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 «Электроника».

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

подпись, дата

Студент УБ22-08Б, 432218064 \_\_\_\_\_\_\_\_\_\_\_ А.А. Иванова

подпись, дата

Красноярск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc154443562)

[1 Теоретическая часть 5](#_Toc154443563)

[1.1 HTML и CSS 5](#_Toc154443564)

[1.2 PHP 6](#_Toc154443565)

[1.3 MySQL 7](#_Toc154443566)

[1.4 JavaScript 8](#_Toc154443567)

[2 Практическая часть 10](#_Toc154443568)

[2.1 Требования и описание реализации 10](#_Toc154443569)

[2.2 Общая структура 11](#_Toc154443570)

[2.3 Создание базы данных 12](#_Toc154443571)

[2.4 Регистрация в web-приложении 15](#_Toc154443572)

[2.5 Авторизация в web-приложении 23](#_Toc154443573)

[2.6 Запись к мастеру в web-приложении 27](#_Toc154443574)

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

[2.8 Функционал обычного пользователя 42](#_Toc154443576)

[ЗАКЛЮЧЕНИЕ 47](#_Toc154443578)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48](#_Toc154443579)

# ВВЕДЕНИЕ

В наше современное время электроника играет важную роль в нашей повседневной жизни. От мобильных телефонов до компьютеров, от бытовой техники до автомобилей - мы полностью зависим от электроники. Однако, как и любая другая технология, электроника может иногда выходить из строя и требовать ремонта.

В связи с этим, создание удобного и эффективного web-приложения становится все более актуальным и востребованным. Такое приложение поможет пользователям, столкнувшимся со сломанной электроникой, быстро найти и связаться с квалифицированными мастерами, специализирующимися на ремонте электроники.

Такое web-приложение будет использовать самые современные web-технологии, чтобы обеспечить удобный и интуитивно понятный интерфейс для пользователей. Оно будет иметь простую и понятную навигацию, чтобы любой пользователь мог с легкостью найти необходимую информацию и услуги.

Важным аспектом разработки этого web-приложения будет интеграция с базой данных, где хранится вся необходимая информация о мастерах и их услугах. Создание базы данных MySQL для web-приложения для записи к мастеру, 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 клиентах, дате и времени записи, количестве техники и других атрибутах заказа. Это позволяет упростить процесс поиска и обработки данных, что способствует более эффективной записи к мастеру.
* Хранение большого объема данных: web-приложение, связанное с записей к мастеру, может содержать значительное количество информации о клиентах, мастерах, 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: запись для починки необходимой техники может быть доступно для множества пользователей, как для клиентов, 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 записи к мастеру для починки электроники - 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 записи к мастеру.

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 - web-приложение с применением различных web-технологий.

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

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

# Теоретическая часть

## 1.1 HTML и CSS

HTML - язык для создания структуры и содержания веб-страниц. Он состоит из тегов, которые определяют различные элементы страницы. Это позволяет структурировать содержимое и правильно отображать страницу в браузере.

CSS - язык для описания внешнего вида веб-страницы. Он позволяет задавать различные свойства элементов HTML, такие как шрифт, цвет, размер и позиционирование. CSS помогает создавать стильные и привлекательные веб-страницы, упрощая работу с HTML.

Применение HTML и CSS в веб-разработке имеет важное значение по нескольким причинам:

1. Структура и семантика: HTML помогает определить структуру и логическую семантику веб-страницы. Это делает содержимое доступным для поисковых систем и улучшает доступность для людей с ограниченными возможностями.
2. Внешний вид и пользовательский опыт: CSS позволяет создавать стильные и привлекательные веб-страницы. Он позволяет настроить внешний вид элементов HTML, что важно для создания индивидуального дизайна и положительного пользовательского опыта.
3. Адаптивность и отзывчивость: HTML и CSS помогают создавать адаптивные веб-страницы, которые могут адаптироваться к различным устройствам и экранам. Это важно для удобства использования и привлечения большей аудитории.
4. Разделение ответственности: HTML отвечает за структуру и содержание, а CSS - за внешний вид страницы. Это разделение позволяет разработчикам более эффективно работать в команде, облегчая обслуживание и повторное использование кода.

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

## 1.2 PHP

PHP (Hypertext Preprocessor) представляет собой универсальный язык программирования, который широко применяется в разработке динамических веб-страниц и веб-приложений. Этот скриптовый язык выполняется на сервере, что означает, что код PHP обрабатывается на сервере, а результаты отображаются на устройстве клиента.

Значимость использования PHP в веб-разработке проявляется в нескольких аспектах:

1. Простота использования: PHP обладает простым и понятным синтаксисом, что делает его доступным для начинающих разработчиков. Большое сообщество PHP-разработчиков также предоставляет обширные обучающие материалы и поддержку.
2. Широкие возможности: PHP поддерживает различные базы данных (MySQL, PostgreSQL, Oracle и др.) и протоколы (HTTP, FTP, SMTP и др.), что позволяет создавать 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 дает разработчикам свободу использовать и изменять его по своему усмотрению. Это также означает наличие огромного количества бесплатных расширений и фреймворков для PHP.

Функции PHP включают:

* Генерация динамического контента: PHP позволяет создавать страницы, которые автоматически обновляются и адаптируются в зависимости от действий пользователя.
* Работа с формами: Легкая обработка данных, полученных от пользователей через веб-формы, что позволяет создавать интерактивные приложения.
* Работа с базами данных: Мощные функции для работы с базами данных, обеспечивая хранение, изменение и получение данных.
* Организация и обработка файлов: Возможности работы с файлами на сервере, включая чтение, запись и удаление данных.
* Обработка ошибок и исключений: Контроль и обработка ошибок и исключений, что способствует созданию более надежных и безопасных приложений.

В целом, PHP представляет собой мощный инструмент для разработки веб-приложений, и его популярность продолжает расти благодаря простоте использования и обширным возможностям.

## MySQL

MySQL - это популярная система управления базами данных, которая обеспечивает организацию и управление большими объемами информации. Ее широко используют для создания веб-приложений.

Функционал MySQL включает в себя следующее:

* Создание и управление базами данных: MySQL позволяет создавать новые базы данных и управлять уже существующими, включая добавление, изменение и удаление таблиц и полей.
* Хранение и обработка информации: MySQL позволяет сохранять и обрабатывать различные типы данных, такие как числа, текст, даты и другие. Его возможности включают операции добавления, обновления и выборки информации из таблицы.
* Защита данных: MySQL обеспечивает защиту информации, предоставляя функции шифрования и аутентификации. Кроме того, он поддерживает модель доступа с разграничением ролей, что позволяет контролировать доступ пользователей к базе данных и ее объектам.
* Высокая производительность: MySQL разработан с учетом высокой скорости обработки и возможности работы с большим количеством запросов одновременно.

Применение MySQL в разработке веб-приложений, таких как системы записи к мастерам для починки электроники, имеет важное значение по следующим причинам:

Хранение информации: MySQL обеспечивает надежное хранение данных о клиентах, записях, мастерах и других сведениях.

Обработка запросов: с помощью MySQL легко выполнять запросы на выборку, обновление и удаление информации, что позволяет эффективно управлять записями к мастерам.

Масштабируемость: MySQL поддерживает возможность горизонтального и вертикального масштабирования, что позволяет расширять функционал системы вместе с ростом бизнеса и количества записей.

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

В общем, использование MySQL для системы записи к мастерам для починки электроники обеспечивает эффективное управление и обработку данных, защиту информации и развитие функционала системы.

## JavaScript

JavaScript - язык программирования, который часто используется для создания динамических и интерактивных веб-сайтов. Он позволяет веб-разработчикам добавлять разные функции и взаимодействие на веб-страницах, что делает пользовательский интерфейс более интересным и привлекательным.

JavaScript имеет несколько важных функций. Во-первых, он позволяет вам изменять и модифицировать HTML-элементы на веб-странице, добавлять, удалять или изменять элементы, классы, стили и атрибуты. Во-вторых, с помощью JavaScript можно реагировать на различные действия пользователя на веб-странице, такие как нажатие кнопок, клики мыши и наведение курсора. Это позволяет создавать интерактивные и отзывчивые пользовательские интерфейсы.

Также JavaScript позволяет загружать данные с сервера без перезагрузки всей страницы, что помогает создавать динамические приложения, обновляющиеся без прерывания работы пользователя. Он предоставляет возможности для обработки и манипуляции данными, включая чтение, запись, редактирование и преобразование. Кроме того, JavaScript дает возможность создавать анимацию на веб-страницах, изменяя стили и свойства элементов.

JavaScript важен для веб-разработки, потому что он добавляет динамичность и интерактивность на веб-страницы. Он позволяет создавать более привлекательные визуальные эффекты, улучшать пользовательский опыт, функциональность и время загрузки страницы. JavaScript широко используется в современных веб-сайтах и приложениях, поэтому знание этого языка программирования является важным навыком для веб-разработчиков.

JavaScript имеет несколько преимуществ. Во-первых, он поддерживается всеми современными веб-браузерами, что обеспечивает совместимость и доступность веб-сайтов для всех пользователей. Во-вторых, он относительно прост в изучении, имеет синтаксис, похожий на другие языки программирования, такие как Java и C++, и множество ресурсов и учебных материалов для начинающих. Также JavaScript легко интегрируется с HTML и CSS, что позволяет создавать привлекательные и кроссплатформенные веб-приложения. И, наконец, JavaScript имеет большое сообщество разработчиков и фреймворков, что обеспечивает быстрое решение проблем и обмен опытом.

# 2 Практическая часть

## 2.1 Требования и описание реализации

Курсовой проект предполагает создание web-приложение на тему «Электроника», 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).

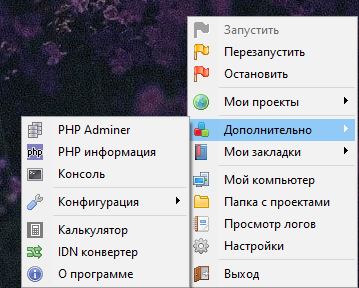


Рисунок 1 — Open server: встроенный функционал

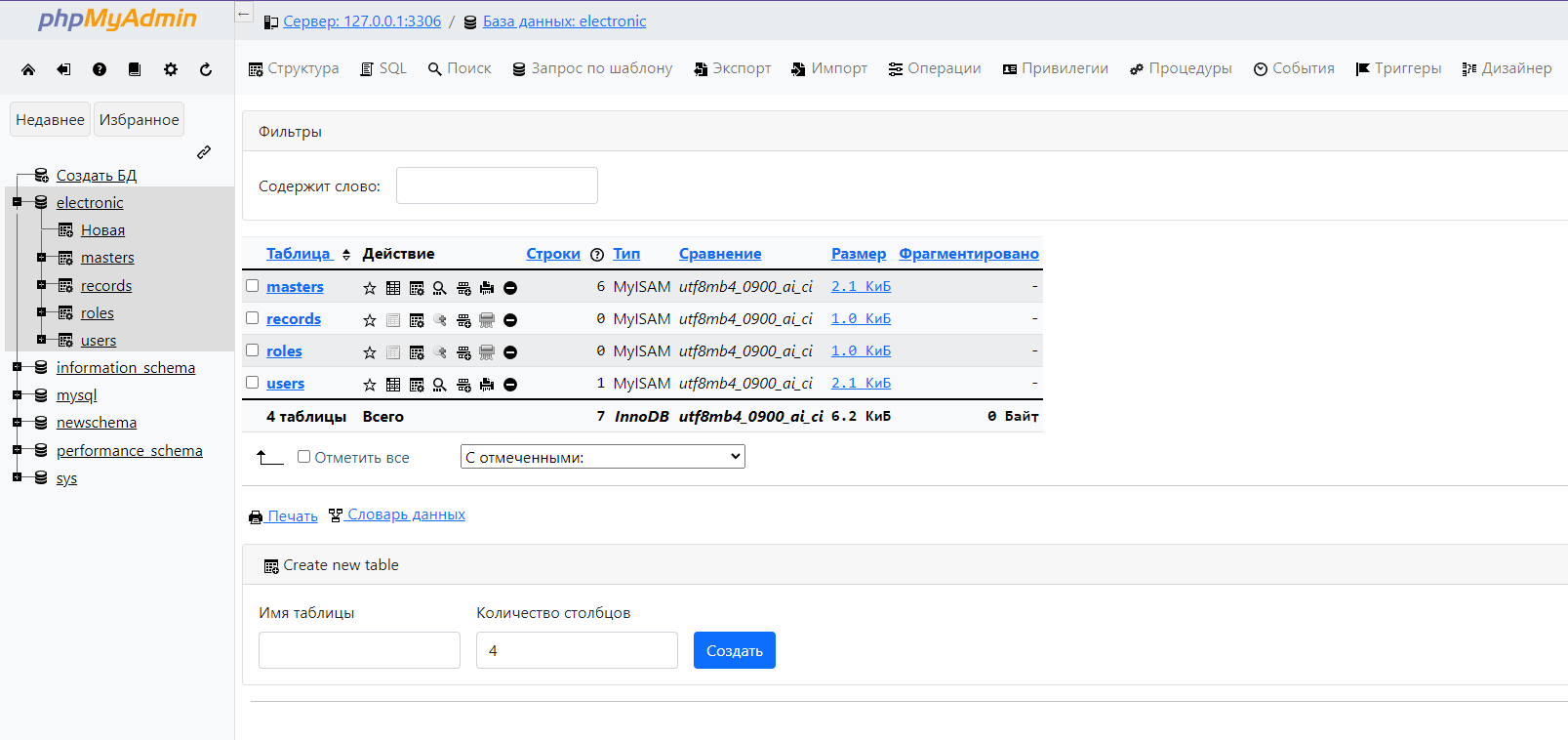


Рисунок 2 — База данных

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

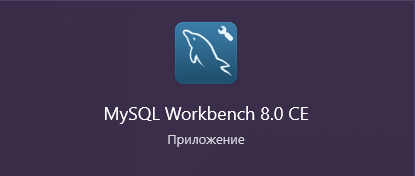


Рисунок 3 — Установка MySQL

## 2.3 Создание базы данных

Листинг создания базы данных:

-- Создание базы данных

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

-- Использование базы данных

USE electronic;

-- 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(20) NOT NULL,

password VARCHAR(100) NOT NULL,

email VARCHAR(110) NOT NULL,

role\_ID INT,

unique (email),

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

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

-- Таблица мастеров

CREATE TABLE IF NOT EXISTS masters (

master\_ID INT AUTO\_INCREMENT PRIMARY KEY,

master\_name VARCHAR(20) NOT NULL,

ability INT NOT NULL,

UNIQUE (master\_name)

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

-- Таблица записей

CREATE TABLE IF NOT EXISTS records (

records\_ID INT AUTO\_INCREMENT PRIMARY KEY,

user\_ID INT,

master\_ID INT,

electronics INT,

records\_date DATE NOT NULL,

records\_time TIME NOT NULL,

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

FOREIGN KEY (master\_ID) REFERENCES master(master\_ID)

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

Название базы данных “electronic” 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. Электронная почта должна быть уникальной, зарегистрировать двух пользователей на одну и ту же почту невозможно. После создаётся таблица мастеров, которая содержит в себе ID мастера, его имя и сколько электроники может починить тот или иной мастер. И создаётся таблица записи, которая связывает пользователя с его мастером (содержит в себе дату, время и количество техники).

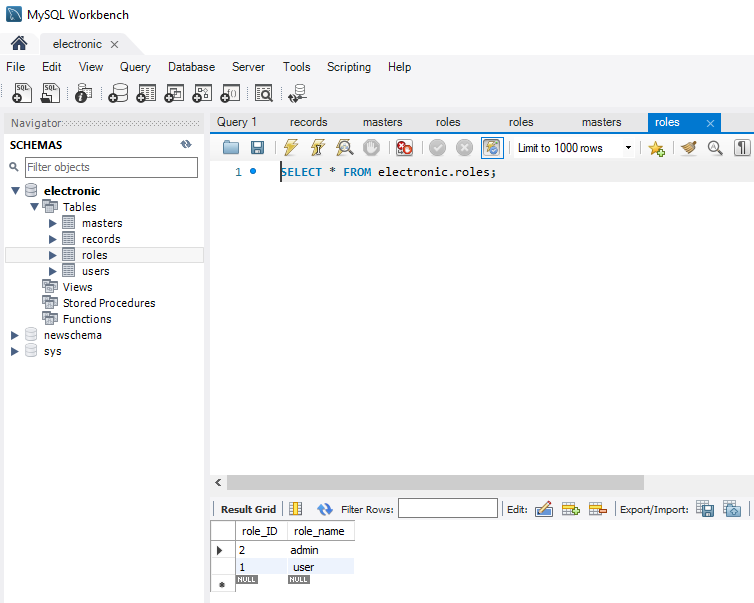


Рисунок 4 — База данных: роль пользователя и админа

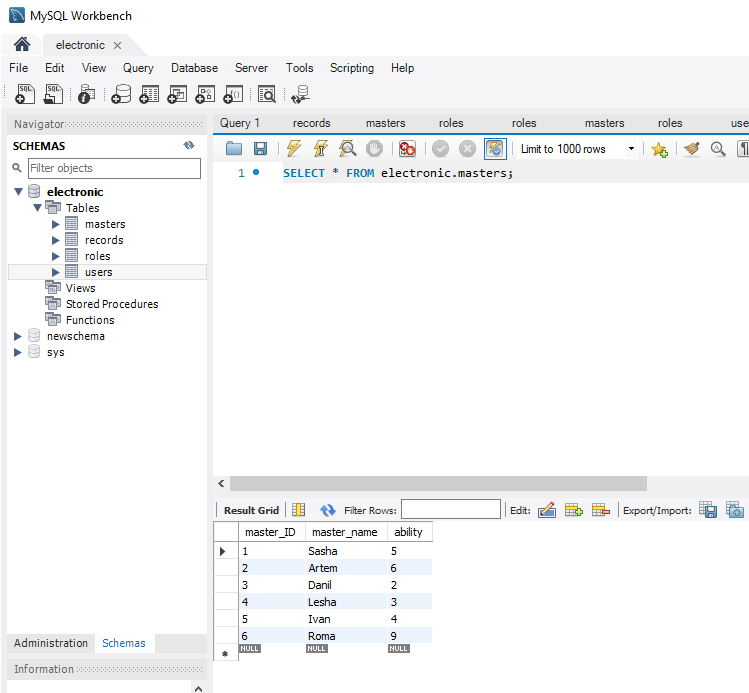


Рисунок 5 — База данных: имена мастеров и количество техники

На рисунке 6 показываются записи. ID пользователя, ID мастера, количество техники, дата и время бронирования.

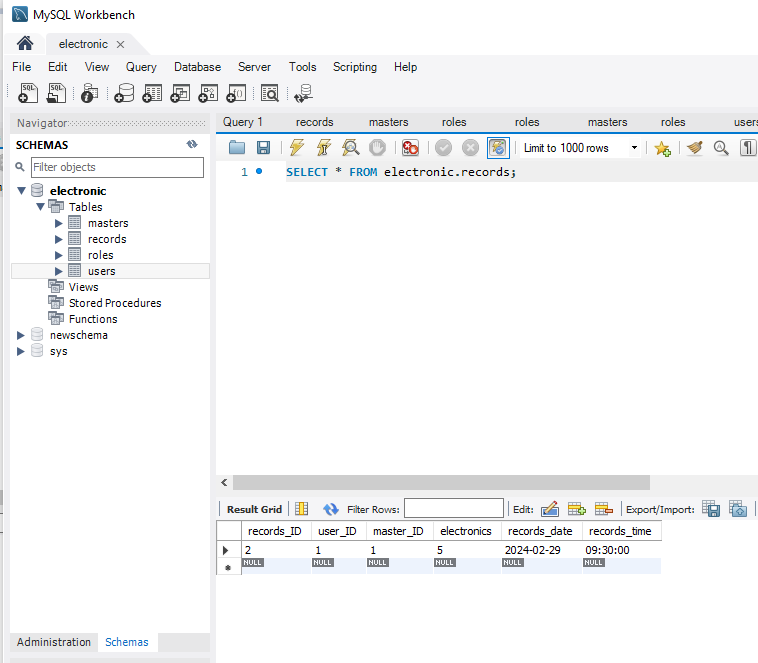


Рисунок 6 — База данных: брони

## 2.4 Регистрация в web-приложении

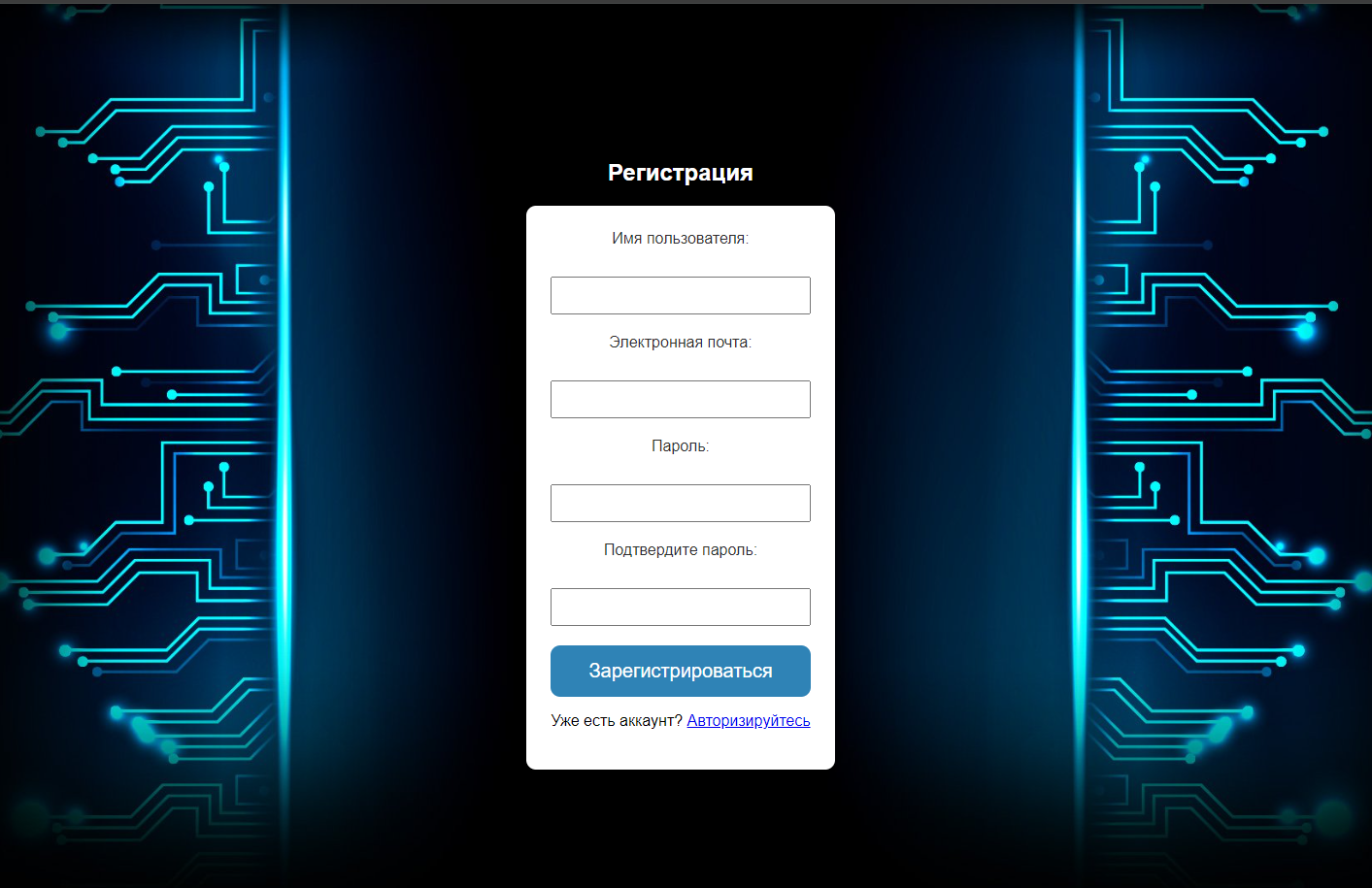


Рисунок 7 — Регистрация в web-приложении

Листинг HTML составляющей панели регистрации:

<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>

</body>

</html>

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('1.jpg');

background-size: contain;

background-position: center;

}

.container {

text-align: center;

}

h2 {

color: #ffffff;

}

form {

max-width: 400px;

margin: 0 auto;

background: #fff;

padding: 25px;

border-radius: 10px;

box-shadow: 0 0 15px #000;

}

label {

display: block;

margin-bottom: 30px;

color: #2e2e2e;

}

input {

width: 100%;

padding: 10px;

margin-bottom: 20px;

box-sizing: border-box;

}

button {

background-color: #2F83B6;

color: white;

padding: 15px 20px;

border: none;

border-radius: 10px;

cursor: pointer;

font-size: 20px;

width: 100%;

}

button:hover {

background-color: #2F83B6;

}

.error {

color: red;

margin-top: 10px;

}

Добавляем функцию JavaScript — function validateForm(). При нажатии на кнопку «зарегистрироваться» активируется данная функция и 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>

При отправке формы активируется PHP код, устанавливается подключение к базе данных. Указываем имя базы данных, имя пользователя, имеющего доступ и пароль. Листинг кода:

<?php

session\_start();

$host = 'localhost';

$db\_name = 'electronic';

$username = 'kot';

$password = 'owo';

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();

if (!$roleId) {

$role\_ID = 1;

} else {

$role\_ID = $roleId;

}

// 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, $role\_ID]);

// 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. Листинг кода:

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

die("Пароли не совпадают");

}

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

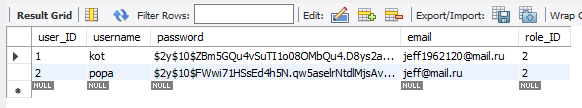


Рисунок 8 — Сохранение регистрации пользователя в базе данных и хеширование пароля

После 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 кода:

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

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

$emailCount = $stmtCheckEmail->fetchColumn();

if ($emailCount > 0) {

die("Адрес электронной почты уже занят");

}

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

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

$stmtRole->execute();

$roleId = $stmtRole->fetchColumn();

if (!$roleId) {

$role\_ID = 1;

} else {

$role\_ID = $roleId;

}

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

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

## 2.5 Авторизация в web-приложении

Завершив регистрацию, пользователя перенаправит на страницу авторизации. Листинг кода:

header("Location: login.php");

exit();

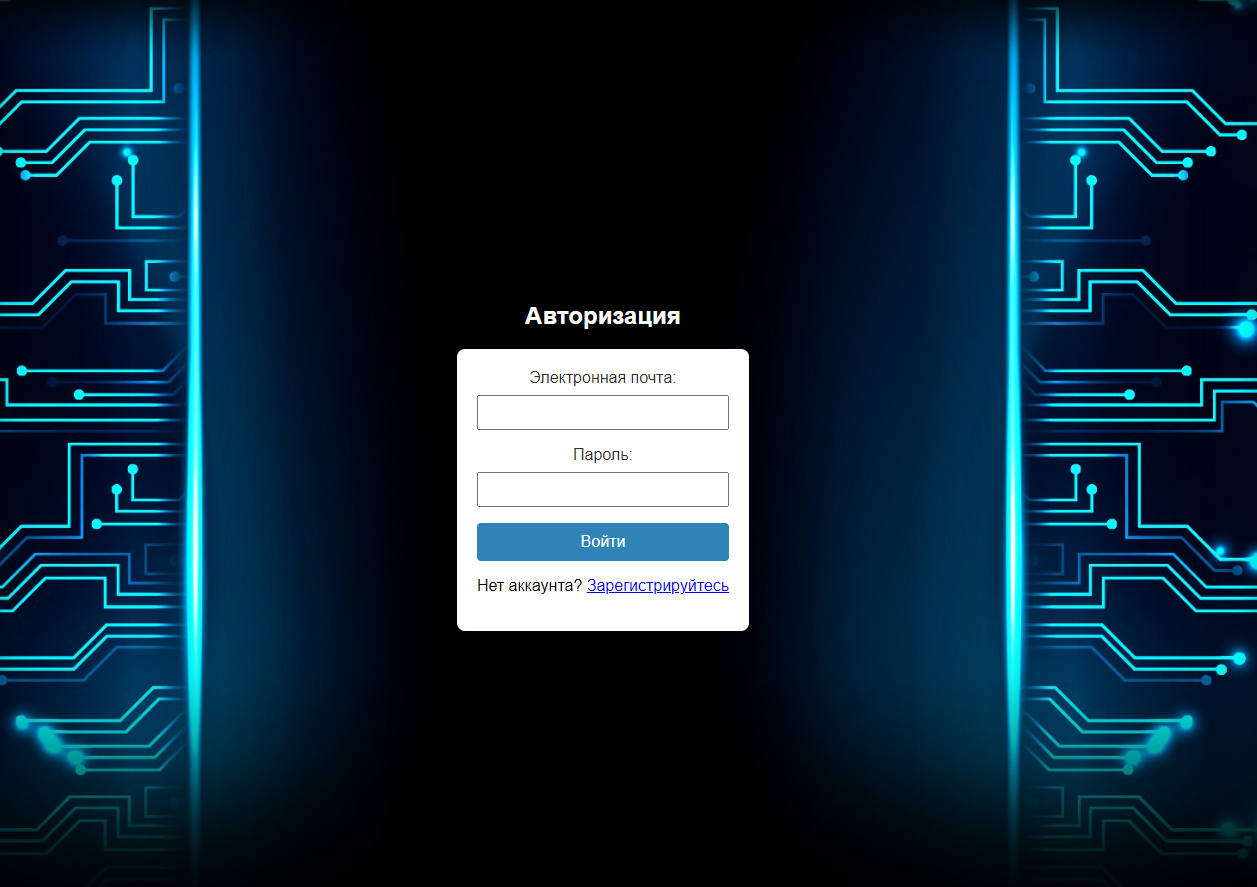


Рисунок 9 — Авторизация в web-приложении

Листинг HTML составляющей панели авторизации:

<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 кода:

<?php

session\_start();

$host = 'localhost';

$db\_name = 'electronic';

$username = 'kot';

$password = 'owo';

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();

}

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

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'];

header("Location: recording.php");

exit();

} else {

}

}

?>

Листинг стилей для панели авторизации:

body{

height: 100vh;

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

background-color: #2e2e2e;

margin: 0;

padding: 0;

display: flex;

align-items: center;

justify-content: center;

background-image: url('1.jpg');

background-size: contain;

background-position: center;

}

.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: #2e2e2e;

}

input {

width: 100%;

padding: 8px;

margin-bottom: 16px;

box-sizing: border-box;

}

button {

background-color: #2F83B6;

color: white;

padding: 10px 15px;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

width: 100%;

}

button:hover {

background-color: #2F83B6;

}

.error {

color: red;

margin-top: 8px;

}

## 2.6 Запись к мастеру в web-приложении

После авторизации, пользователь попадает на страницу записи к мастеру, где он может указать удобные для себя дату, время и количество техники или перейти к себе в профиль.

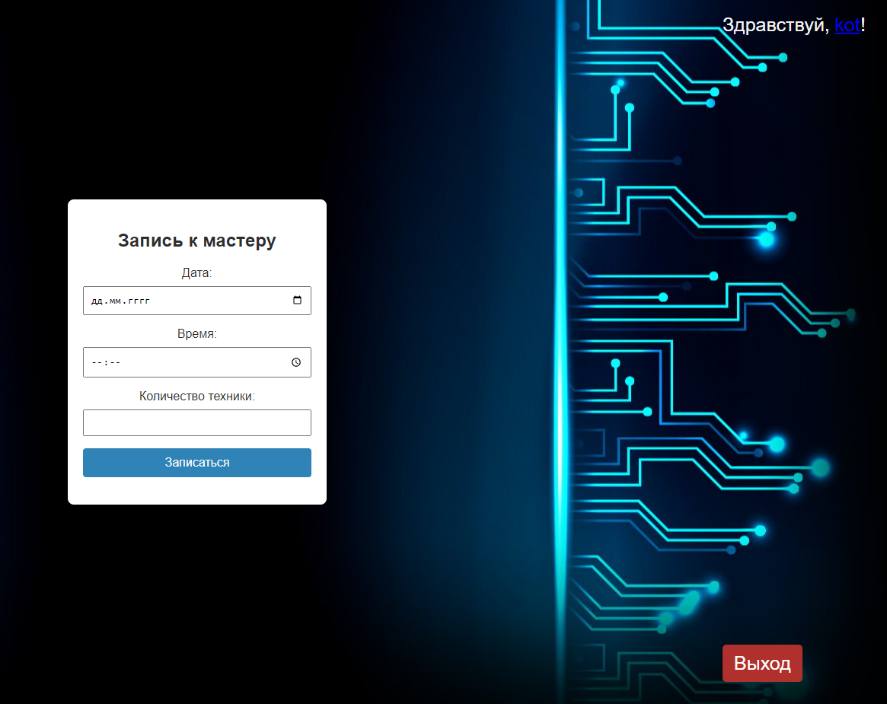


Рисунок 10 — Бронирование столика в web-приложении

Листинг HTML составляющей панели для записи:

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>Запись к мастеру</title>

</head>

<body>

<div class="user-info">

</div>

<div class="container">

<h2>Запись к мастеру</h2>

<form action="recording.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="electronics">Количество техники:</label>

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

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

</form>

</div>

<script>

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

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

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

// Проверка наличия авторизации

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

// Установка состояния кнопки в зависимости от авторизации

recordingButton.disabled = !isUserAuthenticated;

});

</script>

</body>

</html>

Приветствие пользователя, за это отвечает следующий PHP код, встроенный в HTML:

<?php

// Проверка существования сессии пользователя

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

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

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

}

?>

Мы получаем данные об авторизованном пользователе (его имя) и используем их для создания персонализированного приветствия. При нажатии на имя пользователя, он будет перенаправлен в свой личный кабинет. HTML-форма содержит поля для ввода информации о дате, времени и количестве электроники, а также кнопку "Записаться". Важно отметить, что каждый мастер имеет определенное количество чинимой техники, поэтому пользователь может записать количество электроники, соответствующее возможностей выбранного мастера.

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

<?php

session\_start();

// Удаление всех данных сессии

session\_unset();

session\_destroy();

header("Location: login.php");

exit();

?>

Листинг проверки авторизации пользователя:

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

header("Location: login.php");

exit();

}

После ввода данных они отправляются на сервер. Мы получаем информацию о ID пользователя, дате, времени и количестве электроники (Рисунок 11). Затем мы проверяем полученный список мастеров, чтобы найти подходящего по количеству техники. Если подходящего мастера нет, система выдаёт ошибку. После этого мы проверяем, есть ли свободный мастер на указанное время. Если такого мастера нет, система также выдаёт ошибку. При полностью корректном вводе информации создаётся запись в базе данных. После успешной записи пользователь будет перенаправлен на страницу личного кабинета. Листинг кода:

$stmtTables = $pdo->prepare("SELECT master\_ID, ability FROM masters");

$stmtTables->execute();

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

// Обработка формы записи

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

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

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

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

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

// Поиск подходящего мастера в зависимости от поломки электроники

$stmtFindMaster = $pdo->prepare("SELECT master\_ID FROM masters WHERE ability >= ? ORDER BY ability ASC LIMIT 1");

$stmtFindMaster->execute([$electronics]);

$master = $stmtFindMaster->fetch(PDO::FETCH\_ASSOC);

if (!$master) {

die("Извините, нет подходящего мастера для указанного количества устройств.");

}

$master\_ID = $master['master\_ID'];

// Проверка доступности выбранного времени и мастера

$stmtCheckAvailAbility = $pdo->prepare("SELECT COUNT(\*) FROM records WHERE records\_date = ? AND records\_time = ? AND master\_ID = ?");

$stmtCheckAvailAbility->execute([$date, $time, $master\_ID]);

$availAbilityCount = $stmtCheckAvailAbility->fetchColumn();

if ($availAbilityCount > 0) {

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

}

// Вставка записи о записи в базу данных

$stmtInsertRecords = $pdo->prepare("INSERT INTO records (user\_ID, master\_ID, electronics, records\_date, records\_time) VALUES (?, ?, ?, ?, ?)");

$stmtInsertRecords->execute([$user\_ID, $master\_ID, $electronics, $date, $time ]);

// Редирект после успешной записи

header("Location: dashboard.php");

exit();

}

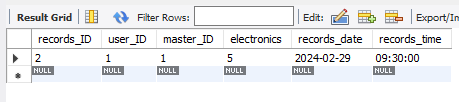


Рисунок 11 — Данные о брони на сервере

В HTML коде применен JavaScript для установки кнопки "записать" в недоступное состояние для неавторизованных пользователей. Однако, можно считать, что это излишне, так как код в любом случае предотвратит доступ к данной странице неавторизованным пользователям. Листинг кода:

<script>

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

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

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

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

recordingButton.disabled = !isUserAuthenticated;

});

</script>

Листинг стилей для панели записей:

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('1.jpg');

background-size: cover;

background-position: center;

}

.container {

text-align: center;

max-width: 300px;

width: 100%;

background: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px #000000;

}

h2 {

color: #2e2e2e;

}

form {

margin-top: 20px;

}

label {

display: block;

margin-bottom: 8px;

color: #2e2e2e;

}

input, select {

width: 100%;

padding: 8px;

margin-bottom: 16px;

box-sizing: border-box;

}

.user-info {

position: absolute;

top: 20px;

left: 1650px;

color: #f4f4f4;

font-size: 25px;

}

button {

background-color: #2F83B6;

color: white;

padding: 10px 15px;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

width: 100%;

}

button:hover {

background-color: #2F83B6;

}

a.logout-button {

display: inline-block;

background-color: #af302c;

color: #fff;

padding: 10px 15px;

text-decoration: none;

border-radius: 5px;

margin-top: 800px;

}

a.logout-button:hover {

background-color: #af302c;

}

## 2.7 Личный кабинет обычного пользователя

Рассмотрим личный кабинет обычного пользователя (рисунок 12).



Рисунок 12 — Личный кабинет пользователя в web-приложении

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

<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: #2e2e2e;

}

</style>

<h1>Панель пользователя</h1>

<h2>Здраствуй, <?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']; ?>

<!-- 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>

Листинг стилей для личного кабинета пользователя:

body {

font-family: Arial, sans-serif;

background-color: #363636;

margin: 0;

padding: 0;

height: 100vh;

box-sizing: border-box;

background-image: url('2.jpg');

background-size: cover;

background-position: center;

}

header {

background-color: #c2c2c2;

color: #fff;

font-size: 25px;

}

h1 {

color: #ffffff;

padding-left: 1350px;

font-size: 34px;

}

h2 {

color: #ffffff;

padding-left: 1300px;

font-size: 25px;

}

p {

color: #ffffff;

padding-left: 1350px;

font-size: 22px;

}

.userListItem{

max-width: 40px;

text-align: center;

padding-left: 50px;

}

button {

background-color: #2F83B6;

color: white;

padding: 10px 10px;

border: none;

border-radius: 9px;

cursor: pointer;

}

form {

margin-bottom: 25px;

}

ul {

list-style-type: none;

padding: 0;

margin: 0;

padding-left: 1350px;

}

li {

background-color: #fff;

padding: 15px;

margin: 10px;

border-radius: 10px;

cursor: pointer;

}

li:hover {

background-color: #f2f2f2;

}

label{

color: #d8d6f5;

}

#userInfoModal p{

color: #363636;

padding-left: 1px;

}

#userInfoModal h2{

color: #363636;

padding-left: 1px;

}

#userInfoModal label{

color: #363636;

padding-left: 1px;

}

.modal {

display: none;

position: fixed;

z-index: 1;

left: 0;

top: 0;

width: 100%;

height: 100%;

overflow: auto;

background-color: rgb(0, 0, 0, 0.6);

}

.modal-content {

background-color: #fefefe;

margin: 20% auto;

padding: 30px;

border: 5px solid #f2f2f2;

width: 40%;

}

.close {

color: #bababa;

float: right;

font-size: 60px;

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: #d24b46;

color: #fff;

padding: 15px 20px;

text-decoration: none;

border-radius: 10px;

top: 10px;

font-size: 22px;

margin-left: 1300px;

}

a.logout-button:hover {

background-color: #9f2723;

}

#editUsernameForm,

#editEmailForm,

#editRoleForm {

margin-top: 25px;

display: none;

}

#editUsernameForm label,

#editEmailForm label,

#editRoleForm label {

display: block;

margin-bottom: 10px;

}

#editUsernameForm input,

#editEmailForm input,

#editRoleForm input {

width: 100%;

padding: 10px;

margin-bottom: 15px;

box-sizing: border-box;

}

#editUsernameForm button,

#editEmailForm button,

#editRoleForm button {

background-color: #2F83B6;

color: white;

padding: 15px;

border: none;

cursor: pointer;

width: 100%;

box-sizing: border-box;

}

#editUsernameForm button:hover,

#editEmailForm button:hover,

#editRoleForm button:hover {

opacity: 0.9;

}

Когда вы нажимаете на кнопку «Изменить» на сайте, это вызывает JavaScript функцию "showUpdateUsernameForm()". А при нажатии кнопки «Сохранить» также вызывается JavaScript функция "updateUsername()". При этом скрытый JavaScript код создает переменную var updateUsernameForm = document.getElementById('updateUsernameForm'), чтобы обратиться к форме, и меняет CSS стиль на 'block' с помощью строки updateUsernameForm.style.display = 'block'. Также используется функция для отправки AJAX-запроса на обновление панели, чтобы обновление имени пользователя на сайте происходило быстрее. Листинг настройки AJAX-запроса:

function updateUsername() {

var newUsername = document.getElementById('updateUsernameForm').elements.newUsername.value;

// Отправка AJAX-запроса

var xhr = new XMLHttpRequest();

xhr.open('POST', 'dashboard.php', true); xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.onreadystatechange = function() {

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

// Обновляем страницу после успешного обновления имени

location.reload();

}

};

xhr.send('action=updateUsername&newUsername=' + newUsername);

}

Происходит отправка запроса с клиентской части на сервер, этот запрос активирует действие вызова PHP функции, которая принимает ID пользователя в качестве параметра. Затем в эту функцию передается запрос, содержащий новое имя пользователя в качестве параметра.

Листинг вывода информации о записи:

<h2>Ваши записи:</h2>

<?php

$userRecordings = getUserRecordings($pdo, $\_SESSION['user\_ID']);

if ($userRecordings) {

foreach ($userRecordings as $recording) {

?>

<div>

<p>Запись ID: <?php echo $recording['records\_ID']; ?></p>

<p>Кол-во техники: <?php echo $recording['electronics']?></p>

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

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

<input type="hidden" name="recordingId" value="<?php echo $recording['records\_ID']; ?>">

<button type="submit" style = "margin-left: 1400px">Отменить запись</button>

</form>

</div>

<?php

}

} else {

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

}

?>

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

// Получаем ID записи из POST-запроса

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

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

$userId = $\_SESSION['user\_ID'];

if (isRecordingBelongsToUser($pdo, $recordingId, $userId)) {

// Если запись принадлежит пользователю, удаляем ее

cancelRecording($pdo, $recordingId);

echo "Запись успешно отменена.";

} else {

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

}

} else {

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

header("Location: dashboard.php");

exit();

}

Выводим функции записи пользователя и функцию отмены записи. Листинг кода:

function isRecordingBelongsToUser($pdo, $recordingId, $userId) {

$query = "SELECT COUNT(\*) FROM records WHERE records\_ID = :recording\_ID AND user\_ID = :user\_ID";

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

$statement->bindParam(':recording\_ID', $recordingId, PDO::PARAM\_INT);

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

$statement->execute();

return $statement->fetchColumn() > 0;

}

// Функция для отмены записи

function cancelRecording($pdo, $recordingId) {

$query = "DELETE FROM records WHERE records\_ID = :recording\_ID";

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

$statement->bindParam(':recording\_ID', $recordingId, PDO::PARAM\_INT);

$statement->execute();

}

?>

После отмены записи пользователя будет перенаправлен на страницу с сообщением "Запись успешно отменена". Если пользователь обновит страницу на панели пользователя, то он увидит уведомление "У вас нет активных записей". Также там будет доступна кнопка выхода и кнопка для создания новой записи. Листинг кода:

<?php endif; ?>

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

<a href="recording.php" class="recording-link" style="font-size: 22px;">Записаться к мастеру</a>

## 2.8 Личный кабинет 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" style = "margin-left: 1350px" 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 вводе имени на сайте происходит фильтрация списка пользователей с использованием скрытого JavaScript кода в функции searchUsers(). Кроме того, с помощью JavaScript может запускаться функция ShowUserInfo для отображения списка всех пользователей из базы данных. Чтобы удалить пользователя, на сайте вводится его ID, который затем автоматически удаляется с использованием функции deleteUser(). Этот метод получает ID пользователя и с помощью AJAX-запроса запускает PHP-код для удаления пользователя. Листинг 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";

}

}

Отображение информации о пользователе в отдельном окне (рисунок 13).

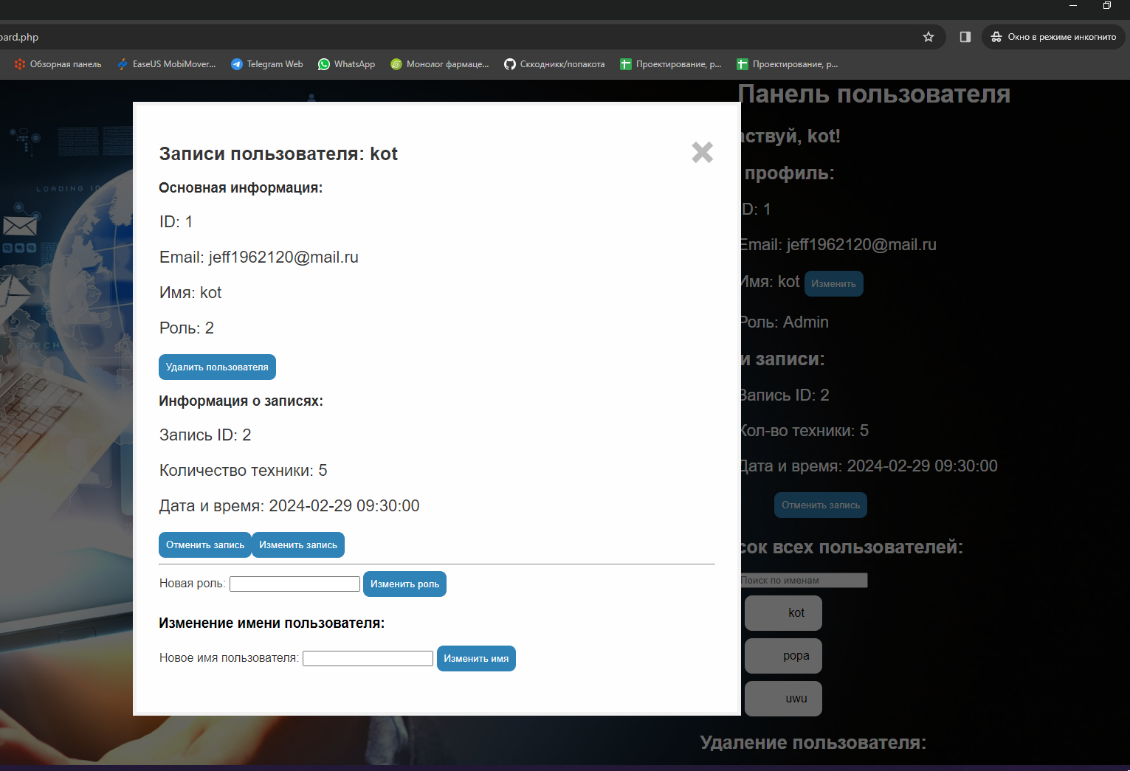
****

Рисунок 13 — отображение информации о пользователе в отдельном окне

У нас имеется HTML код модального окна, который представляет собой блок div с классом userInfoModal. Внутри этого блока находятся также элементы userInfoTitle - заголовок модального окна и userInfoDetails - блок с информацией о пользователе. При инициализации эти элементы пусты. Листинг кода:

<div ID="userInfoModal" class="modal">

<div class="modal-content">

<span class="close" onclick="closeUserInfoModal()">&times;</span>

<h2 ID="userInfoTitle"></h2>

<div ID="userInfoDetails"></div>

<form ID="updateRecordingForm" 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 техники:</label>

<input type="number" name="newElectronics" ID="newElectronics" required>

<button type="button" ID="updateRecordingButton" onclick="updateRecording()">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>

</div>

Выбираем кого-нибудь, запускается JS функция showUserInfo:

function showUserInfo(userId) {

currentUserId = userId;

// Отправка AJAX-запроса для получения информации о пользователе

var xhr = new XMLHttpRequest();

xhr.open('POST', 'dashboard.php', true);

xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.onreadystatechange = function () {

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

var userInfo = JSON.parse(xhr.responseText);

displayUserInfoModal(userInfo);

}

};

xhr.send('action=getUserInfoAndRecordings&userId=' + userId);

}

Здесь мы инициализируем переменную для хранения идентификатора выбранного пользователя, чтобы в последствии взаимодействовать с его данными. После выполнения этого действия будет вызвана функция displayUserInfoModal на языке JavaScript:

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 = 'Записи пользователя: ' + 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 + '</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.recordings.length > 0) {

details.innerHTML += '<h3>Информация о записях:</h3>';

userInfo.recordings.forEach(function (recording) {

details.innerHTML += '<p>Запись ID: ' + recording.records\_ID + '</p>';

details.innerHTML += '<p>Количество техники: ' + recording.electronics + '</p>';

details.innerHTML += '<p>Дата и время: ' + recording.records\_date + ' ' + recording.records\_time + '</p>';

details.innerHTML += '<button onclick="cancelRecording(' + recording.records\_ID + ')">Отменить запись</button>';

details.innerHTML += '<button onclick="showUpdateRecordingForm(' + recording.records\_ID + ')">Изменить запись</button>';

details.innerHTML += '<hr>';

});

} else {

details.innerHTML += '<p>Пользователь не делал записей.</p>';

}

// Отобразить модальное окно

modal.style.display = 'block';

}

В ходе практической части было разработано web-приложение, предназначенное для записи к мастеру по ремонту электроники с использованием современных web-технологий. Приложение включает в себя функции записи, отображения доступных дат и времени, а также предоставляет информацию о пользоввателе.

# ЗАКЛЮЧЕНИЕ

В заключение, создание web-приложения с применением современных web-технологий по тематике "Электроника", направленное на запись для починки сломанной электроники к мастеру, является важным и востребованным шагом в развитии сферы ремонта электроники.

В современном мире потребление электронных устройств стремительно растет, и, соответственно, возникает все больше неисправностей, требующих ремонта. Важность создания такого web-приложения заключается в удобстве и доступности для пользователей, которые ищут надежного специалиста для починки своих сломанных устройств.

Применение современных web-технологий позволяет создать удобный, интуитивно понятный и функциональный интерфейс для пользователей. Он предоставляет возможность быстро ознакомиться с услугами. Кроме того, такое приложение позволяет осуществить запись на услугу онлайн, что существенно экономит время и удобно для клиентов.

Создание web-приложения по тематике "Электроника" является актуальным решением, которое помогает сократить временные и финансовые затраты. Запись к мастеру через интернет экономит время пользователя и улучшает его опыт общения с сервисом ремонта электроники.

В целом, создание web-приложения с применением современных web-технологий по тематике "Электроника" является неотъемлемой частью развития сферы ремонта сломанной электроники. Такое приложение упрощает процесс записи к мастеру и делает его более удобным и доступным для пользователей. Оно способствует повышению качества обслуживания клиентов и значительно экономит их время и усилия.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алексеев Ю.М. Быстро и легко создаем, программируем, шлифуем и раскручиваем web-сайт: учеб. пособие / Ю.М. Алексеев. - М.: Лучшие книги, 2011. - 189 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. Гольцман В. Г63 MySQL 5.0. Библиотека программиста. — СПб.: Питер, 2010. — 256 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-49807-135-0
5. Гончаров А.Ю. Web-дизайн: HTML, JavaScript и CSS. Карманный справочник / А.Ю. Гончаров. - М.: «КУДИЦ-ПРЕСС», 2010. - 320 с.
6. Дуванов А. Web-конструирование. HTML / А. Дуванов. - СПб.: БХВ-Петербург, 2013. - 321 с.
7. Дунаев В. JavaScript: учеб. пособие / В. Дунаев. - СПб.: Питер, 2010. - 394 с.
8. Кириченко А.В., Никольский А.П., Дубовик Е.В. Web на практике. CSS, HTML, JavaScript, MySQL, РНР для fullstасk-разработчиков - СПв.: НАУКА и ТЕхникА, 2021. - 432 с., ил. ISBN 978-5-94387-271-6
9. Кожемякин А. HTML и CSS в примерах. Создание Web-страниц: учебник / А. Кожемякин. - М.: Альтекс-А, 2011. - 415 с.
10. Кузнецов М. В., Симдянов И. В. К89 Самоучитель MySQL 5. — СПб.: БХВ-Петербург, 2006. — 560 с.: ил. ISBN 978-5-94157-754-5
11. 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.