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 «Портфолио»

Преподаватель

подпись, дата

Никитин А.И.

Студент УБ23-09Б, 432320749

подпись, дата

Красноярск 2024

Кузьменко Н.В.

**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 7](#_Toc156153026)

[2 Практическая часть 9](#_Toc156153027)

[2.1 Требования к работе 9](#_Toc156153028)

[2.2 Создание базы данных 11](#_Toc156153029)

[2.3 Регистрация и авторизация 12](#_Toc156153030)

[2.4 Главная страница 15](#_Toc156153031)

[2.5 Личный кабинет 23](#_Toc156153032)

[Заключение 31](#_Toc156153033)

[Список использованных источников 32](#_Toc156153034)

# Введение

В современном мире веб-программирование играет ключевую роль в развитии цифровых технологий. Особое место занимает разработка веб-приложений, направленных на самопрезентацию и профессиональное развитие. В рамках данной курсовой работы рассматривается актуальное направление — создание веб-приложения «Портфолио на стене» с использованием современных веб-технологий и MySQL для хранения данных. В эпоху цифровизации потребность в эффективном представлении своих профессиональных достижений и работ становится все более важной, что делает разработку подобных платформ особенно актуальной.

Создание веб-приложения для публикации портфолио представляет собой комплексную задачу, включающую разработку удобного пользовательского интерфейса, системы управления контентом и надежной базы данных. Разработка с использованием MySQL обеспечивает эффективное хранение и управление данными пользователей, их работами и достижениями. Этот подход способствует созданию масштабируемого решения, которое может адаптироваться под растущие потребности пользователей.

Целью данной курсовой работы является создание современного веб-приложения для публикации портфолио с использованием HTML, CSS, JavaScript, PHP и MySQL. Особое внимание уделяется разработке интуитивно понятного интерфейса и надежной системы управления пользователями с помощью СУБД MySQL. В работе применяются методы анализа требований, проектирования архитектуры приложения и практической реализации с использованием современных веб-технологий.

Объект исследования: Веб-приложение для создания и публикации профессиональных портфолио.

Предмет исследования: Процесс разработки веб-приложения с использованием современных технологий и создания эффективной базы данных для управления контентом пользователей.

Задачи исследования:

Анализ существующих решений и определение ключевых требований к приложению

Проектирование архитектуры приложения и структуры базы данных

Разработка функционального пользовательского интерфейса

Реализация системы управления пользователями и их контентом

Внедрение механизмов безопасности и защиты данных

Тестирование и оптимизация работы приложения

Практическая значимость работы заключается в создании удобной платформы для публикации профессиональных достижений, которая может быть использована как частными лицами, так и организациями для презентации своих работ и проектов.

# 1 Теоретическая часть

# 1.1 HTML И CSS

# HTML (HyperText Markup Language) — это основной язык разметки, который используется для создания структуры веб-страниц. Он состоит из множества элементов, известных как теги, которые определяют различные компоненты страницы, такие как заголовки, параграфы, изображения и таблицы. HTML позволяет организовать содержимое так, чтобы браузеры могли его правильно отображать.

# HTML играет ключевую роль в структурировании информации на веб-странице. С его помощью создаются различные элементы, включая заголовки, параграфы, списки, изображения и гиперссылки. Каждый тег HTML имеет свое назначение, например, <h1> используется для заголовков первого уровня, а <p> — для параграфов.

# CSS (Cascading Style Sheets) — это язык, который используется для описания внешнего вида веб-страницы. Он позволяет задавать стили для элементов HTML, такие как шрифт, цвет, размер, отступы и позиционирование. CSS помогает разработчикам создавать стильные и привлекательные веб-страницы, упрощая работу с HTML-разметкой.

# CSS отвечает за стилизацию веб-страницы. Он позволяет определять внешний вид и оформление элементов HTML, включая цвета, шрифты, расположение, отступы и рамки. CSS также способствует созданию адаптивного и отзывчивого дизайна, который позволяет сайту хорошо выглядеть на различных устройствах и экранах.

# Кроме того, CSS предоставляет возможности для создания анимаций, переходов, градиентов и других эффектов, делая веб-страницы более интерактивными и привлекательными для пользователей.

# Совместное использование HTML и CSS позволяет веб-разработчикам создавать динамичные и креативные веб-страницы. Семантическая разметка HTML улучшает восприятие контента поисковыми системами, а гибкость CSS позволяет адаптировать внешний вид страницы под различные устройства и разрешения экранов. Эти инструменты являются основополагающими для веб-разработчиков, открывая возможности для создания привлекательных, удобных и современных веб-приложений.

# В современной веб-разработке HTML и CSS играют важную роль в обеспечении доступности и адаптивности веб-приложений. Стандарты HTML5 вводят новые элементы и атрибуты, улучшая структурирование контента и обеспечивая большую гибкость для 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 итоге, HTML и CSS работают вместе для создания привлекательного, удобного и эстетически приятного веб-сайта, который обеспечивает хорошее визуальное впечатление и удовлетворяет потребности пользователей.

# 1.2 JavaScript

JavaScript — это высокоуровневый язык программирования, который играет ключевую роль в создании интерактивных веб-страниц. Разработанный в 1995 году компанией Netscape, он изначально назывался LiveScript, но вскоре был переименован в JavaScript.

Этот язык является неотъемлемой частью технологий, обеспечивающих динамическое взаимодействие на веб-страницах. JavaScript позволяет создавать интерактивные элементы, анимации, обрабатывать данные, валидировать формы и обновлять содержимое страницы без необходимости её перезагрузки.

JavaScript был разработан для добавления динамики на стороне клиента, что позволяет веб-страницам реагировать на действия пользователя без обновления всей страницы. Он поддерживает событийно-ориентированное программирование, что позволяет коду реагировать на различные события, такие как нажатия клавиш, клики мыши и изменения элементов на странице.

Основные характеристики JavaScript:

Язык сценариев: JavaScript выполняется на стороне клиента, в браузере пользователя, что позволяет создавать динамичные интерфейсы и взаимодействовать с пользователем без обращения к серверу.

2. Объектно-ориентированный: Поддержка объектно-ориентированной парадигмы позволяет создавать модульные и масштабируемые приложения.

Мультипарадигмальный: JavaScript поддерживает различные стили программирования, включая объектно-ориентированный, функциональный и императивный, что дает разработчикам гибкость в выборе подхода.

4. Динамическая типизация: Переменные в JavaScript динамически типизированы, что означает, что их тип определяется во время выполнения программы.

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

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

# 1.3 PHP

# PHP (Hypertext Preprocessor) — это язык программирования, разработанный специально для веб-разработки. Он широко используется для создания динамических веб-приложений, работы с базами данных и обработки данных на стороне сервера. PHP позволяет встраивать код непосредственно в HTML, что упрощает создание динамических веб-страниц. Это делает его незаменимым инструментом для создания интерактивных приложений, так как он эффективно обрабатывает данные, отправляемые с веб-форм.

# PHP обеспечивает удобное взаимодействие с различными системами управления базами данных, такими как MySQL и PostgreSQL. Это позволяет разработчикам легко управлять данными и интегрировать их в веб-приложения. Кроме того, PHP предоставляет механизмы для создания сессий и работы с куки, что позволяет хранить информацию о состоянии пользователя между запросами и работать с данными на стороне клиента.

# Работа с файлами также является одной из сильных сторон PHP. Язык предлагает средства для работы с файловой системой сервера, включая загрузку, скачивание и обработку файлов. Это делает PHP универсальным решением для создания разнообразных веб-приложений.

# PHP поддерживается большинством хостинг-провайдеров и работает на большинстве серверов, что делает его доступным для широкого круга разработчиков. Язык обладает активным сообществом, множеством библиотек и фреймворков, что облегчает решение различных задач. Простота синтаксиса делает PHP доступным для новичков, но при этом он достаточно гибок для опытных разработчиков.

# PHP остается одним из наиболее популярных языков для серверной веб-разработки благодаря своей простоте и мощным возможностям. Однако, чтобы использовать его эффективно, разработчики должны быть в курсе современных практик безопасности и оптимизации. Это включает в себя защиту от уязвимостей, таких как SQL-инъекции и XSS, а также использование современных фреймворков, которые помогают ускорить разработку и улучшить структуру кода.

# 1.4 MySQL

# MySQL — это популярная реляционная система управления базами данных с открытым исходным кодом, разработанная и поддерживаемая компанией Oracle. Она предоставляет широкий спектр возможностей для хранения, обработки и извлечения данных, что делает её востребованной в различных сферах.

# Одной из ключевых особенностей MySQL является поддержка множества типов данных, включая числовые, строковые, даты/время и JSON. Это позволяет эффективно хранить разнообразную информацию. MySQL также обеспечивает возможность создания индексов, что значительно ускоряет поиск и сортировку данных, повышая производительность при выполнении запросов.

# Система поддерживает транзакции, что позволяет обеспечить целостность данных при выполнении операций. Это особенно важно для приложений, где требуется надежность и согласованность данных. Благодаря модульной архитектуре, MySQL легко расширяется с помощью различных типов расширений, хранимых процедур и триггеров.

# Высокая производительность MySQL достигается за счет оптимизированного ядра и механизмов кэширования, что позволяет эффективно обрабатывать запросы. Поддержка кластеризации и репликации делает MySQL подходящим для создания высокодоступных и масштабируемых конфигураций баз данных.

# MySQL совместима со стандартными языками запросов (SQL) и имеет обширную документацию, активное сообщество пользователей и широкий набор инструментов для администрирования и разработки. Это делает её идеальной для использования в веб-приложениях, системах управления контентом, интернет-магазинах и аналитических системах, где необходимо эффективное управление и обработка больших объемов информации.

# 2 Практическая часть

# 2.1 Требования к работе

Необходимо разработать web-приложения с применением современных web-технологий по тематике «Портфолио», 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:

- Регистрация пользователей: Неавторизованные пользователи могут зарегистрироваться на сайте, заполнив форму добавления нового пользователя. При регистрации каждому пользователю автоматически присваивается статус "авторизованный пользователь".

- Авторизация и управление доступом: Система разделяет пользователей на две группы: администраторы и авторизованные пользователи. Администраторы имеют расширенные права, включая просмотр, добавление, редактирование и удаление пользователей. Авторизованные пользователи могут просматривать и редактировать только свои данные.

- Управление пользователями: Администраторы могут управлять всеми пользователями сайта, включая возможность добавления новых пользователей и редактирования существующих. Это позволяет поддерживать актуальность и безопасность данных.

- Редактирование профиля: Зарегистрированные пользователи имеют возможность редактировать свои личные данные, включая информацию о себе и загруженные работы.

- Просмотр услуг: Неавторизованные пользователи могут просматривать список доступных услуг, предоставляемых на сайте. Это позволяет потенциальным пользователям ознакомиться с возможностями платформы.

Требования к проекту:

- Авторизация и регистрация: Сайт должен обеспечивать возможность регистрации и авторизации пользователей, чтобы разграничить доступ к различным функциям.

- Редактирование данных: Зарегистрированные пользователи могут редактировать свои данные и просматривать свои услуги, что обеспечивает персонализацию и удобство использования.

- Администрирование: Администраторы имеют право редактировать пользователей и добавлять услуги, что позволяет поддерживать актуальность информации на сайте.

- Доступность и адаптивность: Сайт должен быть выдержан в едином стиле и корректно отображаться на устройствах с различными разрешениями экрана.

- Минимальные требования: Проект должен содержать все необходимые функции, даже если он будет максимально простым. Основное внимание уделяется функциональности и удобству использования.

Дополнительные возможности: Бонусы предусмотрены за дополнительный функционал и привлекательный дизайн, что может улучшить пользовательский опыт.

Инструменты разработки:

Для разработки проекта использовались следующие инструменты:

Текстовый редактор: Visual Studio Code для написания и редактирования кода.

Локальный сервер: XAMPP для создания локального сервера и тестирования приложения.

Управление базами данных: phpMyAdmin для администрирования СУБД MySQL, что позволяет легко управлять данными и структурой базы данных.

# 2.2 Создание базы данных

Перед началом работы необходимо запустить приложение xampp

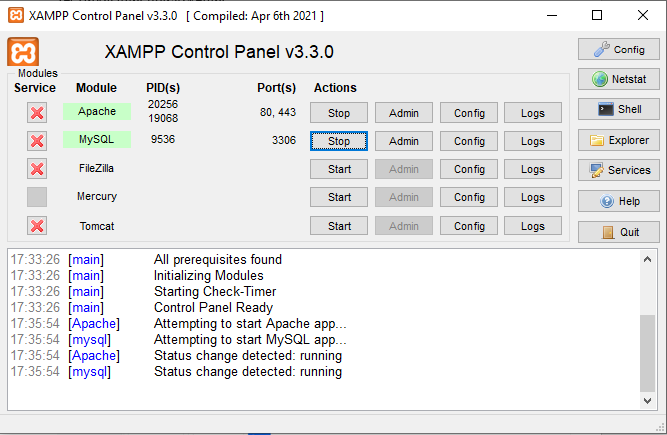


Рисунок 1 – запуск xampp

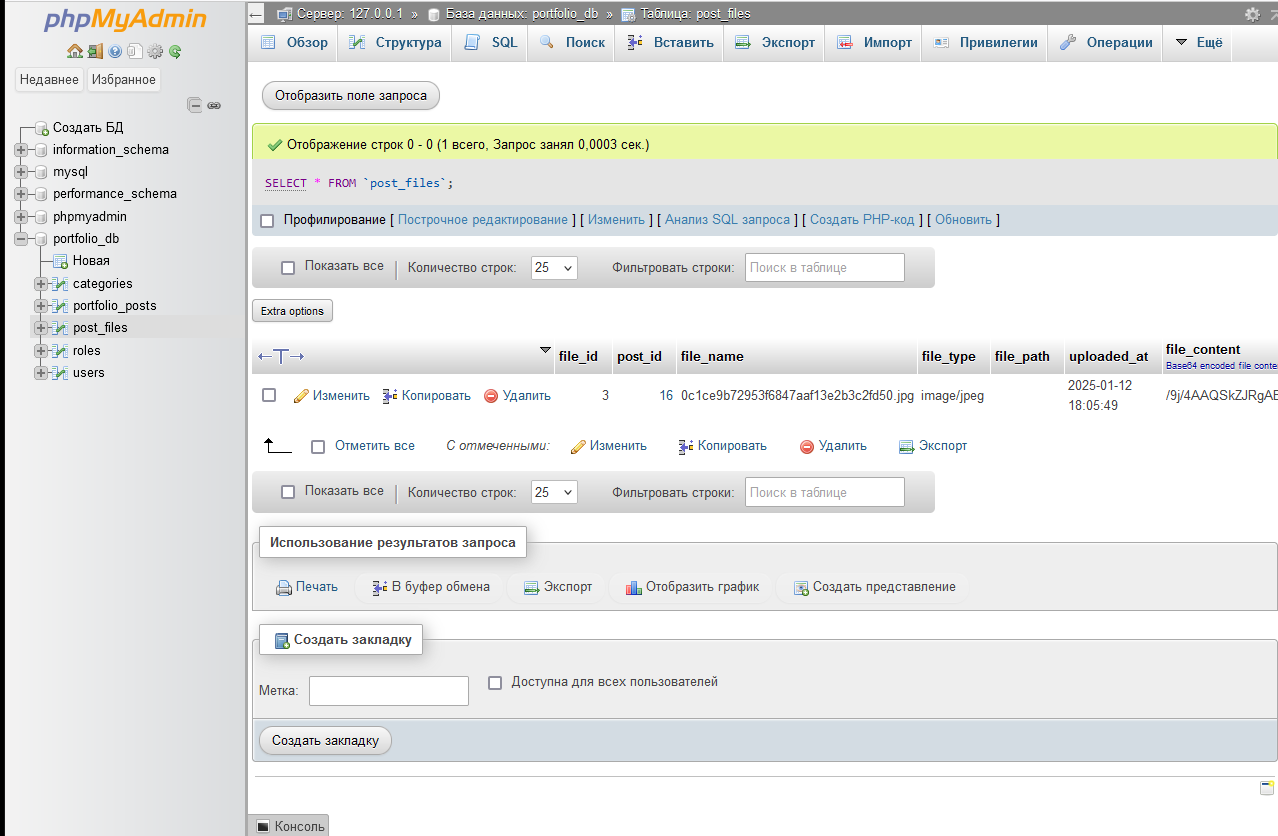
Далее следует открытьphpMyAdmin (рис.2) 

Рисунок 2 – phpMyAdmin

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

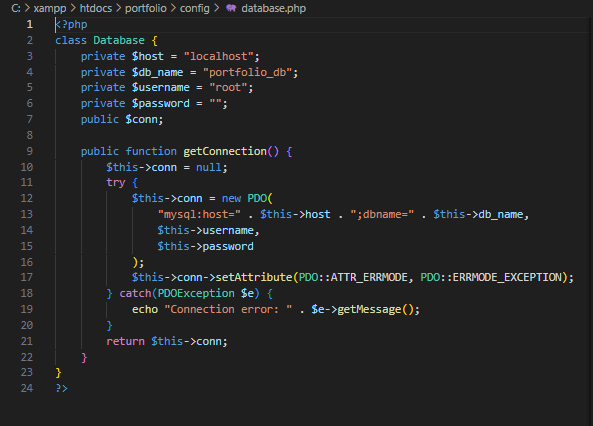


Рисунок 3 – код подключения

Теперь, при ошибке, связанной с подключением базы данных, выведется соответствующее сообщение.

# 2.3 Регистрация и авторизация

Регистрация представляет собой белый фон с окном по середине, со всеми нужными функциями. Также неавторизованный пользователь может сразу перейти в веб-приложение

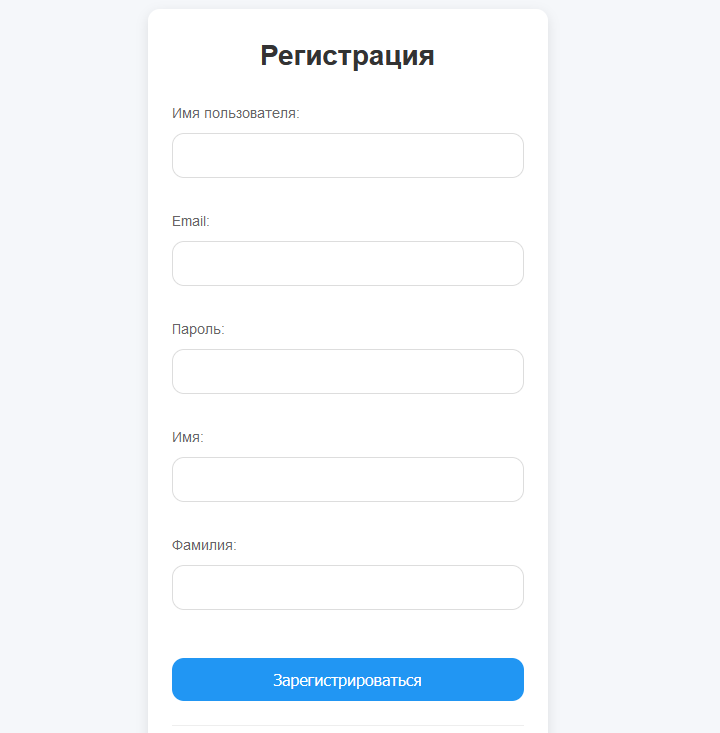


Рисунок 4 – окно регистрации и авторизации

Листинг register.php:

<?php

require\_once 'config/database.php';

require\_once 'models/User.php';

$page\_title = 'Регистрация';

$error = '';

$success = '';

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

    $database = new Database();

    $db = $database->getConnection();

    $user = new User($db);

    $user->username = $\_POST['username'];

    $user->email = $\_POST['email'];

    $user->password = $\_POST['password'];

    $user->first\_name = $\_POST['first\_name'];

    $user->last\_name = $\_POST['last\_name'];

    $result = $user->register();

    if($result['success']) {

        header("Location: login.php");

        exit;

    } else {

        $error = $result['message'];

    }

}

require\_once 'includes/header.php';

?>

<div class="auth-container">

    <h2>Регистрация</h2>

    <?php if ($error): ?>

        <div class="auth-error"><?php echo $error; ?></div>

    <?php endif; ?>

    <?php if ($success): ?>

        <div class="auth-success"><?php echo $success; ?></div>

    <?php endif; ?>

    <form method="POST">

        <div class="form-group">

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

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

        </div>

        <div class="form-group">

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

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

        </div>

        <div class="form-group">

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

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

        </div>

        <div class="form-group">

            <label for="first\_name">Имя:</label>

            <input type="text" id="first\_name" name="first\_name">

        </div>

        <div class="form-group">

            <label for="last\_name">Фамилия:</label>

            <input type="text" id="last\_name" name="last\_name">

        </div>

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

    </form>

    <div class="auth-links">

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

    </div>

</div>

Также для связи функций используются следующие PHP файлы:

Листинг login.php:

<?php

require\_once 'config/database.php';

require\_once 'models/User.php';

require\_once 'auth/check\_auth.php';

// Если пользователь уже авторизован, перенаправляем на профиль

if (isLoggedIn()) {

    header("Location: profile.php");

    exit;

}

$page\_title = 'Вход';

$error = '';

// Обрабатываем форму до подключения header.php

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

    $database = new Database();

    $db = $database->getConnection();

    $user = new User($db);

    $login\_result = $user->login($\_POST['username'], $\_POST['password']);

    if($login\_result) {

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

        $\_SESSION['username'] = $login\_result['username'];

        $\_SESSION['role\_id'] = $login\_result['role\_id'];

        header("Location: profile.php");

        exit;

    } else {

        $error = "Неверное имя пользователя или пароль";

    }

}

// Подключаем header после обработки формы

require\_once 'includes/header.php';

?>

<div class="auth-container">

    <h2>Вход в систему</h2>

    <?php if ($error): ?>

        <div class="auth-error"><?php echo $error; ?></div>

    <?php endif; ?>

    <form method="POST">

        <div class="form-group">

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

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

        </div>

        <div class="form-group">

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

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

        </div>

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

    </form>

    <div class="auth-links">

        <p>Нет аккаунта? <a href="register.php">Зарегистрироваться</a></p>

    </div>

</div>

Листинг logout.php:

<?php

session\_start();

session\_destroy();

header("Location: login.php");

exit;

# 2.4 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, используемом в качестве примера для курсовой работы.

Листинг index.php:

<?php

require\_once 'auth/check\_auth.php';

require\_once 'config/database.php';

require\_once 'models/User.php';

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

$users = $user->getAllUsers();

?>

<!DOCTYPE html>

<html>

<head>

    <title>Главная</title>

    <link rel="stylesheet" href="assets/css/style.css">

</head>

<body>

    <header>

        <nav>

            <div class="nav-left">

                <a href="index.php">Главная</a>

                <?php if (isLoggedIn()): ?>

                    <a href="profile.php">Мой профиль</a>

                <?php endif; ?>

            </div>

            <div class="nav-right">

                <?php if (isLoggedIn()): ?>

                    <?php if(isAdmin()): ?>

                        <a href="admin/users.php">Управление</a>

                    <?php endif; ?>

                    <a href="logout.php" class="btn-logout">Выйти</a>

                <?php else: ?>

                    <a href="login.php" class="btn-login">Войти</a>

                    <a href="register.php" class="btn-register">Регистрация</a>

                <?php endif; ?>

            </div>

        </nav>

    </header>

    <main class="users-grid">

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

            <div class="user-card">

                <div class="user-avatar">

                    <?php if (!empty($user['avatar\_image'])): ?>

                        <img src="data:image/jpeg;base64,<?php echo $user['avatar\_image']; ?>"

                             alt="<?php echo htmlspecialchars($user['username']); ?>"

                             class="avatar-image">

                    <?php else: ?>

                        <div class="avatar-placeholder">

                            <?php echo strtoupper(substr($user['username'], 0, 1)); ?>

                        </div>

                    <?php endif; ?>

                </div>

                <h2 class="user-name"><?php echo htmlspecialchars($user['username']); ?></h2>

                <a href="view\_profile.php?id=<?php echo $user['user\_id']; ?>"

                   class="view-profile-btn">

                    Просмотреть профиль

                </a>

            </div>

        <?php endforeach; ?>

    </main>

</body>

</html>

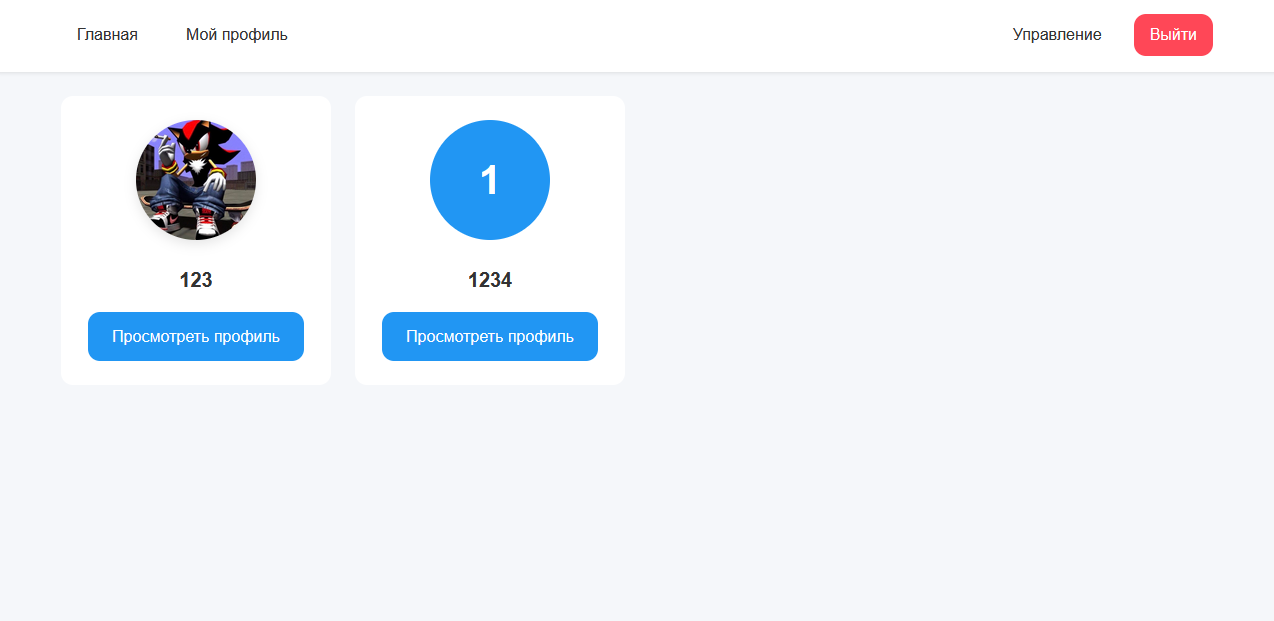


Рисунок 5 – главная страница веб-приложения

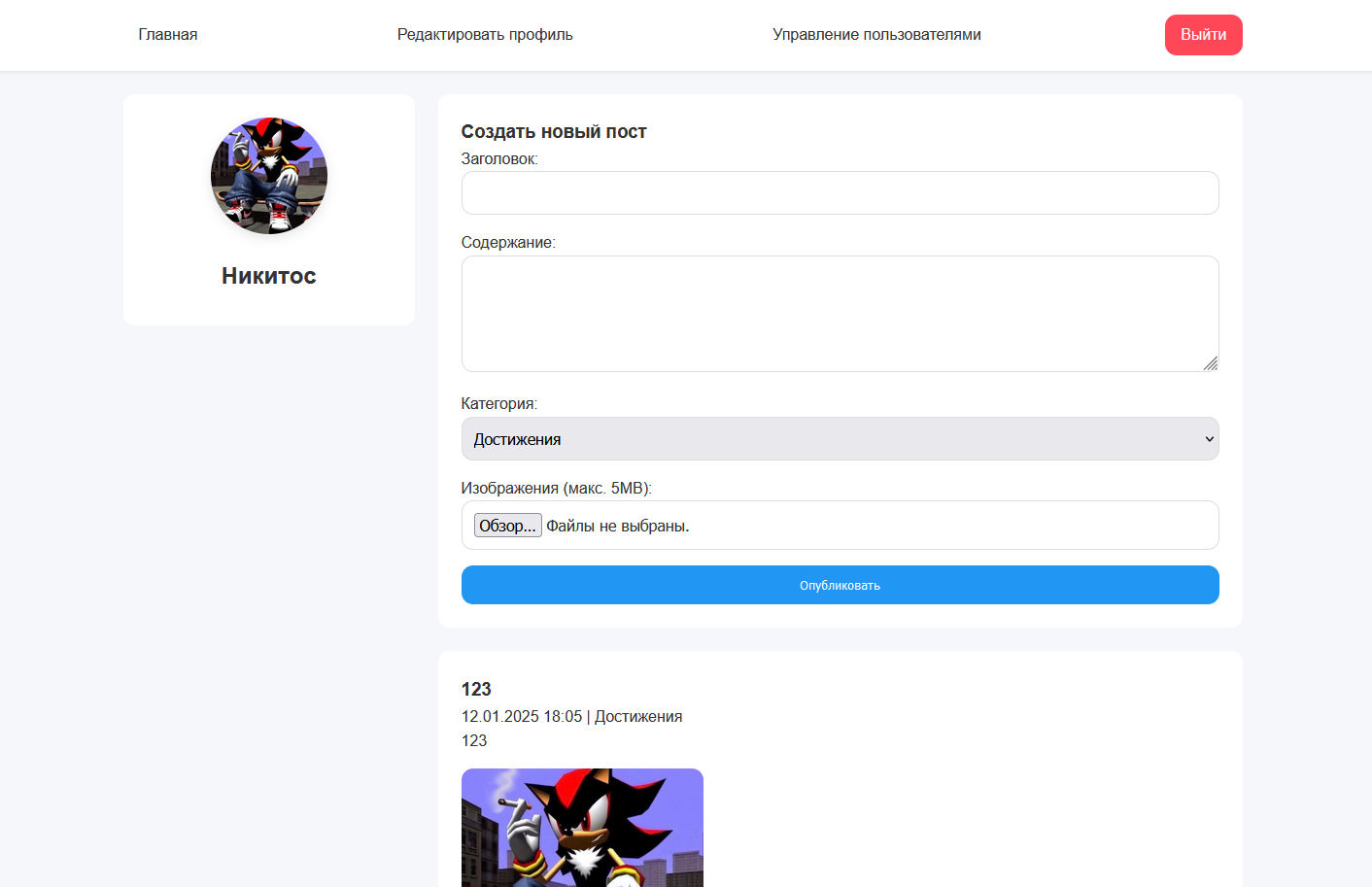


Рисунок 6 – отображение профиля с главной страницы

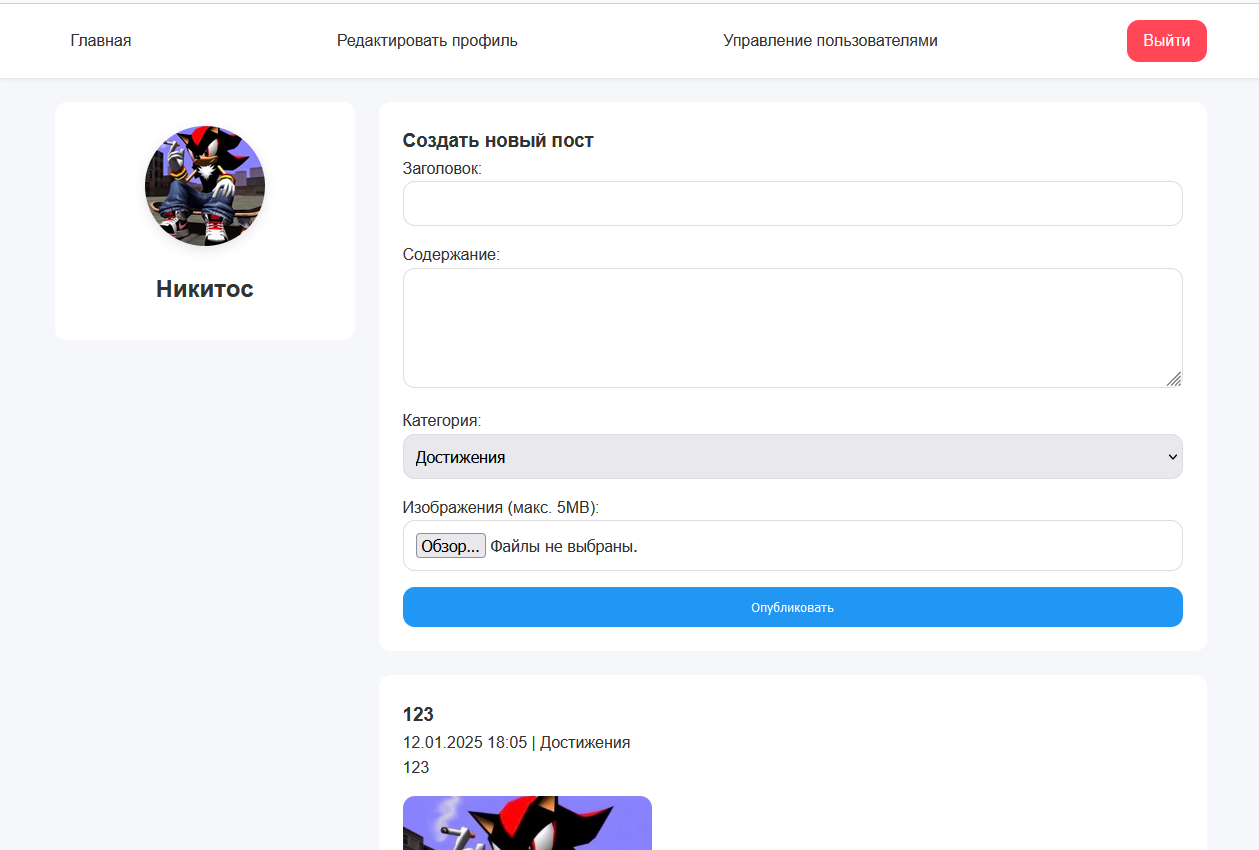


Рисунок 7 – отображение личной информации и загрузка постов

Используется файл create\_post.php:

<?php

require\_once 'auth/check\_auth.php';

require\_once 'config/database.php';

require\_once 'models/Post.php';

if (!isLoggedIn()) {

    header('Location: login.php');

    exit();

}

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

    $database = new Database();

    $db = $database->getConnection();

    $post = new Post($db);

    $files = [];

    if(isset($\_FILES['files'])) {

        foreach($\_FILES['files']['tmp\_name'] as $key => $tmp\_name) {

            if($\_FILES['files']['error'][$key] === UPLOAD\_ERR\_OK) {

                $files[] = [

                    'name' => $\_FILES['files']['name'][$key],

                    'type' => $\_FILES['files']['type'][$key],

                    'tmp\_name' => $tmp\_name,

                    'content' => base64\_encode(file\_get\_contents($tmp\_name))

                ];

            }

        }

    }

    if ($post->create(

        $\_SESSION['user\_id'],

        $\_POST['title'],

        $\_POST['content'],

        $\_POST['category'],

        $files

    )) {

        header('Location: profile.php');

        exit();

    }

}

header('Location: profile.php?error=1');

exit();

# 2.5 Личный кабинет

При переходе в личный кабинет (файл profile.php) перед пользователем представлена возможность изменить все необходимые данные (edit\_profile.php). Также как и для регистрации, здесь используется функция, помогающая определить корректность ввода пароля. Имеется возможность выхода из аккаунта. Сама страница представлена на рисунке 10.

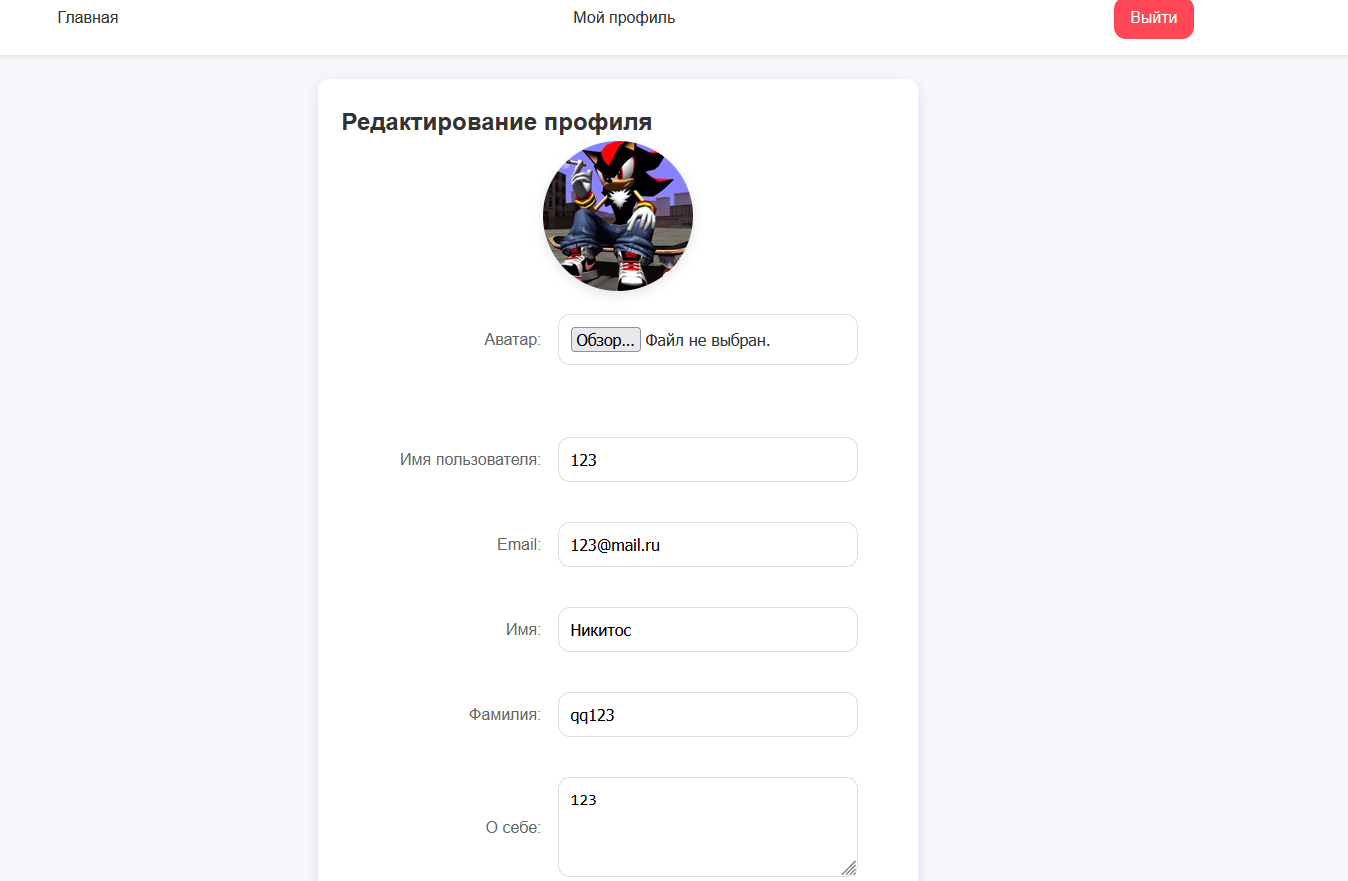


Рисунок 10 – личный кабинет

Для администраторов имеется дополнительная функция, позволяющая работать с панелью администратора (рис.11).

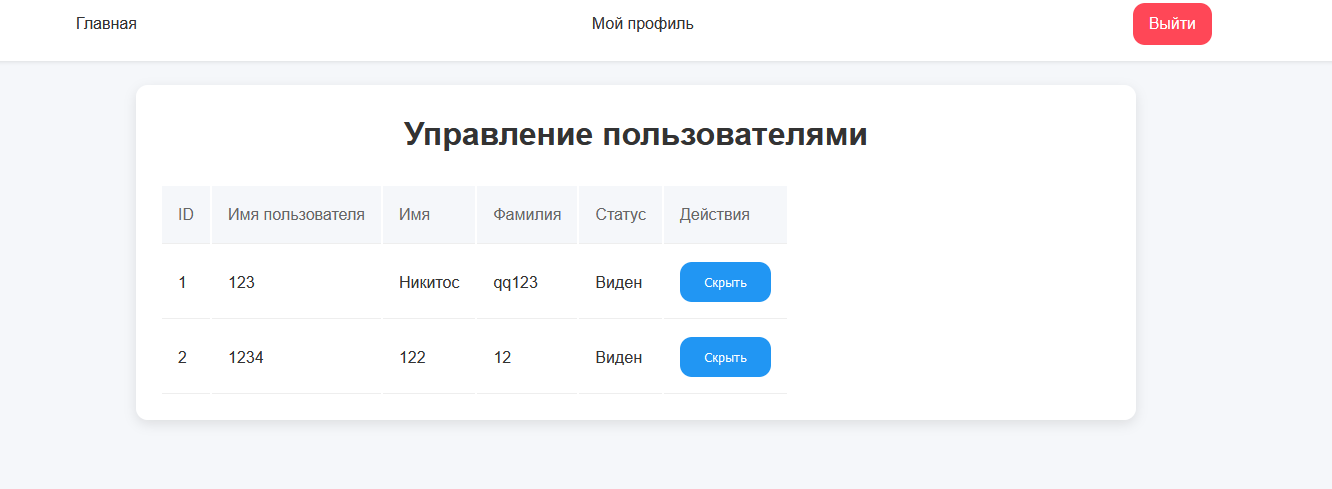


Рисунок 11 – панель администратора

Для объяснения принципа работы всей страницы стоит ознакомиться с кодом:

Листинг profile.php:

<?php

require\_once 'auth/check\_auth.php';

require\_once 'config/database.php';

require\_once 'models/User.php';

require\_once 'models/Post.php';

// Проверяем авторизацию

if (!isLoggedIn()) {

    header('Location: login.php');

    exit();

}

// Получаем данные пользователя

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

$post = new Post($db);

$user\_data = $user->getUserById($\_SESSION['user\_id']);

$user\_posts = $post->getUserPosts($\_SESSION['user\_id']);

// В начале файла после получения данных пользователя

if (isset($\_POST['delete\_post'])) {

    $post\_id = $\_POST['delete\_post'];

    if ($post->deletePost($post\_id, $\_SESSION['user\_id'])) {

        // Обновляем список постов после удаления

        $user\_posts = $post->getUserPosts($\_SESSION['user\_id']);

    }

}

?>

<!DOCTYPE html>

<html>

<head>

    <title>Профиль</title>

    <link rel="stylesheet" href="assets/css/style.css">

</head>

<body>

    <header>

        <nav>

            <a href="index.php">Главная</a>

            <a href="edit\_profile.php" class="edit-btn">Редактировать профиль</a>

            <?php if(isAdmin()): ?>

                <a href="admin/users.php">Управление пользователями</a>

            <?php endif; ?>

            <a href="logout.php" class="btn-logout">Выйти</a>

        </nav>

    </header>

    <div class="profile">

        <div class="profile-info">

            <?php if (!empty($user\_data['avatar\_image'])): ?>

                <div class="avatar-container">

                    <img src="data:image/jpeg;base64,<?php echo $user\_data['avatar\_image']; ?>"

                         alt="Avatar" class="profile-avatar">

                </div>

            <?php endif; ?>

            <h2><?php echo htmlspecialchars($user\_data['first\_name']); ?></h2>

        </div>

        <div class="profile-wall">

            <div class="create-post">

                <h3>Создать новый пост</h3>

                <form action="create\_post.php" method="POST" enctype="multipart/form-data">

                    <div>

                        <label for="title">Заголовок:</label>

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

                    </div>

                    <div>

                        <label for="content">Содержание:</label>

                        <textarea id="content" name="content" required></textarea>

                    </div>

                    <div>

                        <label for="category">Категория:</label>

                        <select id="category" name="category" required>

                            <?php foreach ($post->getAllCategories() as $category): ?>

                                <option value="<?php echo $category['category\_id']; ?>">

                                    <?php echo htmlspecialchars($category['name']); ?>

                                </option>

                            <?php endforeach; ?>

                        </select>

                    </div>

                    <div>

                        <label for="files">Изображения (макс. 5MB):</label>

                        <input type="file" id="files" name="files[]" multiple accept="image/jpeg,image/png,image/gif">

                        <div class="progress-bar" style="display: none;">

                            <div class="progress"></div>

                        </div>

                        <div id="uploadProgress"></div>

                    </div>

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

                </form>

            </div>

            <div class="posts-wall">

                <?php foreach ($user\_posts as $post): ?>

                    <article class="post-card">

                        <h3><?php echo htmlspecialchars($post['title']); ?></h3>

                        <div class="post-meta">

                            <?php echo date('d.m.Y H:i', strtotime($post['created\_at'])); ?>

                            | <?php echo htmlspecialchars($post['category']); ?>

                        </div>

                        <div class="post-content">

                            <?php echo nl2br(htmlspecialchars($post['content'])); ?>

                        </div>

                        <?php if (!empty($post['files'])): ?>

                            <div class="post-files">

                                <?php foreach ($post['files'] as $file): ?>

                                    <?php if (in\_array($file['file\_type'], ['image/jpeg', 'image/png', 'image/gif'])): ?>

                                        <img src="data:<?php echo $file['file\_type']; ?>;base64,<?php echo $file['file\_content']; ?>"

                                             alt="Image" class="post-image">

                                    <?php else: ?>

                                        <a href="data:<?php echo $file['file\_type']; ?>;base64,<?php echo $file['file\_content']; ?>"

                                           download="<?php echo htmlspecialchars($file['file\_name']); ?>">

                                            <?php echo htmlspecialchars($file['file\_name']); ?>

                                        </a>

                                    <?php endif; ?>

                                <?php endforeach; ?>

                            </div>

                        <?php endif; ?>

                        <?php if ($post['user\_id'] == $\_SESSION['user\_id'] || isAdmin()): ?>

                            <div class="post-actions">

                                <a href="edit\_post.php?id=<?php echo $post['post\_id']; ?>" class="btn-edit">Редактировать</a>

                                <form method="POST" style="display: inline;" onsubmit="return confirm('Вы уверены, что хотите удалить этот пост?');">

                                    <input type="hidden" name="delete\_post" value="<?php echo $post['post\_id']; ?>">

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

                                </form>

                            </div>

                        <?php endif; ?>

                    </article>

                <?php endforeach; ?>

            </div>

        </div>

    </div>

    <script>

    document.getElementById('files').onchange = async function(e) {

        const preview = document.createElement('div');

        preview.className = 'upload-preview';

        // Очищаем предыдущий предпросмотр

        const oldPreview = document.querySelector('.upload-preview');

        if (oldPreview) oldPreview.remove();

        for (let file of this.files) {

            // Проверяем тип файла

            if (!file.type.startsWith('image/')) {

                continue;

            }

            // Проверяем размер

            if (file.size > 5 \* 1024 \* 1024) {

                alert('Файл ' + file.name + ' слишком большой. Максимальный размер 5MB');

                continue;

            }

            // Создаем контейнер для изображения

            const imgContainer = document.createElement('div');

            imgContainer.className = 'preview-container';

            const img = document.createElement('img');

            img.className = 'upload-preview-image';

            // Добавляем кнопку удаления

            const deleteBtn = document.createElement('button');

            deleteBtn.className = 'delete-preview';

            deleteBtn.innerHTML = '×';

            deleteBtn.onclick = function(e) {

                e.preventDefault();

                imgContainer.remove();

            };

            // Сжимаем изображение

            const compressedImage = await compressImage(file);

            const reader = new FileReader();

            reader.onload = function(e) {

                img.src = e.target.result;

            };

            reader.readAsDataURL(compressedImage);

            imgContainer.appendChild(img);

            imgContainer.appendChild(deleteBtn);

            preview.appendChild(imgContainer);

        }

        this.parentNode.appendChild(preview);

    };

    // Функция сжатия изображения

    async function compressImage(file) {

        return new Promise((resolve) => {

            const reader = new FileReader();

            reader.onload = function(e) {

                const img = new Image();

                img.onload = function() {

                    const canvas = document.createElement('canvas');

                    let width = img.width;

                    let height = img.height;

                    // Максимальные размеры

                    const MAX\_WIDTH = 1200;

                    const MAX\_HEIGHT = 1200;

                    if (width > height) {

                        if (width > MAX\_WIDTH) {

                            height \*= MAX\_WIDTH / width;

                            width = MAX\_WIDTH;

                        }

                    } else {

                        if (height > MAX\_HEIGHT) {

                            width \*= MAX\_HEIGHT / height;

                            height = MAX\_HEIGHT;

                        }

                    }

                    canvas.width = width;

                    canvas.height = height;

                    const ctx = canvas.getContext('2d');

                    ctx.drawImage(img, 0, 0, width, height);

                    canvas.toBlob((blob) => {

                        resolve(new File([blob], file.name, {

                            type: 'image/jpeg',

                            lastModified: Date.now()

                        }));

                    }, 'image/jpeg', 0.7); // Качество 0.7

                };

                img.src = e.target.result;

            };

            reader.readAsDataURL(file);

        });

    }

    // Добавляем обработку отправки формы

    document.querySelector('form').onsubmit = function(e) {

        const progressBar = document.querySelector('.progress-bar');

        const progress = document.querySelector('.progress');

        progressBar.style.display = 'block';

        let width = 0;

        const interval = setInterval(() => {

            if (width >= 100) {

                clearInterval(interval);

            } else {

                width++;

                progress.style.width = width + '%';

            }

        }, 10);

    };

    </script>

</body>

</html>

Также используются следующие PHP файлы:

Листинг edit\_profile.php:

<?php

require\_once 'auth/check\_auth.php';

require\_once 'config/database.php';

require\_once 'models/User.php';

// Проверяем авторизацию

if (!isLoggedIn()) {

    header("Location: login.php");

    exit;

}

// Получаем данные пользователя

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

$user\_data = $user->getUserById($\_SESSION['user\_id']);

// Обработка формы

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

    $update\_data = [

        'user\_id' => $\_SESSION['user\_id'],

        'username' => trim($\_POST['username']),

        'email' => trim($\_POST['email']),

        'first\_name' => trim($\_POST['first\_name']),

        'last\_name' => trim($\_POST['last\_name']),

        'bio' => trim($\_POST['bio'])

    ];

    // Обработка загруженного аватара

    if (isset($\_FILES['avatar']) && $\_FILES['avatar']['error'] === UPLOAD\_ERR\_OK) {

        $image\_info = getimagesize($\_FILES['avatar']['tmp\_name']);

        if ($image\_info !== false) {

            $update\_data['avatar\_image'] = base64\_encode(file\_get\_contents($\_FILES['avatar']['tmp\_name']));

        }

    }

    // Обработка нового пароля

    if (!empty($\_POST['new\_password'])) {

        $update\_data['new\_password'] = $\_POST['new\_password'];

    }

    if ($user->updateProfile($update\_data)) {

        $success = "Профиль успешно обновлен";

        $user\_data = $user->getUserById($\_SESSION['user\_id']);

    } else {

        $error = "Ошибка при обновлении профиля";

    }

}

?>

<!DOCTYPE html>

<html>

<head>

    <title>Редактирование профиля</title>

    <link rel="stylesheet" href="assets/css/style.css">

</head>

<body>

    <header>

        <nav>

            <a href="index.php">Главная</a>

            <a href="profile.php">Мой профиль</a>

            <a href="logout.php" class="btn-logout">Выйти</a>

        </nav>

    </header>

    <div class="edit-profile">

        <h2>Редактирование профиля</h2>

        <?php if (isset($error)): ?>

            <div class="error"><?php echo htmlspecialchars($error); ?></div>

        <?php endif; ?>

        <?php if (isset($success)): ?>

            <div class="success"><?php echo htmlspecialchars($success); ?></div>

        <?php endif; ?>

        <form method="POST" enctype="multipart/form-data">

            <div class="avatar-section">

                <?php if (!empty($user\_data['avatar\_image'])): ?>

                    <img src="data:image/jpeg;base64,<?php echo $user\_data['avatar\_image']; ?>"

                         alt="Avatar" class="current-avatar">

                <?php endif; ?>

                <div class="form-group">

                    <label for="avatar">Аватар:</label>

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

                </div>

            </div>

            <div class="form-group">

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

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

                       value="<?php echo htmlspecialchars($user\_data['username']); ?>" required>

            </div>

            <div class="form-group">

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

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

                       value="<?php echo htmlspecialchars($user\_data['email']); ?>" required>

            </div>

            <div class="form-group">

                <label for="first\_name">Имя:</label>

                <input type="text" id="first\_name" name="first\_name"

                       value="<?php echo htmlspecialchars($user\_data['first\_name']); ?>">

            </div>

            <div class="form-group">

                <label for="last\_name">Фамилия:</label>

                <input type="text" id="last\_name" name="last\_name"

                       value="<?php echo htmlspecialchars($user\_data['last\_name']); ?>">

            </div>

            <div class="form-group">

                <label for="bio">О себе:</label>

                <textarea id="bio" name="bio"><?php echo htmlspecialchars($user\_data['bio'] ?? ''); ?></textarea>

            </div>

            <div class="form-group">

                <label for="new\_password">Новый пароль (оставьте пустым, если не хотите менять):</label>

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

            </div>

            <div class="form-actions">

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

                <a href="profile.php" class="btn-cancel">Отмена</a>

            </div>

        </form>

    </div>

</body>

</html>

Листинг edit\_user.php:

<?php

require\_once '../auth/check\_auth.php';

require\_once '../config/database.php';

require\_once '../models/User.php';

// Проверяем права администратора или владельца профиля

if (!isAdmin() && !checkUserAccess($\_GET['id'])) {

    header("Location: ../login.php");

    exit;

}

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

$message = '';

$messageType = '';

// Получаем данные пользователя

$user\_data = $user->getUserById($\_GET['id']);

// Обработка формы редактирования

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

    $user->user\_id = $\_GET['id'];

    $user->email = $\_POST['email'];

    $user->first\_name = $\_POST['first\_name'];

    $user->last\_name = $\_POST['last\_name'];

    if ($user->update()) {

        $message = "Данные успешно обновлены";

        $messageType = "success";

        $user\_data = $user->getUserById($\_GET['id']); // Обновляем данные для отображения

    } else {

        $message = "Ошибка при обновлении данных";

        $messageType = "error";

    }

}

?>

<!DOCTYPE html>

<html>

<head>

    <title>Редактирование пользователя</title>

    <style>

        .error { color: red; }

        .success { color: green; }

        form {

            margin: 20px;

        }

        input {

            display: block;

            margin: 10px 0;

            padding: 5px;

        }

    </style>

</head>

<body>

    <h2>Редактирование пользователя</h2>

    <?php if($message): ?>

        <div class="<?php echo $messageType; ?>">

            <?php echo $message; ?>

        </div>

    <?php endif; ?>

    <form method="POST">

        <input type="email" name="email" value="<?php echo htmlspecialchars($user\_data['email']); ?>" required>

        <input type="text" name="first\_name" value="<?php echo htmlspecialchars($user\_data['first\_name']); ?>">

        <input type="text" name="last\_name" value="<?php echo htmlspecialchars($user\_data['last\_name']); ?>">

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

    </form>

    <p>

        <?php if(isAdmin()): ?>

            <a href="users.php">Вернуться к списку пользователей</a>

        <?php else: ?>

            <a href="../profile.php">Вернуться в профиль</a>

        <?php endif; ?>

    </p>

</body>

</html>

Листинг users.php:

<?php

require\_once '../auth/check\_auth.php';

require\_once '../config/database.php';

require\_once '../models/User.php';

// Проверяем права администратора

if (!isAdmin()) {

    header('Location: ../index.php');

    exit();

}

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

// Обработка изменения видимости

if (isset($\_POST['toggle\_visibility']) && isset($\_POST['user\_id'])) {

    $user->toggleVisibility($\_POST['user\_id']);

}

// Получаем список всех пользователей, включая скрытые

$users = $user->getAllUsers(true);

?>

<!DOCTYPE html>

<html>

<head>

    <title>Управление пользователями</title>

    <link rel="stylesheet" href="../assets/css/style.css">

</head>

<body>

    <header>

        <nav>

            <a href="../index.php">Главная</a>

            <a href="../profile.php">Мой профиль</a>

            <a href="../logout.php" class="btn-logout">Выйти</a>

        </nav>

    </header>

    <div class="admin-panel">

        <h2>Управление пользователями</h2>

        <div class="users-table">

            <table>

                <thead>

                    <tr>

                        <th>ID</th>

                        <th>Имя пользователя</th>

                        <th>Имя</th>

                        <th>Фамилия</th>

                        <th>Статус</th>

                        <th>Действия</th>

                    </tr>

                </thead>

                <tbody>

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

                        <tr>

                            <td><?php echo $user['user\_id']; ?></td>

                            <td><?php echo htmlspecialchars($user['username']); ?></td>

                            <td><?php echo htmlspecialchars($user['first\_name']); ?></td>

                            <td><?php echo htmlspecialchars($user['last\_name']); ?></td>

                            <td>

                                <?php echo $user['is\_hidden'] ? 'Скрыт' : 'Виден'; ?>

                            </td>

                            <td>

                                <form method="POST" style="display: inline;">

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

                                    <button type="submit" name="toggle\_visibility" class="btn">

                                        <?php echo $user['is\_hidden'] ? 'Показать' : 'Скрыть'; ?>

                                    </button>

                                </form>

                            </td>

                        </tr>

                    <?php endforeach; ?>

                </tbody>

            </table>

        </div>

    </div>

</body>

</html>

Далее рассмотрим созданную базу данных.

Созданная база данных имеет следующие таблицы:

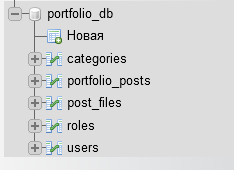


Рисунок 12 – база данных

# Заключение

В рамках данного курсового проекта было разработано веб-приложение "Портфолио на стене", использующее современные веб-технологии, такие как HTML, CSS, JavaScript, PHP и MySQL. Также была создана база данных для управления пользователями и их портфолио.

Дизайн проекта выполнен в минималистичном стиле, сохраняя при этом все необходимые функциональные возможности. В приложении реализованы формы регистрации и авторизации, где применены базовые алгоритмы безопасности, такие как хеширование паролей и проверка их минимальной длины.

Пользователи имеют доступ к личному кабинету, где они могут редактировать свои данные, включая логин и пароль, а также управлять своим портфолио. Для администраторов предусмотрена специальная панель, позволяющая управлять пользователями, редактировать и удалять аккаунты, а также модерировать контент, представленный в портфолио.

Проект демонстрирует возможности создания удобного и функционального веб-приложения для представления и управления профессиональными достижениями пользователей.

# 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:
5. https://ru.hexlet.io/courses/php-setup-environment/lessons/what-is-php/theory\_unit (дата обращения: 12.01.2024).
6. Что такое MySQL: Макхост [Электронный источник]. – URL: https://mchost.ru/articles/chto-takoe-mysql/ [дата доступа 12.01.2024].
7. Что такое - PHP? [Электронный источник]. – URL: https://archive-ipq-co.narod.ru/ (дата обращения: 12.01.2023).
8. Что такое HTML и CSS [Электронный источник]. – URL: https://shneider-host.ru/blog/chto-takoe-html-i-css.html (дата обращения: 12.01.2024).
9. 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).
10. PHP: википедия [Электронный источник]. – URL: https://ru.wikipedia.org/wiki/PHP [дата доступа 12.01.2024].