <thead>

        <tr>

            <th>ID</th>

            <th>Название</th>

            <th>Описание</th>

            <th>Цена</th>

        </tr>

    </thead>

    <tbody>

    </tbody>

</table>

<!-- Модальное окно для добавления/редактирования записей -->

<div id="modal" class="modal" style="display:none;">

    <div class="modal-content">

        <span class="close-button" onclick="closeModal()">&times;</span>

        <h2 id="formTitle">Добавить маршрут</h2>

        <form id="addEditForm">

            <input type="hidden" id="id" name="id">

            <label for="name">Название:</label>

            <input type="text" id="name" name="name" required><br><br>

            <label for="description">Описание:</label>

            <textarea id="description" name="description" required></textarea><br><br>

            <label for="price">Цена:</label>

            <input type="number" id="price" name="price" step="0.01" required><br><br>

            <button type="submit">Сохранить</button>

        </form>

    </div>

</div>

<script>

let selectedRows = [];

document.getElementById('routeTable').addEventListener('click', function(e) {

    if (e.target.tagName === 'TD') {

        const currentRow = e.target.parentNode;

        const rowIndex = selectedRows.indexOf(currentRow);

        if (rowIndex > -1) {

            selectedRows.splice(rowIndex, 1);

            currentRow.classList.remove('selectedRow');

        } else {

            selectedRows.push(currentRow);

            currentRow.classList.add('selectedRow');

        }

    }

    // Обновляем состояние кнопок

    document.getElementById('editButton').disabled = selectedRows.length !== 1;

    document.getElementById('deleteButton').disabled = selectedRows.length === 0;

});

document.getElementById('deleteButton').addEventListener('click', function() {

    if (selectedRows.length > 0 && confirm('Вы уверены, что хотите удалить выбранные записи?')) {

        const ids = selectedRows.map(row => row.children[0].textContent); // Собираем ID выделенных записей

        deleteRoutes(ids.join(',')); // Преобразуем массив в строку, разделенную запятыми

    }

});

function deleteRoutes(ids) {

    const formData = new FormData();

    formData.append('ids', ids); // Отправляем строку с ID

    formData.append('action', 'deleteMultipleRoutes'); // Действие для множественного удаления

    fetch('./DAO/route\_DAO.php', {

        method: 'POST',

        body: formData

    })

    .then(response => response.json())

    .then(data => {

        if (data.success) {

            alert('Выбранные записи успешно удалены.');

            loadRouteList();

        }

        if (data.errors) {

            data.errors.forEach(error => {

                alert(`Ошибка при удалении записи с ID ${error.id}: ${error.message}`);

            });

            loadRouteList();

        }

    })

    .catch(error => console.error('Ошибка при удалении:', error));

}

document.getElementById('editButton').addEventListener('click', function() {

    if (selectedRows.length > 1) {

        alert('Редактировать можно только одну запись.');

        return;

    }

    if (selectedRows.length === 1) {

        const id = selectedRows[0].children[0].textContent;

        openModal();

        fetch(`./DAO/route\_DAO.php?action=getRouteById&id=${id}`)

            .then(response => response.json())

            .then(data => {

                document.getElementById('id').value = data.id;

                document.getElementById('name').value = data.name;

                document.getElementById('description').value = data.description;

                document.getElementById('price').value = data.price;

                document.getElementById('formTitle').innerText = 'Редактировать маршрут';

            })

            .catch(error => console.error('Ошибка:', error));

    }

});

// Обработка нажатия на кнопку "Добавить"

document.querySelector('.addButton').addEventListener('click', function() {

    // Очистить для новой записи

    document.getElementById('id').value = '';

    document.getElementById('name').value = '';

    document.getElementById('description').value = '';

    document.getElementById('price').value = '';

    document.getElementById('formTitle').innerText = 'Добавить маршрут';

    openModal(); // Открыть модальное окно

});

function loadRouteList() {

    // Сброс выделенных строк

    selectedRows = []; // Очищаем массив выделенных строк

    document.querySelectorAll('#routeTable .selectedRow').forEach(row => {

        row.classList.remove('selectedRow'); // Убираем класс выделения со всех строк

    });

    // Обновляем состояние кнопок

    document.getElementById('editButton').disabled = true;

    document.getElementById('deleteButton').disabled = true;

    fetch('./DAO/route\_DAO.php?action=getAllRoutesJson')

    .then(response => response.json())

    .then(data => {

        const tableBody = document.getElementById('routeTable').getElementsByTagName('tbody')[0];

        tableBody.innerHTML = ''; // Очищаем текущее содержимое tbody

        data.forEach(row => {

            const tr = document.createElement('tr');

            Object.values(row).forEach(text => {

                const td = document.createElement('td');

                td.appendChild(document.createTextNode(text));

                tr.appendChild(td);

            });

            tableBody.appendChild(tr);

        });

    })

    .catch(error => console.error('Ошибка:', error));

}

function openModal() {

    document.getElementById('modal').style.display = 'block';

}

document.addEventListener('DOMContentLoaded', function() {

    // Добавляем обработчик событий для кнопки закрытия модального окна

    const closeButton = document.querySelector('.close-button');

    if (closeButton) {

        closeButton.addEventListener('click', function() {

            closeModal();

        });

    }

});

function closeModal() {

    document.getElementById('modal').style.display = 'none';

}

document.getElementById('addEditForm').addEventListener('submit', function(e) {

e.preventDefault();

const formData = new FormData(this);

formData.append('action', formData.get('id') ? 'updateRoute' : 'addRoute');

fetch('./DAO/route\_DAO.php', {

    method: 'POST',

    body: formData

})

.then(response => response.json())

.then(data => {

    alert(data.message);

    if (data.success) {

        closeModal(); // Закрыть модальное окно

        loadRouteList(); // Перезагрузить список маршрутов

    }

})

.catch(error => {

    console.error('Ошибка:', error);

    closeModal(); // Закрыть модальное окно в случае ошибки

});

});

loadRouteList();

</script>

</body>

</html>

Листинг редактирования маршрутов:

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Список маршрутов</title>

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

</head>

<body>

<div class="sticky-header">

    <h2>Список маршрутов</h2>

    <h2></h2>

</div>

<table id="routeTable">

    <thead>

        <tr>

            <th>ID</th>

            <th>Название</th>

            <th>Описание</th>

            <th>Цена</th>

        </tr>

    </thead>

    <tbody>

    </tbody>

</table>

<!-- Модальное окно для добавления/редактирования записей -->

<div id="modal" class="modal" style="display:none;">

    <div class="modal-content">

        <span class="close-button" onclick="closeModal()">&times;</span>

        <h2 id="formTitle">Добавить маршрут</h2>

        <form id="addEditForm">

            <input type="hidden" id="id" name="id">

            <label for="name">Название:</label>

            <input type="text" id="name" name="name" required><br><br>

            <label for="description">Описание:</label>

            <textarea id="description" name="description" required></textarea><br><br>

            <label for="price">Цена:</label>

            <input type="number" id="price" name="price" step="0.01" required><br><br>

            <button type="submit">Сохранить</button>

        </form>

    </div>

</div>

<script>

let selectedRows = [];

document.getElementById('routeTable').addEventListener('click', function(e) {

    if (e.target.tagName === 'TD') {

        const currentRow = e.target.parentNode;

        const rowIndex = selectedRows.indexOf(currentRow);

        if (rowIndex > -1) {

            selectedRows.splice(rowIndex, 1);

            currentRow.classList.remove('selectedRow');

        } else {

            selectedRows.push(currentRow);

            currentRow.classList.add('selectedRow');

        }

    }

});

function loadRouteList() {

    // Сброс выделенных строк

    selectedRows = []; // Очищаем массив выделенных строк

    document.querySelectorAll('#routeTable .selectedRow').forEach(row => {

        row.classList.remove('selectedRow'); // Убираем класс выделения со всех строк

    });

    fetch('./DAO/route\_DAO.php?action=getAllRoutesJson')

    .then(response => response.json())

    .then(data => {

        const tableBody = document.getElementById('routeTable').getElementsByTagName('tbody')[0];

        tableBody.innerHTML = ''; // Очищаем текущее содержимое tbody

        data.forEach(row => {

            const tr = document.createElement('tr');

            Object.values(row).forEach(text => {

                const td = document.createElement('td');

                td.appendChild(document.createTextNode(text));

                tr.appendChild(td);

            });

            tableBody.appendChild(tr);

        });

    })

    .catch(error => console.error('Ошибка:', error));

}

loadRouteList();

</script>

</body>

</html>

# **ЗАКЛЮЧЕНИЕ**

В заключение данной курсовой работы можно с уверенностью сказать, что достигнута поставленная цель. В результате было разработано web-приложение, ориентированное на тематику "туристические маршруты", с использованием передовых web-технологий. В процессе реализации данного проекта были эффективно применены современные web-технологии, включая html, css, javascript, php и mysql.

Использование html и css позволило создать эстетически привлекательный дизайн и верстку, а добавление javascript добавило интерактивности и динамического поведения на веб-страницах. Особое внимание было уделено возможности эффективной обработки данных и взаимодействия с базой данных, что достигнуто благодаря применению php.

Реализация базы данных обеспечивает хранение информации о продуктах, пользователях и сообщениях от посетителей сайта. Это позволяет эффективно управлять информацией и реализовать функциональность, такую как оформление заказов, регистрация пользователей и обратная связь.

Использование передовых web-технологий позволяет создавать отзывчивые и мобильно-дружественные интерфейсы, которые подстраиваются под различные устройства и обеспечивают комфортное взаимодействие с сайтом на любом устройстве.

Создание web-приложения, связанного с "туристическими маршрутами", с использованием передовых web-технологий открывает широкие возможности для продвижения и продажи различных путевок в онлайн-среде, привлечения новых клиентов и создания уникального пользовательского опыта.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Дуванов А. Web-конструирование. HTML / А. Дуванов. - СПб.: БХВ-Петербург, 2013. - 321 с.
2. Белунцов В. Новейший самоучитель по созданию Web-страниц / В. Белунцов. - М.: NT Press, 2012. - 185 с.
3. Браун Брэдли Oracle Database. Создание Web-приложений: учебник / Брэдли Браун, Ричард Ниемик, Джозеф С. Треззо. - СПб.: Лори, 2011. - 722 с.
4. Работа с MySQL в PHP: сайт. – URL: https://htmlacademy.ru/blog/php/mysql?ysclid=lrnhk4qz9b568571317 (дата обращения: 06.01.2024)
5. Руководства по использованию HTML, CSS, JavaScript: сайт. – URL: https://developer.mozilla.org/ (дата обращения: 05.01.2024)
6. Соединение с базой данных: сайт. – URL: https://lite.host/faq/hosting/soedinenie-bazoy-dannih?ysclid=lrdlj2xa3t541902095 (дата обращения: 12.01.2024)
7. PHP: сайт. – URL: https://www.php.net/manual/ru/intro-whatis.php (дата обращения: 11.01.2024)
8. Кожемякин А. HTML и CSS в примерах. Создание Web-страниц: учебник / А. Кожемякин. - М.: Альтекс-А, 2011. - 415 с.