Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт управления бизнес-процессами

Кафедра «Бизнес информатика и моделирование бизнес-процессов»

**КУРСОВАЯ РАБОТА**

«Создание web-приложения с применением современных web-технологий» на тему «Театр»»

тема

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

подпись, дата инициалы, фамилия

Студент УБ22-08Б, 432218194 \_\_\_\_\_\_\_\_ Д.А. Филимонова

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc156580927)

[1 Теоретическая часть 5](#_Toc156580928)

[2 Практическая часть 6](#_Toc156580929)

[2.1 Требования и описание реализации 6](#_Toc156580930)

[2.2 Общая структура 7](#_Toc156580931)

[2.3 База данных сайта театр 7](#_Toc156580932)

[2.4 Регистрация 10](#_Toc156580933)

[2.5 Авторизация 15](#_Toc156580934)

[2.6 Личный кабинет 18](#_Toc156580935)

[2.7 Админ – панель 23](#_Toc156580936)

[2.8 Информация о сеансе 27](#_Toc156580937)

[2.9 Детали пользователя 33](#_Toc156580938)

[ЗАКЛЮЧЕНИЕ 39](#_Toc156580939)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 41](#_Toc156580940)

# ВВЕДЕНИЕ

Веб-программирование - это процесс создания и поддержки веб-сайтов с использованием различных технологий и языков программирования. Эта область программирования играет важную роль в современном мире, так как веб-сайты являются неотъемлемой частью коммуникации и представления информации в интернете.

Одной из важных областей применения веб-программирования является разработка сайтов театра. Сайты театра имеют свои особенности, так как они должны передать атмосферу и эстетику самого театра, а также предоставить посетителям всю необходимую информацию о представлениях, актерах, билетах и так далее.

Веб-программирование позволяет создавать сайты театра, которые могут включать в себя интерактивные элементы, такие как расписание представлений, онлайн-бронирование билетов, видео подкасты или трансляции спектаклей в режиме реального времени. Также возможно создание специальных страниц для актеров, режиссеров и других создателей спектаклей, на которых можно разместить их биографии, интервью и другую дополнительную информацию.

Благодаря веб-программированию сайты театра могут стать современными платформами для продвижения и привлечения новой аудитории, а также инструментами удобной и оперативной коммуникации с посетителями. Реализация таких сайтов требует знания и умения работать с различными языками программирования, такими как HTML, CSS, JavaScript, MySQL, PHP а также понимание принципов дизайна и пользовательского опыта.

Целью курсовой работы является разработка сайта, используя технологии HTML, CSS, Javascript, PHP, MySQL, на тему «Театр», а также разработка базы данных для управления пользователями, используя СУБД MySQL.

Объектом исследования является web-приложение с применением таких web-технологий HTML, CSS, Javascript, PHP, MySQL.

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

Методы исследования включают в себя анализ и синтез информации, проектирование и практическую реализацию.

# 1 Теоретическая часть

Веб-программирование является процессом создания и разработки интернет-приложений и сайтов с использованием различных языков программирования, технологий и инструментов. Сайты являются основным средством коммуникации в онлайн-мире и используются для представления информации, проведения транзакций, взаимодействия с пользователями и выполняют другие функции в различных сферах деятельности.

Пример, языков программирования:

HTML&CSS: HTML (HyperText Markup Language) - это язык разметки, который используется для создания веб-страниц. CSS (Cascading Style Sheets) - это язык стилей, который используется для определения внешнего вида веб-страниц, созданных с помощью HTML.

JavaScript: JavaScript - это язык программирования, который используется для создания интерактивных элементов на веб-страницах. Он позволяет добавлять функциональность, обработку событий, взаимодействие с пользователями и другие динамические элементы на странице.

PHP: PHP (Hypertext Preprocessor) - это язык программирования, который используется для разработки веб-приложений и динамических веб-страниц. Он может быть встроен непосредственно в HTML-код и выполнять действия на сервере, такие как обработка форм, взаимодействие с базами данных и генерация динамического контента.

MySQL: MySQL - это открытая реляционная система управления базами данных (СУБД). Она используется для хранения и управления данными на сервере. MySQL позволяет создавать, изменять и удалять таблицы, а также выполнять запросы для получения, изменения или удаления информации из базы данных.

# 2 Практическая часть

## 2.1 Требования и описание реализации

Создать web-приложение согласно выбранной теме, используя технологии HTML, CSS, Javascript, PHP, MySQL. Требуется реализовать добавление, редактирование, просмотр и удаление пользователей. Также предусмотреть функции авторизации и регистрации.

**Описание реализации:**

- неавторизированные пользователи должны иметь возможность регистрироваться. (форма добавления пользователя)

- авторизированные пользователи разделить на 2 группы: администраторы и авторизированные пользователи.

- администраторы имеют право просматривать, добавлять, редактировать и удалять пользователей.

- авторизированные пользователи имеют право просматривать и редактировать только свои данные.

- при регистрации пользователю присваивается статус авторизированного пользователя.

**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

Open Server устанавливаем для возможности локального воспроизведения веб-приложений. Встроенный функционал позволяет создавать базу данных с кодировкой utf8\_general\_ci. (Рисунок 1)

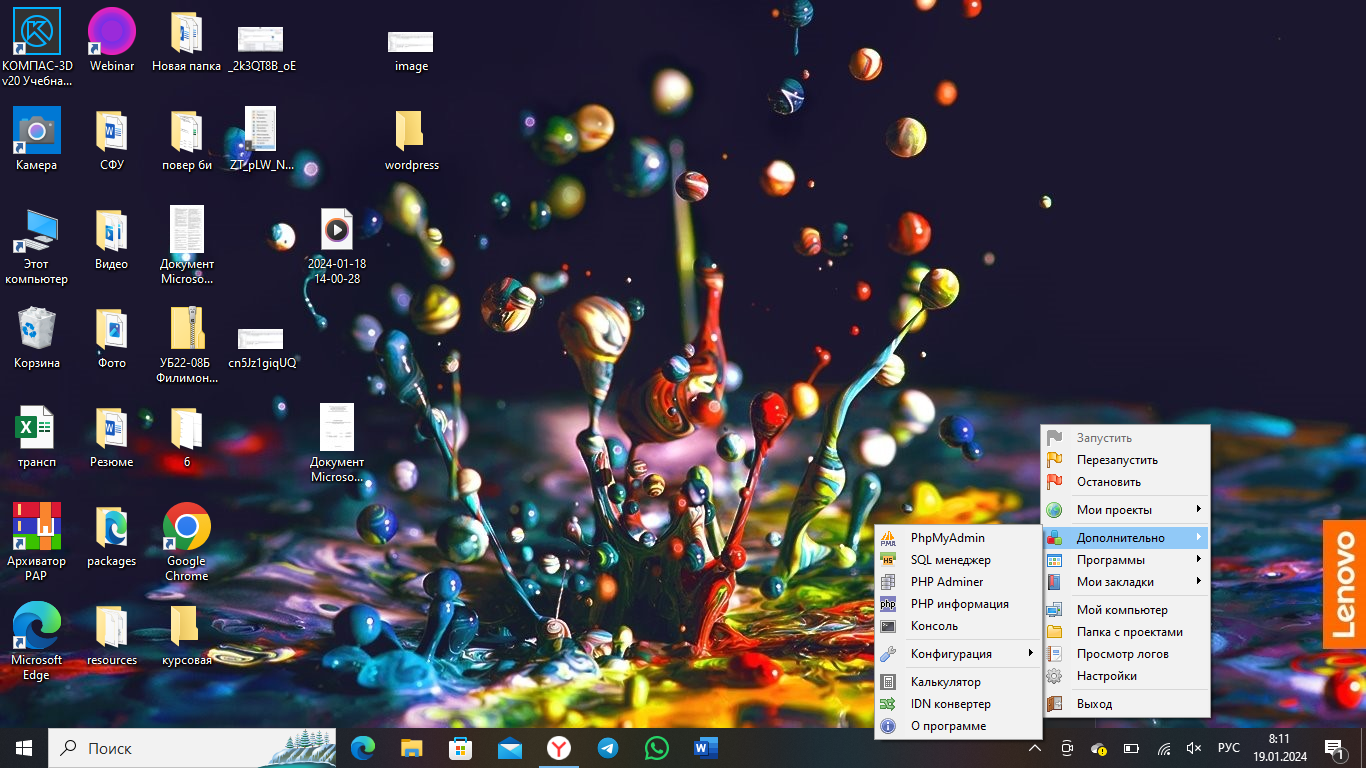


Рисунок 1 — Open server: встроенный функционал

## 2.3 База данных сайта театр

**Листинг создания базы данных:**

CREATE DATABASE IF NOT EXISTS theater;

USE theater;

CREATE TABLE performances (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    title VARCHAR(255) NOT NULL,

    description TEXT

);

CREATE TABLE halls (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    name VARCHAR(255) NOT NULL

);

CREATE TABLE sessions (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    performance\_id INT,

    hall\_id INT,

    date\_time DATETIME NOT NULL,

    FOREIGN KEY (performance\_id) REFERENCES performances(id),

    FOREIGN KEY (hall\_id) REFERENCES halls(id)

);

CREATE TABLE users (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    username VARCHAR(50) unique NOT NULL,

    email VARCHAR(255) unique not null,

    password\_hash VARCHAR(255) NOT NULL

);

CREATE TABLE bookings (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    user\_id INT,

    session\_id INT,

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

    FOREIGN KEY (session\_id) REFERENCES sessions(id)

);

CREATE TABLE reviews (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    user\_id INT,

    performance\_id INT,

    review TEXT,

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

    FOREIGN KEY (performance\_id) REFERENCES performances(id)

);

CREATE TABLE roles (

    id INT PRIMARY KEY AUTO\_INCREMENT,

    name VARCHAR(50) NOT NULL

);

INSERT INTO roles (name) VALUES ('user'), ('admin');

CREATE TABLE user\_roles (

    user\_id INT,

    role\_id INT,

    PRIMARY KEY (user\_id, role\_id),

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

    FOREIGN KEY (role\_id) REFERENCES roles(id)

);

*Название базы данных* **theatre** *с кодировкой* **utf8\_general\_ci.** (Рисунок 2)

1. Таблица **"performances"** используется для хранения информации о спектаклях, которые можно заказать в театре. Она содержит следующие столбцы: **id** (уникальный идентификатор спектакля), **title** (название спектакля), **description** (описание спектакля).

2. Таблица **"halls"** используется для хранения информации о залах театра. Она содержит следующие столбцы: **id** (уникальный идентификатор зала), **name** (название зала).

3. Таблица **"sessions"** используется для хранения информации о сеансах спектаклей. Она содержит следующие столбцы: **id** (уникальный идентификатор сеанса), **performance\_id** (ссылка на спектакль из таблицы **"performances"**), **hall\_id** (ссылка на зал из таблицы **"halls"**), **date\_time** (дата и время сеанса).

4. Таблица **"users"** используется для хранения информации о пользователях системы. Она содержит следующие столбцы: **id** (уникальный идентификатор пользователя), **username** (имя пользователя), **email** (электронная почта пользователя), **password\_hash** (хэш пароля пользователя).

5. Таблица **"bookings"** используется для хранения информации о бронированиях пользователей. Она содержит следующие столбцы: id (уникальный идентификатор бронирования), **user\_id** (ссылка на пользователя из таблицы **"users"**), **session\_id** (ссылка на сеанс из таблицы **"sessions"**).

6. Таблица **"reviews"** используется для хранения информации о отзывах пользователей о спектаклях. Она содержит следующие столбцы: id (уникальный идентификатор отзыва), **user\_id** (ссылка на пользователя из таблицы **"users"**), **performance\_id** (ссылка на спектакль из таблицы **"performances"**), **review** (текст отзыва).

7. Таблица **"roles"** используется для хранения информации о ролях пользователей системы. Она содержит следующие столбцы: **id** (уникальный идентификатор роли), **name** (название роли).

8. Таблица **"user\_roles"** используется для связи пользователей с их ролями. Она содержит следующие столбцы: **user\_id** (ссылка на пользователя из таблицы **"users"**), **role\_id** (ссылка на роль из таблицы **"roles"**).



Рисунок 2 – База данных

## 2.4 Регистрация

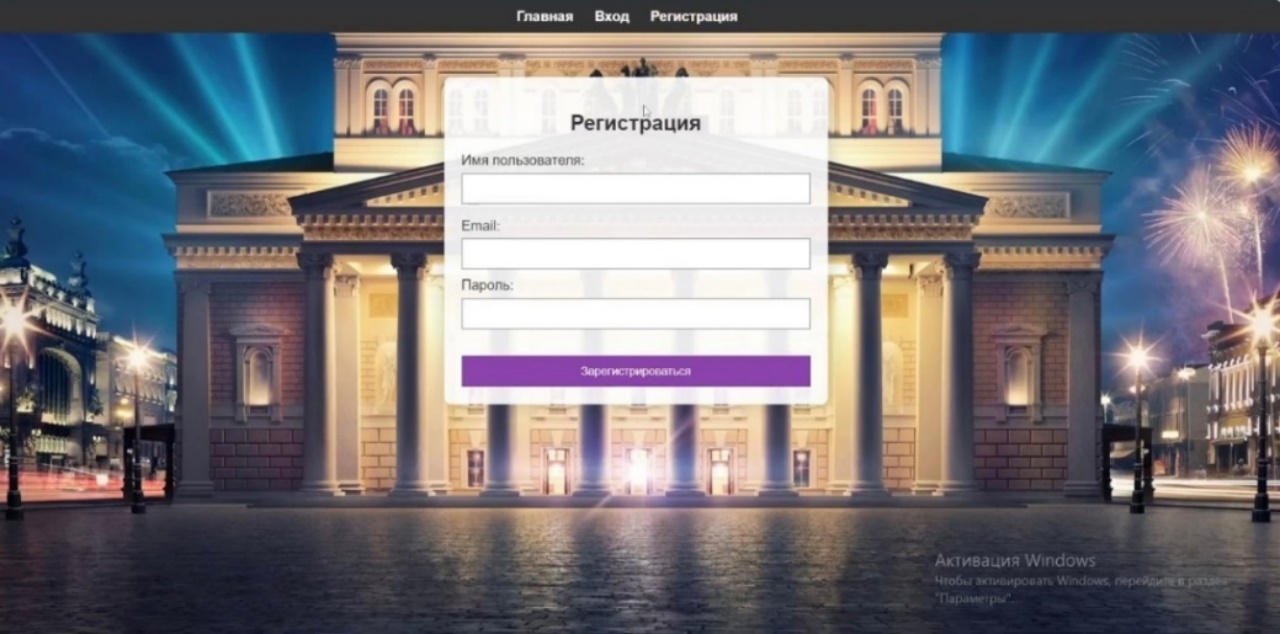


Рисунок 3 – Форма регистрации

Код представляет собой PHP-скрипт, который обрабатывает регистрацию пользователя.

1. Скрипт запускается с проверки того, вошел ли пользователь в систему. Если это так, он перенаправляет их на страницу профиля.

2. Затем он определяет детали подключения к базе данных и устанавливает соединение с базой данных MySQL, используя расширение PDO (PHP Data Objects).

3. Скрипт проверяет, является ли метод запроса POST (т.е. форма была отправлена). Если это так, он извлекает имя пользователя, адрес электронной почты и пароль из данных формы.

4. Затем он проверяет, существует ли имя пользователя или адрес электронной почты в базе данных. Если это так, он выводит сообщение об ошибке и останавливает выполнение скрипта.

5. Если имя пользователя и адрес электронной почты уникальны, скрипт вставляет новую запись пользователя в таблицу "пользователи", используя предоставленные имя пользователя, адрес электронной почты и хэшированный пароль.

6. Он также вставляет соответствующую запись в таблицу "user\_roles", присваивая пользователю роль с идентификатором 1 (при условии, что идентификатор роли относится к роли обычного пользователя).

7. Наконец, если регистрация прошла успешно, скрипт перенаправляет пользователя на страницу входа в систему.

8. В случае возникновения каких-либо ошибок в процессе регистрации скрипт выводит сообщение об ошибке и останавливает выполнение скрипта.

9. HTML-разметка под PHP-кодом представляет собой регистрационную форму, которая содержит поля ввода имени пользователя, электронной почты и пароля.

10. Также включен некоторый код JavaScript, который проверяет вводимые данные перед отправкой формы. Он проверяет, содержит ли имя пользователя 3-16 буквенно-цифровых символов или подчеркиваний и содержит ли пароль не менее 8 символов, включая по крайней мере одну букву и одну цифру.

11. HTML-разметка также включает ссылку на CSS-файл для оформления регистрационной формы.

В целом, этот скрипт обрабатывает процесс регистрации, проверяя вводимые данные в форму, проверяя существующие имена пользователей или электронные письма, вставляя новую запись пользователя в базу данных и перенаправляя пользователя на страницу входа в систему.

**Листинг регистрации:**

<?php

session\_start();

if (isset($\_SESSION['user\_id'])) {

    header("Location: profile.php");

    exit();

}

$host = 'localhost';

$db\_name = 'theater';

$username = 'root';

$password = '';

try {

    $pdo = new PDO("mysql:host=$host;dbname=$db\_name;charset=utf8", $username, $password);

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

} catch (PDOException $e) {

    die("Ошибка подключения к базе данных: " . $e->getMessage());

}

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

    $username = $\_POST['username'];

    $email = $\_POST['email'];

    $password = password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

    try {

        $check\_username\_query = "SELECT \* FROM users WHERE username=:username";

        $stmt = $pdo->prepare($check\_username\_query);

        $stmt->bindParam(':username', $username, PDO::PARAM\_STR);

        $stmt->execute();

        if ($stmt->rowCount() > 0) {

            die("Имя пользователя уже занято.");

        }

        $check\_email\_query = "SELECT \* FROM users WHERE email=:email";

        $stmt = $pdo->prepare($check\_email\_query);

        $stmt->bindParam(':email', $email, PDO::PARAM\_STR);

        $stmt->execute();

        if ($stmt->rowCount() > 0) {

            die("Адрес почты уже используется.");

        }

        $insert\_user\_query = "INSERT INTO users (username, email, password\_hash) VALUES (:username, :email, :password)";

        $stmt = $pdo->prepare($insert\_user\_query);

        $stmt->bindParam(':username', $username, PDO::PARAM\_STR);

        $stmt->bindParam(':email', $email, PDO::PARAM\_STR);

        $stmt->bindParam(':password', $password, PDO::PARAM\_STR);

        $stmt->execute();

        $user\_id = $pdo->lastInsertId();

        $insert\_user\_role\_query = "INSERT INTO user\_roles (user\_id, role\_id) VALUES (:user\_id, 1)";

        $stmt = $pdo->prepare($insert\_user\_role\_query);

        $stmt->bindParam(':user\_id', $user\_id, PDO::PARAM\_INT);

        $stmt->execute();

        header("Location: login.php");

        exit();

    } catch (PDOException $e) {

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

    }

}

$pdo = null;

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Регистрация</title>

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

</head>

<body>

<?php include "layout.php"?>

<div class="container">

    <h2>Регистрация</h2>

    <form action="registration.php" method="post" onsubmit="return validateForm()">

        <label for="username">Имя пользователя:</label>

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

        <p id="usernameError" class="error"></p>

        <label for="email">Email:</label>

        <input type="email" id="email" name="email" required>

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

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

        <p id="passwordError" class="error"></p>

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

    </form>

</div>

<script>

 function validateForm() {

 var username = document.getElementById('username').value;

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

 var usernameRegex = /^[a-zA-Z0-9\_]{3,16}$/;

 if (!usernameRegex.test(username)) {

            document.getElementById('usernameError').innerText = 'Имя пользователя должно содержать от 3 до 16 символов: буквы, цифры, знаки подчеркивания';

 return false;

        } else {

            document.getElementById('usernameError').innerText = '';

        }

 var passwordRegex = /^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{8,}$/;

if (!passwordRegex.test(password)) {

            document.getElementById('passwordError').innerText = 'Пароль должен содержать минимум 8 символов, хотя бы одну цифру и одну букву';

 return false;

        } else {

            document.getElementById('passwordError').innerText = '';

        }

        return true;

    }

</script>

</body>

</html>

## 2.5 Авторизация

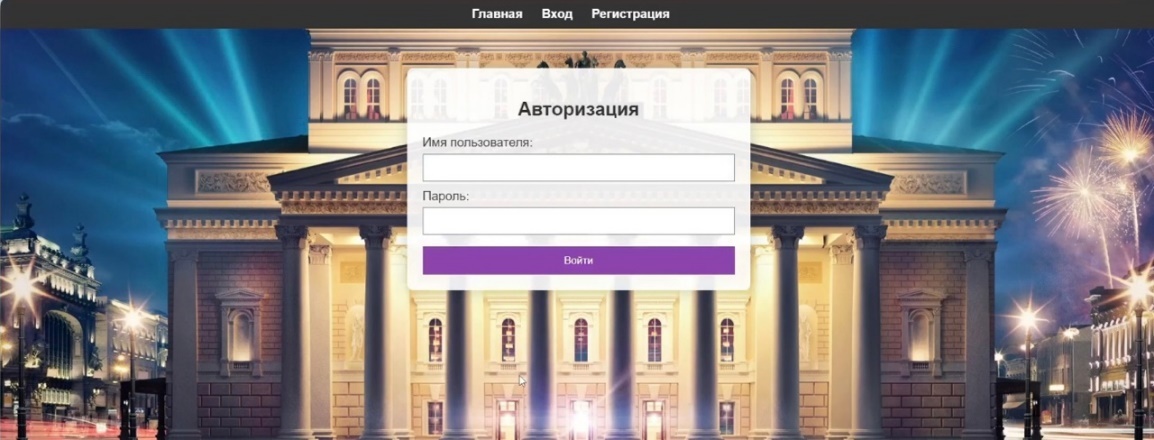


Рисунок 4 – Форма авторизации

Это PHP-код, который обрабатывает аутентификацию пользователя и функции входа в систему.

1. Код начинается с проверки того, что пользователь уже вошел в систему, используя переменную сеанса 'user\_id'. Если пользователь вошел в систему, он перенаправляет его на страницу профиля.

2. Далее код устанавливает соединение с базой данных MySQL с помощью PDO (PHP Data Objects) и устанавливает режим обработки ошибок в exception.

3. Если методом HTTP-запроса является POST, он извлекает значения имени пользователя и пароля из формы отправки.

4. Затем он подготавливает запрос SELECT для извлечения информации о пользователе из таблицы "пользователи" на основе предоставленного имени пользователя. Запрос выполняется, и, если пользователь найден, он проверяет пароль с помощью функции password\_verify.

5. Если пароль подтвержден, он присваивает переменным сеанса 'user\_id', 'username' и 'user\_email' соответствующие значения и перенаправляет пользователя на страницу профиля.

6. Если имя пользователя или пароль указаны неверно, отображается сообщение об ошибке.

7. Если во время процесса аутентификации или подключения к базе данных возникает какая-либо ошибка, отображается сообщение об ошибке.

Наконец, подключение к базе данных закрыто.

Код также включает HTML-разметку для отображения формы входа в систему и сообщений об ошибках. Включен файл "layout.php", который содержит общие элементы макета, используемые на нескольких страницах.

**Листинг аторизации:**

<?php

session\_start();

if (isset($\_SESSION['user\_id'])) {

    header("Location: profile.php");

    exit();

}

$host = 'localhost';

$db\_name = 'theater';

$username = 'root';

$password = '';

try {

    $pdo = new PDO("mysql:host=$host;dbname=$db\_name;charset=utf8", $username, $password);

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

} catch (PDOException $e) {

    die("Ошибка подключения к базе данных: " . $e->getMessage());

}

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

    $username = $\_POST['username'];

    $password = $\_POST['password'];

    try {

        $find\_user\_query = "SELECT \* FROM users WHERE username=:username";

        $stmt = $pdo->prepare($find\_user\_query);

        $stmt->bindParam(':username', $username, PDO::PARAM\_STR);

        $stmt->execute();

        if ($stmt->rowCount() == 1) {

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

            if (password\_verify($password, $user['password\_hash'])) {

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

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

                $\_SESSION['user\_email'] = $user['email'];

                header("Location: profile.php");

                exit();

            } else {

                $error\_message = "Неверное имя пользователя или пароль.";

            }

        } else {

            $error\_message = "Неверное имя пользователя или пароль.";

        }

    } catch (PDOException $e) {

        die("Ошибка при авторизации: " . $e->getMessage());

    }

}

$pdo = null;

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Авторизация</title>

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

</head>

<body>

<?php include "layout.php";?>

<div class="container">

<h2>Авторизация</h2>

<?php if (isset($error\_message)) : ?>

<p style="color: red;"><?php echo $error\_message; ?></p>

 <?php endif; ?>

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

     <label for="username">Имя пользователя:</label>

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

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

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

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

    </form>

</div>

</body>

</html>

## 2.6 Личный кабинет

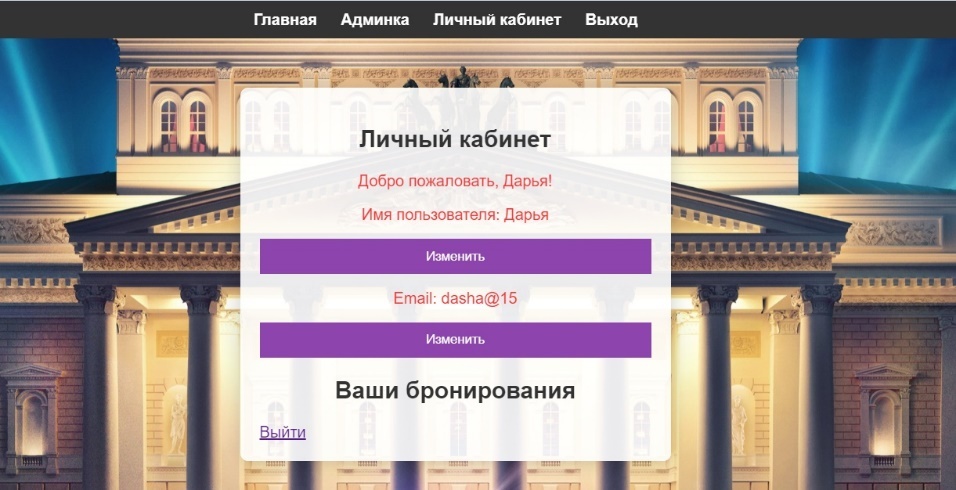


Рисунок 5 – Форма личного кабинета

Этот код написан на PHP и выполняет различные задачи, связанные с профилями пользователей и бронированиями. Описание кода:

1. Код запускается с помощью функции session\_start() для запуска новой сессии.

2. Он подключается к базе данных MySQL с использованием расширения PDO (PHP Data Objects). Сведения о подключении к базе данных хранятся в переменных $host, $db\_name, $username и $password. Если во время подключения к базе данных возникает какая-либо ошибка, она перехватывает исключение с помощью блока try-catch и выводит сообщение об ошибке.

3. Код проверяет, установлена ли переменная сеанса user\_id. Если не установлена, он перенаправляет пользователя на страницу входа в систему и завершает работу скрипта.

4. Он извлекает данные пользователя (имя пользователя и адрес электронной почты) из базы данных на основе переменной сеанса user\_id.

5. Код определяет две функции: fetchuserbookings() и cancelbooking().

6. Функция fetchuserbookings() извлекает все бронирования, сделанные пользователем, из базы данных и возвращает их в виде ассоциативного массива.

7. Функция cancelbooking() удаляет бронирование из базы данных на основе предоставленного идентификатора бронирования.

8. Он вызывает функцию fetchuserbookings() для извлечения бронирований пользователя и сохраняет их в переменной $userbookings.

9. Если отправлен запрос POST и установлен параметр cancel\_booking, он вызывает функцию cancelbooking() с указанным идентификатором бронирования и перенаправляет пользователя обратно на страницу профиля.

10. Если отправляется запрос POST, он проверяет, задан ли параметр new\_username. Если задан, он проверяет, занято ли новое имя пользователя другим пользователем в базе данных. Если не введено, он обновляет имя пользователя в базе данных, обновляет переменную сеанса username и перенаправляет пользователя обратно на страницу профиля. Если имя пользователя уже введено, он отображает сообщение об ошибке и завершает работу скрипта.

11. Аналогично, если установлен параметр new\_email, он проверяет, не занят ли новый адрес электронной почты уже другим пользователем. Если не принято, он обновляет адрес электронной почты пользователя в базе данных, обновляет переменную сеанса user\_email и перенаправляет пользователя обратно на страницу профиля. Если электронное письмо уже принято, он отображает сообщение об ошибке и завершает работу скрипта.

В целом, этот код обрабатывает аутентификацию пользователя, извлекает пользовательские данные из базы данных, позволяет пользователям просматривать и отменять свои бронирования, а также обновлять свое имя пользователя и адрес электронной почты.

**Листинг личного кабинета:**

<?php

session\_start();

try {

    $pdo = new PDO("mysql:host=$host;dbname=$db\_name;charset=utf8", $username, $password);

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

} catch (PDOException $e) {

    die("Ошибка подключения к базе данных: " . $e->getMessage());

}

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

    header("Location: login.php");

    exit();

}

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

$user\_query = "SELECT username, email FROM users WHERE id = :user\_id";

$stmt = $pdo->prepare($user\_query);

$stmt->bindParam(':user\_id', $user\_id, PDO::PARAM\_INT);

$stmt->execute();

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

if ($user\_data) {

    $username = $user\_data['username'];

    $user\_email = $user\_data['email'];

} else {

    header("Location: login.php");

    exit();

}

function fetchUserBookings($pdo, $userId)

{

    $query = "SELECT bookings.\*, sessions.date\_time, performances.title as performance\_title

              FROM bookings

              JOIN sessions ON bookings.session\_id = sessions.id

              JOIN performances ON sessions.performance\_id = performances.id

              WHERE bookings.user\_id = :user\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

    $stmt->execute();

    return $stmt->fetchAll(PDO::FETCH\_ASSOC);

}

function cancelBooking($pdo, $bookingId)

{

    $query = "DELETE FROM bookings WHERE id = :booking\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':booking\_id', $bookingId, PDO::PARAM\_INT);

    $stmt->execute();

}

$userBookings = fetchUserBookings($pdo, $\_SESSION['user\_id']);

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['cancel\_booking'])) {

    $bookingIdToCancel = $\_POST['cancel\_booking'];

    cancelBooking($pdo, $bookingIdToCancel);

    header("Location: profile.php");

    exit();

}

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

    if (isset($\_POST['new\_username'])) {

        $new\_username = $\_POST['new\_username'];

        $check\_username\_query = "SELECT COUNT(\*) FROM users WHERE username = :new\_username AND id != :user\_id";

        $stmt = $pdo->prepare($check\_username\_query);

        $stmt->bindParam(':new\_username', $new\_username, PDO::PARAM\_STR);

        $stmt->bindParam(':user\_id', $user\_id, PDO::PARAM\_INT);

        $stmt->execute();

        if ($stmt->fetchColumn() == 0) {

            $update\_username\_query = "UPDATE users SET username = :new\_username WHERE id = :user\_id";

            $stmt = $pdo->prepare($update\_username\_query);

            $stmt->bindParam(':new\_username', $new\_username, PDO::PARAM\_STR);

            $stmt->bindParam(':user\_id', $user\_id, PDO::PARAM\_INT);

            $stmt->execute();

            $\_SESSION['username'] = $new\_username;

            header("Location: profile.php");

            exit();

        } else {

            die("Имя уже занято");

        }

    }

    if (isset($\_POST['new\_email'])) {

        $new\_email = $\_POST['new\_email'];

        $check\_email\_query = "SELECT COUNT(\*) FROM users WHERE email = :new\_email AND id != :user\_id";

        $stmt = $pdo->prepare($check\_email\_query);

        $stmt->bindParam(':new\_email', $new\_email, PDO::PARAM\_STR);

        $stmt->bindParam(':user\_id', $user\_id, PDO::PARAM\_INT);

        $stmt->execute();

        if ($stmt->fetchColumn() == 0) {

            $update\_email\_query = "UPDATE users SET email = :new\_email WHERE id = :user\_id";

            $stmt = $pdo->prepare($update\_email\_query);

            $stmt->bindParam(':new\_email', $new\_email, PDO::PARAM\_STR);

            $stmt->bindParam(':user\_id', $user\_id, PDO::PARAM\_INT);

            $stmt->execute();

            $\_SESSION['user\_email'] = $new\_email;

            header("Location: profile.php");

            exit();

        } else {

            die("Адрес почты уже занят");

        }

    }

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Личный кабинет</title>

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

</head>

<body>

<?php include "layout.php";?>

<div class="container">

    <h2>Личный кабинет</h2>

    <p>Добро пожаловать, <?php echo $username; ?>!</p>

    <p>Имя пользователя: <?php echo $username; ?>

        <button onclick="toggleForm('changeUsernameForm')">Изменить</button>

    </p>

    <form id="changeUsernameForm" action="profile.php" method="post" style="display: none;">

        <label for="new\_username">Новое имя пользователя:</label>

        <input type="text" id="new\_username" name="new\_username" required>

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

    </form>

    <p>Email: <?php echo $user\_email; ?>

        <button onclick="toggleForm('changeEmailForm')">Изменить</button>

    </p>

    <!-- Форма для изменения адреса электронной почты -->

    <form id="changeEmailForm" action="profile.php" method="post" style="display: none;">

   <label for="new\_email">Новый Email:</label>

   <input type="email" id="new\_email" name="new\_email" required>

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

    </form>

    <h2>Ваши бронирования</h2>

    <?php foreach ($userBookings as $booking) : ?>

   <div class="booking-section">

   <p><?= $booking['performance\_title'] ?> - <?= $booking['date\_time'] ?></p>

    <form action="profile.php" method="post" style="display: inline;">

    <input type="hidden" name="cancel\_booking" value="<?= $booking['id'] ?>">

    <button type="submit">Отменить бронь</button>

    </form>

    </div>

    <?php endforeach; ?>

    <a href="/methods/logout.php">Выйти</a>

</div>

<script>

    function toggleForm(formId) {

var form = document.getElementById(formId);

form.style.display = (form.style.display === 'none') ? 'block' : 'none';

    }

</script>

</body>

</html>

## 2.7 Админ – панель

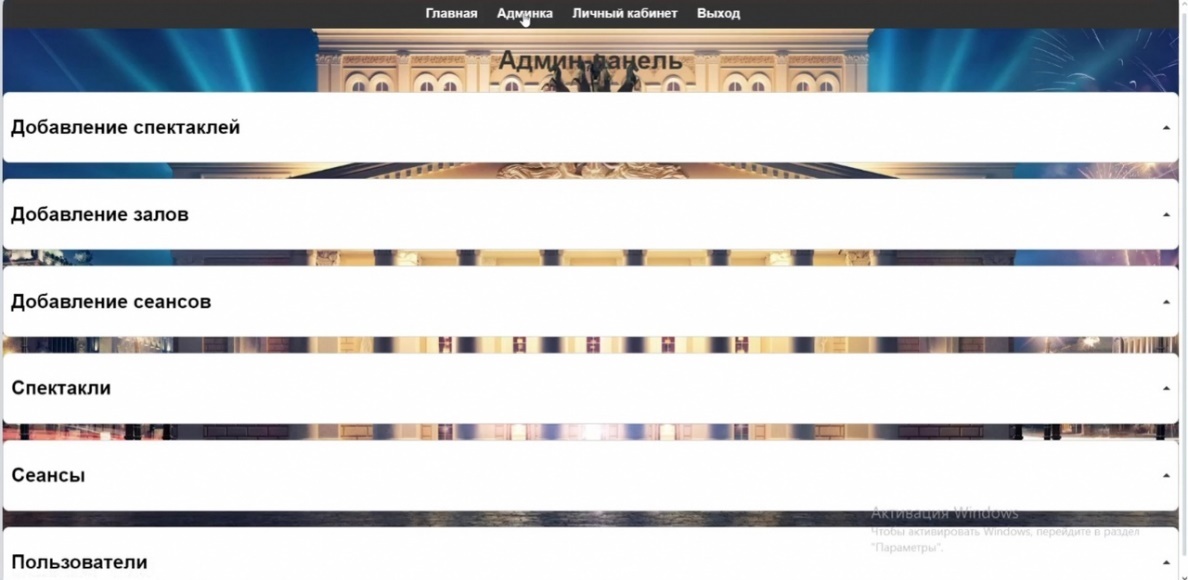


Рисунок 6 – Админ панель

Данный код является PHP скриптом, который выполняет подключение к базе данных и содержит несколько функций для работы с данными о пользователях и спектаклях в театре.

В самом начале скрипта происходит вызов функции session\_start(), которая стартует новую сессию или возобновляет текущую. Затем задаются переменные с данными для подключения к базе данных.

Далее идет блок try-catch, в котором происходит подключение к базе данных и установка атрибута ошибок.

Если происходит ошибка подключения, скрипт прекращает свою работу и выводит сообщение об ошибке.

Функция isuseradmin() принимает подключение к базе данных $pdo и проверяет, является ли текущий пользователь администратором.

Для этого она проверяет наличие пользователя в таблице user\_roles с соответствующей ролью 'admin'.

Если пользователь является администратором, функция возвращает true, в противном случае - false.

Функция fetchperformances() выполняет запрос к базе данных и возвращает все записи из таблицы performances.

Функция fetchsessions() выполняет запрос к базе данных и возвращает все записи из таблицы sessions вместе с соответствующими данными о спектаклях и зале.

Подключение роли администратора, с помощью настроек двух таблиц:

В таблице roles, мы выбираем роль для пользователя

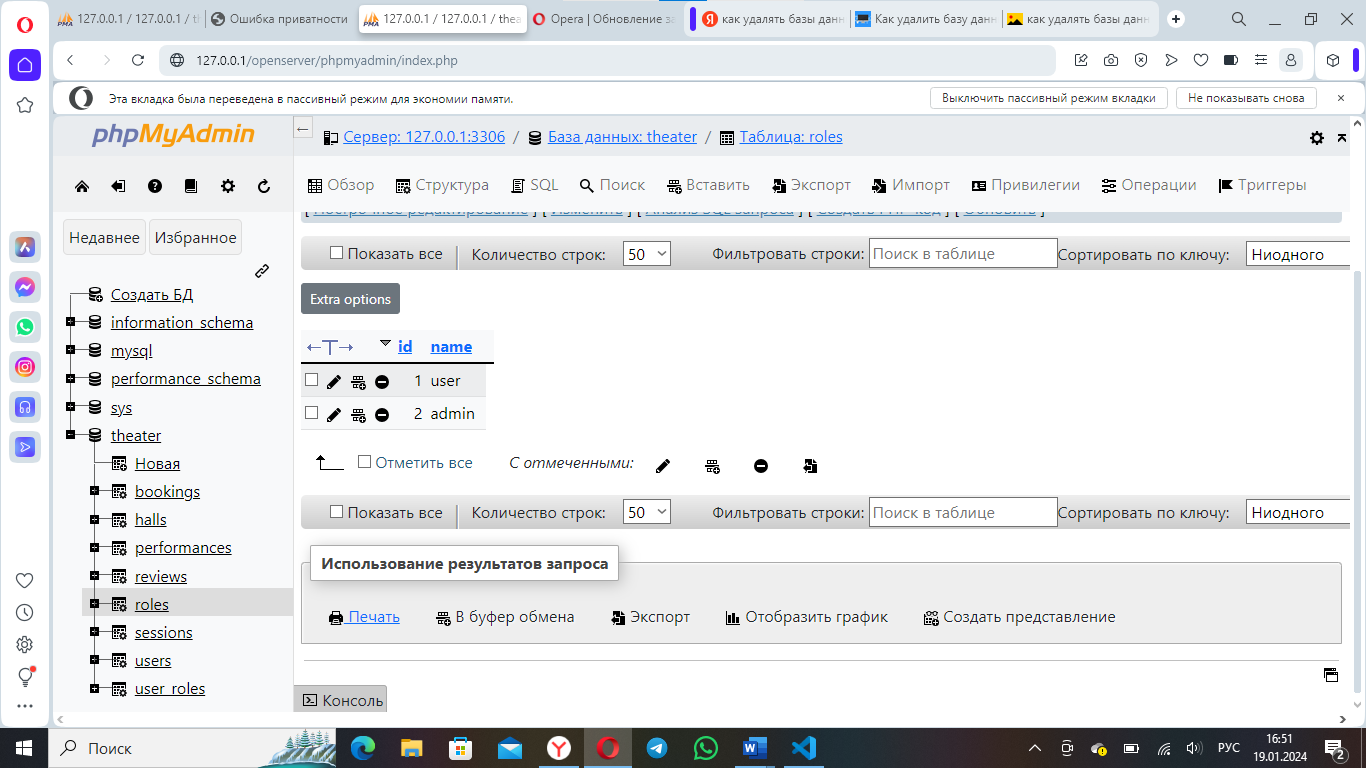


Рисунок 7 – Таблица roles

Из таблицы users, мы берем id пользователя, которому хотим дать роль администратора. (Рисунок 8,9)

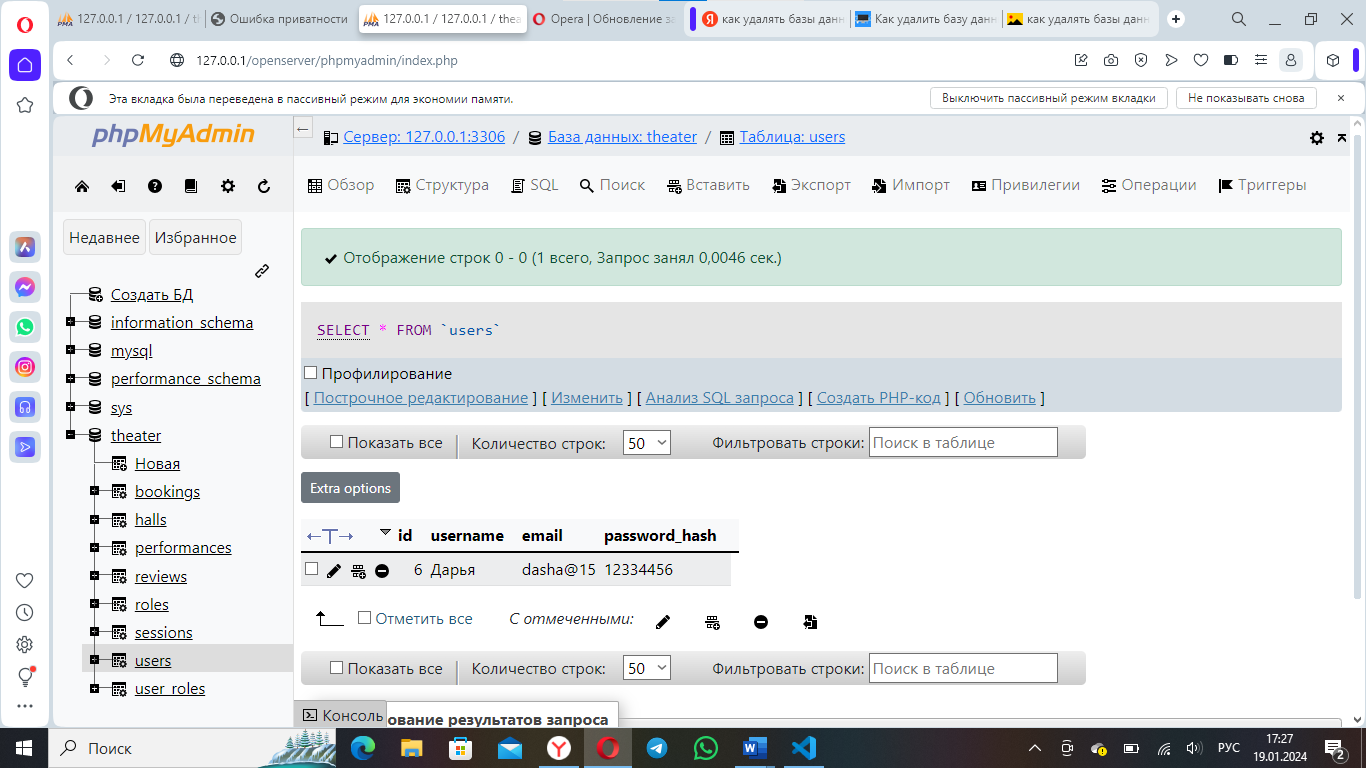


Рисунок 8 – Таблица users

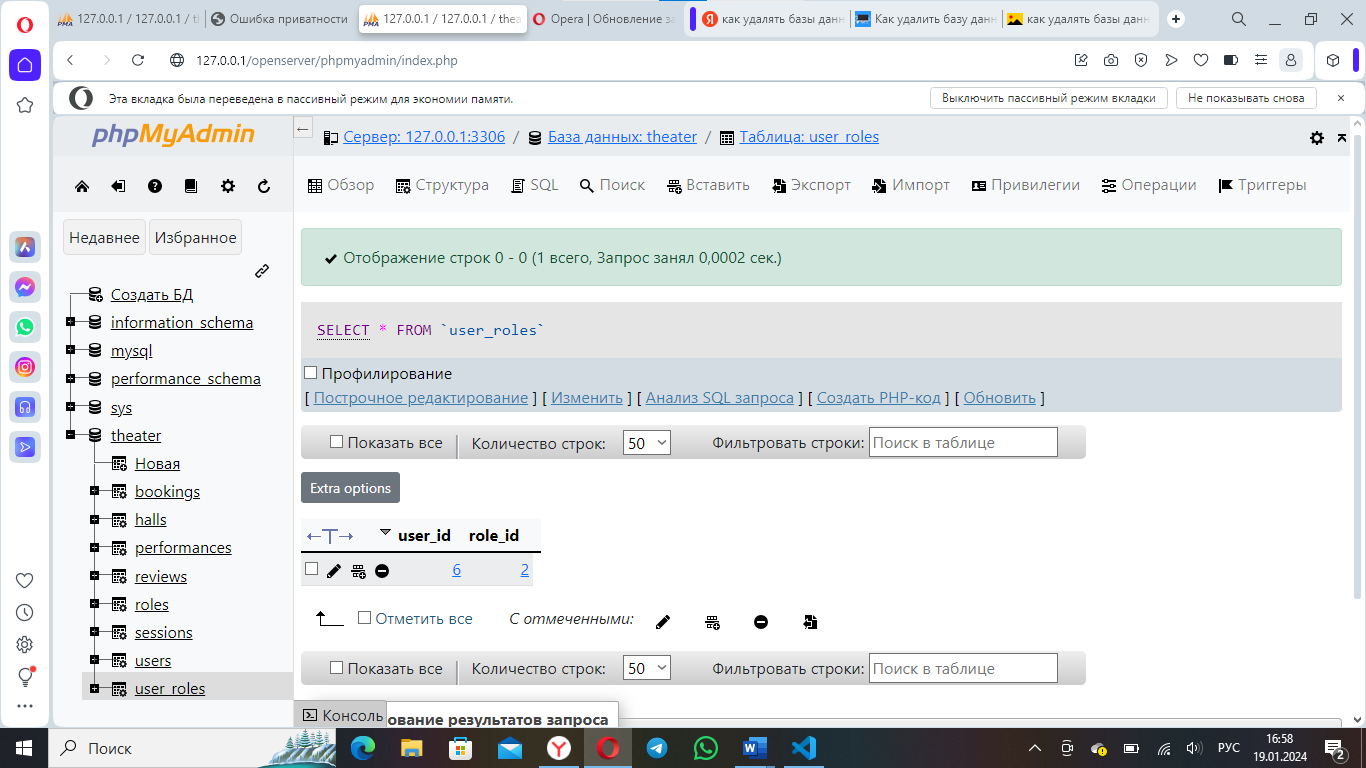


Рисунок 9 – Таблица user\_roles

Теперь пользователь имеет доступ как администратор и может редактировать данные.

**Листинг админ панели:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Админ-панель</title>

    <style>

</head>

<body>

<?php include "layout.php";?>

<h1>Админ-панель</h1>

<div class="add-section" onclick="toggleContent('performances')">

    <h2>Добавление спектаклей <span class="arrow" id="arrowPerformances"></span></h2>

    <div class="add-section-content" id="performances">

   <form action="admin\_panel.php" method="post">

<label for="title">Название представления:</label>

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

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

<textarea id="description" name="description" rows="4" required></textarea>

 <button type="submit" name="add\_performance">Добавить</button>

</form>

</div>

</div>

<div class="add-section" onclick="toggleContent('halls')">

    <h2>Добавление залов <span class="arrow" id="arrowHalls"></span></h2>

    <div class="add-section-content" id="halls">

        <form action="admin\_panel.php" method="post">

       <label for="hall\_name">Название зала:</label>

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

<button type="submit" name="add\_hall">Добавить</button>

</form>

</div>

</div>

<div class="add-section" onclick="toggleContent('sessions')">

    <h2>Добавление сеансов <span class="arrow" id="arrowSessions"></span></h2>

    <div class="add-section-content" id="sessions">

        <form action="admin\_panel.php" method="post">

       <label for="performance\_id">Выберите представление:</label>

     <select id="performance\_id" name="performance\_id" required>

                <?php foreach ($performances as $performance) : ?>

                    <option value="<?php echo $performance['id']; ?>"><?php echo $performance['title']; ?></option>

                <?php endforeach; ?>

            </select>

<label for="hall\_id">Выберите зал:</label>

<select id="hall\_id" name="hall\_id" required>

<?php foreach ($halls as $hall) : ?>

<option value="<?php echo $hall['id']; ?>"><?php echo $hall['name']; ?></option>

                <?php endforeach; ?>

</select>

<label for="date\_time">Дата и время:</label>

<input type="datetime-local" id="date\_time" name="date\_time" required>

<button type="submit" name="add\_session">Добавить</button>

</form>

</div>

</div>

<div class="add-section" onclick="toggleSection('performancelist')">

    <h2>Спектакли <span class="arrow" id="arrowPerformances"></span></h2>

    <div class="add-section-content" id="performancelist">

        <ul>

            <?php foreach ($performances as $performance) : ?>

                <li>

                    <?php echo $performance['title']; ?>

   <form action="admin\_panel.php" method="post" style="display: inline;">

     <input type="hidden" name="performance\_id" value="<?php echo $performance['id']; ?>">

       <button type="submit" name="delete\_performance" class="delete-button">Удалить</button>

</form>

</li>

            <?php endforeach; ?>

        </ul>

    </div>

</div>

<div class="add-section" onclick="toggleSection('sessionlist')">

    <h2>Сеансы <span class="arrow" id="arrowSessions"></span></h2>

    <div class="add-section-content" id="sessionlist">

       <ul>

       <?php foreach ($sessions as $session) : ?>

       <li>

       <?php echo $session['performance\_title'] . ' - ' . $session['hall\_name'] . ' - ' . $session['date\_time']; ?>

       <form action="admin\_panel.php" method="post" style="display: inline;">

       <input type="hidden" name="delete\_session" value="<?php echo $session['id']; ?>">

       <button type="submit" class="delete-button">Удалить</button>

</form>

</li>

<?php endforeach; ?>

</ul>

</div>

</div>

<div class="add-section" onclick="toggleContent('users')">

    <h2>Пользователи <span class="arrow" id="arrowUsers"></span></h2>

    <div class="add-section-content" id="users">

        <ul>

     <?php foreach ($users as $user) : ?>

     <li>

     <a href="/views/user\_details.php?user\_id=<?= $user['id'] ?>">

                        <?= $user['username'] ?>

     </a>

    <form action="admin\_panel.php" method="post" style="display: inline;">

     <input type="hidden" name="delete\_user" value="<?= $user['id'] ?>">

     <button type="submit" class="delete-button">Удалить</button>

     </form>

     </li>

            <?php endforeach; ?>

    </ul>

    </div>

</div>

</body>

</html>

## 2.8 Информация о сеансе

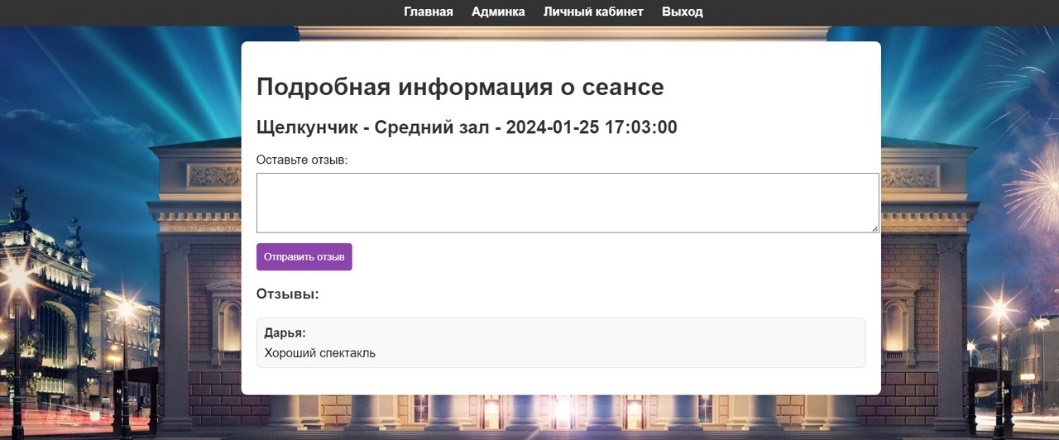


Рисунок 10 – информации о сеансе

Данный код представляет собой PHP-скрипт, который подключается к базе данных MySQL и отображает подробную информацию о сеансе, включая его выступления, залы, дату и время. Он также позволяет пользователям отправлять отзывы о сеансе и отображает существующие отзывы.

1. Скрипт запускается вызовом session\_start() для запуска нового или возобновления существующего сеанса.

2. Сведения о подключении к базе данных (хост, имя базы данных, имя пользователя, пароль) определяются как переменные.

3. Создается объект PDO для подключения к базе данных MySQL с использованием предоставленных сведений о подключении.

4. Если во время подключения к базе данных возникает исключение, отображается сообщение об ошибке и сценарий завершается.

5. Идентификатор сеанса извлекается из параметров запроса ($\_GET['session\_id']) для идентификации просматриваемого сеанса.

6. Если методом запроса является "POST" и отправлена форма submit\_review, сведения об отзыве извлекаются из параметров POST ($\_POST) и вставляются в таблицу отзывов в базе данных. Затем пользователь перенаправляется обратно на страницу сведений о сеансе.

7. Если метод запроса 'GET' и указан идентификатор сеанса, скрипт извлекает сведения о сеансе и отзывы из базы данных, используя подготовленные инструкции. Сведения о сеансе хранятся в переменной $session, а отзывы хранятся в переменной $reviews.

8. Затем генерируется HTML-разметка для отображения сведений о сеансе и отзывов. Если сеанс найден (задано значение $session), отображаются название выступления, название зала и дата/время.

10. Если пользователь вошел в систему (задано значение $\_SESSION['user\_id']), отображается форма для отправки отзыва о сеансе.

11. Существующие отзывы отображаются в цикле с использованием переменной $reviews.

Таким образом, этот код подключается к базе данных, извлекает и отображает сведения о сеансе и отзывы, а также позволяет пользователям отправлять новые отзывы.

**Листинг информации о сеансе:**

<?php

session\_start();

$host = 'localhost';

$db\_name = 'theater';

$username = 'root';

$password = '';

try {

    $pdo = new PDO("mysql:host=$host;dbname=$db\_name;charset=utf8", $username, $password);

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

} catch (PDOException $e) {

    die("Ошибка подключения к базе данных: " . $e->getMessage());

}

$session\_id = $\_GET['session\_id'] ?? null;

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['submit\_review'])) {

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

    $performance\_id = $\_POST['performance\_id'];

    $review = $\_POST['review'];

    // Вставка отзыва в базу данных

    $insert\_review\_query = "INSERT INTO reviews (user\_id, performance\_id, review) VALUES (:user\_id, :performance\_id, :review)";

    $stmt\_review = $pdo->prepare($insert\_review\_query);

    $stmt\_review->bindParam(':user\_id', $user\_id, PDO::PARAM\_INT);

    $stmt\_review->bindParam(':performance\_id', $performance\_id, PDO::PARAM\_INT);

    $stmt\_review->bindParam(':review', $review, PDO::PARAM\_STR);

    $stmt\_review->execute();

    // Перенаправление на страницу сеанса

    header("Location: session\_details.php?session\_id=$session\_id");

    exit();

}

if ($\_SERVER["REQUEST\_METHOD"] == "GET" && isset($\_GET['session\_id'])) {

    $session\_id = $\_GET['session\_id'];

    // Получение информации о сеансе

    $session\_query = "SELECT sessions.\*, performances.title as performance\_title, halls.name as hall\_name

                        FROM sessions

                        JOIN performances ON sessions.performance\_id = performances.id

                        JOIN halls ON sessions.hall\_id = halls.id

                        WHERE sessions.id = :session\_id";

    $stmt\_session = $pdo->prepare($session\_query);

    $stmt\_session->bindParam(':session\_id', $session\_id, PDO::PARAM\_INT);

    $stmt\_session->execute();

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

    // Получение отзывов о сеансе

    $reviews\_query = "SELECT reviews.\*, users.username

                        FROM reviews

                        JOIN users ON reviews.user\_id = users.id

                        WHERE reviews.performance\_id = :performance\_id";

    $stmt\_reviews = $pdo->prepare($reviews\_query);

    $stmt\_reviews->bindParam(':performance\_id', $session['performance\_id'], PDO::PARAM\_INT);

    $stmt\_reviews->execute();

    $reviews = $stmt\_reviews->fetchAll(PDO::FETCH\_ASSOC);

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Подробная информация о сеансе</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

<?php include "layout.php"; ?>

<style>

    body {

        font-family: Arial, sans-serif;

        margin: 0;

        padding: 0;

        background-color: #f4f4f4;

    }

    .container {

        max-width: 800px;

        margin: 20px auto;

        background-color: #fff;

        padding: 20px;

        border-radius: 8px;

        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    }

    h1, h2, h3 {

        color: #333;

    }

    ul {

        list-style-type: none;

        padding: 0;

    }

    li {

        margin-bottom: 10px;

    }

    form {

        margin-top: 20px;

    }

    label {

        display: block;

        margin-bottom: 8px;

    }

    textarea {

        width: 100%;

        padding: 8px;

        margin-bottom: 10px;

    }

    button {

        background-color: #8e44ad;

        color: #fff;

        padding: 10px;

        border: none;

        border-radius: 4px;

        cursor: pointer;

    }

    button:hover {

        background-color: #6c3483;

    }

    h3 {

        margin-top: 20px;

    }

    .reviews-wrapper {

        margin-top: 20px;

    }

    .review {

        background-color: #f9f9f9;

        padding: 10px;

        border-radius: 8px;

        border: 1px solid #ddd;

        margin-bottom: 15px;

    }

    .review .username {

        font-weight: bold;

        color: #333;

    }

    .review .content {

        margin-top: 8px;

    }

    body::before {

        content: "";

        position: fixed;

        top: 0;

        left: 0;

        width: 100%;

        height: 100%;

        background: url('/img/theater.jpg') no-repeat center center fixed;

        background-size: cover;

        z-index: -1;

    }

</style>

<div class="container">

    <h1>Подробная информация о сеансе</h1>

    <?php if (isset($session)) : ?>

        <h2><?php echo $session['performance\_title'] . ' - ' . $session['hall\_name'] . ' - ' . $session['date\_time']; ?></h2>

        <?php if (isset($\_SESSION['user\_id'])) : ?>

        <form action="session\_details.php?session\_id=<?php echo $session\_id; ?>" method="post">

       <input type="hidden" name="performance\_id" value="<?php echo $session['performance\_id']; ?>">

       <label for="review">Оставьте отзыв:</label>

       <textarea id="review" name="review" rows="4" required></textarea>

       <button type="submit" name="submit\_review">Отправить отзыв</button>

       </form>

       <?php endif; ?>

       <h3>Отзывы:</h3>

       <div class="reviews-wrapper">

       <?php foreach ($reviews as $review) : ?>

      <div class="review">

       <span class="username"><?php echo $review['username']; ?>:</span>

       <div class="content"><?php echo $review['review']; ?></div>

   </div>

   <?php endforeach; ?>

   </div>

   <?php else : ?>

<p>Сеанс не найден.</p>

<?php endif; ?>

</div>

</body>

</html>

## 2.9 Детали пользователя

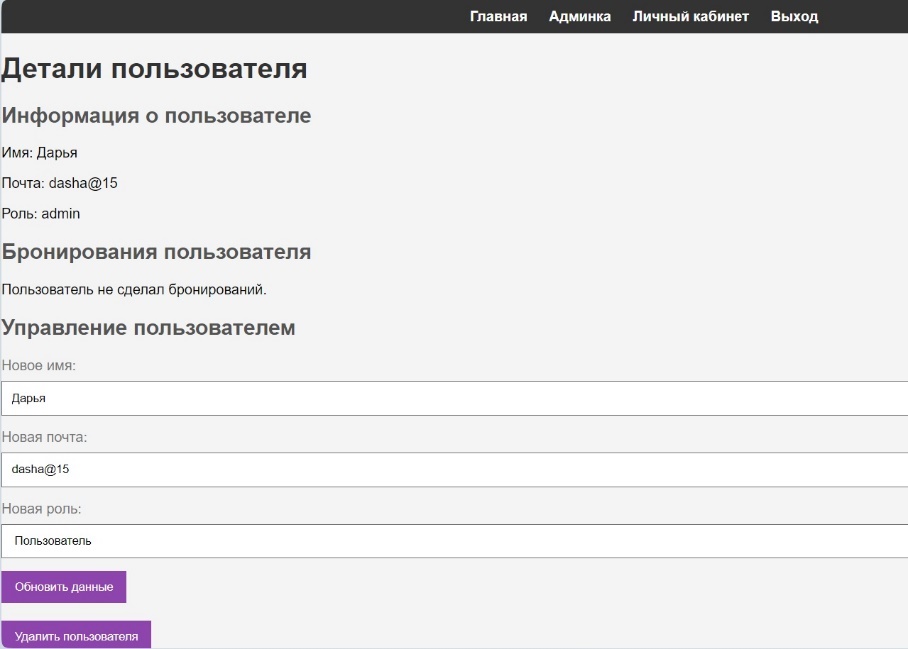


Рисунок 11 – Деталий пользователя

Этот код на языке PHP начинается с инициализации сеанса и установки параметров подключения к базе данных. Затем он проверяет, является ли текущий пользователь администратором, и, если нет, перенаправляет его на страницу входа.

Далее определены функции, включая функции для проверки, является ли пользователь администратором, извлечения данных пользователя, обновления данных пользователя, удаления пользователя и отмены бронирования.

Если метод запроса равен "POST", код проверяет, было ли отправлено обновление данных пользователя, удаление пользователя или отмена бронирования, и вызывает соответствующие функции. Затем он перенаправляет пользователя на соответствующие страницы.

Если в URL указан параметр "user\_id", код извлекает данные пользователя с помощью функции "fetchuserdata" и сохраняет их в переменную "userdata". В противном случае он перенаправляет пользователя на страницу панели администратора.

Также в этом годе представлена верстка HTML страницы, которая отображает детали пользователя. В нем используются различные стили, а также включается файл "layout.php".

**Листинг деталей пользователей:**

<?php

session\_start();

$host = 'localhost';

$db\_name = 'theater';

$username = 'root';

$password = '';

try {

    $pdo = new PDO("mysql:host=$host;dbname=$db\_name;charset=utf8", $username, $password);

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

} catch (PDOException $e) {

    die("Ошибка подключения к базе данных: " . $e->getMessage());

}

if (!isUserAdmin($pdo)) {

    header("Location: login.php");

    exit();

}

function isUserAdmin($pdo)

{

    return isset($\_SESSION['user\_id']) && isAdmin($pdo, $\_SESSION['user\_id']);

}

function isAdmin($pdo, $userId)

{

    $query = "SELECT \* FROM user\_roles WHERE user\_id = :user\_id AND role\_id = (SELECT id FROM roles WHERE name = 'admin')";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

    $stmt->execute();

    return $stmt->rowCount() > 0;

}

function fetchUserData($pdo, $userId)

{

$query = "SELECT users.\*, roles.name as role,

GROUP\_CONCAT(performances.title ORDER BY bookings.id SEPARATOR ', ') as booked\_performances,

GROUP\_CONCAT(bookings.id ORDER BY bookings.id SEPARATOR ', ') as booked\_performances\_ids,

GROUP\_CONCAT(performances.title ORDER BY bookings.id SEPARATOR ', ') as performance\_titles

FROM users

LEFT JOIN user\_roles ON users.id = user\_roles.user\_id

LEFT JOIN roles ON user\_roles.role\_id = roles.id

LEFT JOIN bookings ON users.id = bookings.user\_id

LEFT JOIN sessions ON bookings.session\_id = sessions.id

LEFT JOIN performances ON sessions.performance\_id = performances.id

WHERE users.id = :user\_id

GROUP BY users.id, roles.name";

$stmt = $pdo->prepare($query);

$stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

$stmt->execute();

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

    if (!empty($result['booked\_performances\_ids'])) {

        $bookedPerformancesIds = explode(', ', $result['booked\_performances\_ids']);

        $performanceTitles = explode(', ', $result['performance\_titles']);

    $bookedPerformancesData = array();

        foreach ($bookedPerformancesIds as $index => $bookingId) {

     $bookedPerformancesData[] = array(

                'booking\_id' => $bookingId,

                'performance\_title' => $performanceTitles[$index]

            );

        }

    $result['booked\_performances\_data'] = $bookedPerformancesData;

    } else {

        // Если бронирований нет, устанавливаем пустой массив

    $result['booked\_performances\_data'] = array();

    }

    return $result;

}

function updateUserData($pdo, $userId, $username, $email, $roleId)

{

    $query = "UPDATE users SET username = :username, email = :email WHERE id = :user\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

    $stmt->bindParam(':username', $username, PDO::PARAM\_STR);

    $stmt->bindParam(':email', $email, PDO::PARAM\_STR);

    $stmt->execute();

    // Обновляем роль пользователя

    $query = "UPDATE user\_roles SET role\_id = :role\_id WHERE user\_id = :user\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

    $stmt->bindParam(':role\_id', $roleId, PDO::PARAM\_INT);

    $stmt->execute();

}

function deleteUser($pdo, $userId)

{

    $query = "DELETE FROM bookings WHERE user\_id = :user\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

    $stmt->execute();

    $query = "DELETE FROM users WHERE id = :user\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

    $stmt->execute();

    $query = "DELETE FROM user\_roles WHERE user\_id = :user\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':user\_id', $userId, PDO::PARAM\_INT);

    $stmt->execute();

}

function cancelBooking($pdo, $bookingId)

{

    $query = "DELETE FROM bookings WHERE id = :booking\_id";

    $stmt = $pdo->prepare($query);

    $stmt->bindParam(':booking\_id', $bookingId, PDO::PARAM\_INT);

    $stmt->execute();

}

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

    if (isset($\_POST['update\_user'])) {

        $userIdToUpdate = $\_POST['update\_user'];

        $newUsername = $\_POST['new\_username'];

        $newEmail = $\_POST['new\_email'];

        $newRoleId = $\_POST['new\_role'];

    updateUserData($pdo, $userIdToUpdate, $newUsername, $newEmail, $newRoleId);

    header("Location: user\_details.php?user\_id=$userIdToUpdate");

    exit();

    } elseif (isset($\_POST['delete\_user'])) {

    $userIdToDelete = $\_POST['delete\_user'];

    deleteUser($pdo, $userIdToDelete);

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

    exit();

    } elseif (isset($\_POST['cancel\_booking'])) {

    $bookingIdToCancel = $\_POST['cancel\_booking'];

    cancelBooking($pdo, $bookingIdToCancel);

    header("Location: user\_details.php?user\_id=$\_GET[user\_id]");

    exit();

    }

}

if (isset($\_GET['user\_id'])) {

    $userId = $\_GET['user\_id'];

    $userData = fetchUserData($pdo, $userId);

} else {

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

    exit();

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Детали пользователя</title>

</head>

<body>

<style>

    body {

        font-family: Arial, sans-serif;

        background-color: #f4f4f4;

        margin: 0;

        padding: 0;

    }

    .container {

        max-width: 800px;

        margin: 20px auto;

        background-color: #fff;

        padding: 20px;

        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    }

    h1 {

        color: #333;

    }

    h2 {

        color: #555;

    }

    form {

        margin-top: 20px;

    }

    label {

        display: block;

        margin-bottom: 8px;

        color: #777;

    }

    input, select {

        width: 100%;

        padding: 10px;

        margin-bottom: 15px;

        box-sizing: border-box;

    }

    button {

        background-color: #8e44ad;

        color: #fff;

        padding: 10px 15px;

        border: none;

        cursor: pointer;

    }

    button:hover {

        background-color: #6c3483;

    }

</style>

<?php include "layout.php";?>

<h1>Детали пользователя</h1>

<h2>Информация о пользователе</h2>

<p>Имя: <?= $userData['username'] ?></p>

<p>Почта: <?= $userData['email'] ?></p>

<p>Роль: <?= $userData['role'] ?></p>

<h2>Бронирования пользователя</h2>

<?php if (!empty($userData['booked\_performances'])) : ?>

    <ul>

    <?php foreach (explode(', ', $userData['booked\_performances']) as $performance) : ?>

    <li><?= $performance ?></li>

    <?php endforeach; ?>

    </ul>

<?php else : ?>

    <p>Пользователь не сделал бронирований.</p>

<?php endif; ?>

<h2>Управление пользователем</h2>

<form action="user\_details.php?user\_id=<?= $userId ?>" method="post">

    <label for="new\_username">Новое имя:</label>

    <input type="text" id="new\_username" name="new\_username" value="<?= $userData['username'] ?>" required>

    <br>

    <label for="new\_email">Новая почта:</label>

    <input type="email" id="new\_email" name="new\_email" value="<?= $userData['email'] ?>" required>

    <br>

    <label for="new\_role">Новая роль:</label>

    <select id="new\_role" name="new\_role" required>

    <option value="1" <?= ($userData['role'] == 'Пользователь') ? 'selected' : '' ?>>Пользователь</option>

        <option value="2" <?= ($userData['role'] == 'Администратор') ? 'selected' : '' ?>>Администратор</option>

 </select>

 <br>

 <button type="submit" name="update\_user" value="<?= $userId ?>">Обновить данные</button>

</form>

<form action="user\_details.php" method="post" onsubmit="return confirm('Вы уверены, что хотите удалить этого пользователя?');">

<button type="submit" name="delete\_user" value="<?= $userId ?>">Удалить пользователя</button>

</form>

<h2>Отмена бронирования</h2>

<form action="user\_details.php?user\_id=<?= $userId ?>" method="post">

    <label for="cancel\_booking">Выберите бронь для отмены:</label>

    <select id="cancel\_booking" name="cancel\_booking" required>

    <?php foreach ($userData['booked\_performances\_data'] as $booking) : ?>

    <option value="<?= $booking['booking\_id'] ?>">

    <?= $booking['performance\_title'] ?>

    </option>

    <?php endforeach; ?>

</select>

<br>

<button type="submit">Отменить бронь</button>

</form>

</body>

</html>

# ЗАКЛЮЧЕНИЕ

Web-программирование играет важную роль в современном мире, позволяя создавать интерактивные и функциональные веб-сайты для различных целей. В частности, сайты театров являются отличным примером успешного применения web-программирование.

Сайт театра - это удобный и доступный инструмент для пользователей, который позволяет им получать всю необходимую информацию о спектаклях, актерах, билетах и расписании. Преимущества сайтов театра для пользователей включают:

1. Удобство использования: на сайте театра пользователи могут легко найти все необходимые сведения о предстоящих спектаклях, исходя из своих предпочтений и графика.

2. Быстрый доступ к информации: данные о билетах, ценах и местах проведения актуализируются мгновенно, что позволяет пользователям быть в курсе текущего состояния и сделать выбор, наиболее удовлетворяющий их потребностям.

3. Возможность покупки билетов онлайн: многие сайты театров предоставляют возможность покупки билетов онлайн, что экономит время посетителей и дает возможность избежать очередей.

4. Дополнительные сервисы и контент: пользователи могут получать дополнительную информацию об актерах, режиссерах, постановщиках и самом театре. Также, сайты театров предлагают фото- и видеоматериалы, представляющие атмосферу и характер спектаклей.

В общем говоря, сайты театра создают комфортную среду для пользователей, облегчая процесс выбора и покупки билетов. Они предоставляют возможность быстро и легко ознакомиться с актуальной информацией о спектаклях и позволяют наслаждаться миром театра с помощью интернета.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Веллинг, Л. Разработка веб-приложений с помощью PHP и MySQL / Л.Веллинг, Л. Томсон. М.: Вильямс, 2012. – 848 с.
2. Дакетт, Дж. HTML и CSS. Разработка и дизайн веб-сайтов / Дж. Дакетт – М.: ЭКСМО, 2013. – 480 с.
3. Белунцов В. Новейший самоучитель по созданию Web-страниц / В. Белунцов. - М.: NT Press, 2012. - 185 с.
4. Колисниченко Д.Н. Современный сайт на PHP и JavaScript. - СПб.: Питер, 2009.
5. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство. – СПб.: Питер, 2013. – 512 с..