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 «Сауны»»

тема

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

подпись, дата инициалы, фамилия

Студент УБ22-08Б, 432217710 \_\_\_\_\_\_\_\_ А.А. Дугаров

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc155983464)

[1 Теоретическая часть 5](#_Toc155983465)

[1.1 HTML и CSS 5](#_Toc155983466)

[1.2 JavaScript 6](#_Toc155983467)

[1.3 PHP 7](#_Toc155983468)

[1.4 MySQL 8](#_Toc155983469)

[2 Практическая часть 10](#_Toc155983470)

[2.1 Требования и описание реализации 10](#_Toc155983471)

[2.2 Выполнение работы 11](#_Toc155983472)

[Заключение 27](#_Toc155983473)

[Список использованных источников 28](#_Toc155983474)

# Введение

В современном мире, сфера web-программирования приобретает особую значимость. Одним из важных направлений в этой области является создание веб-приложений, направленных на улучшение и автоматизацию различных бизнес-процессов. В рамках данной курсовой работы рассматривается перспективное направление – создание сайта, посвященного саунам, с основным функционалом, связанным с бронированием саун.

Подобно системе общественного питания, сфера досуга также испытывает потребность в современных решениях для автоматизации внутренних процессов. Внимание уделяется оптимизации и улучшению процесса бронирования саун. Web-разработка открывает перед нами возможность создания технологичного продукта с увлекательным функционалом, предназначенного как для посетителей, так и для администраторов.

Создание базы данных MySQL для веб-приложения по бронированию саун является ключевым шагом к обеспечению эффективности функционирования данного приложения. Ниже представлены основные аспекты, обосновывающие важность создания такой базы данных:

1. Эффективное управление данными: База данных позволяет эффективно управлять информацией о посетителях и саунах. Это существенно упрощает процесс поиска и обработки данных, способствуя более эффективному бронированию саун.

2. Хранение большого объема данных: Веб-приложение, связанное с бронированием саун, может содержать обширную информацию о клиентах, саунах, услугах и прочих связанных данных. MySQL обеспечивает надежное хранение этой информации, обеспечивая эффективное управление даже при значительных объемах данных.

3. Поддержка множества пользователей: Бронирование саун может быть доступно для множества пользователей, включая посетителей и администраторов. MySQL обеспечивает многопользовательскую поддержку, что позволяет без проблем работать с базой данных нескольким пользователям одновременно, исключая проблемы с доступом и интеграцией данных.

Создание базы данных MySQL для веб-приложения по бронированию саун является неотъемлемой частью обеспечения эффективности и гибкости функционирования приложения. Это ключевой компонент, способствующий эффективному управлению данными, обеспечивающий безопасность и защиту информации, а также создающий условия для взаимодействия с другими системами, обеспечивая более гибкое управление процессом бронирования саун.

Целью курсовой работы является разработка сайта, используя технологии HTML, CSS, Javascript, PHP, MySQL, согласно заданной теме, а также разработка базы данных для управления пользователями, используя СУБД MySQL.

Объектом исследования является web-приложения, использующего разнообразные web-технологии.

Предметом исследования выступает процесс разработки веб-приложения и соответствующей базы данных.

В данной работе применяются методы анализа и синтеза информации, проектирование, а также практическая реализация.

# 1 Теоретическая часть

## HTML и CSS

HTML — это язык разметки, который используется для создания структуры веб-страниц. Он состоит из тегов, которые обозначают различные элементы страницы, такие как заголовки, абзацы, списки, ссылки и множество других элементов.

CSS — это язык стилей, используемый для описания внешнего вида веб-страницы. Он позволяет задавать различные стили для HTML-элементов, такие как цвет, размер шрифта, отступы и многое другое. CSS-стили можно применять к HTML-элементам с помощью селекторов.

Использование HTML и CSS в веб-разработке играет критически важную роль. Вот несколько аргументов, подчеркивающих этот факт:

1. HTML способствует определению структуры и логической семантики веб-страницы. Это делает контент более доступным для поисковых систем и улучшает взаимодействие со скринридерами, обеспечивая удобство использования для людей с ограниченными возможностями.

2. CSS позволяет разработчикам создавать привлекательные и стильные веб-страницы. Этот инструмент предоставляет возможность настройки визуального оформления элементов HTML, что позволяет создавать уникальный дизайн в соответствии с брендингом и предпочтениями пользователей. Это имеет фундаментальное значение для формирования положительного пользовательского опыта.

3. HTML и CSS позволяют создавать адаптивные и отзывчивые веб-страницы, которые могут легко адаптироваться к различным устройствам и экранам, включая мобильные телефоны, планшеты и настольные компьютеры. Это важно для обеспечения удобства использования и достижения более широкой аудитории.

4. HTML отвечает за структуру и содержание веб-страницы, а CSS – за внешний вид. Такое разделение ответственности позволяет разработчикам эффективнее работать в команде, повышает уровень обслуживания и облегчает повторное использование кода.

В общем, применение HTML и CSS в веб-разработке является неотъемлемым элементом создания высококачественных и стильных веб-страниц. Эти технологии обеспечивают ключевые функции, способствующие повышению доступности, улучшению пользовательского опыта и обеспечению удобства использования веб-сайтов.

## JavaScript

JavaScript ("JS" для краткости) — это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript невероятно универсален и дружелюбен к новичкам. Обладая большим опытом, вы сможете создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое!

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

Программные интерфейсы приложения (API), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов.

Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook.

Также вы можете применить к вашему HTML сторонние фреймворки и библиотеки, что позволит вам ускорить создание сайтов и приложений.

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 (рекурсивный акроним словосочетания PHP: Hypertext Preprocessor) - это распространённый язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

Вместо рутинного вывода HTML-кода командами языка (как это происходит, например, в Perl или C), скрипт PHP содержит HTML с встроенным кодом (в нашем случае, это вывод текста "Привет, я - скрипт PHP!"). Код PHP отделяется специальными начальным и конечным тегами <?php и ?>, которые позволяют "переключаться" в "PHP-режим" и выходить из него.

PHP отличается от JavaScript тем, что PHP-скрипты выполняются на сервере и генерируют HTML, который посылается клиенту. Если бы у вас на сервере был размещён скрипт, подобный вышеприведённому, клиент получил бы только результат его выполнения, но не смог бы выяснить, какой именно код его произвёл. Вы даже можете настроить свой сервер таким образом, чтобы обычные HTML-файлы обрабатывались процессором PHP, так что клиенты даже не смогут узнать, получают ли они обычный HTML-файл или результат выполнения скрипта.

PHP крайне прост для освоения, но вместе с тем способен удовлетворить запросы профессиональных программистов. Не пугайтесь длинного списка возможностей PHP. Вы можете быстро начать, и уже в течение нескольких часов сможете создавать простые PHP-скрипты.

## MySQL

MySQL — это реляционная база данных (СУБД). Реляционная — значит внутри неё есть данные, которые связаны между собой, и эту связь можно представить в виде таблиц.

Технически MySQL — это много таблиц, как-то связанных между собой. Например, одна отвечает за товары, другая — за покупки, третья — за клиентов.

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.

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

# 2 Практическая часть

## 2.1 Требования и описание реализации

Курсовой проект предполагает Создание web-приложения с применением современных web-технологий по тематике «Сауны», 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 Выполнение работы

Описание функционала сайта:

1. Во-первых, необходимо создать стартовую страницу, на которой будет возможность зарегистрироваться или авторизоваться, если пользователь уже имеется в базе данных. На данном этапе создается файл index.html. Он представляет собой белый фон с окном по середине, со всеми нужными функциями. Также неавторизованный пользователь может сразу перейти на страницу с саунами.

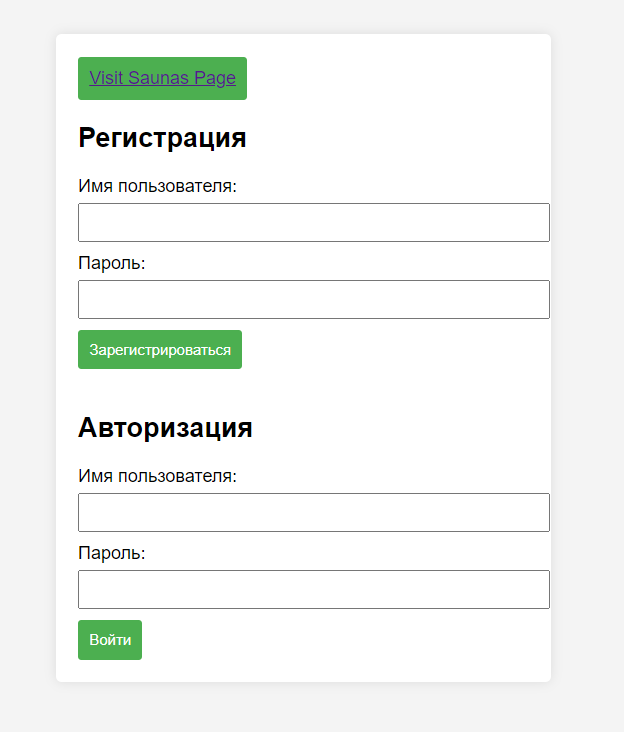


Рисунок 1 – Главная страница index.html

Листинг index.html:

<!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="styles.css">

<title>User Management System</title>

</head>

<body>

<div class="container">

<button type="submit"><a class="visit-saunas" href="nosaunas.php" style="font-size: 16px;">Visit Saunas Page</a></button>

<br>

<h2>Регистрация</h2>

<form action="register.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">Зарегистрироваться</button>

</form>

<br>

<h2>Авторизация</h2>

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

<label for="login\_username">Имя пользователя:</label>

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

<label for="login\_password">Пароль:</label>

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

<button type="submit">Войти</button>

</form>

</div>

</body>

</html>

Также используется следующие PHP файлы:

1. register.php:

<?php

require\_once('db.php');

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

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

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

$sql = "INSERT INTO users (username, password, role) VALUES ('$username', '$password', 'user')";

if ($conn->query($sql) === TRUE) {

header("Location: index.html");

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

$conn->close();

?>

2. login.php

<?php

session\_start();

require\_once('db.php');

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

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

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

$sql = "SELECT \* FROM users WHERE username='$login\_username'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

$row = $result->fetch\_assoc();

if (password\_verify($login\_password, $row['password'])) {

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

header("Location: dashboard.php");

} else {

echo "Invalid password";

}

} else {

echo "User not found";

}

}

$conn->close();

?>

3. logout.php:

<?php

session\_start();

session\_destroy();

header("Location: index.html");

?>

2. Во-вторых, создадим «Личный кабинет». В нем пользователь может увидеть свой ID, перейти на страницу с саунами, а также изменить свои данные. Также данная страница предусматривает возможность роли администратора. Администратор помимо своих данных, также может изменять, удалять и добавлять пользователей. Для этого вверху страницы отображена админ-панель.

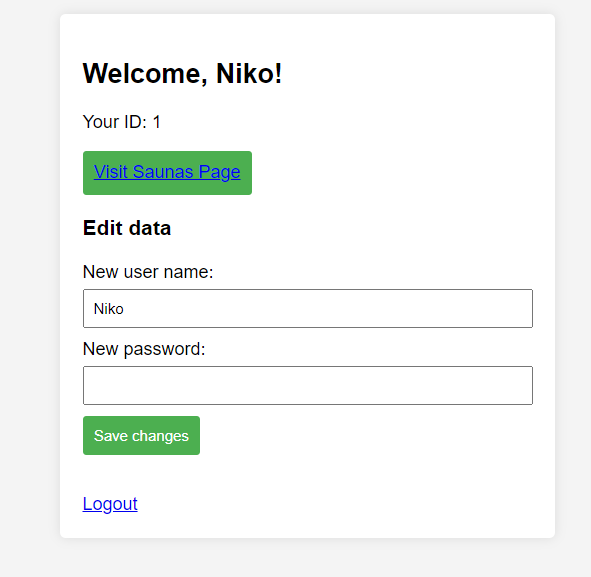


Рисунок 2 – Страница dashboard.php

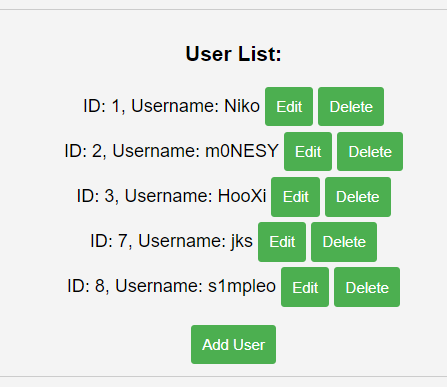


Рисунок 3 – Админ-панель

Листинг dashboard.php:

<?php

session\_start();

require\_once('db.php');

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

header("Location: index.html");

exit();

}

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

$sql\_role = "SELECT role FROM users WHERE id=$user\_id";

$result\_role = $conn->query($sql\_role);

if ($result\_role->num\_rows > 0) {

$user\_role = $result\_role->fetch\_assoc()['role'];

if ($user\_role === 'admin') {

$sql\_users = "SELECT id, username FROM users";

$result\_users = $conn->query($sql\_users);

if ($result\_users->num\_rows > 0) {

echo "<div class='admin-panel'>";

echo "<h3>User List:</h3>";

echo "<ul>";

while ($row = $result\_users->fetch\_assoc()) {

echo "<li>ID: " . $row['id'] . ", Username: " . $row['username'] . "

<button onclick='editUser(" . $row['id'] . ")'>Edit</button>

<button onclick='confirmDelete(" . $row['id'] . ")'>Delete</button></li>";

}

echo "</ul>";

echo "<button onclick='showAddUserForm()'>Add User</button>";

echo "</div>";

} else {

echo "No registered users.";

}

}

$sql = "SELECT \* FROM users WHERE id=$user\_id";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

$user = $result->fetch\_assoc();

echo "<h2> </h2>";

} else {

echo "User not found";

}

} else {

echo "Error fetching user role.";

}

$conn->close();

?>

<!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="styles.css">

<title>Dashboard</title>

</head>

<body>

<div class="container">

<h2>Welcome, <?php echo $user['username']; ?>!</h2>

<p>Your ID: <?php echo $user['id']; ?></p>

<button type="submit"><a class="visit-saunas" href="saunas.php" style="font-size: 16px;">Visit Saunas Page</a></button>

<h3>Edit data</h3>

<form action="edit\_user.php" method="post">

<input type="hidden" name="edit\_user\_id" id="edit\_user\_id" value="">

<label for="new\_username">New user name:</label>

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

<label for="new\_password">New password:</label>

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

<button type="submit">Save changes</button>

</form>

<br>

<a href="logout.php">Logout</a>

</div>

<script>

function editUser(userId) {

document.getElementById("edit\_user\_id").value = userId;

document.getElementById("new\_username").value = prompt("Enter new username:", "");

document.getElementById("new\_password").value = prompt("Enter new password:", "");

document.forms[0].submit();

}

function confirmDelete(userId) {

var confirmDelete = confirm("Are you sure you want to delete this user?");

if (confirmDelete) {

var deleteForm = document.createElement("form");

deleteForm.method = "post";

deleteForm.action = "delete\_user.php";

var input = document.createElement("input");

input.type = "hidden";

input.name = "user\_id";

input.value = userId;

deleteForm.appendChild(input);

document.body.appendChild(deleteForm);

deleteForm.submit();

}

}

function showAddUserForm() {

var username = prompt("Enter new username:", "");

var password = prompt("Enter new password:", "");

var addUserForm = document.createElement("form");

addUserForm.method = "post";

addUserForm.action = "add\_user.php";

var usernameInput = document.createElement("input");

usernameInput.type = "text";

usernameInput.name = "username";

usernameInput.value = username;

var passwordInput = document.createElement("input");

passwordInput.type = "password";

passwordInput.name = "new\_password";

passwordInput.value = password;

var addButton = document.createElement("button");

addButton.type = "submit";

addButton.textContent = "Add User";

addUserForm.appendChild(usernameInput);

addUserForm.appendChild(passwordInput);

addUserForm.appendChild(addButton);

document.body.appendChild(addUserForm);

addUserForm.submit();

}

</script>

</body>

</html>

Также используются следующие PHP файлы:

1. add\_user.php:

<?php

require\_once('db.php');

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

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

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

$sql = "INSERT INTO users (username, password) VALUES ('$newUsername', '$newPassword')";

if ($conn->query($sql) === TRUE) {

header("Location: dashboard.php");

exit();

} else {

echo "Error adding new user: " . $conn->error;

}

} else {

echo "Invalid request";

}

$conn->close();

?>

1. edit\_user.php:

<?php

require\_once('db.php');

if ($\_SERVER["REQUEST\_METHOD"] === "POST" && isset($\_POST['edit\_user\_id'])) {

$editUserId = $\_POST['edit\_user\_id'];

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

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

$sql = "UPDATE users SET username=?, password=? WHERE id=?";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("ssi", $newUsername, $newPassword, $editUserId);

if ($stmt->execute()) {

header("Location: dashboard.php");

exit();

} else {

echo "Error updating user data: " . $conn->error;

}

$stmt->close();

} else {

echo "Invalid request";

}

$conn->close();

?>

1. delete\_user.php:

<?php

require\_once('db.php');

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

$userToDelete = $\_POST['user\_id'];

$sql = "DELETE FROM users WHERE id = $userToDelete";

if ($conn->query($sql) === TRUE) {

header("Location: dashboard.php");

exit();

} else {

echo "Error deleting user: " . $conn->error;

}

} else {

echo "Invalid request";

}

$conn->close();

?>

3. В-третьих, создадим страницу с саунами, на которой будут представлены основные сауны, а также те, что были добавлены администраторами. Для этого создадим «панель Менеджера», куда сможет перейти только пользователь с ролью admin. На этой странице можно создавать новые сауны.

Также должна быть возможность для пользователя оставлять свой номер телефона, который автоматически добавляется в базу данных.

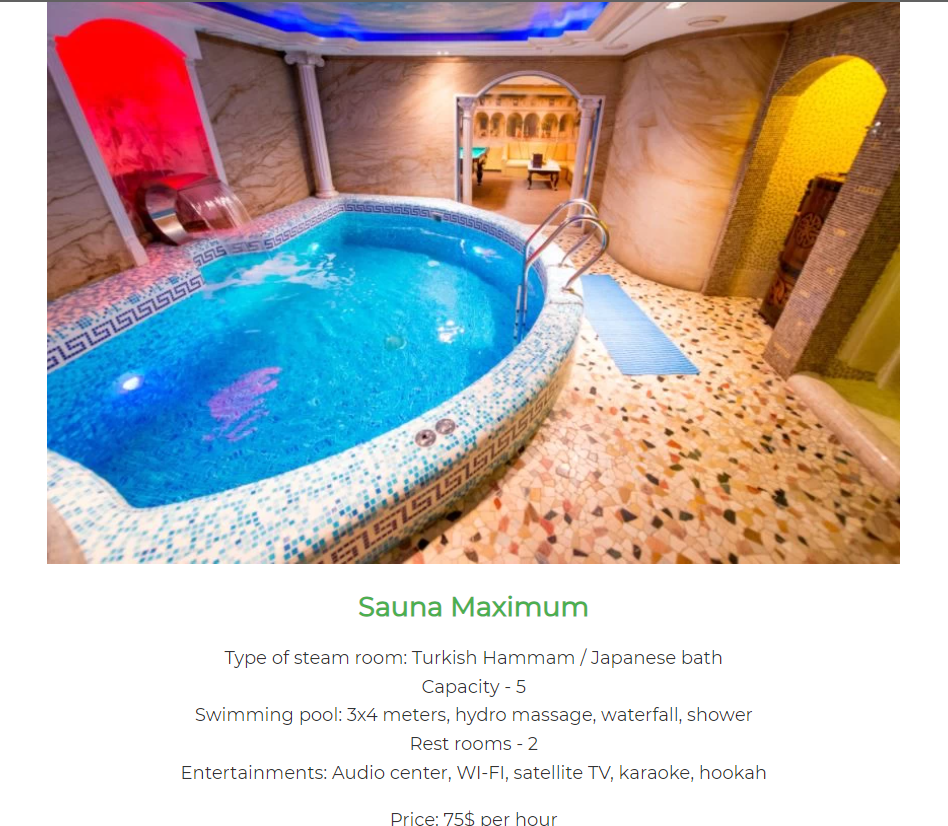


Рисунок 4 – Пример сауны со страницы saunas.php



Рисунок 5 – Кнопка перехода на страницу «панель Менеджера»

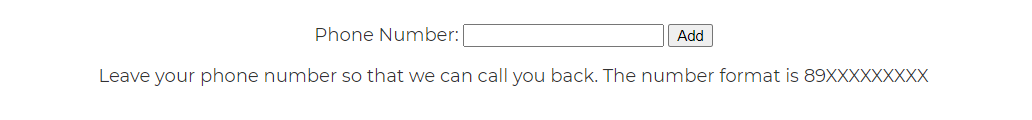


Рисунок 6 – Поле для ввода номера телефона

Листинг saunas.php:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<style>

body {

text-align: center;

}

a {

font-size: 24px;

text-decoration: none;

color: #4CAF50;

}

</style>

</head>

<body>

<?php

session\_start();

require\_once('db.php');

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

header("Location: index.html");

exit();

}

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

$sql\_role = "SELECT role FROM users WHERE id=$user\_id";

$result\_role = $conn->query($sql\_role);

if ($result\_role->num\_rows > 0) {

$user\_role = $result\_role->fetch\_assoc()['role'];

if ($user\_role === 'admin') {

echo '<a href="admin.php"><br>Go to Manager Page</a>';

}

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

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

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

<title>Saunas</title>

</head>

<body>

<div class="saunacontainer">

<h2>Our Saunas</h2>

</div>

<div class="container">

<form id="phoneForm">

<label for="phoneNumber">Phone Number:</label>

<input type="text" id="phoneNumber" name="phoneNumber" pattern="89\d{9}" required>

<?php

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

echo '<button type="button" onclick="openApplication()">Add</button>';

} else {

echo '<p>Login to add a phone number</p>';

}

?>

<p> Leave your phone number so that we can call you back. The number format is 89XXXXXXXXX</p>

</form>

</div>

<div class="saunacontainer">

<h3>Main Saunas</h3>

</div>

<div class="saunacontainer">

<div class="sauna">

<img src="sauna1.jpg" alt="Sauna 1" width="800" height="700">

<h3><a href="Odyssey.html">Odyssey Sauna</a></h3>

<p> Type of steam room: Turkish Hammam / Finnish sauna / Japanese bathhouse

<br>Capacity - 8

<br>Swimming pool: 3x6 meters, waterfall, geyser, artificial current, font, pouring bucket, shower

<br>Rest rooms - 2

<br>Entertainments: Billiards, board games, hookah, satellite TV, karaoke, audio center, WI-FI, home theater, wood-burning fireplace, massage chair</p>

<p>Price: 100$ per hour</p>

</div>

<div class="sauna">

<img src="sauna2.jpg" alt="Sauna 2" width="600" height="400">

<h3><a href="hype.html">Sauna HYPE</a></h3>

<p> Type of steam room: Finnish sauna

<br>Capacity - 15

<br>Swimming pool: 2.5x 3.5x 1.5 meters, warm, shower

<br>Rest rooms - 1

<br>Entertainments: Board games, Wi-Fi, audio center, satellite TV, karaoke, light music, massage chair, hookah</p>

<p>Price: 140$ per hour</p>

</div>

<div class="sauna">

<img src="sauna3.jpg" alt="Sauna 3" width="800" height="700">

<h3><a href="maximum.html">Sauna Maximum</a></h3>

<p> Type of steam room: Turkish Hammam / Japanese bath

<br>Capacity - 5

<br>Swimming pool: 3x4 meters, hydro massage, waterfall, shower

<br>Rest rooms - 2

<br>Entertainments: Audio center, WI-FI, satellite TV, karaoke, hookah</p>

<p>Price: 75$ per hour</p>

</div>

<div class="sauna">

<img src="sauna4.jpg" alt="Sauna 4" width="800" height="700">

<h3><a href="extra.html">Sauna Extra</a></h3>

<p> Type of steam room: Finnish sauna

<br>Capacity - 5

<br>Swimming pool: 2x3 meters, geyser, shower

<br>Rest rooms - 1

<br>Entertainments: Wi-Fi, audio center, satellite TV, karaoke, home theater, massage chair, board games</p>

<p>Price: 90$ per hour</p>

</div>

<div class="container">

<p align="center" style="color:#4CAF50">new saunas</p>

</div>

<?php

$sql\_saunas = "SELECT \* FROM saunas";

$result\_saunas = $conn->query($sql\_saunas);

if ($result\_saunas->num\_rows > 0) {

while ($row = $result\_saunas->fetch\_assoc()) {

echo '<div class="sauna">';

echo '<img src="' . $row['photo\_path'] . '" alt="' . $row['name'] . '" width="800" height="700">';

echo '<h3>' . $row['name'] . '</h3>';

echo '<p>Type of steam room: ' . $row['type\_of\_steam\_room'] . '<br>';

echo 'Capacity - ' . $row['capacity'] . '<br>';

echo 'Swimming pool: ' . $row['swimming\_pool'] . '<br>';

echo 'Rest rooms - ' . $row['rest\_rooms'] . '<br>';

echo 'Entertainments: ' . $row['entertainments'] . '</p>';

echo '<p>Price: ' . $row['price'] . '$ per hour</p>';

echo '</div>';

}

} else {

echo 'No saunas available.';

}

$conn->close();

?>

</div>

<div class="container">

<div class="contacts">

<div class="contact-block">

<h4>Our Phone Number</h4>

<p>+7 995 388-24-51</p>

</div>

<div class="contact-block">

<h4>Second Phone Number</h4>

<p>+7 950 387-52-39</p>

</div>

<div class="contact-block">

<h4>Social Media</h4>

<p><a href="https://vk.com/thebestbrawler"> My VK page</a></p>

</div>

<div class="contact-block">

<h4>Your profile</h4>

<a href="dashboard.php">Back to Dashboard</a>

</div>

</div>

</div>

<script>

function openApplication() {

var phoneNumberInput = document.getElementById('phoneNumber');

var phoneNumber = phoneNumberInput.value;

var phoneNumberRegex = /^89\d{9}$/;

if (phoneNumberRegex.test(phoneNumber)) {

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

alert("Your phone number has been added to the database.");

}

};

xmlhttp.open("GET", "add\_phone\_number.php?phone=" + phoneNumber, true);

xmlhttp.send();

} else {

alert("Invalid phone number format. Please enter a valid number.");

}

}

</script>

</body>

</html>

Используется файл add\_phone\_number.php:

<?php

session\_start();

require\_once('db.php');

if (isset($\_GET['phone'])) {

$phone = mysqli\_real\_escape\_string($conn, $\_GET['phone']);

$current\_time = date('Y-m-d H:i:s'); // Получаем текущее время

$sql = "INSERT INTO phone\_numbers (phone\_number, time\_added) VALUES ('$phone', '$current\_time')";

if ($conn->query($sql) === TRUE) {

echo "Phone number added successfully.";

} else {

echo "Error adding phone number: " . $conn->error;

}

} else {

echo "Invalid request.";

}

$conn->close();

?>

Листинг страницы «Панель Менеджера» (admin.php):

<?php

session\_start();

require\_once('db.php');

require\_once('add\_sauna.php');

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

$name = $\_POST['name'];

$type\_of\_steam\_room = $\_POST['type\_of\_steam\_room'];

$capacity = $\_POST['capacity'];

$swimming\_pool = $\_POST['swimming\_pool'];

$rest\_rooms = $\_POST['rest\_rooms'];

$entertainments = $\_POST['entertainments'];

$price = $\_POST['price'];

// Обработайте загрузку файла (фотографии)

$uploadDir = 'uploads/'; // Директория для загрузки файлов

$uploadFile = $uploadDir . basename($\_FILES['photo']['name']);

if (move\_uploaded\_file($\_FILES['photo']['tmp\_name'], $uploadFile)) {

// Файл успешно загружен

$photoPath = $uploadFile;

} else {

// Ошибка загрузки файла

$photoPath = ''; // или другой путь по умолчанию

}

// Добавьте новую сауну в базу данных

addSauna($name, $type\_of\_steam\_room, $capacity, $swimming\_pool, $rest\_rooms, $entertainments, $price, $photoPath);

}

$conn->close();

?>

<!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="saunastyles.css">

<title>Manager Panel</title>

</head>

<body>

<div class="container">

<h2>Manager Panel</h2>

<form action="admin.php" method="post" enctype="multipart/form-data">

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

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

<br>

<label for="type\_of\_steam\_room">Type of Steam Room:</label>

<input type="text" id="type\_of\_steam\_room" name="type\_of\_steam\_room" required>

<br>

<label for="capacity">Capacity:</label>

<input type="number" id="capacity" name="capacity" required>

<br>

<label for="swimming\_pool">Swimming Pool:</label>

<input type="text" id="swimming\_pool" name="swimming\_pool" required>

<br>

<label for="rest\_rooms">Rest Rooms:</label>

<input type="number" id="rest\_rooms" name="rest\_rooms" required>

<br>

<label for="entertainments">Entertainments:</label>

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

<br>

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

<input type="number" id="price" name="price" required>

<br>

<label for="photo">Photo:</label>

<input type="file" id="photo" name="photo" accept="image/\*" required>

<br>

<br>

<button type="submit" name="add\_sauna">Add Sauna</button>

</form>

<br>

<br>

<a href="saunas.php">Back to Saunas</a>

</div>

</body>

</html>

Также используется файл add\_sauna.php:

<?php

function addSauna($name, $type\_of\_steam\_room, $capacity, $swimming\_pool, $rest\_rooms, $entertainments, $price, $photoPath) {

global $conn;

// Подготовка данных перед добавлением в базу данных (защита от SQL-инъекций)

$name = mysqli\_real\_escape\_string($conn, $name);

$type\_of\_steam\_room = mysqli\_real\_escape\_string($conn, $type\_of\_steam\_room);

$swimming\_pool = mysqli\_real\_escape\_string($conn, $swimming\_pool);

$entertainments = mysqli\_real\_escape\_string($conn, $entertainments);

// SQL-запрос для добавления сауны

$sql = "INSERT INTO saunas (name, type\_of\_steam\_room, capacity, swimming\_pool, rest\_rooms, entertainments, price, photo\_path)

VALUES ('$name', '$type\_of\_steam\_room', $capacity, '$swimming\_pool', $rest\_rooms, '$entertainments', $price, '$photoPath')";

// Выполнение запроса

if ($conn->query($sql) === TRUE) {

// Успешно добавлено

$conn->close();

// Переход на страницу saunas.php

if (file\_exists("saunas.php")) {

header("Location: saunas.php");

exit();

} else {

echo "Error: saunas.php not found!";

}

} else {

// В случае ошибки выводим сообщение

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

?>

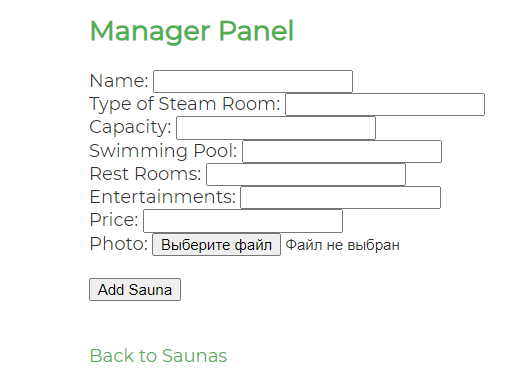


Рисунок 7 – Manager Panel

Далее рассмотрим созданную базу данных.

Созданная база данных имеет следующие таблицы:

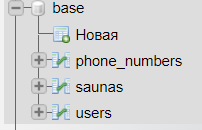


Рисунок 8 – База данных base

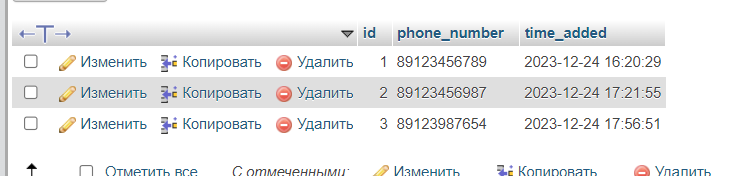


Рисунок 9 – Таблица phone\_numbers



Рисунок 10 – Таблица saunas

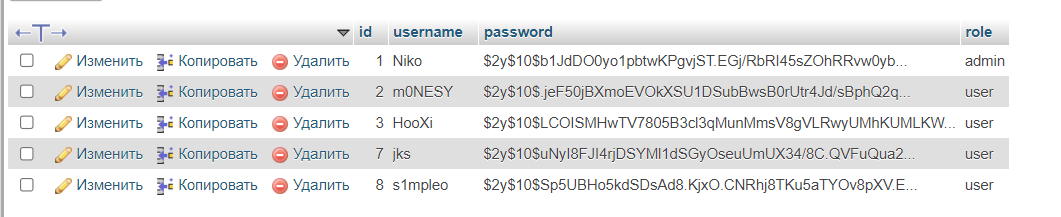


Рисунок 11 – Таблица users

# Заключение

В заключении, хочется подчеркнуть, что разработка web-приложений имеет важное значение в контексте современного общества, поскольку они предоставляют значительные возможности для усовершенствования бизнес-процессов и удовлетворения потребностей в различных областях.

Web-приложения, такие как сервис по бронированию саун, играют ключевую роль в улучшении процессов заказа и резервирования саун, предоставляя клиентам эффективные инструменты для удобного оформления бронирования.

Для владельцев саун web-приложение представляет мощный инструмент управления бронированиями. Они могут эффективно контролировать доступность саун в режиме реального времени. Приложение также упрощает взаимодействие с клиентами, обеспечивая удобное управление базой данных.

В общем, web-приложение в сфере бронирования саун играет решающую роль в улучшении опыта клиентов, автоматизации процессов бронирования и управления заказами. Оно способствует повышению эффективности и рентабельности бизнеса в данной области, а также улучшает взаимодействие между клиентами и владельцами саун.

# Список использованных источников

1. MySQL — царица баз // Журнал Яндекс Практикума URL: https://thecode.media/mysql/ (дата обращения: 19.12.2023).
2. Основы JavaScript // JavaScript.ru URL: https://learn.javascript.ru/first-steps (дата обращения: 21.12.2023).
3. Основы JavaScript // mdn web docs\_ URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics (дата обращения: 21.12.2023).
4. Подробнее о PHP // hexlet URL: https://ru.hexlet.io/courses/php-setup-environment/lessons/what-is-php/theory\_unit (дата обращения: 20.12.2023).
5. Что такое - PHP? // archive-irq-co URL: https://archive-ipq-co.narod.ru/ (дата обращения: 20.12.2023).
6. Что такое HTML и CSS // shneider host URL: https://shneider-host.ru/blog/chto-takoe-html-i-css.html (дата обращения: 21.12.2023).
7. Что такое HTML, CSS и JavaScript // skypro URL: https://sky.pro/media/chto-takoe-html-css-i-javascript/ (дата обращения: 21.12.2023).
8. Что такое PHP? // php 8.3 URL: https://www.php.net/manual/ru/intro-whatis.php (дата обращения: 21.12.2023).
9. Язык разметки HTML, таблицы стилей CSS и язык программирования JavaScript // web creator URL: https://web-creator.ru/articles/about\_client\_side (дата обращения: 21.12.2023).