Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное учреждение высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт управления бизнес-процессами

Кафедра бизнес-информатики и моделирования бизнес-процессов

**КУРСОВАЯ РАБОТА**

по дисциплине «Проектирование, разработка и оптимизация web-приложений»

«Создание web-приложения с применением современных web-технологий по тематике «Медицинский центр»

Преподаватель

подпись, дата

Никитин А.И.

Студент УБ22-08Б, 432217040

подпись, дата

Красноярск 2024

Муравьёв Н.М.

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc156153021)

[1 Теоретическая часть 4](#_Toc156153022)

[1.1 HTML И CSS 4](#_Toc156153023)

[1.2 JavaScript 5](#_Toc156153024)

[1.3 PHP 6](#_Toc156153025)

[1.4 MySQL 8](#_Toc156153026)

[2 Практическая часть 9](#_Toc156153027)

[2.1 Требования к работе 9](#_Toc156153028)

[2.2 Создание базы данных 10](#_Toc156153029)

[2.3 Регистрация и авторизация 12](#_Toc156153030)

[2.4 Главная страница 16](#_Toc156153031)

[2.5 Личный кабинет 23](#_Toc156153032)

[Заключение 31](#_Toc156153033)

[Список использованных источников 32](#_Toc156153034)

# Введение

В современном мире веб-программирование становится ключевой областью, особенно в контексте создания веб-приложений, направленных на оптимизацию бизнес-процессов. Данная курсовая работа фокусируется на перспективном направлении – разработке веб-сайта "Музыкальный портал" с использованием MySQL для хранения данных. В отрасли развлечений, так же как и в общественном питании, существует потребность в инновационных решениях для автоматизации внутренних процессов. Web-разработка открывает возможности для создания продукта с интересным функционалом, полезным для посетителей и администраторов. Создание базы данных MySQL для веб-приложения по размещению музыкальных композиций является ключевым этапом, обеспечивающим эффективное функционирование приложения.

Разработка базы данных MySQL для веб-приложения, посвященного музыке, считается неотъемлемой частью обеспечения его эффективности и гибкости. Этот компонент обеспечивает эффективное управление данными, гарантирует безопасность информации и открывает возможности для взаимодействия с другими системами, обеспечивая более гибкое управление процессами.

Цель данной курсовой работы заключается в создании сайта с использованием HTML, CSS, JavaScript, PHP, MySQL, а также в разработке базы данных для управления пользователями с помощью MySQL. Объектом исследования выступает веб-приложение, использующее различные web-технологии, а предметом исследования является процесс создания веб-приложения и соответствующей базы данных. В работе используются методы анализа, синтеза информации, проектирования и практической реализации.

Объект исследования: Веб-приложение.

Предмет исследования: Процесс создания веб-приложения и связанной с ним базы данных.

# 1 Теоретическая часть

# 1.1 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.

HTML является языком разметки, предназначенным для структурирования информации на веб-странице. С его помощью создаются различные элементы, такие как заголовки, параграфы, списки, изображения и гиперссылки. HTML предоставляет универсальные теги, каждый из которых определяет определенный тип контента. Например, тег <h1> используется для создания заголовков первого уровня, а тег <p> - для параграфов.

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.

CSS, с другой стороны, используется для стилизации веб-страницы. Он позволяет определять внешний вид и оформление элементов HTML, включая цвета, шрифты, расположение, отступы, рамки и другие аспекты дизайна. CSS также способствует созданию адаптивного и отзывчивого дизайна, который позволяет сайту хорошо выглядеть на различных устройствах и экранах.

Кроме того, CSS позволяет создавать анимации, переходы, градиенты и другие эффекты, делая веб-страницы более интерактивными и привлекательными для пользователей.

Сочетание HTML и CSS позволяет веб-разработчикам создавать динамичные и креативные веб-страницы. Применение семантической разметки с использованием HTML способствует лучшему восприятию контента поисковыми системами, а гибкость CSS дает возможность адаптировать внешний вид страницы под различные устройства и разрешения экранов. Эти инструменты являются основополагающими для веб-разработчика, открывая двери к созданию привлекательных, удобных в использовании и технологически современных веб-приложений.

В современной веб-разработке HTML и CSS также играют ключевую роль в обеспечении доступности и адаптивности веб-приложений. Стандарты разметки HTML5 вводят новые элементы и атрибуты, улучшая структурирование контента и обеспечивая большую гибкость для веб-разработчиков. Новые возможности, такие как мультимедийные элементы <audio> и <video>, а также API для работы с локальным хранилищем, значительно расширяют функционал веб-приложений.

CSS3, в свою очередь, вводит множество новых свойств и селекторов, что позволяет создавать более сложные и интерактивные дизайны. Технологии, такие как Flexbox и Grid Layout, предоставляют разработчикам мощные инструменты для управления расположением элементов на веб-странице.

В целом, HTML и CSS работают вместе для создания привлекательного, удобного в использовании и эстетически приятного веб-сайта, который обеспечивает хорошее визуальное впечатление и удовлетворяет потребности пользователей.

# 1.2 JavaScript

JavaScript — это высокоуровневый, интерпретируемый язык программирования, который широко используется для создания интерактивных веб-страниц. Он был создан в 1995 году компанией Netscape и изначально назывался LiveScript, но затем был переименован в JavaScript.

JavaScript является частью технологий, которые обеспечивают динамичное взаимодействие на веб-страницах. Он используется для создания интерактивных элементов, анимаций, обработки данных, валидации форм, обновления содержимого страницы без перезагрузки и многих других возможностей.

JavaScript был создан для добавления возможности создания динамичных элементов на стороне клиента, что позволяет веб-страницам реагировать на действия пользователя без необходимости обновления всей страницы. Этот язык поддерживает событийно-ориентированное программирование, что означает, что код может реагировать на различные события, такие как нажатия клавиш, клики мыши и изменения веб-элементов.

Основные особенности JavaScript:

1. Язык сценариев. JavaScript предназначен для выполнения на стороне клиента, в браузере пользователя. Это позволяет создавать динамичные интерфейсы и взаимодействовать с пользователем без необходимости отправлять запросы на сервер.

2. Объектно-ориентированный. JavaScript поддерживает объектно-ориентированную парадигму программирования, что позволяет создавать модульные и масштабируемые приложения.

3. Мультипарадигмальный. JavaScript поддерживает не только объектно-ориентированный подход, но также функциональное программирование и императивный стиль. Это позволяет разработчикам выбирать наиболее подходящий подход в зависимости от конкретной задачи.

4. Динамически типизированный. Переменные в JavaScript являются динамически типизированными, что означает, что тип данных переменной определяется во время выполнения программы.

5. Широкая поддержка. JavaScript поддерживается всеми современными браузерами и используется не только на веб-страницах, но также в различных фреймворках и библиотеках для разработки клиентской и серверной части приложений.

Благодаря своим возможностям, JavaScript стал одним из самых популярных языков программирования и широко используется в веб-разработке для создания динамичных и интерактивных пользовательских интерфейсов.

# 1.3 PHP

PHP (Hypertext Preprocessor) представляет собой язык программирования, специально разработанный для веб-разработки, и широко используется для создания динамических веб-приложений, работы с базами данных и обработки данных на стороне сервера.

Основные возможности PHP:

- Интеграция с HTML: PHP позволяет встраивать свой код непосредственно в HTML, что позволяет создавать динамические веб-страницы.

- Работа с базами данных: PHP обеспечивает удобное взаимодействие с различными системами управления базами данных.

- Обработка форм и данных: PHP является эффективным инструментом для обработки данных, отправляемых с веб-форм.

- Создание сессий и куки: PHP предоставляет механизмы для хранения информации о состоянии пользователя между запросами и для хранения информации на стороне клиента.

- Работа с файлами: PHP предоставляет средства для работы с файловой системой сервера, включая загрузку, скачивание и обработку файлов.

Актуальность использования PHP в веб-приложениях:

- Широкое распространение: PHP широко поддерживается хостинг-провайдерами и работает на большинстве серверов.

- Сообщество и ресурсы: PHP обладает активным сообществом разработчиков, множеством библиотек и фреймворков.

- Простота и гибкость: Язык предоставляет простой синтаксис, доступный для новичков, но при этом гибок для опытных разработчиков.

- Обширный инструментарий: PHP легко интегрируется с другими технологиями и инструментами, что делает его универсальным решением для создания различных веб-приложений.

PHP остается одним из наиболее популярных языков для серверной веб-разработки благодаря своей простоте использования и мощным возможностям. Однако разработчики должны быть в курсе его ограничений и использовать современные и безопасные методы.

# 1.4 MySQL

MySQL — это открытая реляционная система управления базами данных (СУБД), разработанная и поддерживаемая компанией Oracle. Она предлагает множество возможностей для хранения, обработки и извлечения данных. Ниже приведены некоторые ключевые особенности и характеристики MySQL:

1. Множество типов данных: MySQL поддерживает различные типы данных, включая числовые, строковые, даты/время, JSON и другие, что позволяет эффективно хранить разнообразную информацию.

2. Индексация данных: MySQL обеспечивает возможность создания индексов для ускорения поиска и сортировки данных. Это повышает производительность при выполнении запросов к базе данных.

3. Транзакционная обработка: MySQL поддерживает транзакции, что позволяет обеспечить целостность данных при выполнении операций в рамках базы данных.

4. Расширяемость: MySQL имеет модульную архитектуру, что облегчает расширение его функциональности с помощью различных типов расширений, хранимых процедур, триггеров и других объектов базы данных.

5. Высокая производительность: MySQL обладает оптимизированным ядром и механизмами кэширования, что позволяет обеспечить высокую производительность при обработке запросов к базе данных.

6. Кластеризация и репликация: MySQL поддерживает кластеризацию и репликацию, что позволяет создавать высокодоступные и масштабируемые конфигурации баз данных.

7. Совместимость и поддержка: MySQL поддерживает стандартные языки запросов (SQL), а также имеет обширную документацию, активное сообщество пользователей и широкий набор инструментов для администрирования и разработки.

MySQL широко применяется в веб-приложениях, системах управления контентом, системах электронной коммерции, аналитических системах и других областях, где требуется эффективная обработка больших объемов данных.

# 2 Практическая часть

# 2.1 Требования к работе

Необходимо разработать web-приложения с применением современных web-технологий по тематике «Музыкальный портал», используя технологии HTML, CSS, JavaScript, PHP и MySQL. Кроме того для своего проекта необходимо разработать базу данных для управления пользователями. 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:

– должна быть предусмотрена авторизация и регистрация пользователей;

– зарегистрированный пользователь имеет право редактировать свои данные и просматривать свои услуги;

– администраторы имеют право редактировать пользователей и добавлять услуги;

– неавторизированный пользователи имеют право просматривать список услуг;

– сайт должен быть выдержан в определенном стиле и корректно открываться на любом разрешении;

– общий функционал может быть продуман самостоятельно;

– важно продемонстрировать не большой объем функционала, а минимальные требования к проекту – он может быть максимально простым, но при этом содержать все необходимые функции;

– отдельные бонусы предусмотрены за доп. функционал и красивый дизайн.

В качестве инструментов в работе будет использован текстовый редактор VScode, приложение для создания локального сервера OPEN server, в котором также можно найти веб-приложение phpMyAdmin для администрирования СУБД MySQL.

# 2.2 Создание базы данных

Перед началом работы необходимо запустить приложение OPEN server и подождать, С помощью встроенного функционала (Рисунок 1) создалась база данных с кодировкой utf8\_general\_ci.

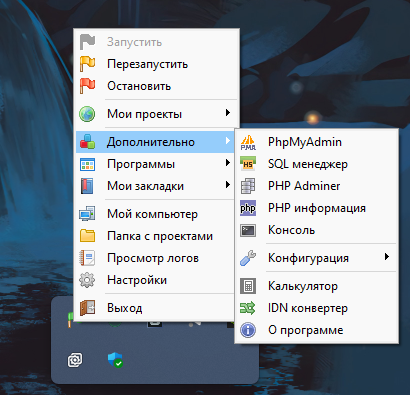


Рисунок 1 – запуск серверов через OPEN server

Далее следует открыть веб-приложение phpMyAdmin (рис.2), в котором можно создать базу данных через вспомогательные кнопки или напрямую через консоль.

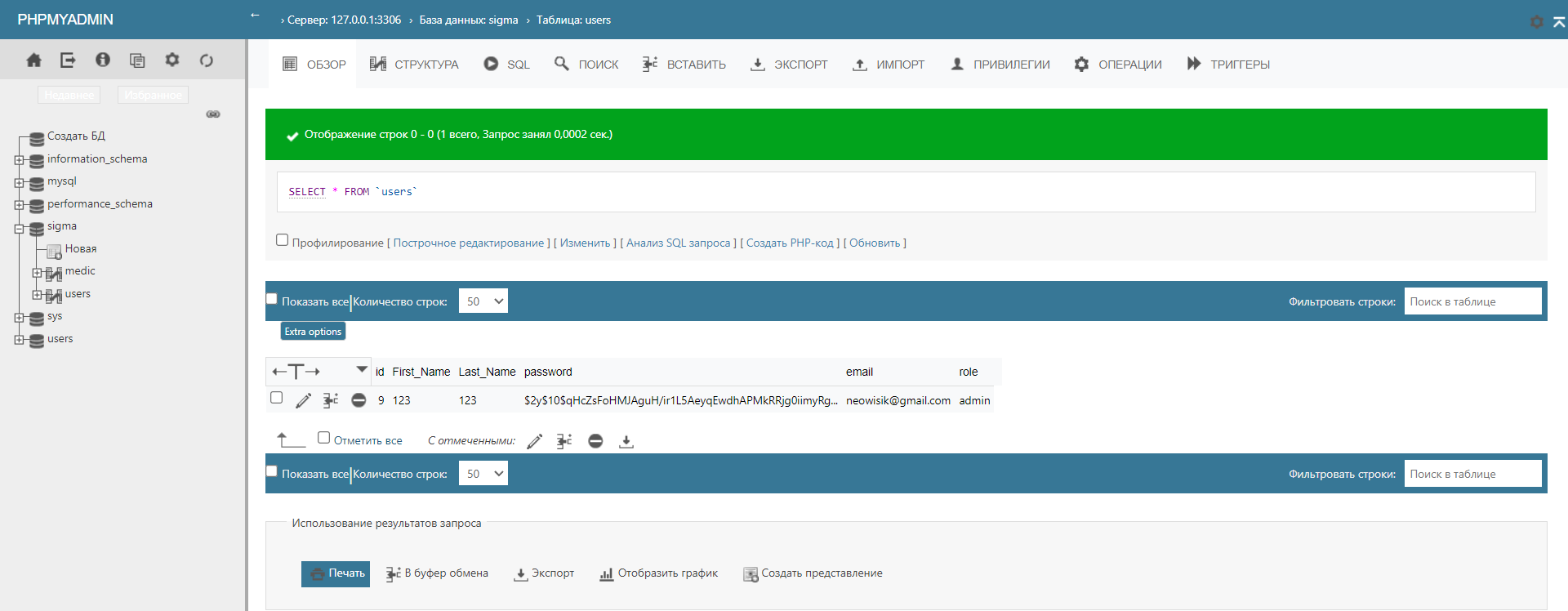


Рисунок 2 – веб-приложение phpMyAdmin

После создания базы данных следует проверить корректность ее подключения. Для этого можно использовать серверный язык php. В текстовом редакторе достаточно ввести следующий код:

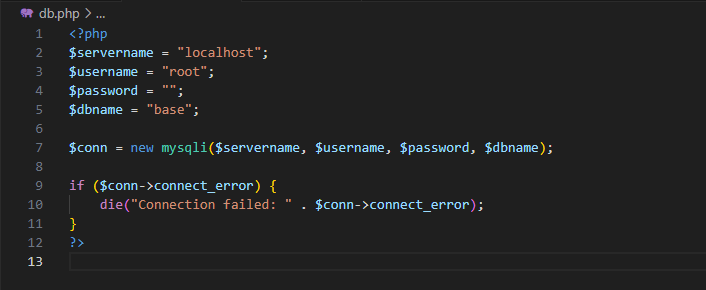


Рисунок 3 – код подключения

Теперь, при ошибке, связанной с подключением базы данных, выведется сообщение “Connection failed”.

# 2.3 Главная страница

Попасть на главную страницу могут как авторизованные, так и неавторизованные пользователи. Сам документ называется index.php и содержит весь основной контент об веб-приложении Medical Services, используемом в качестве примера для курсовой работы.

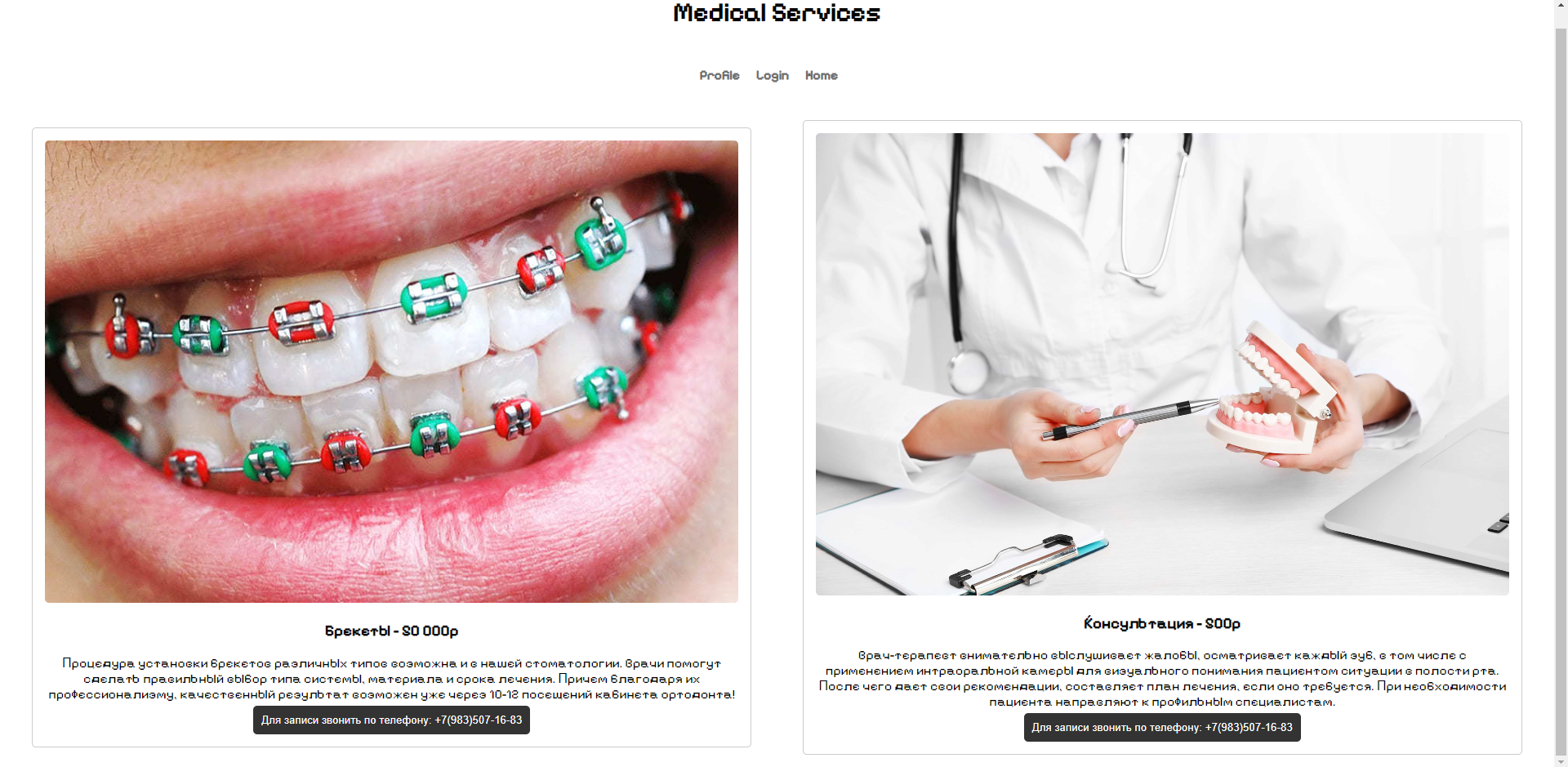


Рисунок 4 – главная страница для обычного пользователя

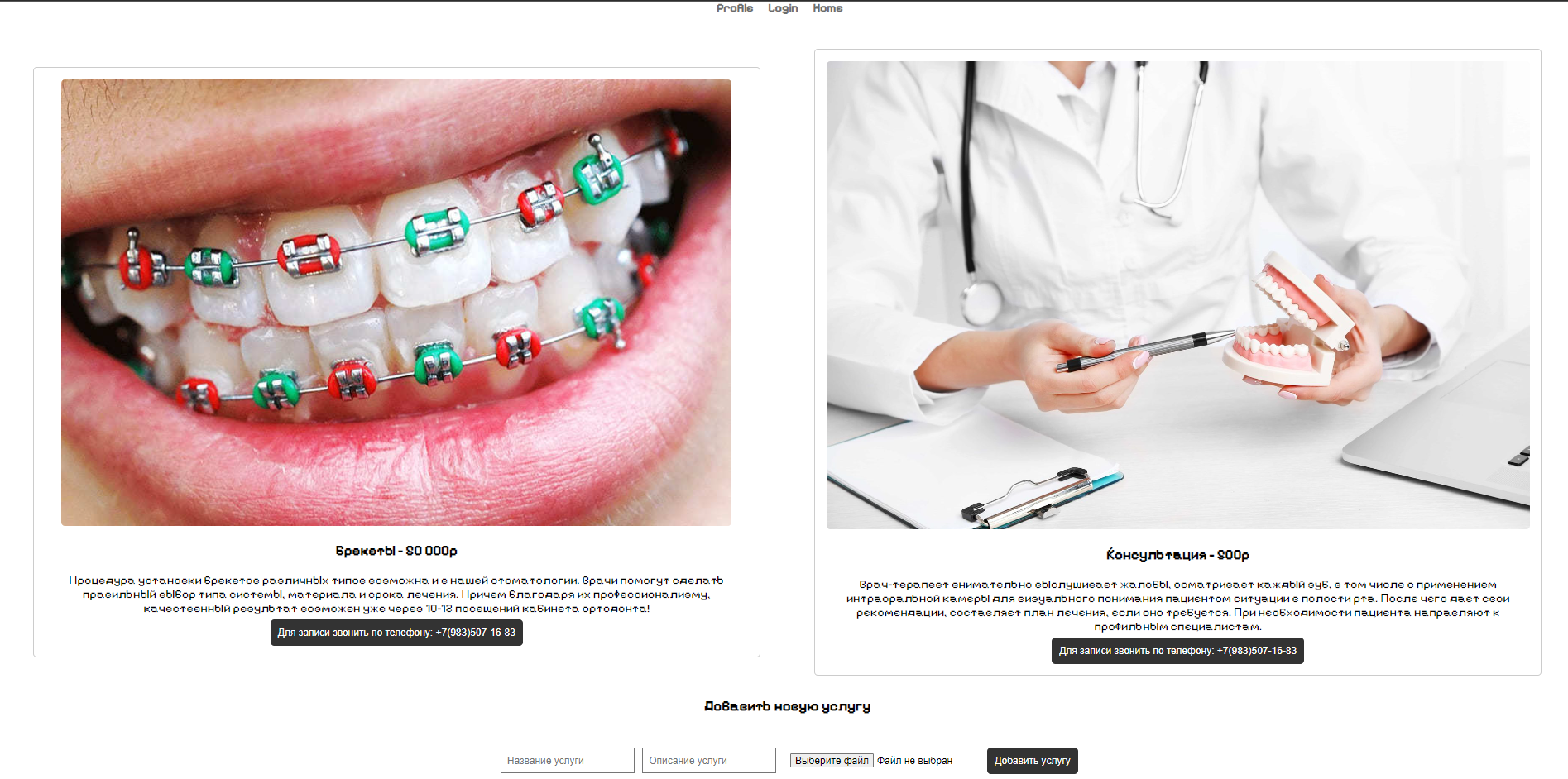


Рисунок 5- главная страница для администратора

Листинг index.php:

<?php

session\_start();

require\_once('db.php');

// Получение услуг из базы данных

$sql = "SELECT \* FROM medic";

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

$services\_from\_db = [];

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

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

        $services\_from\_db[] = $row;

    }

}

// Добавление нескольких услуг, не связанных с базой данных

$services\_not\_from\_db = [

    [

        'id' => 0,

        'name' => 'Брекеты - 50 000р',

        'opisanie' => 'Процедура установки брекетов различных типов возможна и в нашей стоматологии. Врачи помогут сделать правильный выбор типа системы, материала и срока лечения. Причем благодаря их профессионализму, качественный результат возможен уже через 10-12 посещений кабинета ортодонта!',

        'photo\_path' => 'pic1.jpg',

    ],

    [

        'id' => 0,

        'name' => 'Консультация - 500р',

        'opisanie' => 'Врач-терапевт внимательно выслушивает жалобы, осматривает каждый зуб, в том числе с применением интраоральной камеры для визуального понимания пациентом ситуации в полости рта. После чего дает свои рекомендации, составляет план лечения, если оно требуется. При необходимости пациента направляют к профильным специалистам.',

        'photo\_path' => 'pic2.jpg',

    ],

];

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

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

    $sql\_check\_admin = "SELECT role FROM users WHERE id='$user\_id'";

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

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

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

        if ($user\_data['role'] == 'admin' && isset($\_POST['add\_service'])) {

            $new\_service\_name = $\_POST['new\_service\_name'];

            $new\_service\_description = $\_POST['new\_service\_description'];

            $new\_service\_photo\_path = $\_POST['service\_image'];

            $sql\_add\_service = "INSERT INTO medic (name, opisanie, photo\_path) VALUES ('$new\_service\_name', '$new\_service\_description', '$new\_service\_photo\_path')";

            if ($conn->query($sql\_add\_service) !== TRUE) {

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

            }

        }

    }

}

$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="css/style.css">

    <title>Your Medical Services</title>

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Pixelify+Sans&display=swap" rel="stylesheet">

</head>

<body>

    <header>

        <h1 align="center">Medical Services</h1>

        <div class="block">

            <nav>

                <ul>

                    <li><a href="profile.php">Profile</a></li>

                    <li><a href="login.php">Login</a></li>

                    <li><a href="index.php">Home</a></li>

                </ul>

            </nav>

        </div>

    </header>

    <div class="service-container">

        <?php

        foreach ($services\_from\_db as $service) {

            echo '<div class="service-card">';

            echo '<img src="' . $service['photo\_path'] . '" alt="' . $service['name'] . '">';

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

            echo '<p>' . $service['opisanie'] . '</p>';

            echo '<button>Записаться</button>';

            echo '</div>';

        }

        foreach ($services\_not\_from\_db as $service) {

            echo '<div class="service-card">';

            echo '<img src="' . $service['photo\_path'] . '" alt="' . $service['name'] . '">';

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

            echo '<p>' . $service['opisanie'] . '</p>';

            echo '<button>Для записи звонить по телефону: +7(983)507-16-83</button>';

            echo '</div>';

        }

        ?>

    </div>

    <?php

    if (isset($\_SESSION['user\_id']) && $user\_data['role'] == 'admin') {

        echo '<div class="add-service-form">';

        echo '<h3>Добавить новую услугу</h3>';

        echo '<form method="post" action="">';

        echo '<input type="text" name="new\_service\_name" placeholder="Название услуги" required>';

        echo '<input type="text" name="new\_service\_description" placeholder="Описание услуги" required>';

        echo '<input type="file" name="service\_image" accept="image/\*" required>';

        echo '<button type="submit" name="add\_service">Добавить услугу</button>';

        echo '</form>';

        echo '</div>';

    }

    ?>

</body>

</html>

Также для связи функций используюется следующий PHP файлы:

Листинг add\_service.php:

<?php

session\_start();

require\_once('db.php');

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

    $new\_service\_name = $\_POST['new\_service\_name'];

    $new\_service\_description = $\_POST['new\_service\_description'];

    $new\_service\_photo = $\_POST['new\_service\_photo'];

    $sql = "INSERT INTO medic (name, opisanie, photo\_path) VALUES ('$new\_service\_name', '$new\_service\_description', '$new\_service\_photo')";

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

        header("Location: index.php");

        exit();

    } else {

        echo "Ошибка при добавлении новой услуги: " . $conn->error;

    }

}

$conn->close();

?>

# 2.4 Регистрация и авторизация

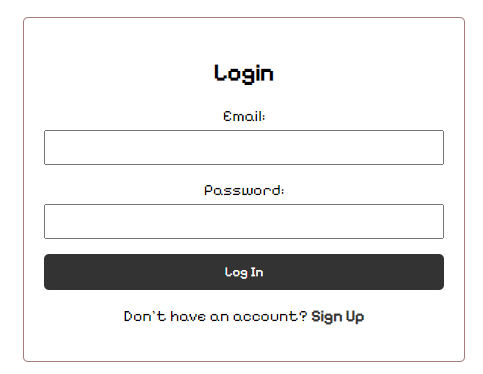
Документ называется login.php. Он представляет собой белый фон с окном по середине при помощи которого пользователь сможет авторизироваться.  


Рисунок 6 – окно авторизации

Листинг login.php:

<?php

session\_start();

require\_once('db.php');

if ($conn->connect\_error) {

    die("Ошибка подключения к базе данных: " . $conn->connect\_error);

}

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

    $email = $\_POST['email'];

    $password = $\_POST['password'];

    $sql = "SELECT \* FROM users WHERE email='$email'";

    $result = $conn->query($sql);

    if ($result->num\_rows > 0) {

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

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

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

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

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

            header("Location: profile.php");

            exit();

        } else {

            echo "Неверный пароль";

        }

    } else {

        echo "Пользователь с указанным email не найден";

    }

}

$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="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Pixelify+Sans&display=swap" rel="stylesheet">

    <title>Login</title>

</head>

<style>

    body {

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100vh;

    margin: 0;

    font-family: 'Pixelify Sans', sans-serif;

}

.login-container {

    width: 400px;

    margin: 50px auto;

    padding: 20px;

    border: 1px solid #a67c7c;

    border-radius: 5px;

    text-align: center;

}

form {

    display: flex;

    flex-direction: column;

}

label {

    margin-bottom: 5px;

}

input {

    margin-bottom: 15px;

    padding: 8px;

}

button {

    padding: 10px;

    background-color: #333;

    color: white;

    border: none;

    cursor: pointer;

    font-family: 'Pixelify Sans', sans-serif;

    border-radius:5px;

}

a {

    text-decoration: none;

    color: #333;

    font-weight: bold;

}

</style>

<body>

    <div class="login-container">

        <h2>Login</h2>

        <form method="post" action="">

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

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

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

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

            <button type="submit">Log In</button>

        </form>

        <p>Don't have an account? <a href="registration.php">Sign Up</a></p>

    </div>

</body>

</html>

Также со вкладки авторизация можно перейти на вкладку регистрация. Для этого используется файл registration.php.

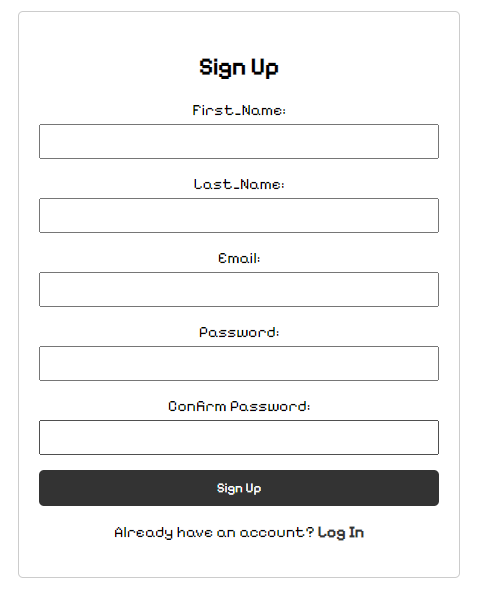


Рисунок 7 -окно регистрации

Листинг страницы registration.php:

<?php

require\_once('db.php');

if ($conn->connect\_error) {

    die("Ошибка подключения к базе данных: " . $conn->connect\_error);

}

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

$First\_name = $\_POST['First\_Name'];

$Last\_name = $\_POST['Last\_Name'];

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

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

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

$avatar = $\_POST["avatar"];

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

    die("Пароли не совпадают");

}

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

$sql = "INSERT INTO users (First\_Name, Last\_Name, email, password, role) VALUES ('$First\_name', '$Last\_name', '$email', '$hashed\_password', 'user')";

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

    header("Location: login.php");

} else {

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

}

}

$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="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Pixelify+Sans&display=swap" rel="stylesheet">

    <title>Registration</title>

    <style>

.registration-container {

    width: 400px;

    margin: 50px auto;

    padding: 20px;

    border: 1px solid #ccc;

    border-radius: 5px;

    text-align: center;

}

form {

    display: flex;

    flex-direction: column;

}

label {

    margin-bottom: 5px;

}

input {

    margin-bottom: 15px;

    padding: 8px;

}

button {

    padding: 10px;

    background-color: #333;

    color: white;

    border: none;

    cursor: pointer;

    border-radius: 5px;

    font-family: 'Pixelify Sans', sans-serif;

}

a {

    text-decoration: none;

    color: #333;

    font-weight: bold;

}

body {

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100vh;

    margin: 0;

    font-family: 'Pixelify Sans', sans-serif;

}

.avatar {

    margin-bottom: 5px;

    background-color: #333;

    border-radius: 5px;

    color: white;

    height: 20px;

}

.knopka{

    display: none;

}

h2{

    font-family: 'Pixelify Sans', sans-serif;

}

    </style>

</head>

<body>

    <div class="registration-container">

        <h2>Sign Up</h2>

        <form method="post" action="">

            <label for="First\_Name">First\_Name:</label>

            <input type="text" id="First\_Name" name="First\_Name" required>

            <label for="Last\_Name">Last\_Name:</label>

            <input type="text" id="Last\_Name" name="Last\_Name" required>

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

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

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

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

            <label for="confirm\_password">Confirm Password:</label>

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

            <button type="submit">Sign Up</button>

        </form>

        <p>Already have an account? <a href="login.php">Log In</a></p>

    </div>

</body>

</html>

# 

# 2.5 Личный кабинет

При переходе в личный кабинет (файл profile.php) перед пользователем представлена возможность изменить необходимые данные. Также как и для регистрации, здесь используется функция, помогающая определить корректность ввода пароля.

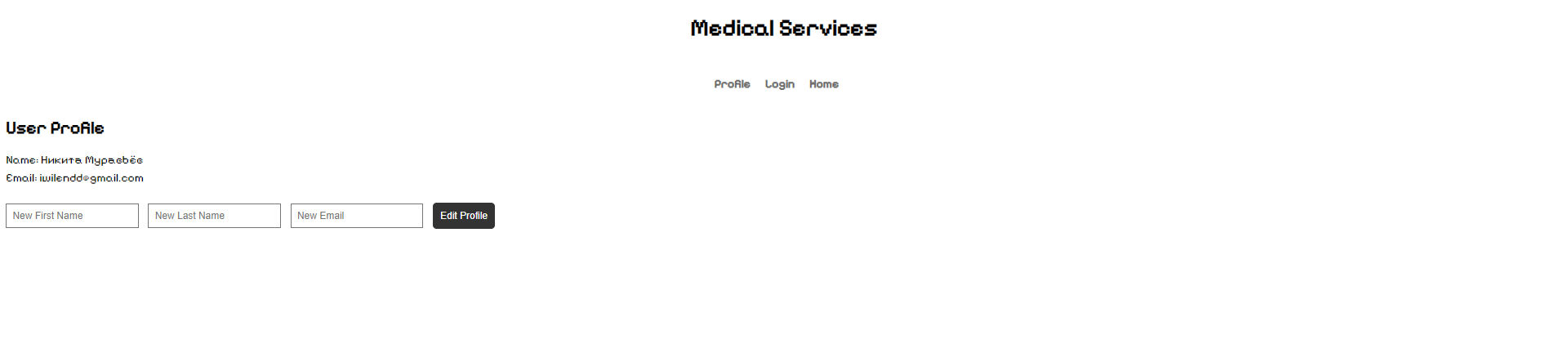


Рисунок 8 – личный кабинет для обычного пользователя

Для администраторов страница выглядит иначе.

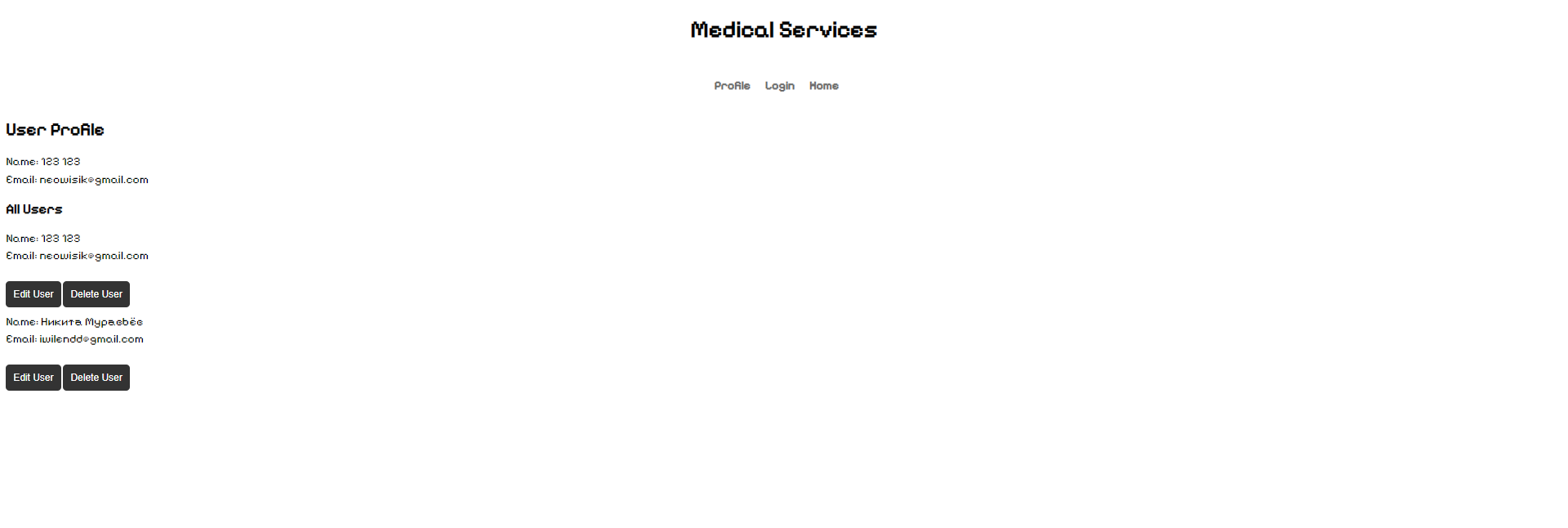


Рисунок 9 – профиль администратора

Для объяснения принципа работы всей страницы стоит ознакомиться с кодом:

Листинг profile.php:

<?php

session\_start();

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

    header("Location: login.php");

    exit();

}

require\_once('db.php');

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

$sql = "SELECT First\_Name, Last\_Name, email, role FROM users WHERE id='$user\_id'";

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

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

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

} else {

    echo "Данные пользователя не найдены";

}

$all\_users = [];

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

    $sql\_all\_users = "SELECT id, First\_Name, Last\_Name, email FROM users";

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

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

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

            $all\_users[] = $row;

        }

    }

}

$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="css/style.css">

    <title>Your Music Website</title>

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Pixelify+Sans&display=swap" rel="stylesheet">

</head>

<body>

    <header>

        <h1 align="center">Medical Services</h1>

        <div class="block">

            <nav>

                <ul>

                    <li><a href="profile.php">Profile</a></li>

                    <li><a href="login.php">Login</a></li>

                    <li><a href="index.php">Home</a></li>

                </ul>

            </nav>

        </div>

    </header>

    <div class="profile-container">

        <h2>User Profile</h2>

        <div>

            <p >Name: <?php echo $user\_data['First\_Name'] . ' ' . $user\_data['Last\_Name']; ?></p>

            <p >Email: <?php echo $user\_data['email']; ?></p>

            <?php if ($user\_data['role'] == 'admin'): ?>

                <div class="user-list" >

                    <h3>All Users</h3>

                    <?php foreach ($all\_users as $user): ?>

                        <div class="user-item">

                            <p>Name: <?php echo $user['First\_Name'] . ' ' . $user['Last\_Name']; ?></p>

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

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

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

                                <button type="submit">Edit User</button>

                            </form>

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

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

                                <button type="submit">Delete User</button>

                            </form>

                        </div>

                    <?php endforeach; ?>

                </div>

            <?php else: ?>

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

                    <input type="text" name="new\_first\_name" placeholder="New First Name">

                    <input type="text" name="new\_last\_name" placeholder="New Last Name">

                    <input type="text" name="new\_email" placeholder="New Email">

                    <button type="submit">Edit Profile</button>

                </form>

            <?php endif; ?>

        </div>

    </div>

</body>

</html>

Также используются следующие PHP файлы:

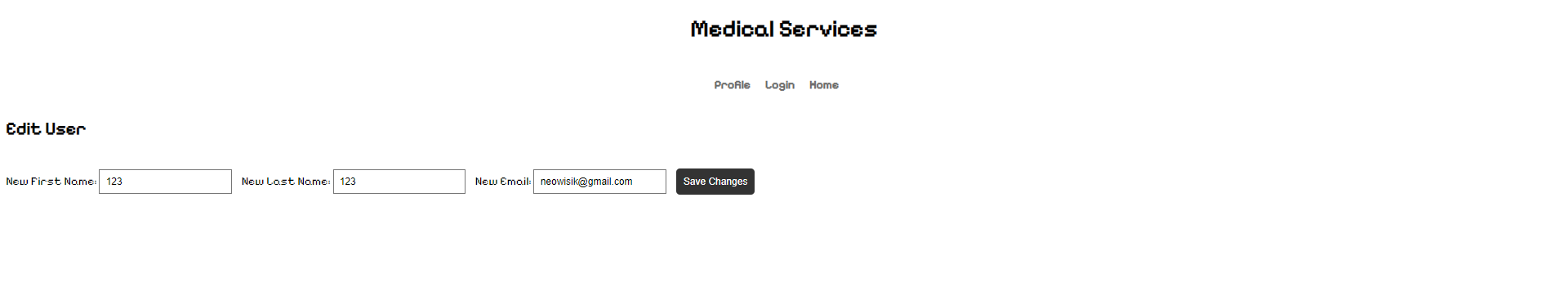


Рисунок 10 - окно редактирования пользователя

Листинг edit\_user.php:

<?php

session\_start();

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

    header("Location: login.php");

    exit();

}

require\_once('db.php');

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

    $user\_id\_to\_edit = $\_POST['user\_id'];

    $sql\_get\_user = "SELECT First\_Name, Last\_Name, email FROM users WHERE id='$user\_id\_to\_edit'";

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

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

        $user\_data\_to\_edit = $result\_get\_user->fetch\_assoc();

    } else {

        echo "Данные пользователя не найдены";

    }

}

$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="css/style.css">

    <title>Edit User</title>

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Pixelify+Sans&display=swap" rel="stylesheet">

</head>

<body>

    <header>

        <h1 align="center">Medical Services</h1>

        <div class="block">

            <nav>

                <ul>

                    <li><a href="profile.php">Profile</a></li>

                    <li><a href="login.php">Login</a></li>

                    <li><a href="index.html">Home</a></li>

                </ul>

            </nav>

        </div>

    </header>

    <div class="edit-user-container">

        <h2>Edit User</h2>

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

            <input type="hidden" name="user\_id" value="<?php echo $user\_id\_to\_edit; ?>">

            <label for="new\_first\_name">New First Name:</label>

            <input type="text" id="new\_first\_name" name="new\_first\_name" value="<?php echo $user\_data\_to\_edit['First\_Name']; ?>" required>

            <label for="new\_last\_name">New Last Name:</label>

            <input type="text" id="new\_last\_name" name="new\_last\_name" value="<?php echo $user\_data\_to\_edit['Last\_Name']; ?>" required>

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

            <input type="text" id="new\_email" name="new\_email" value="<?php echo $user\_data\_to\_edit['email']; ?>" required>

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

        </form>

    </div>

</body>

</html>

Листинг delete\_user.php:

<?php

session\_start();

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

    header("Location: login.php");

    exit();

}

require\_once('db.php');

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

    $user\_id\_to\_delete = $\_POST['user\_id'];

    $sql = "DELETE FROM users WHERE id='$user\_id\_to\_delete'";

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

        header("Location: profile.php");

        exit();

    } else {

        echo "Ошибка при удалении пользователя: " . $conn->error;

    }

}

$conn->close();

?>

Листинг 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();

?>

Далее рассмотрим созданную базу данных.

Созданная база данных имеет следующие таблицы:

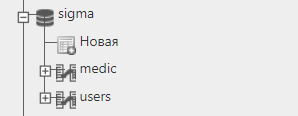


Рисунок 11 – база данных

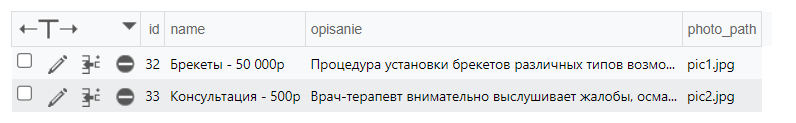


Рисунок 12 – Таблица выложенных услуг на сайт

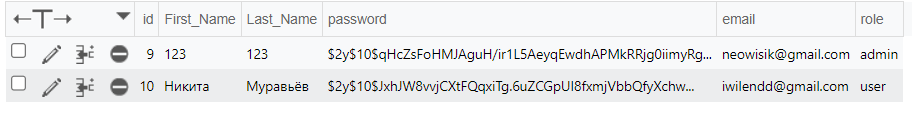


Рисунок 13 – Таблица пользователей

# Заключение

В ходе выполнения данного курсового проекта было разработано веб-приложение, использующее современные веб-технологии, такие как HTML, CSS, JavaScript, PHP и MySQL. Создана база данных для управления пользователями и внутренними услугами. Проект обладает минималистичным дизайном, но в то же время обеспечивает полный функционал.

Приложение включает в себя формы регистрации и авторизации, где применены простые алгоритмы безопасности, такие как хеширование паролей и проверка их минимальной длины. Также разработаны основные и вспомогательные страницы интернет-магазина.

Личный кабинет доступен только для авторизованных пользователей, предоставляя им возможность изменять необходимые данные, такие как логин и пароль, а также выходить из аккаунта или удалять его полностью. При наличии роли администратора пользователю предоставляется специальная панель управления, позволяющая изменять данные пользователей, удалять и добавлять аккаунты, а также добавлять новые музыкальные композиции на главной странице интернет-магазина.

# Список использованных источников

1. Основы CSS: mdn web docs [Электронный источник]. – URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics [дата доступа 12.01.2024].
2. Основы HTML: mdn web docs [Электронный источник]. – URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics [дата доступа 12.01.2024].
3. Основы JavaScript: mdn web docs [Электронный источник]. – URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics [дата доступа 12.01.2024].
4. Подробнее о PHP [Электронный источник]. – URL:

https://ru.hexlet.io/courses/php-setup-environment/lessons/what-is-php/theory\_unit (дата обращения: 12.01.2024).

1. Что такое MySQL: Макхост [Электронный источник]. – URL: https://mchost.ru/articles/chto-takoe-mysql/ [дата доступа 12.01.2024].
2. Что такое - PHP? [Электронный источник]. – URL: https://archive-ipq-co.narod.ru/ (дата обращения: 12.01.2023).
3. Что такое HTML и CSS [Электронный источник]. – URL: https://shneider-host.ru/blog/chto-takoe-html-i-css.html (дата обращения: 12.01.2024).
4. Язык разметки HTML, таблицы стилей CSS и язык программирования JavaScript [Электронный источник]. – URL: https://web-creator.ru/articles/about\_client\_side (дата обращения: 12.01.2024).
5. PHP: википедия [Электронный источник]. – URL: https://ru.wikipedia.org/wiki/PHP [дата доступа 12.01.2024].