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 по тематике «Бронирование столиков в ресторане»»

тема

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

подпись, дата инициалы, фамилия

Студент УБ22-08Б, 43221546 \_\_\_\_\_\_\_\_ Е.В. Чистякова

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc154443562)

[1 Теоретическая часть 5](#_Toc154443563)

[1.1 HTML и CSS 5](#_Toc154443564)

[1.2 JavaScript 6](#_Toc154443565)

[1.3 PHP 8](#_Toc154443566)

[1.4 MySQL 10](#_Toc154443567)

[2 Практическая часть 12](#_Toc154443568)

[2.1 Требования и описание реализации 12](#_Toc154443569)

[2.2 Общая структура 13](#_Toc154443570)

[2.3 Создание базы данных 14](#_Toc154443571)

[2.4 Регистрация в web-приложении 17](#_Toc154443572)

[2.5 Авторизация в web-приложении 24](#_Toc154443573)

[2.6 Бронирование столика в web-приложении 27](#_Toc154443574)

[2.7 Личный кабинет пользователя в web-приложении 33](#_Toc154443575)

[2.1.7 Функционал обычного пользователя 33](#_Toc154443576)

[2.2.7 Функционал администратора 41](#_Toc154443577)

[ЗАКЛЮЧЕНИЕ 46](#_Toc154443578)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc154443579)

# ВВЕДЕНИЕ

В настоящее время в связи с глобальным развитием сети Интернет, в программировании все более резко начала выделяться отдельная его отрасль web-программирование. Система общественного питания как никогда нуждается в автоматизации различных внутренних процессов, в частности в процессе бронирования столиков. Web-разработка позволяет сделать высокотехнологичный продукт с интересным функционалом как для пользователя, так и для администратора.

Создание базы данных MySQL для web-приложения по бронированию столиков в ресторане является важным шагом для эффективной работы приложения. Ниже перечислены несколько важных аспектов, которые обосновывают важность создания такой базы данных:

База данных позволяет эффективно управлять данными, хранить информацию о посетителях, дате и времени бронирования, количестве гостей и других атрибутах заказа. Это позволяет упростить процесс поиска и обработки данных, что способствует более эффективному бронированию столиков.

Нельзя не упомянуть о хранении большого объема данных: web-приложение, связанное с бронированием столиков в ресторане, может содержать значительное количество информации о клиентах, столиках, меню, событиях и других связанных данных. База данных MySQL обеспечивает надежное хранение этих данных и позволяет эффективно управлять ими, даже при большом объеме информации.

Поддержка множества пользователей: Бронирование столиков в ресторане может быть доступно для множества пользователей, как для посетителей, так и для администраторов. MySQL обеспечивает многопользовательскую поддержку, позволяя одновременно работать с базой данных нескольким пользователям без проблем с доступом и интеграцией данных.

В целом, создание базы данных MySQL для веб-приложения по бронированию столиков в ресторане - это ключевой аспект, который способствует эффективной работе приложения, улучшает управление данными, обеспечивает безопасность и защиту данных, а также позволяет взаимодействие с другими системами для более гибкого управления процессом бронирования столиков.

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.

Методами иcследования являются: методы анализа и синтеза информации, проектирование, практическая реализация.

# 1 Теоретическая часть

## HTML и CSS

HTML (HyperText Markup Language) - это язык разметки, используемый для создания структуры и содержания web-страниц. Он состоит из серии элементов (тегов), которые определяют различные части страницы, такие как заголовки, параграфы, изображения, таблицы и т.д. HTML позволяет структурировать содержимое, чтобы браузер мог правильно отображать страницу.

CSS (Cascading Style Sheets) - это язык таблиц стилей, используемый для описания внешнего вида web-страницы. Он позволяет определить различные свойства элементов HTML, такие как шрифт, цвет, размер, отступы, позиционирование и др. CSS позволяет разработчикам создавать стильные и привлекательные веб-страницы, облегчая работу с разметкой HTML.

Применение HTML и CSS в веб-разработке является критически важным. Вот несколько причин, почему:

1. Структура и семантика: HTML помогает определить структуру и логическую семантику веб-страницы. Это делает содержимое более доступным для поисковых систем и улучшает взаимодействие со скринридерами для людей с ограниченными возможностями.

2. Внешний вид и пользовательский опыт: CSS позволяет разработчикам создавать привлекательные и стильные веб-страницы. Он дает возможность настраивать внешний вид элементов HTML, что позволяет создавать индивидуальный и уникальный дизайн в соответствии с брендингом и предпочтениями пользователей. Это важно для создания положительного пользовательского опыта.

3. Адаптивность и отзывчивость: HTML и CSS позволяют создавать адаптивные и отзывчивые веб-страницы. Это означает, что веб-сайты могут автоматически адаптироваться к различным устройствам и экранам, таким как мобильные телефоны, планшеты и настольные компьютеры. Это важно для обеспечения удобства использования и достижения большей аудитории.

4. Разделение ответственности: HTML отвечает за структуру и содержание веб-страницы, а CSS за внешний вид. Это разделение позволяет разработчикам более эффективно работать в команде, улучшает обслуживание и повторное использование кода.

В целом, применение HTML и CSS в веб-разработке является неотъемлемой частью создания высококачественных и стильных веб-страниц. Они обеспечивают важные функции, которые улучшают доступность, пользовательский опыт и удобство использования веб-сайтов.

## JavaScript

JavaScript - это интерпретируемый язык программирования, используемый для создания динамических и интерактивных web-сайтов. Он позволяет web-разработчикам добавлять различные функции и взаимодействие на веб-страницах, что помогает создавать более интересные и привлекательные пользовательские интерфейсы.

Несколько важных функций JavaScript:

1. Манипуляция HTML-элементами: JavaScript позволяет изменять и модифицировать HTML-код страницы, добавлять, удалять или изменять элементы, классы, стили и атрибуты.

2. Обработка событий: JavaScript позволяет реагировать на различные действия пользователя на web-странице, такие как нажатие кнопки, клик мыши, наведение курсора и многое другое. Это позволяет создавать интерактивные и отзывчивые пользовательские интерфейсы.

3. Асинхронная загрузка данных: JavaScript позволяет загружать данные с сервера без перезагрузки всей страницы. Это позволяет создавать динамические приложения, которые могут обновляться без прерывания работы пользователя.

4. Работа с данными: JavaScript предоставляет множество возможностей для обработки и манипуляции данными, включая их чтение, запись, редактирование и преобразование. Это особенно полезно при создании форм, валидации данных и обработке результатов пользовательского взаимодействия.

5. Создание анимаций: JavaScript позволяет создавать анимацию на web-страницах, изменяя стили и свойства элементов, перемещая их или изменяя их размеры. Это позволяет создавать более привлекательные и интерактивные пользовательские интерфейсы.

Важность JavaScript для web-разработки состоит в его способности добавлять динамичность и интерактивность на веб-страницы. Он позволяет создавать более привлекательные визуальные эффекты, улучшать пользовательский опыт, улучшать функциональность и сокращать время загрузки страницы. Большинство современных web-сайтов и приложений используют JavaScript, поэтому знание этого языка программирования является важным навыком для веб-разработчиков.

Некоторые достоинства JavaScript:

1. Широкая поддержка: JavaScript поддерживается всеми современными web-браузерами, что позволяет создавать совместимые и доступные web-сайты для всех пользователей.

2. Простота в изучении: JavaScript относительно прост в изучении и понимании. Он имеет синтаксис, похожий на другие языки программирования, такие как Java и C++, и множество ресурсов и учебных материалов для начинающих.

3. Интеграция с HTML и CSS: JavaScript легко интегрируется с HTML и CSS, что позволяет создавать привлекательные и кросс-платформенные web-приложения.

4. Большое сообщество: JavaScript имеет очень активное и большое сообщество разработчиков и фреймворков, что позволяет быстро решать проблемы и обмениваться опытом.

## 1.3 PHP

PHP (Hypertext Preprocessor) - это язык программирования общего назначения, который широко используется для разработки динамических web-страниц и web-приложений. Он скриптовый язык, выполняемый на сервере, что означает, что код PHP обрабатывается на сервере, а результаты отображаются на клиентском устройстве.

Важность применения PHP в веб-разработке заключается в следующем:

1. Простота использования: PHP обладает простым и понятным синтаксисом, что делает его доступным для начинающих разработчиков. Большое сообщество PHP разработчиков также означает наличие большого количества обучающих материалов и поддержки.

2. Широкие возможности: PHP поддерживает множество баз данных (MySQL, PostgreSQL, Oracle и другие) и протоколов (HTTP, FTP, SMTP и другие), что позволяет разрабатывать разнообразные веб-приложения.

3. Интеграция со сторонними системами: PHP легко интегрируется с другими технологиями, такими как HTML, CSS, JavaScript, что позволяет создавать сложные веб-приложения и веб-сайты.

4. Открытый и свободный: PHP является Open Source, что означает, что разработчики могут свободно использовать и изменять его в соответствии с собственными потребностями. Это также означает наличие огромной базы бесплатных расширений и фреймворков для PHP.

Функции PHP включают:

1. Генерация динамического контента: благодаря PHP, веб-разработчики могут создавать динамические страницы, которые автоматически обновляются и адаптируются в зависимости от действий пользователя.

2. Работа с формами: PHP позволяет легко обрабатывать данные, полученные от пользователей через формы на веб-страницах. Это позволяет создавать интерактивные приложения, такие как регистрация пользователей, отправка сообщений и т. д.

3. Работа с базами данных: PHP предоставляет мощные функции для работы с базами данных, что позволяет хранить, изменять и получать данные из баз данных.

4. Организация и обработка файлов: PHP предоставляет возможности для работы с файлами на сервере, включая чтение, запись и удаление данных из файлов.

5. Обработка ошибок и исключений: PHP позволяет разработчикам контролировать и обрабатывать ошибки и исключения, что помогает создавать более надежные и безопасные приложения.

В целом, PHP является мощным инструментом для разработки веб-приложений, и его популярность продолжает расти благодаря своей простоте использования и большому количеству возможностей.

## MySQL

MySQL - это система управления базами данных, которая позволяет организовывать и управлять большими объемами данных. Это одна из наиболее популярных реляционных СУБД, используемых для работы с веб-приложениями.

Функции, предоставляемые MySQL, включают в себя:

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.

Применение MySQL для веб-разработки, в частности для бронирования столиков в ресторане, имеет важное значение по следующим причинам:

1. Хранение данных: MySQL позволяет сохранять информацию о клиентах, резервациях, столиках и т. д. и обеспечивает надежность хранения данных.

2. Обработка запросов: С помощью MySQL можно легко выполнять запросы на выборку, обновление и удаление данных, что позволяет эффективно управлять бронированием столиков.

3. Масштабируемость: MySQL поддерживает горизонтальное и вертикальное масштабирование, что позволяет расширять функциональность системы по мере роста бизнеса и увеличения числа резерваций.

4. Безопасность данных: MySQL обеспечивает защиту данных, позволяя контролировать доступ пользователей к базе данных и предоставляя возможности шифрования для защиты конфиденциальной информации.

В целом, использование MySQL для бронирования столиков в ресторане позволяет эффективно управлять и обрабатывать данные, обеспечивает безопасность хранимых данных и способствует развитию функционала системы.

# 2 Практическая часть

## 2.1 Требования и описание реализации

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

**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

Установка open server позволяет воспроизводиться web-приложению локально. С помощью встроенного функционала (Рисунок 1) создалась база данных с кодировкой *utf8\_general\_ci* (Рисунок 2).

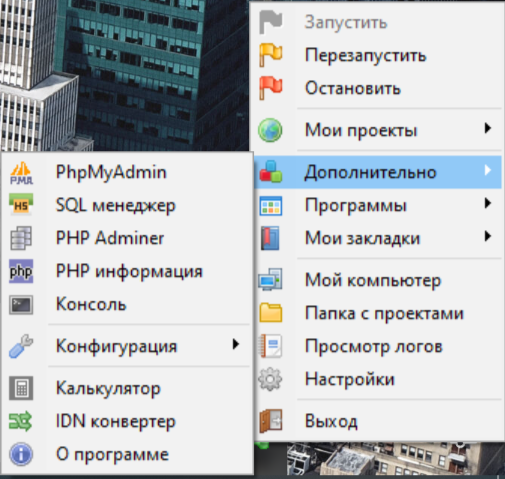


Рисунок 1 — Open server: встроенный функционал

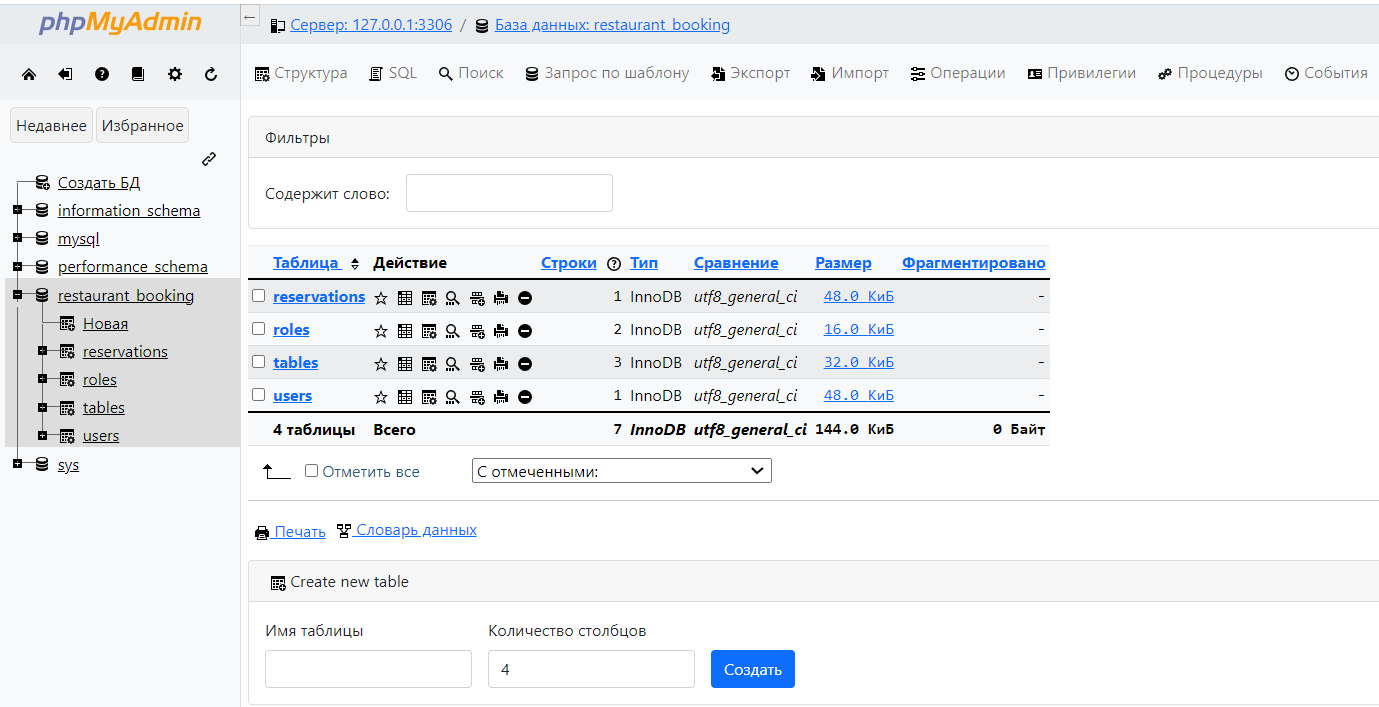


Рисунок 2 — База данных

Далее была установлена система управления базами данных MySQL, которая позволяет организовывать и управлять большими объемами данных.



Рисунок 3 — Установка MySQL

## 2.3 Создание базы данных

Листинг создания базы данных:

-- Создание базы данных

CREATE DATABASE IF NOT EXISTS restaurant\_booking CHARACTER SET utf8 COLLATE utf8\_general\_ci;

-- Использование базы данных

USE restaurant\_booking;

-- Таблица ролей

CREATE TABLE IF NOT EXISTS roles (

    role\_id INT AUTO\_INCREMENT PRIMARY KEY,

    role\_name ENUM('user', 'admin') NOT NULL

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

-- Таблица пользователей

CREATE TABLE IF NOT EXISTS users (

    user\_id INT AUTO\_INCREMENT PRIMARY KEY,

    username VARCHAR(50) NOT NULL,

    password VARCHAR(255) NOT NULL,

    email VARCHAR(255) NOT NULL,

    role\_id INT,

    unique (email),

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

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

-- Таблица столиков

CREATE TABLE IF NOT EXISTS tables (

    table\_id INT AUTO\_INCREMENT PRIMARY KEY,

    table\_number INT NOT NULL,

    capacity INT NOT NULL,

    UNIQUE (table\_number)

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

-- Таблица бронирований

CREATE TABLE IF NOT EXISTS reservations (

    reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,

    user\_id INT,

    table\_id INT,

    guests INT,

    reservation\_date DATE NOT NULL,

    reservation\_time TIME NOT NULL,

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

    FOREIGN KEY (table\_id) REFERENCES tables(table\_id)

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

Название базы данных *restaurant\_booking* с кодировкой *utf8\_general\_ci*. Создаётся таблица ролей, которая имеет role\_id и role\_name (роль админа или пользователя). После создаём таблицу пользователей, которая содержит в себе user\_id, username, password, email, role\_id. Примечание: электронная почта должна быть уникальной, зарегистрировать двух пользователей на одну и ту же почту невозможно. Далее создаётся таблица столиков, которая содержит в себе id столика, его номер и сколько человек может вмещать тот или иной столик. После создаётся таблица брони, которая связывает пользователя с его столиком (содержит в себе дату, время, кол-во гостей).

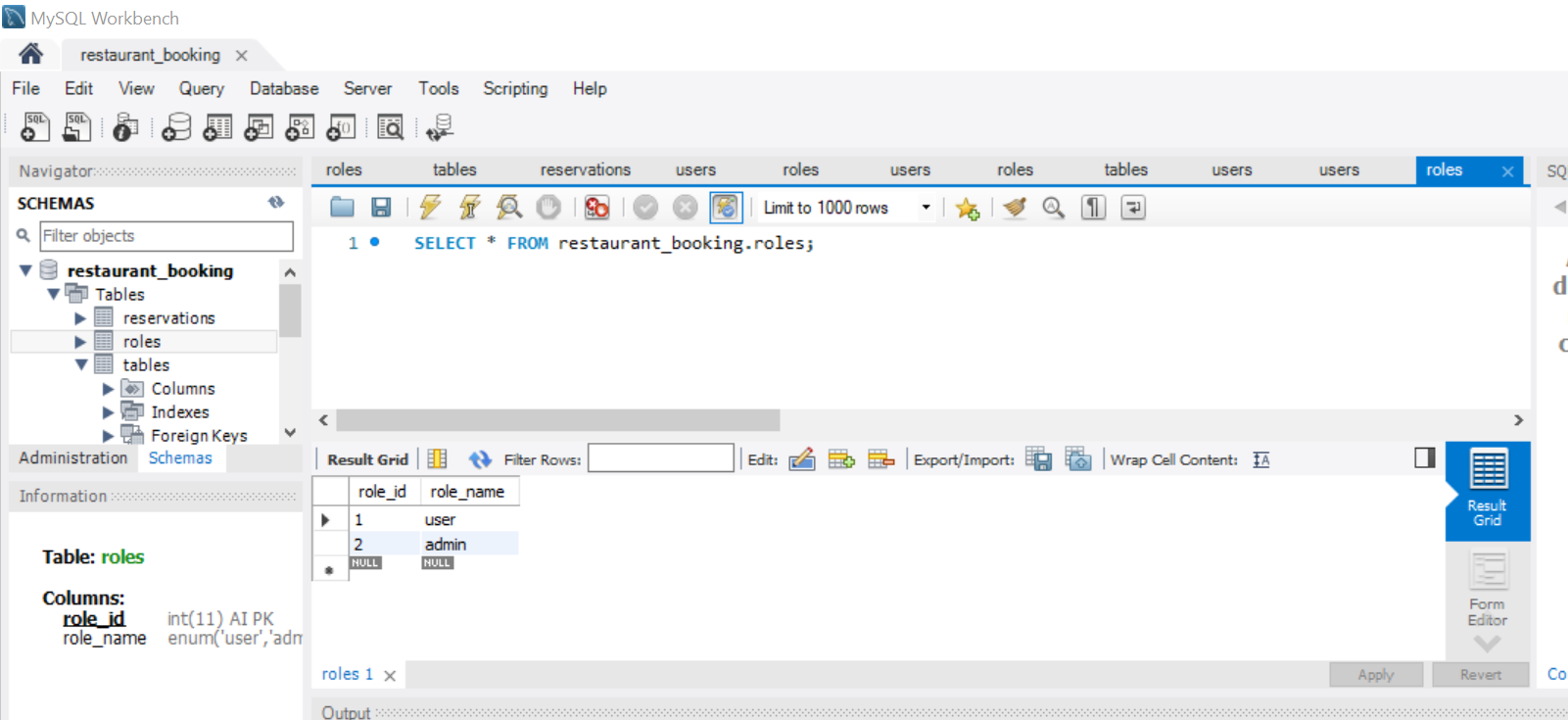


Рисунок 4 — База данных: роль пользователя и админа

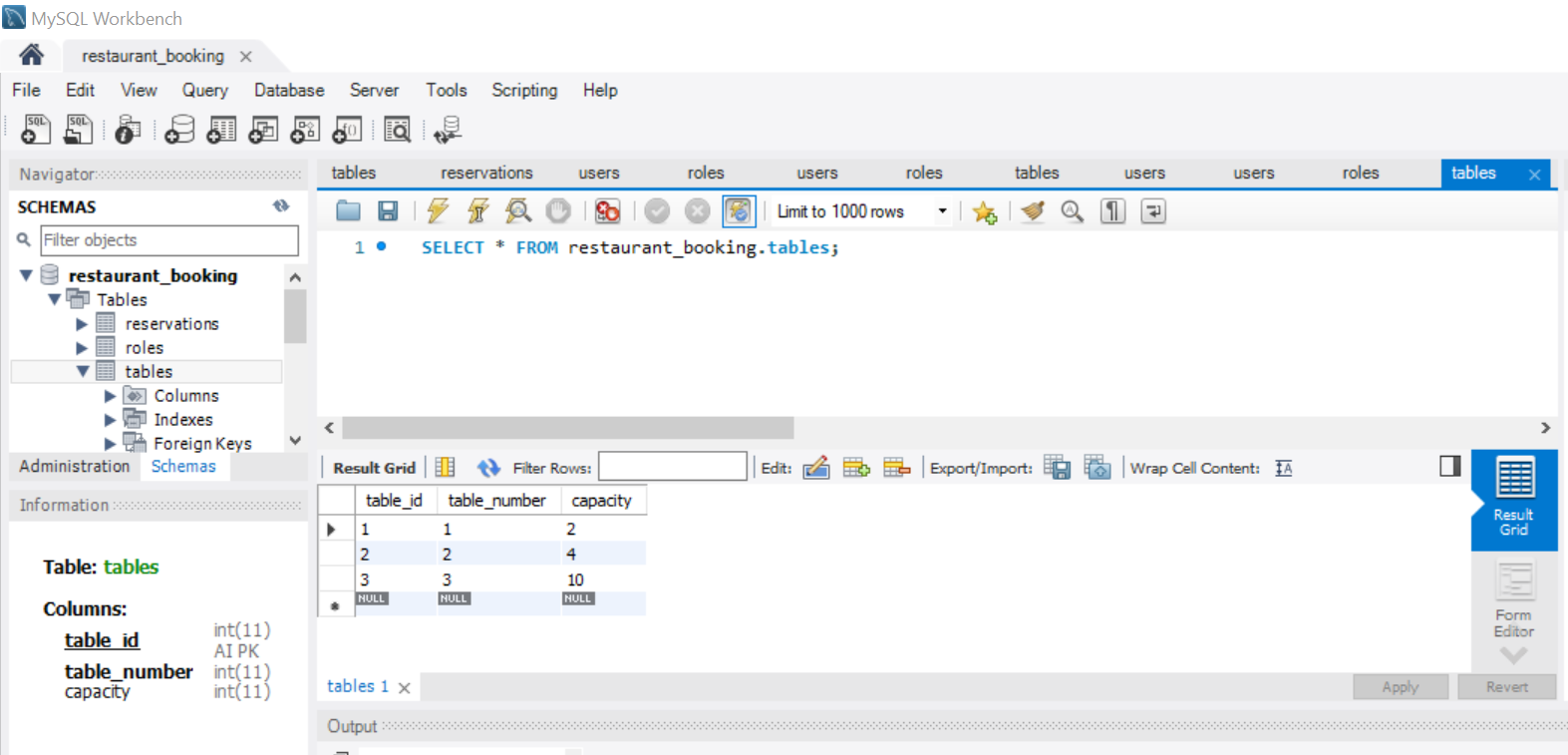


Рисунок 5 — База данных: нумерация столов и их вместимость

Важное примечание: большую роль играет количество человек, которое мы внесём в строку capacity, то есть в строку вместимости. Если пользователь захочет забронировать столик на 3,7,12 человек — код автоматически найдёт столик, который сможет вместить столько человек. Например на 3 человека ближайший такой столик - это столик, вмещающий 4 человека. С 7 то же самое, код найдёт столик, который вмещает 10 человек, а вот с 12 уже не получится, т.к. максимальная вместительность 10 из тех столиков, что добавлено в базу данных.

На рисунке 6 показываются брони. ID пользователя номер стола, количество гостей, дата и время бронирования.

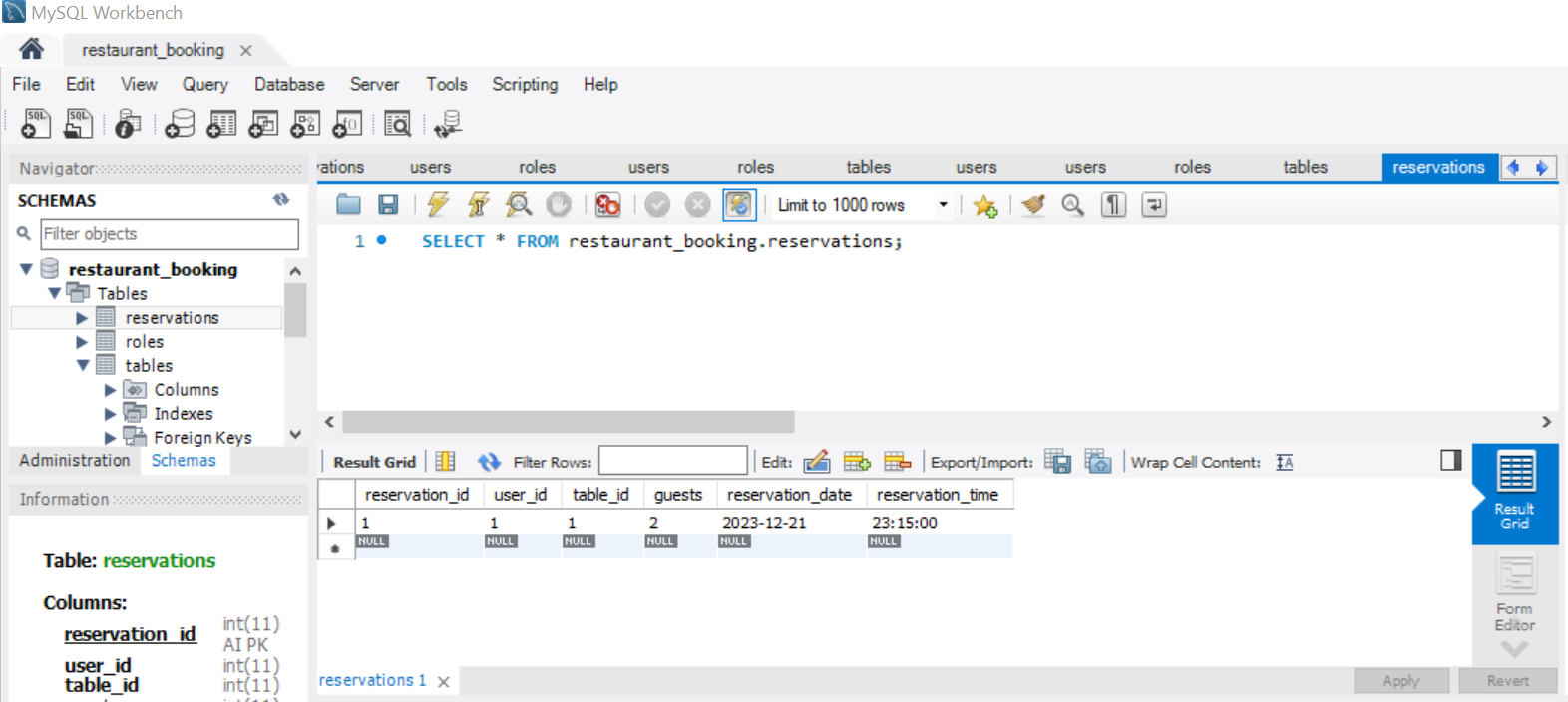


Рисунок 6 — База данных: брони

## 2.4 Регистрация в web-приложении

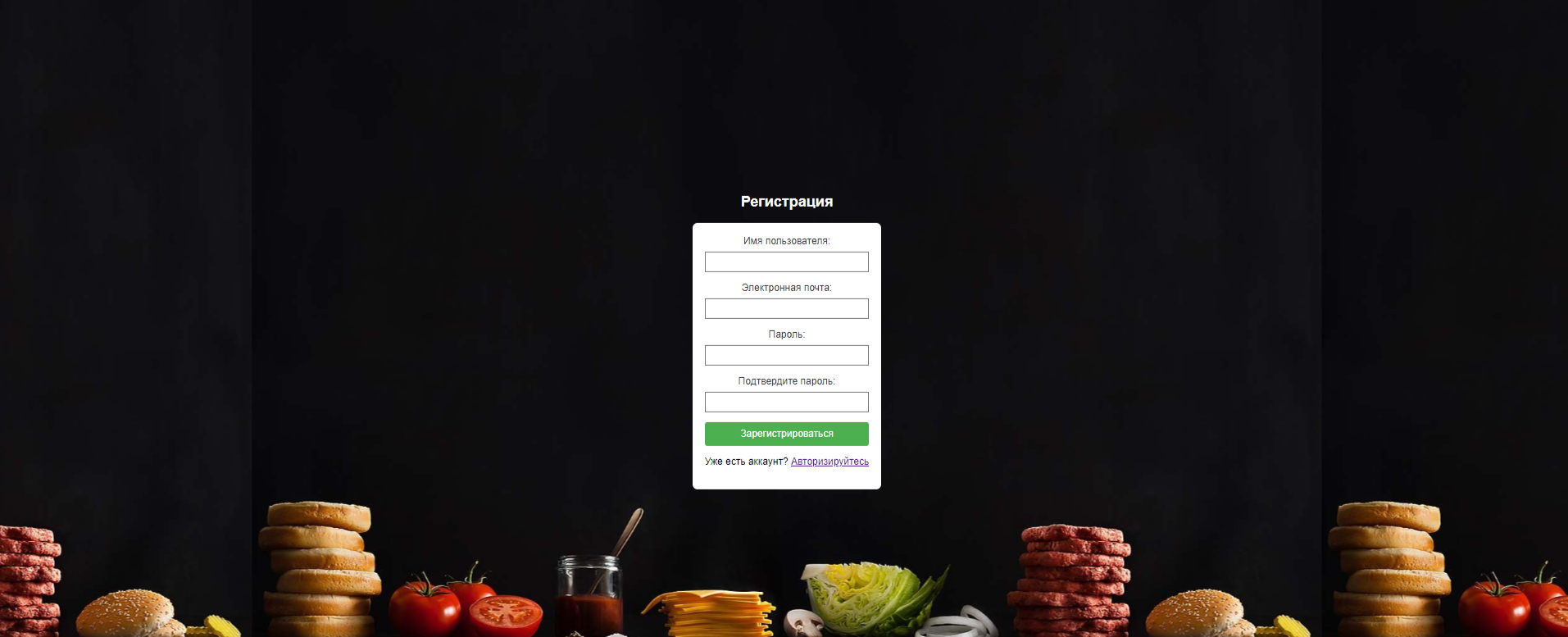


Рисунок 7 — Регистрация в web-приложении

Листинг HTML составляющей панели регистрации:

<!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="registration.css">

</head>

<body>

<div class="container">

    <h2>Регистрация</h2>

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

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

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

        <label for="email">Электронная почта:</label>

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

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

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

        <span id="passwordError" class="error"></span><br>

        <label for="confirm\_password">Подтвердите пароль:</label>

        <input type="password" name="confirm\_password" id="confirm\_password" required>

        <span id="confirmPasswordError" class="error"></span><br>

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

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

    </form>

</div>

Форма регистрации содержит в себе поля для ввода имени, электронной почты, пароля, строки подтверждения пароля, а также кнопки «зарегистрироваться». Отдельно хочется добавить удобное перемещение по сайту: если пользователь уже зарегистрирован, то его перебросит на страницу авторизации. «Уже есть аккаунт? Авторизируйтесь».

С помощью тега <link rel="stylesheet" href="registration.css"> подключаем стили. Листинг стилей для панели регистрации:

body {

    font-family: 'Arial', sans-serif;

    background-color: #f4f4f4;

    margin: 0;

    padding: 0;

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100vh;

    background-image: url('fotka\_na\_sayt\_4.jpg'); /\* Укажите путь к вашей картинке \*/

    background-size: contain; /\* Заменено на contain \*/

    background-position: center; /\* Добавлено для центрирования изображения \*/

}

.container {

    text-align: center;

}

h2 {

    color: #ffffff;

}

form {

    max-width: 300px;

    margin: 0 auto;

    background: #fff;

    padding: 20px;

    border-radius: 8px;

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

}

label {

    display: block;

    margin-bottom: 8px;

    color: #333;

}

input {

    width: 100%;

    padding: 8px;

    margin-bottom: 16px;

    box-sizing: border-box;

}

button {

    background-color: #4caf50;

    color: white;

    padding: 10px 15px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    font-size: 16px;

    width: 100%;

}

button:hover {

    background-color: #45a049;

}

.error {

    color: red;

    margin-top: 8px;

}

При нажатии на кнопку «зарегистрироваться» активируется функция JavaScript — function validateForm(). Данная функция устанавливает правила и ограничения для полей «пароль» и «электронная почта». Пароль должен содержать в себе 8+ символов, цифры и буквы. Электронная почта должна содержать в себе символ «@» и название электронной почты. Листинг функции:

<script>

    function validateForm() {

        var username = document.getElementsByName("username")[0].value;

        var email = document.getElementsByName("email")[0].value;

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

        var confirmPassword = document.getElementById("confirm\_password").value;

        // Регулярное выражение для проверки адреса электронной почты

        var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

        if (!emailRegex.test(email)) {

            alert("Пожалуйста, введите корректный адрес электронной почты.");

            return false;

        }

        // Регулярное выражение для проверки пароля

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

        if (!passwordRegex.test(password)) {

            document.getElementById("passwordError").innerHTML = "Пароль должен содержать минимум 8 символов, включая буквы и цифры.";

            return false;

        } else {

            document.getElementById("passwordError").innerHTML = "";

        }

        if (password !== confirmPassword) {

            document.getElementById("confirmPasswordError").innerHTML = "Пароли не совпадают.";

            return false;

        } else {

            document.getElementById("confirmPasswordError").innerHTML = "";

        }

        checkEmailUnique(email);

        return true;

    }

    function checkEmailUnique(email) {

        $.ajax({

            type: 'POST',

            url: 'check\_email.php',

            data: { email: email },

            success: function (response) {

                if (response === 'exists') {

                    document.getElementById("emailError").innerHTML = "Этот email уже зарегистрирован.";

                } else {

                    document.getElementById("emailError").innerHTML = "";

                }

            }

        });

    }

</script>

Далее при отправке формы активируется PHP код, устанавливается подключение к базе данных (подключение к локальному хосту). Указывается имя базы данных, имя пользователя, имеющего доступ и пароль. Листинг:

<?php

session\_start();

// Подключение к базе данных (замените значения на свои)

$host = 'localhost';

$db\_name = 'restaurant\_booking';

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

    header("Location: dashboard.php");

    exit();

}

// Обработка формы регистрации

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

    $username = $\_POST['username'];

    $email = $\_POST['email'];

    $password = $\_POST['password'];

    $confirm\_password = $\_POST['confirm\_password'];

    // Проверка пароля

    if ($password !== $confirm\_password) {

        die("Пароли не совпадают");

    }

    // Хеширование пароля

    $hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);

    // Проверка уникальности адреса электронной почты

    $stmtCheckEmail = $pdo->prepare("SELECT COUNT(\*) FROM users WHERE email = ?");

    $stmtCheckEmail->execute([$email]);

    $emailCount = $stmtCheckEmail->fetchColumn();

    if ($emailCount > 0) {

        die("Адрес электронной почты уже занят");

    }

    // Получение идентификатора роли "user"

    $stmtRole = $pdo->prepare("SELECT role\_id FROM roles WHERE role\_name = 'user'");

    $stmtRole->execute();

    $roleId = $stmtRole->fetchColumn();

    // Вставка пользователя в базу данных с ролью "user"

    $stmt = $pdo->prepare("INSERT INTO users (username, email, password, role\_id) VALUES (?, ?, ?, ?)");

    $stmt->execute([$username, $email, $hashed\_password, $roleId]);

    // Редирект после успешной регистрации

    header("Location: login.php");

    exit();

}

?>

При проверке пароля, если он неверный, система оповестит о том, что пароли не совпадают. Если же пароли совпали, запись отобразится в базе данных и пароль хешируется (Делается это для безопасности пользователей). Листинг:

// Проверка пароля

if ($password !== $confirm\_password) {

die("Пароли не совпадают");

}

// Хеширование пароля

$hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);

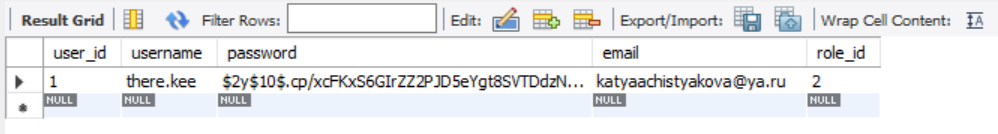


Рисунок 8 — Сохранение регистрации пользователя в базе данных, хеширование пароля

Далее реализована проверка уникальности электронной почты для того, чтобы код не сохранил одну и ту же почту дважды. Если почта уже занята, система выдаст оповещение. Листинг:

// Проверка уникальности адреса электронной почты

$stmtCheckEmail = $pdo->prepare("SELECT COUNT(\*) FROM users WHERE email = ?");

$stmtCheckEmail->execute([$email]);

$emailCount = $stmtCheckEmail->fetchColumn();

if ($emailCount > 0) {

die("Адрес электронной почты уже занят");

}

Далее мы получаем идентификатор роли «user» и присваиваем её только что зарегистрировавшемуся пользователю. Листинг:

// Получение идентификатора роли "user"

$stmtRole = $pdo->prepare("SELECT role\_id FROM roles WHERE role\_name = 'user'");

$stmtRole->execute();

$roleId = $stmtRole->fetchColumn();

// Вставка пользователя в базу данных с ролью "user"

$stmt = $pdo->prepare("INSERT INTO users (username, email, password, role\_id) VALUES (?, ?, ?, ?)");

$stmt->execute([$username, $email, $hashed\_password, $roleId]);

После завершения регистрации пользователя перенаправит на страницу авторизации. Листинг:

// Редирект после успешной регистрации

header("Location: login.php");

exit();

## 2.5 Авторизация в web-приложении

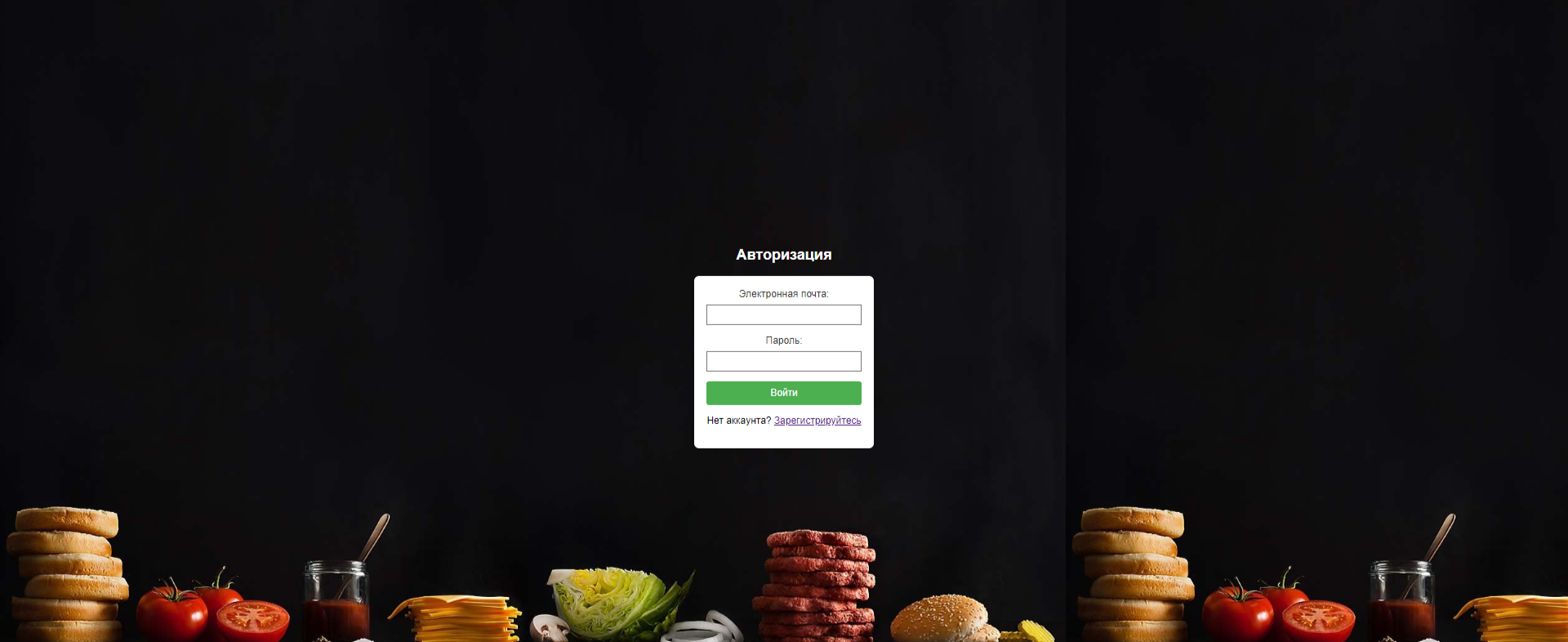


Рисунок 9 — Авторизация в web-приложении

Листинг HTML составляющей панели авторизации:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>Авторизация</title>

</head>

<body>

<div class="container">

    <h2>Авторизация</h2>

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

        <label for="email">Электронная почта:</label>

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

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

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

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

        <?php

        if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && !empty($user)) {

            echo '<div class="error">Неверные электронная почта или пароль.</div>';

        }

        ?>

        <p>Нет аккаунта? <a href="registration.php">Зарегистрируйтесь</a></p>

    </form>

</div>

</body>

</html>

Авторизация включает в себя форму ввода электронной почты, пароль и кнопку «войти».

Установка соединения с базой данных. Листинг:

<?php

session\_start();

// Подключение к базе данных (замените значения на свои)

$host = 'localhost';

$db\_name = 'restaurant\_booking';

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

    header("Location: dashboard.php");

    exit();

}

Далее с помощью PHP кода обрабатывается форма авторизации, проверяется существование пользователя с соответствующими введёнными данными. После успешной регистрации пользователь будет перенаправлен на панель бронирования. Листинг:

// Обработка формы авторизации

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

    $email = $\_POST['email'];

    $password = $\_POST['password'];

    // Получение пользователя по электронной почте

    $stmt = $pdo->prepare("SELECT \* FROM users WHERE email = ?");

    $stmt->execute([$email]);

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

    // Проверка пароля и роли

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

        session\_start();

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

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

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

        // Редирект после успешной авторизации

        header("Location: booking.php");

        exit();

    } else {

    }

}

?>

Листинг стилей для панели авторизации:

body{

    height: 100vh;

    font-family: 'Arial', sans-serif;

    background-color: #f4f4f4;

    margin: 0;

    padding: 0;

    display: flex;

    align-items: center;

    justify-content: center;

    background-image: url('fotka\_na\_sayt\_4.jpg');

    background-size: contain;

}

.container {

    text-align: center;

}

h2 {

    color: #FFFFFF;

}

form {

    max-width: 300px;

    margin: 0 auto;

    background: #fff;

    padding: 20px;

    border-radius: 8px;

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

}

label {

    display: block;

    margin-bottom: 8px;

    color: #333;

}

input {

    width: 100%;

    padding: 8px;

    margin-bottom: 16px;

    box-sizing: border-box;

}

button {

    background-color: #4caf50;

    color: white;

    padding: 10px 15px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    font-size: 16px;

    width: 100%;

}

button:hover {

    background-color: #45a049;

}

.error {

    color: red;

    margin-top: 8px;

}

## 2.6 Бронирование столика в web-приложении

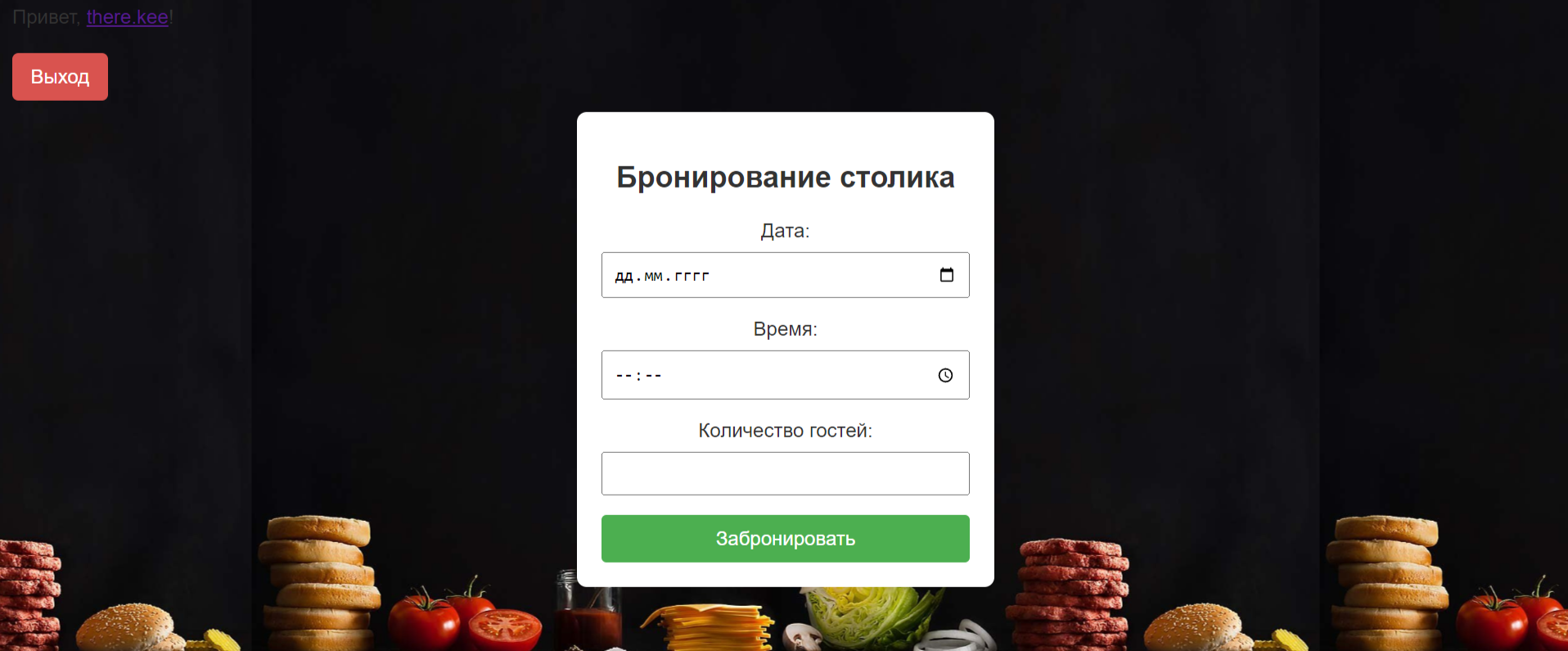


Рисунок 10 — Бронирование столика в web-приложении

Листинг HTML составляющей панели бронирования:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>Бронирование столика</title>

</head>

<body>

<div class="user-info">

    <?php

    // Проверка существования сессии пользователя

    if (isset($\_SESSION['username'])) {

        echo 'Привет, ' .'<a href="dashboard.php">'.htmlspecialchars($\_SESSION['username']).'</a>'. '!';

        echo '<br><a href="logout.php" class="logout-button">Выход</a>';

    }

    ?>

</div>

<div class="container">

    <h2>Бронирование столика</h2>

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

        <label for="date">Дата:</label>

        <input type="date" name="date" required><br>

        <label for="time">Время:</label>

        <input type="time" name="time" required><br>

        <label for="guests">Количество гостей:</label>

        <input type="number" name="guests" min="1" required><br>

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

    </form>

</div>

На рисунке 10 в левом верхнем углу можно увидеть приветствие, за это отвечает следующий PHP код, встроенный в HTML:

<?php

// Проверка существования сессии пользователя

if (isset($\_SESSION['username'])) {

echo 'Привет, ' .'<a href="dashboard.php">'.htmlspecialchars($\_SESSION['username']).'</a>'. '!';

echo '<br><a href="logout.php" class="logout-button">Выход</a>';

}

?>

То есть, мы получаем данные об авторизованном пользователе (его имя) и создаём приветствие. При нажатии на имя, пользователя переносит в его личный кабинет. Сам HTML состоит из формы, которая включает в себя поля ввода следующей информации: дату, время и количество гостей, а также кнопку «забронировать». Важное примечание: как отмечалось ранее, столик имеет определённую вместимость. Следовательно, при бронировании пользователь сможет указать то количество гостей, которое будет удовлетворять вместимости того или иного столика.

Также имеется кнопка «выход». Реализованный код для разрыва пользователя с сайтом (кнопка «выход»). Листинг:

<?php

session\_start();

// Удаление всех данных сессии

session\_unset();

session\_destroy();

// Перенаправление на страницу входа или другую страницу вашего выбора

header("Location: login.php");

exit();

?>

Листинг проверки авторизации пользователя:

// Проверка существования сессии пользователя

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

    header("Location: login.php");

    exit();

}

После бронирования все данные отправляются на сервер. Мы получаем данные о ID пользователя, дате, времени и количестве гостей (Рисунок 11). Из полученного списка столиков мы проверяем те, которые походят по количество гостей. Если подходящего столика нет, то система выдаёт ошибку. Далее идёт проверка о наличии свободного столика конкретно в данное время. Если такого столика нет, то система также выдаёт ошибку. В конечном итоге, при полном корректном вводе создаётся таблица в базе данных о бронировании. После успешного бронирования пользователь будет направлен на страницу личного кабинета. Листинг:

// Получение списка столов для выбора

$stmtTables = $pdo->prepare("SELECT table\_id, capacity FROM tables");

$stmtTables->execute();

$tables = $stmtTables->fetchAll(PDO::FETCH\_ASSOC);

// Обработка формы бронирования

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

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

    $date = $\_POST['date'];

    $time = $\_POST['time'];

    $guests = $\_POST['guests'];

    // Поиск подходящего стола в зависимости от количества гостей

    $stmtFindTable = $pdo->prepare("SELECT table\_id FROM tables WHERE capacity >= ? ORDER BY capacity ASC LIMIT 1");

    $stmtFindTable->execute([$guests]);

    $table = $stmtFindTable->fetch(PDO::FETCH\_ASSOC);

    if (!$table) {

        die("Извините, нет подходящего стола для указанного количества гостей.");

    }

    $table\_id = $table['table\_id'];

    // Проверка доступности выбранного времени и стола (здесь можно добавить дополнительные проверки)

    $stmtCheckAvailability = $pdo->prepare("SELECT COUNT(\*) FROM reservations WHERE reservation\_date = ? AND reservation\_time = ? AND table\_id = ?");

    $stmtCheckAvailability->execute([$date, $time, $table\_id]);

    $availabilityCount = $stmtCheckAvailability->fetchColumn();

    if ($availabilityCount > 0) {

        die("Выбранное время или стол уже заняты. Пожалуйста, выберите другое время или стол.");

    }

    // Вставка записи о бронировании в базу данных

    $stmtInsertReservation = $pdo->prepare("INSERT INTO reservations (user\_id, table\_id,guests, reservation\_date, reservation\_time) VALUES (?, ?, ?, ?, ?)");

    $stmtInsertReservation->execute([$user\_id, $table\_id,$guests, $date, $time ]);

    // Редирект после успешного бронирования

    header("Location: dashboard.php");

    exit();

}

?>

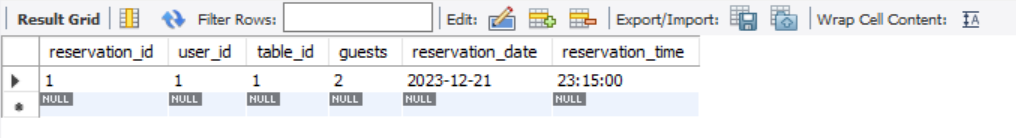


Рисунок 11 — Данные о брони на сервере

Далее в HTML коде применён JavaScript код, который делает кнопку «забронировать» недоступной для неавторизированного пользователя. Но, можно считать, что это теряет смысл, т.к. код в любом случае не даст попасть на данную страницу неавторизированному пользователю. Тем не менее, написание данного кода не будет лишним. Листинг:

<script>

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

        var bookingForm = document.getElementById("bookingForm");

        var bookingButton = document.getElementById("bookingButton");

        // Проверка наличия авторизации

        var isUserAuthenticated = <?php echo isset($\_SESSION['user\_id']) ? 'true' : 'false'; ?>;

        // Установка состояния кнопки в зависимости от авторизации

        bookingButton.disabled = !isUserAuthenticated;

    });

</script>

Листинг стилей для панели бронирования:

body {

    font-family: 'Arial', sans-serif;

    background-color: #f4f4f4;

    margin: 0;

    padding: 0;

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100vh;

    background-image: url('fotka\_na\_sayt\_4.jpg'); /\* Укажите путь к вашей картинке \*/

    background-size: contain; /\* Заменено на contain \*/

    background-position: center; /\* Добавлено для центрирования изображения \*/

}

.container {

    text-align: center;

    max-width: 300px;

    width: 100%;

    background: #fff;

    padding: 20px;

    border-radius: 8px;

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

}

h2 {

    color: #333;

}

form {

    margin-top: 20px;

}

label {

    display: block;

    margin-bottom: 8px;

    color: #333;

}

input, select {

    width: 100%;

    padding: 8px;

    margin-bottom: 16px;

    box-sizing: border-box;

}

.user-info {

    position: absolute;

    top: 10px;

    left: 10px;

    color: #333;

}

button {

    background-color: #4caf50;

    color: white;

    padding: 10px 15px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    font-size: 16px;

    width: 100%;

}

button:hover {

    background-color: #45a049;

}

a.logout-button {

    display: inline-block;

    background-color: #d9534f;

    color: #fff;

    padding: 10px 15px;

    text-decoration: none;

    border-radius: 5px;

    margin-top: 20px;

}

a.logout-button:hover {

    background-color: #c9302c;

}

## 2.7 Личный кабинет пользователя в web-приложении

## 2.1.7 Функционал обычного пользователя

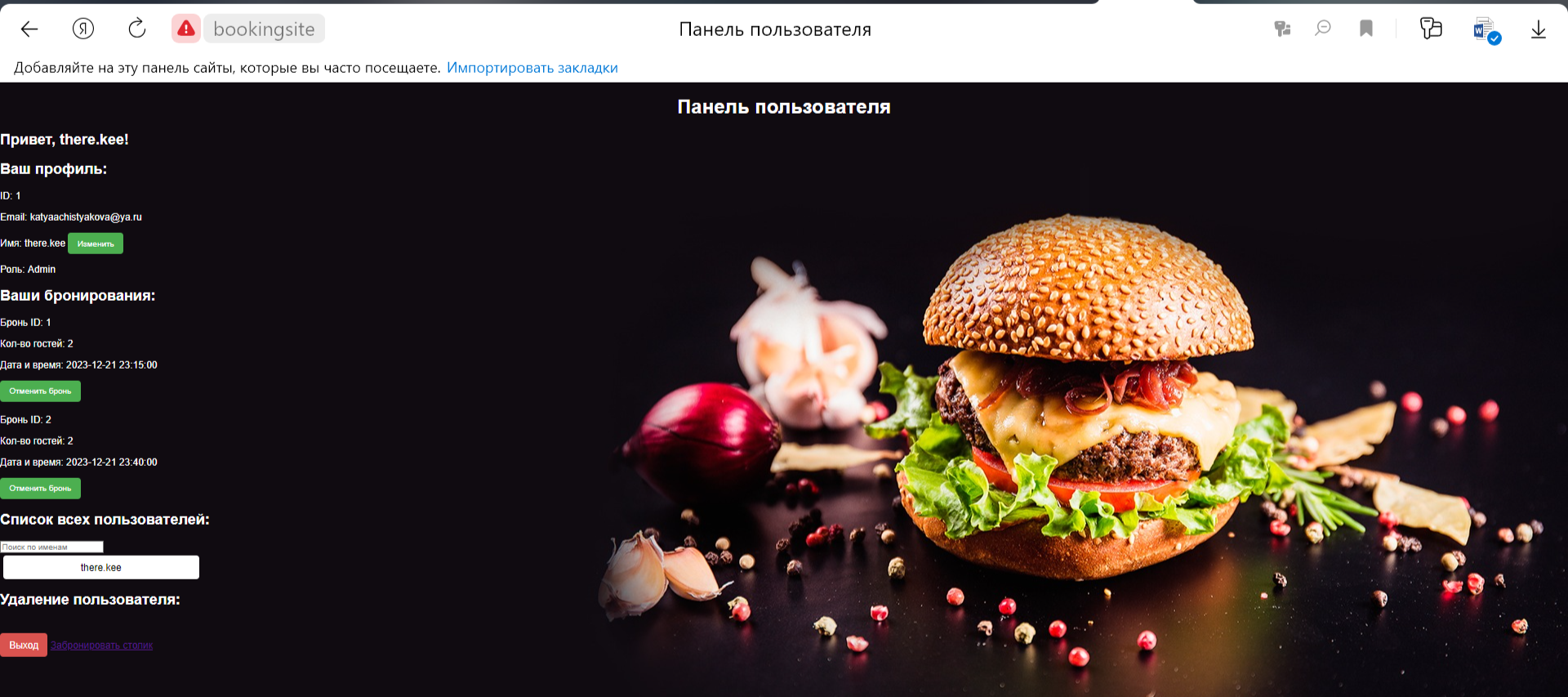


Рисунок 12 — Личный кабинет пользователя в web-приложении

Листинг стилей для личного кабинета пользователя:

body {

    font-family: Arial, sans-serif;

    background-color: #f4f4f4;

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    background-image: url('fotka\_na\_sayt\_8.jpg');

    background-size: cover; /\* Заменено на contain \*/

    background-position: center; /\* Добавлено для центрирования изображения \*/

}

header {

    background-color: #333;

    color: #fff;

    text-align: center;

    padding: 1em;

}

h1 {

    color: #ffffff;

    text-align: center;

}

h2 {

    color: #ffffff;

}

p {

    color: #ffffff;

}

.userListItem{

    max-width: 300px;

    text-align: center;

}

button {

    background-color: #4CAF50;

    color: white;

    padding: 10px 15px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

}

form {

    margin-bottom: 20px;

}

ul {

    list-style-type: none;

    padding: 0;

    margin: 0;

}

li {

    background-color: #fff;

    padding: 10px;

    margin: 5px;

    border-radius: 5px;

    cursor: pointer;

}

li:hover {

    background-color: #ddd;

}

label{

    color: white;

}

#userInfoModal p{

    color: #333333;

}

#userInfoModal h2{

    color: #333333;

}

#userInfoModal label{

    color: #333333;

}

.modal {

    display: none;

    position: fixed;

    z-index: 1;

    left: 0;

    top: 0;

    width: 100%;

    height: 100%;

    overflow: auto;

    background-color: rgba(0, 0, 0, 0.4);

}

.modal-content {

    background-color: #fefefe;

    margin: 15% auto;

    padding: 20px;

    border: 1px solid #888;

    width: 80%;

}

.close {

    color: #aaa;

    float: right;

    font-size: 28px;

    font-weight: bold;

}

.close:hover,

.close:focus {

    color: black;

    text-decoration: none;

    cursor: pointer;

}

.hidden {

    visibility: hidden;

    opacity: 0;

    transition: visibility 0s 0.5s, opacity 0.5s linear;

}

.visible {

    visibility: visible;

    opacity: 1;

    transition: visibility 0s 0s, opacity 0.5s linear;

}

a.logout-button {

    display: inline-block;

    background-color: #d9534f;

    color: #fff;

    padding: 10px 15px;

    text-decoration: none;

    border-radius: 5px;

    margin-top: 20px;

}

a.logout-button:hover {

    background-color: #c9302c;

}

#editUsernameForm,

#editEmailForm,

#editRoleForm {

    margin-top: 20px;

    display: none;

}

#editUsernameForm label,

#editEmailForm label,

#editRoleForm label {

    display: block;

    margin-bottom: 5px;

}

#editUsernameForm input,

#editEmailForm input,

#editRoleForm input {

    width: 100%;

    padding: 8px;

    margin-bottom: 10px;

    box-sizing: border-box;

}

#editUsernameForm button,

#editEmailForm button,

#editRoleForm button {

    background-color: #4CAF50;

    color: white;

    padding: 10px;

    border: none;

    cursor: pointer;

    width: 100%;

    box-sizing: border-box;

}

#editUsernameForm button:hover,

#editEmailForm button:hover,

#editRoleForm button:hover {

    opacity: 0.8;

}

Заголовок «панель пользователя» и приветствие. Мы получаем информацию об авторизированном пользователе и выводим её на панель.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

    <title>Панель пользователя</title>

</head>

<body>

<style>

    label{

        color: white;

    }

    #userInfoDetails{

        color: #333333;

    }

</style>

<h1>Панель пользователя</h1>

<h2>Привет, <?php echo $userInfo['username']; ?>!</h2>

<!-- Информация о пользователе -->

<h2>Ваш профиль:</h2>

<p>ID: <?php echo $userInfo['user\_id']; ?></p>

<p>Email: <?php echo $userInfo['email']; ?></p>

<p>Имя: <?php echo $userInfo['username']; ?>

С помощью PHP кода отображаем информацию о пользователе. На web-странице возле имени есть кнопка «изменить», при нажатии на которую запускается JavaScript код (+ кнопка «сохранить»). Листинг формы обновления имени пользователя:

<!-- Добавляем кнопку "Изменить" и форму для изменения имени -->

    <button onclick="showUpdateUsernameForm()">Изменить</button>

<form id="updateUsernameForm" style="display: none;">

    <label for="newUsername">Новое имя пользователя:</label>

    <input type="text" name="newUsername" required>

    <button type="button" onclick="updateUsername()">Сохранить</button>

</form>

</p>

<p>Роль: <?php echo ucfirst($roleName); ?></p>

При нажатии на кнопку «Изменить» на сайте, вызывается JavaScript функция "showUpdateUsernameForm()". При нажатии на кнопку «Сохранить» так же вызывается JavaScript функция "updateUsername()". Когда мы нажимаем на кнопку «Изменить», у нас запускается скрытый JavaScript код, в котором мы создаём переменную var updateUsernameForm = document.getElementById('updateUsernameForm'); , в неё мы передаём form id="updateUsernameForm", с помощью которого мы можем обращаться к форме. Далее следующей строчкой настраивается css стиль с параметром block : updateUsernameForm.style.display = 'block'. Также используется функция для отправки AJAX-запроса на обновление панели. Это делается для более быстрого обновление имени пользователя на сайте. Листинг настройки AJAX-запроса:

function changeUsername() {  
var newUsername = document.getElementById('newUsername').value;  
  
// Отправка AJAX-запроса  
var xhr = new XMLHttpRequest();  
[xhr.open](https://vk.com/away.php?to=http%3A%2F%2Fxhr.open&cc_key=)('POST', 'dashboard.php', true);  
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  
xhr.onreadystatechange = function () {  
if (xhr.readyState == 4 && xhr.status == 200) {  
// Обновляем информацию в модальном окне после успешного изменения имени  
showUserInfo(currentUserId);  
}  
};  
xhr.send('action=changeUsername&userId=' + currentUserId + '&newUsername=' + newUsername);  
}

Происходит отправка запроса с клиентской части на сервер, в связи с чем активируется действие вызова PHP функции с параметром ID пользователя, далее мы передаём запрос в саму переменную и также передаём параметр нового имени пользователя.

Вывод информации о бронировании (PHP код):

<!-- Ваш код для отображения бронирований пользователя -->

<!-- Форма для отмены бронирования -->

<!-- Список бронирований пользователя -->

<h2>Ваши бронирования:</h2>

<?php

// Получаем бронирования пользователя из базы данных

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

if ($userBookings) {

    foreach ($userBookings as $booking) {

        ?>

        <div>

            <p>Бронь ID: <?php echo $booking['reservation\_id']; ?></p>

            <p>Кол-во гостей: <?php echo $booking['guests']?></p>

            <p>Дата и время: <?php echo $booking['reservation\_date'] . ' ' . $booking['reservation\_time']; ?></p>

            <!-- Добавляем кнопку для отмены бронирования -->

            <form action="cancel\_booking.php" method="post">

                <input type="hidden" name="bookingId" value="<?php echo $booking['reservation\_id']; ?>">

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

            </form>

        </div>

        <?php

    }

} else {

    echo "<p>У вас нет активных бронирований.</p>";

}

?>

При нажатии на кнопку «Отменить бронь» активируется отдельный код, где мы получаем ID брони и ID пользователя. Листинг:

// Получаем ID бронирования из POST-запроса

    $bookingId = isset($\_POST['bookingId']) ? intval($\_POST['bookingId']) : 0;

    // Ваш код для проверки, принадлежит ли бронирование текущему пользователю

    $userId = $\_SESSION['user\_id'];

    if (isBookingBelongsToUser($pdo, $bookingId, $userId)) {

        // Если бронирование принадлежит пользователю, удаляем его

        cancelBooking($pdo, $bookingId);

        echo "Бронирование успешно отменено.";

    } else {

        echo "Ошибка: Бронирование не принадлежит текущему пользователю.";

    }

} else {

    // Если запрос не является POST-запросом, перенаправляем пользователя

    header("Location: dashboard.php");

    exit();

}

Далее выводятся функции брони пользователя и самая функция отмены бронирования. Листинг:

// Функция для проверки, принадлежит ли бронирование пользователю

function isBookingBelongsToUser($pdo, $bookingId, $userId) {

    $query = "SELECT COUNT(\*) FROM reservations WHERE reservation\_id = :booking\_id AND user\_id = :user\_id";

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

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

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

    $statement->execute();

    return $statement->fetchColumn() > 0;

}

// Функция для отмены бронирования

function cancelBooking($pdo, $bookingId) {

    $query = "DELETE FROM reservations WHERE reservation\_id = :booking\_id";

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

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

    $statement->execute();

}

?>

При отмене брони пользователя перенаправит на страницу, где будет написано «Бронирование успешно отменено». При обновлении на панели пользователя будет написано «У вас нет активных броней.» Листинг кнопки выхода и кнопки, чтобы снова сделать бронь:

<a href="logout.php" class="logout-button">Выход</a>

<a href="booking.php" class="booking-link">Забронировать столик</a>

Функционал обычного пользователя закончен, далее будет представлен функционал администратора.

## 2.2.7 Функционал администратора

С помощью PHP кода проверяется, соответствует ли название пользователя роли администратора. При выполнении данного условия администратор получает список всех пользователей. Листинг:

<?php if ($roleName === 'admin'): ?>

    <h2>Список всех пользователей:</h2>

    <input type="text" id="userSearch" placeholder="Поиск по именам" oninput="searchUsers()">

    <ul id="userList">

        <?php

        // Получаем список всех пользователей из базы данных

        $allUsers = getAllUsers($pdo);

        foreach ($allUsers as $user) {

            echo '<li class="userListItem" onclick="showUserInfo(' . $user['user\_id'] . ')">' . $user['username'] . '</li>';

        }

        ?>

    </ul>

Также при введении имени на сайте, происходит фильтрация списка пользователей. Это реализовано с помощью скрытого JavaScript кода: функция function searchUsers(). Помимо этого, может запускаться JavaScript код, с помощью которого отображается список всех пользователей из базы данных: функция ShowUserInfo. Чтобы удалить пользователя, на сайте вводится его ID и он автоматически удаляется: функция function deleteUser(). Метод получает в переменную var deleteUserId его ID и благодаря AJAX-запросу запускает PHP код для удаления пользователя.

// Функция для удаления пользователя

function deleteUser($pdo, $userId) {

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

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

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

    if ($statement->execute()) {

        echo "Пользователь успешно удален";

    } else {

        echo "Ошибка при удалении пользователя";

    }

Также было реализовано отображение информации о пользователе в отдельном окне.

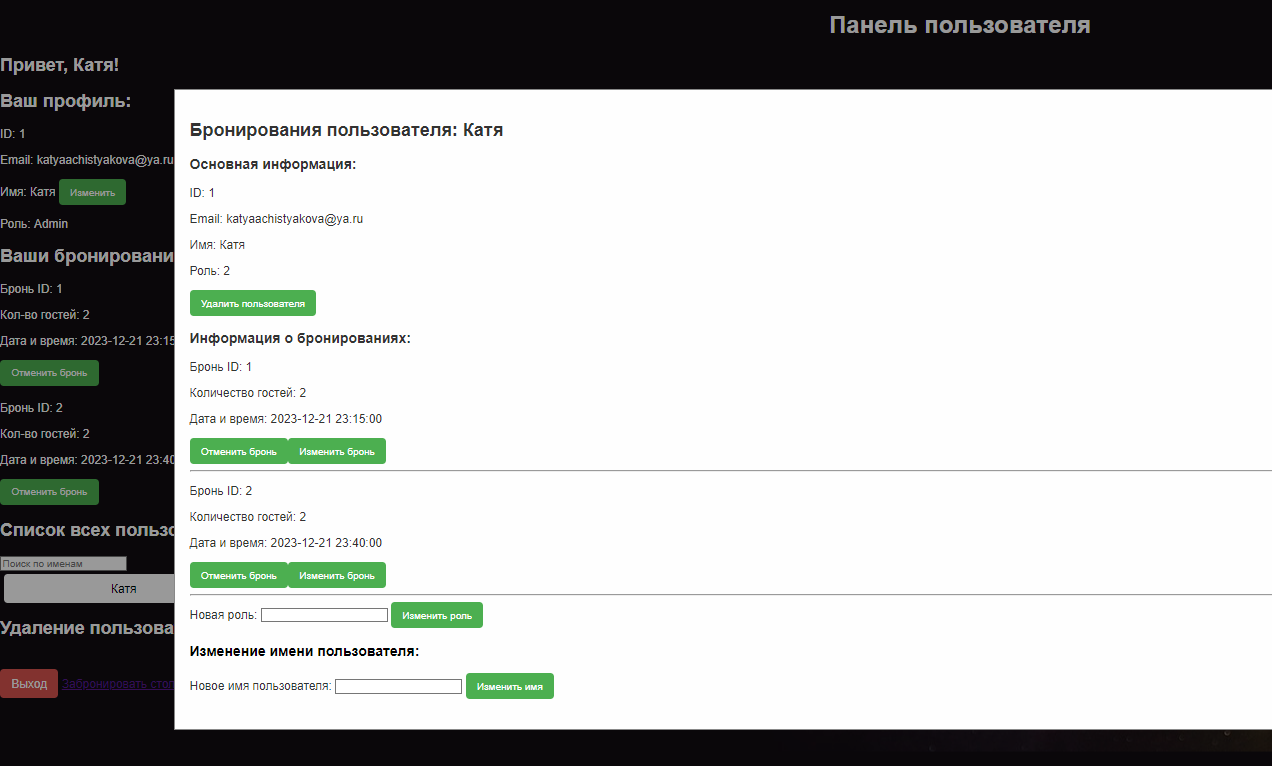


Рисунок 13 — отображение информации о пользователе в отдельном окне

У нас есть такой html код модального окна, проще говоря, это div с классом userInfoModal, внутри которого находятся также блоки userInfoTitle - заголовок модального окна и userInfoDetails - блок с информацией и пользователе. Изначально в них ничего нет, они пустые. Листинг:

<div class="modal-content">  
<span class="close" onclick="closeUserInfoModal()">&times;</span>  
<h2 id="userInfoTitle"></h2>  
<div id="userInfoDetails"></div>  
<form id="updateBookingForm" style="display: none;">  
<label for="newDate">Новая дата:</label>  
<input type="date" name="newDate" id="newDate" required>  
<label for="newTime">Новое время:</label>  
<input type="time" name="newTime" id="newTime" required>  
<label for="newGuests">Новое количество гостей:</label>  
<input type="number" name="newGuests" id="newGuests" required>  
<button type="button" id="updateBookingButton" onclick="updateBooking()">Сохранить изменения</button>  
</form>  
<form id="changeRoleForm">  
<label for="newRole">Новая роль:</label>  
<input type="text" name="newRole" id="newRole" required>  
<button type="button" onclick="changeUserRole(<?php echo $userId; ?>)">Изменить роль</button>  
</form>  
  
<h3>Изменение имени пользователя:</h3>  
<form id="changeUsernameForm">  
<label for="newUsername">Новое имя пользователя:</label>  
<input type="text" name="newUsername" id="newUsername" required>  
<button type="button" onclick="changeUsername()">Изменить имя</button>  
</form>  
  
</div>

Когда мы из списка пользователей выбираем кого-нибудь, запускается JS функция (AJAX запрос) showUserInfo:

function showUserInfo(userId) {  
currentUserId = userId;  
// Отправка AJAX-запроса для получения информации о пользователе  
var xhr = new XMLHttpRequest();  
[xhr.open](https://vk.com/away.php?to=http%3A%2F%2Fxhr.open&cc_key=)('POST', 'dashboard.php', true);  
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  
xhr.onreadystatechange = function () {  
if (xhr.readyState == 4 && xhr.status == 200) {  
// Парсинг JSON-ответа и отображение информации в модальном окне  
var userInfo = JSON.parse(xhr.responseText);  
displayUserInfoModal(userInfo);  
}  
};  
xhr.send('action=getUserInfoAndBookings&userId=' + userId);  
}

Здесь создаётся переменная, которая будет хранить Id пользователя, которого мы выбрали, для дальнейшего взаимодействия с его данными. После выполнения эта функция запустит ещё одну JS функцию displayUserInfoModal:

function displayUserInfoModal(userInfo) {  
var modal = document.getElementById('userInfoModal');  
var title = document.getElementById('userInfoTitle');  
var details = document.getElementById('userInfoDetails');  
  
// Отобразить информацию в модальном окне  
title.innerHTML = 'Бронирования пользователя: ' + (ред.)

userInfo.user\_info.username;  
details.innerHTML = '';  
// Отобразить основную информацию о пользователе  
details.innerHTML += '<h3>Основная информация:</h3>';  
details.innerHTML += '<p>ID: ' + userInfo.user\_info.user\_id + '</p>';  
details.innerHTML += '<p>Email: ' + [userInfo.user\_info.email](https://vk.com/away.php?to=http%3A%2F%2FuserInfo.user_info.email&cc_key=) + '</p>';  
details.innerHTML += '<p>Имя: ' + userInfo.user\_info.username + '</p>';  
details.innerHTML += '<p>Роль: ' + userInfo.user\_info.role\_id + '</p>';  
details.innerHTML += '<button onclick="deleteUser()">Удалить пользователя</button>';  
if (userInfo.bookings.length > 0) {  
details.innerHTML += '<h3>Информация о бронированиях:</h3>';  
  
userInfo.bookings.forEach(function (booking) {  
details.innerHTML += '<p>Бронь ID: ' + booking.reservation\_id + '</p>';  
details.innerHTML += '<p>Количество гостей: ' + booking.guests + '</p>';  
details.innerHTML += '<p>Дата и время: ' + booking.reservation\_date + ' ' + booking.reservation\_time + '</p>';  
details.innerHTML += '<button onclick="cancelBooking(' + booking.reservation\_id + ')">Отменить бронь</button>';  
  
// Добавляем кнопки для изменения даты и времени брони, а также количества гостей  
details.innerHTML += '<button onclick="showUpdateBookingForm(' + booking.reservation\_id + ')">Изменить бронь</button>';  
  
details.innerHTML += '<hr>';  
});  
} else {  
details.innerHTML += '<p>Пользователь не сделал бронирований.</p>';  
}  
  
// Отобразить модальное окно  
modal.style.display = 'block';  
}

Здесь мы просто получаем данные пользователя и с помощью innerHTML добавляем эти данные в те пустые HTML блоки. А с помощью стилей это окно как раз так и выглядит.

Выше был представлен функционал администратора на заключительной панели (панель пользователя web-прилоежния).

# ЗАКЛЮЧЕНИЕ

В заключении хочется сказать, что создание web-приложений играет важную роль в современном обществе, поскольку они обладают большим потенциалом для улучшения работы и удовлетворения потребностей в различных сферах.

Web-приложение в сфере общественного питания и бронирования онлайн имеет важную роль в улучшении процесса заказа и резервирования столов и услуг в ресторанах, кафе и других заведениях.

Одна из основных функций такого приложения - обеспечение удобства и доступности процесса бронирования для клиентов. Пользователи могут удобно выбирать дату, время и количество мест, которые им необходимы. Они могут также просмотреть доступные акции, специальные предложения и узнать дополнительную информацию о заведении, такую как меню, цены и отзывы. Кроме того, клиенты могут сразу же узнать о наличии свободных столов и осуществить бронирование в режиме реального времени.

Для владельцев ресторанов и кафе web-приложение предоставляет мощный инструмент управления бронированиями. Они могут легко контролировать доступность столов в режиме реального времени, управлять расписанием и предлагать специальные предложения для привлечения новых клиентов. Приложение также облегчает взаимодействие с клиентами, позволяя удобно управлять резервациями, изменениями или отменами заказов.

В целом, web-приложение в сфере общественного питания и бронирования онлайн играет роль мощного инструмента для улучшения опыта клиентов, упрощения и автоматизации процесса бронирования и управления заказами. Оно помогает улучшить эффективность и рентабельность бизнеса в этой сфере, а также улучшить коммуникацию между клиентами и владельцами заведений.

# 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. Алексеев Ю.М. Быстро и легко создаем, программируем, шлифуем и раскручиваем web-сайт: учеб. пособие / Ю.М. Алексеев. - М.: Лучшие книги, 2011. - 189 с.
5. Гончаров А.Ю. Web-дизайн: HTML, JavaScript и CSS. Карманный справочник / А.Ю. Гончаров. - М.: «КУДИЦ-ПРЕСС», 2010. - 320 с.
6. Дунаев В. JavaScript: учеб. пособие / В. Дунаев. - СПб.: Питер, 2010. - 394 с.
7. Кожемякин А. HTML и CSS в примерах. Создание Web-страниц: учебник / А. Кожемякин. - М.: Альтекс-А, 2011. - 415 с.
8. Яргер, Р.Дж. MySQL и mSQL: Базы данных для небольших предприятий и Интернета / Р.Дж. Яргер, Дж. Риз, Т. Кинг. - М.: СПб: Символ-Плюс, **2014**. - 560 c.