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**

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

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Н. Никитин

Студент УБ23-08Б, №432326071 \_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Кузьмина

Красноярск 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc8374)

[1 Теоретическая часть 4](#_Toc26651)

[1.1 HTML - язык гипертекстовой разметки текста 4](#_Toc5888)

[1.2 CSS - каскадные таблицы стилей 6](#_Toc10561)

[1.3 PHP - язык с динамической типизацией 8](#_Toc19233)

[1.4 JavaScript — язык управления элементами на сайте 10](#_Toc31558)

[1.5 MySQL - свободная реляционная СУБД 12](#_Toc2683)

[1.6 Веб-приложение phpMyAdmin 13](#_Toc6941)

[2 Проектная часть 14](#_Toc5865)

[2.1 Создание и подключение базы данных 14](#_Toc8480)

[2.2 Создание главной страницы сайта 16](#_Toc13724)

[2.3 Организация авторизации и регистрации пользователей 18](#_Toc6898)

[2.4 Создание и редактирование личного кабинета пользователем 20](#_Toc3580)

[2.5 Создание, добавление, редактирование и удаление статей ..............22](#_Toc26988)

[2.6 Организация админ-панели: просмотр, изменение, добавление и удаление пользователей 27](#_Toc8291)

[2.8 Оформление сайта 32](#_Toc10964)

[ЗАКЛЮЧЕНИЕ 34](#_Toc7155)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc31729)

## ВВЕДЕНИЕ

Современные технологии веб-разработки играют важную роль в создании удобных и функциональных веб-приложений. С каждым годом веб-приложения становятся более востребованными в различных сферах жизни, включая спорт.

Интернет-платформы, посвященные спортивной тематике, позволяют пользователям получать актуальную информацию, взаимодействовать друг с другом и участвовать в обсуждениях, связанных со спортом.

Целью данной курсовой работы является разработка веб-приложения на тему «Спорт» с применением современных технологий веб-разработки, таких как HTML, CSS, PHP и MySQL. Веб-приложение будет представлять собой платформу для публикации статей и контента.

Актуальность проекта обусловлена широким использованием веб-приложений в спортивной индустрии, что позволяет повышать интерес к спорту и улучшать коммуникацию между спортсменами.

Задачи курсовой работы:

1. Изучить возможности HTML и CSS для создания структуры и стилизации веб-приложения.
2. Реализовать динамическую функциональность с использованием PHP.
3. Создать базу данных в MySQL для хранения данных о пользователях и статьях.
4. Обеспечить функциональность регистрации и авторизации пользователей.
5. Разделить пользователей на группы: авторизованные пользователи и администраторы.
6. Разработать функционал для добавления, редактирования и удаления статей, а также для возможности их комментирования.
7. Реализовать адаптивный дизайн для корректного отображения на различных устройствах.

## 1 Теоретическая часть

## HTML - язык гипертекстовой разметки текста

HTML (HyperText Markup Language) [2] — это язык гипертекстовой разметки. Гипертекст — это система текстов, в которой разработчики прописывают элементы сайта и связывают их между собой ссылками. Благодаря гипертексту можно переходить по ссылкам на сайте, открывать видео и картинки. А гипертекстовая разметка — это правила, по которым разработчики создают текст веб-страницы. Основная цель HTML — структурировать и оформлять контент на сайте.

HTML создает иерархическую структуру веб-страницы, используя заголовки, абзацы, списки и таблицы. Такая структура помогает пользователю легче ориентироваться на сайте.

Также с помощью HTML отображается текст, изображения, таблицы, видео и аудио. А еще благодаря коду на HTML мы можем переходить по ссылкам в интернете с одного сайта на другой.

В HTML можно даже создавать простой дизайн: например, устанавливать цвет и шрифт текста или фоновый цвет блока. Но более сложный дизайн страницы разработчики делают с помощью CSS — языка стилей, который создали специально для работы в связке с HTML.

Возможностей HTML не хватает, чтобы «оживлять» сайты, делать их функциональными и интерактивными. Здесь приходит на помощь язык программирования JavaScript, благодаря которому мы можем взаимодействовать с формами на сайте, видеть динамические элементы и анимации.

Теги HTML — это команды, которые говорят браузеру, что и в каком порядке показывать на экране. У каждого тега есть имя, которое расположено в угловых скобках.

HTML-файл всегда начинается с тега <!DOCTYPE html>, по которому браузер понимает, как правильно отобразить страницу.

В теге <head> <head/> хранится служебная информация — заголовок и кодировка страницы.

Внутри тега <body> <body/> помещается содержимое страницы, которое отображается в браузере — текст, изображения, видео.

В теге <title> <title/> — заголовок веб-страницы. Его браузер отобразит как название и сохранит в описании, если сохранить страницу в закладки.

Тег <img> помещает изображение в нужное место страницы.

Есть парные и непарные HTML-теги, первых — большинство. Парный тег состоит из открывающего и закрывающего тега. Закрывающий тег содержит дополнительный элемент / — эта косая черта называется «слэш».

У каждого тега есть атрибуты. Они расширяют возможности тегов: задают стили, добавляют ссылки, управляют мультимедиа.

Заголовки в HTML — это элементы, которые используются для создания заголовков и подзаголовков на веб-странице. Они определяют важность и иерархию информации на странице.

В HTML применяют шесть тегов заголовков: <h1>, <h2>, <h3>, <h4>, <h5> и <h6>. Тег <h1> обычно используется для главного заголовка страницы, а теги <h2>-<h6> используются для подзаголовков и других заголовков на странице.

Абзац — это элемент, который используется для создания текстовых блоков на странице. Каждый абзац текста создается парным тегом <p> <p/>.

Текст, разбитый на абзацы читать и воспринимать легче, чем сплошное «полотно». Абзац делит содержимое на логически связанные блоки.

писки упорядочивают информацию на странице и улучшают ее восприятие.

Обычно используются нумерованные и ненумерованные списки.

Ненумерованные или маркированные списки («буллеты») добавляют на страницу тегом <ul></ul>. Такие списки применяют, когда последовательность элементов не важна.

Для создания нумерованного списка используется тег <ol><ol/>.

Отдельный элемент в списке любого типа вводится тегом <li></li>. Этот тег нужно закрывать после каждого пункта.

## CSS - каскадные таблицы стилей

**CSS (Cascading Style Sheets, каскадные таблицы стилей)** [5] — язык описания внешнего вида HTML-документа. Это одна из базовых технологий в современном интернете. Практически ни один сайт не обходится без CSS, поэтому HTML и CSS действуют в единой связке.

CSS, как и любой язык, имеет свой синтаксис. В нем есть правила — значения, которые определяют внешний вид элементов. CSS-правило состоит из селектора, CSS-свойств и их значений:

Селекторы — это метки, которые помогают браузеру понять, к какой части HTML-кода нужно применить заданные параметры.

CSS-свойства — это определенные параметры оформления, например цвет элемента или текста (color) или цвет фона (background).

Значение — это просто значение, оно выражается текстом или числом, например черный (black).

селектор {

свойство: значение;

}

p {

color: black

}

CSS-правила в коде заключаются в фигурные скобки {…}. Перед открытием скобки обязательно нужно указать селектор, к которому относится это правило.

В примере селектором является <p>, и он выбирает все теги с именем <p>, color — это CSS-свойство а black — значение CSS-свойства. Связка «свойство: значение» называется блоком объявления стилей. Внутри него свойство отделяется от значения двоеточием, а один блок от другого отделяет точка с запятой.

Таблицы называются каскадными, потому что работают по принципу каскада — то есть правило, прописанное ниже, считается приоритетным. Например, если в нашем примере под значением фонового цвета мы пропишем еще одно значение color: red, то цвет текста будет красным, а не черным.

p {

color: black

background: #eeeeee

color: red

}

Чтобы использовать CSS совместно с HTML, можно выбрать один из способов:

1. Весь код, написанный на CSS, прописывается в отдельном внешнем файле с расширением .css. Его подключают к HTML-странице при помощи тега <link href> — это служебный тег, который на странице не будет видно:

<head>

<link href=”style.css” rel=”stylesheet”>

</head>

Атрибут rel со значением stylesheet указывает, что в документе применяются именно стили текста. Это важно, так как, кроме применения стилей, тег <link> используется еще во множестве разных значений.

2. Прописать стили CSS внутри конкретного тега с помощью атрибута style:

<p style=»color: black; background: #eeeeee»> Добавление стиля с помощью атрибута style</p>

В этом случае можно не использовать фигурные скобки, потому что и без них браузеру будет понятно, к какому элементу относится правило.

Некоторые CSS-свойства и их назначение:

1. font-family, font-size, font-weight, font-style — свойства шрифта;
2. color, background-color, border-color — свойства цвета;
3. text-align, text-decoration, text-transform — свойства текста;
4. display, float, position, margin, padding — свойства макета;
5. background-image, background-repeat, background-position — свойства фона.

## **PHP - язык с динамической типизацией**

Некоторые из ключевых возможностей PHP [6]:

1. Динамические веб-страницы: PHP позволяет создавать динамические веб-страницы, которые могут адаптироваться к пользовательскому вводу и предоставлять интерактивные функции.
2. Обработка форм: PHP легко интегрируется с HTML-формами, что делает его идеальным для обработки данных, отправляемых с веб-страниц.
3. Работа с базами данных: PHP имеет широкую поддержку для множества систем управления базами данных (например, MySQL, PostgreSQL, SQLite), что делает его отличным выбором для создания веб-приложений с базами данных.
4. Работа с файлами: PHP предоставляет функции для чтения, записи и обработки файлов на сервере, что полезно для управления содержимым сайта.
5. Работа с изображениями: PHP позволяет манипулировать изображениями, изменять их размеры, форматы и многое другое.
6. Интеграция с другими технологиями: PHP может легко интегрироваться с другими технологиями, такими как HTML, CSS, JavaScript, что позволяет создавать полноценные веб-приложения.
7. Кросс-платформенность: PHP поддерживается на различных операционных системах (Windows, Linux, macOS) и работает с различными веб-серверами (Apache, Nginx).

PHP хранит состояние, созданное в текущий момент времени. В отличие от классических компилируемых языков программирования, которые работают все время от запуска до остановки программы, PHP-скрипт начинает выполняться с момента обращения веб-сервера и заканчивает (забывает все, что знал), когда возвращает сгенерированный пакет данных обратно веб-серверу.

Возможность комбинировать PHP- и HTML-код позволяет создавать динамические страницы, которые показывают каждый раз разную информацию в зависимости от полученных от пользователя данных или настроек.

Синтаксис языка берет начало из C, Java и Perl. Классический PHP-сценарий представляет собой набор выражений. Каждое из них начинается с новой строки и заканчивается точкой с запятой. Выражение — это инструкция, по которой PHP-интерпретатор выполняет одно действие, например умножает числа, выводит информацию на экран.

Вот так выглядит самый простой сценарий, который выводит на экран строку «Hello world!»:

<?php echo("Hello world"); ?>

PHP-сценарий всегда начинается <?php. Так веб-серверу отправляется сигнал, что код в файле написан на языке PHP.

Коды PHP и HTML могут встраиваться вместе в файле с расширением PHP, например в index.php. Вот тот же пример, но расположенный внутри HTML:

<body>

<p><?php echo("Hello world"); ?></p>

</body>

Фрагмент ?> сообщает серверу, где заканчивается PHP-сценарий. Если после кода в документе ничего не следует, фрагмент ?> не обязателен.

## JavaScript — язык управления элементами на сайте

JavaScript [7] — это интерпретируемый язык программирования, который используют для написания frontend- и backend-частей сайтов, а также мобильных приложений.

JavaScript простыми словами называют языком скриптов или сценариев. Скрипты — это набор инструкций, которые выполняются при загрузке страницы. Браузер самостоятельно интерпретирует код на JavaScript.

Скрипты можно прописать внутри кода страницы или подключить к HTML отдельным файлом. Например в стандартной разметке index.html JS-код прописывают внутри тега script, помещенного в тег body

Но обычно для этого создается отдельный файл с расширением .js. Его называют script.js и прописывают код в нем.

JavaScript отвечает за интерактивность и функционал этого интерфейса. Вот примеры задач, которые решает JS:

1. Взаимодействие с пользователем. Всплывающие окна с полем для ввода почты или других данных, уведомления об использовании cookie.
2. Обработка данных HTML. JS помогает заполнить необходимые поля в формах, проверять буквенные и числовые значения, а также управлять HTML-элементами на странице. Например, в конструкторе лендингов Tilda можно перетаскивать блоки и по клику менять шрифты, цвета текста, фонов и другие значения.
3. Анимация. Это не только движущиеся объекты на странице и кнопки, но и карусель в галерее или плавная прокрутка страницы. Например, Pinterest с помощью JavaScript создал эффект бесконечной ленты.
4. Математические вычисления. С помощью JS-скриптов на сайтах реализованы калькуляторы, например подсчет суммы заказа в корзине интернет-магазина.
5. Мобильные приложения: С использованием фреймворков, таких как React Native и NativeScript, JavaScript может быть использован для создания мобильных приложений для разных платформ.
6. Серверный JavaScript: используется на серверной стороне с использованием платформы Node.js для создания высокопроизводительных и масштабируемых веб-приложений.
7. Обработка событий: JavaScript используется для обработки событий, таких как клики мышью, нажатия клавиш, отправка форм и другие действия пользователя. Это позволяет создавать реактивные и отзывчивые интерфейсы.

Действия пользователя на странице вызывают события: клик на кнопке запускает анимацию, захват мышкой и перемещение курсора двигают объект по странице. Каждое из этих действий выполняется с помощью скриптов, написанных на JavaScript. У них есть определенный алгоритм работы:

1. Пользователь совершает действие — нажимает на кнопку «Зарегистрироваться».
2. Браузер фиксирует выполнение определенных условий — клик мыши в конкретной области на странице. За реакцию на действие отвечает команда onclick, которая запускает прописанный JS-код.
3. Затем запускается JS-код, который прописан для этих условий, — анимация нажатия кнопки и открытие формы регистрации.
4. Страница полностью или частично обновляется — в зависимости от настроек форма может открыться в этой же или в соседней вкладке.

## MySQL - свободная реляционная СУБД

MySQL [3] — это реляционная система управления базами данных (СУБД), которая распространяется как свободное программное обеспечение. Является одной из наиболее популярных, так как отличается гибкостью, легкостью, удобством в использовании.

Слово «реляционный» означает, что базы представлены в виде связанной информации и описываются как набор связей. MySQL работает с языком запросов SQL, который традиционно используется в базах данных.

MySQL имеет клиент-серверную архитектуру. Это означает, что база данных хранится в одном источнике — на сервере. А связываться с ней могут клиенты — сторонние устройства. Клиенты отправляют запросы к БД, а потом получают от сервера информацию.

Клиент-серверная архитектура делает хранение данных безопаснее: клиентские компьютеры не могут получить к ним бесконтрольный доступ. Им открывается только часть данных, которую можно получить по запросу. Вся информация находится на сервере, а клиенты не перегружены, поэтому им не нужны большие вычислительные мощности.

MySQL предоставляет множество функций для управления данными. Базовые действия можно выполнять с помощью команд.

* Создание базы данных — команда CREATE DATABASE.
* Вставка данных — команда INSERT.
* Обновление данных — команда UPDATE.
* Удаление данных — команда DELETE.
* Извлечение данных — команда SELECT.

Иногда продукты с открытым исходным кодом вызывают настороженность в части безопасности. Однако MySQL позволяет настраивать права доступа пользователей к БД, поддерживает шифрование данных, предоставляет возможности для мониторинга и логирования активности в БД. В совокупности это обеспечивает высокий уровень безопасности.

## Веб-приложение phpMyAdmin

phpMyAdmin [4] — [веб-приложение](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5" \o "Веб-приложение) с открытым кодом, написанное на языке [PHP](https://ru.wikipedia.org/wiki/PHP" \o "PHP) и представляющее собой веб-интерфейс для администрирования [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85" \o "Система управления базами данных) [MySQL](https://ru.wikipedia.org/wiki/MySQL" \o "MySQL). PhpMyAdmin позволяет через [браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80" \o "Браузер) и не только осуществлять администрирование сервера MySQL, запускать команды [SQL](https://ru.wikipedia.org/wiki/SQL" \o "SQL) и просматривать содержимое таблиц и баз данных. Приложение пользуется большой популярностью у веб-разработчиков, так как позволяет управлять СУБД MySQL без непосредственного ввода SQL команд.

Приложение многие разработчики интегрируют в свои разработки, например [XAMPP](https://ru.wikipedia.org/wiki/XAMPP" \o "XAMPP), [Denwer](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)" \o "Денвер (программа)), [AppServ](https://ru.wikipedia.org/w/index.php?title=AppServ&action=edit&redlink=1" \o "AppServ (страница отсутствует)), [Open Server](https://ru.wikipedia.org/w/index.php?title=Open_Server&action=edit&redlink=1" \o "Open Server (страница отсутствует)).

Использование phpMyAdmin — самый простой способ создавать, редактировать, удалять и скачивать/загружать таблицы с информацией на сервере. В приложении можно просмотреть содержимое баз, добавить новые, внести или изменить сведения, импортировать/экспортировать целые базы или отдельные таблицы. Если все же необходимо выполнить SQL-запрос, в PMA есть командная строка для их ввода.

Вид главной страницы представлен на рисунке 1 [1].

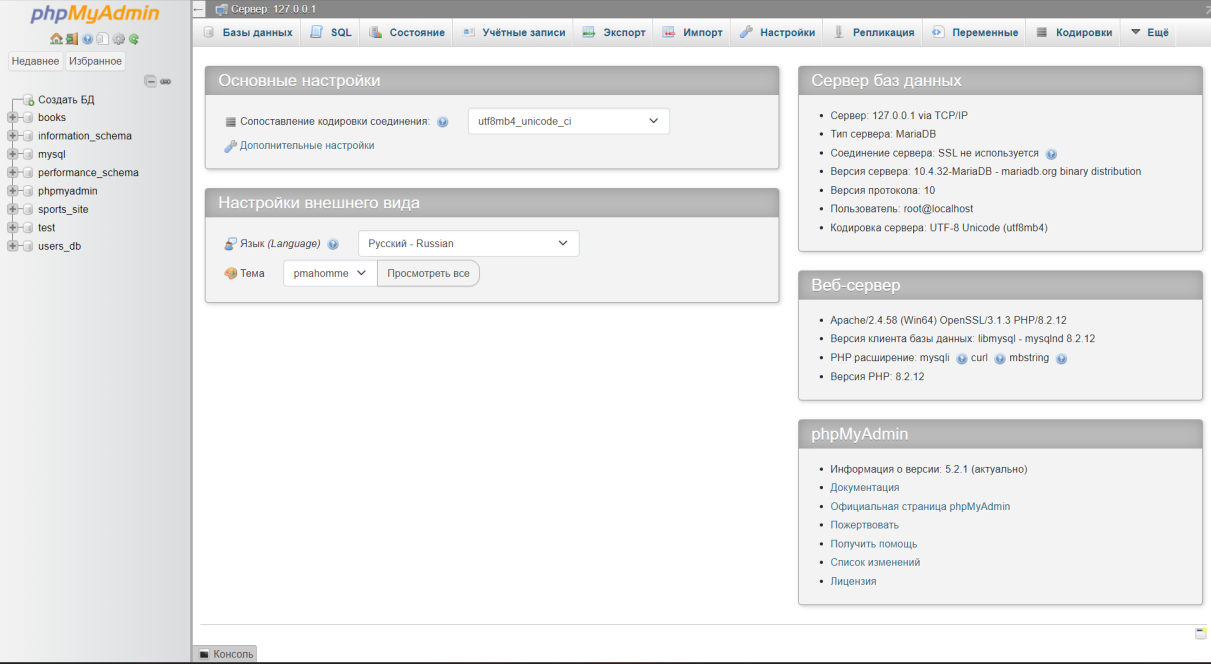


Рисунок 1 - Главная страница phpMyAdmin

## 2 Проектная часть

## 2.1 Создание и подключение базы данных

На phpMyAdmin создается база данных sports\_site.

SQL-код (файл db.sql) создает базу данных для управления пользователями, статьями и комментариями.

Таблица users хранит данные о пользователях (id, имя, пароль, email, роль, даты создания и обновления), где username и email уникальны, а роль по умолчанию — user. В таблицу users добавлены два пользователя: администратор и обычный пользователь.

Таблица articles хранит статьи (id, заголовок, содержание, id автора, изображение и дату создания), при этом author\_id является внешним ключом, ссылающимся на users.id, с каскадным удалением статей при удалении пользователя. Запрос выбирает заголовок, содержание, имя автора и дату создания статьи, объединяя таблицы articles и users по автору, с сортировкой по убыванию даты создания.

Таблица comments (рис. 2) включает поля: уникальный идентификатор комментария id, который автоматически увеличивается, идентификатор статьи article\_id, к которой относится комментарий, и идентификатор пользователя user\_id, оставившего комментарий; оба из которых не могут быть пустыми. Поле comment хранит текст комментария и не может быть пустым, а поле created\_at фиксирует время создания комментария по умолчанию на текущий момент. Также определены внешние ключи: FOREIGN KEY (article\_id) REFERENCES articles(id) ON DELETE CASCADE обеспечивает автоматическое удаление комментариев при удалении статьи, а FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE гарантирует удаление комментариев пользователей при удалении самих пользователей.

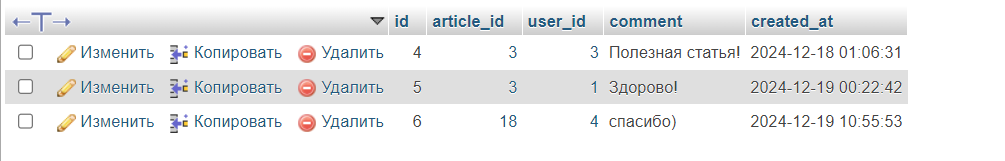


Рисунок 2 - Вид таблицы comments на phpMyAdmin

PHP-код (файл db.php) устанавливает соединение с базой данных MySQL с использованием PDO. Указываются параметры подключения: хост (localhost), имя базы данных (sports\_site), имя пользователя (root) и пароль (пустое значение).

С помощью конструктора new PDO создается объект для работы с базой данных, а строка DSN определяет тип базы данных и адрес сервера.

Устанавливается режим обработки ошибок PDO::ERRMODE\_EXCEPTION, чтобы исключения выбрасывались при возникновении ошибок, упрощая их отладку. Конструкция try-catch перехватывает ошибки подключения: если соединение не удается, блок catch выводит сообщение об ошибке, полученное через $e->getMessage(), и завершает выполнение скрипта. Код представлен на рисунке 3.

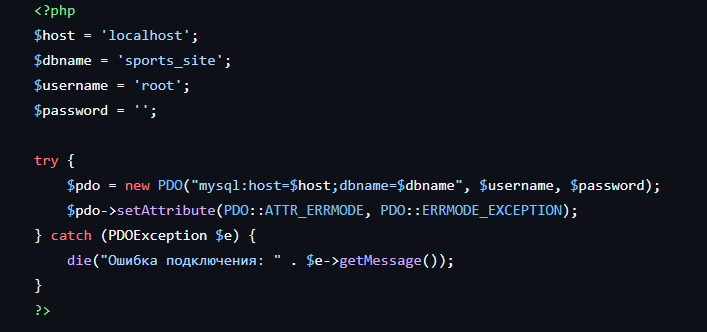


Рисунок 3 - Код из файла db.php

## **2.2 Создание главной страницы сайта**

Веб-страницу спортивного сайта с возможностью отображения статей и управления доступом на основе статуса пользователя представляет собой PHP-код (файл main.php). В начале подключается файл с настройками базы данных (db.php) и запускается сессия. Проверяется, авторизован ли пользователь ($isLoggedIn) и является ли он администратором ($isAdmin). Затем выполняется SQL-запрос для получения списка статей, которые сортируются по дате создания в порядке убывания, и результат сохраняется в массив $articles.

HTML-часть страницы включает заголовок, стили из внешнего файла style.css и приветственный текст. В меню слева отображаются ссылки для зарегистрированных пользователей на просмотр профиля и личные статьи, а для незарегистрированных — на вход и регистрацию. Если пользователь администратор, ему предоставляется доступ к админ-панели. Справа меню включает ссылку на контакт в ВКонтакте и кнопку выхода для авторизованных пользователей.

Список статей выводится в виде заголовка, изображения, содержания и даты публикации, причем текст статьи и заголовка фильтруется через htmlspecialchars, чтобы предотвратить внедрение вредоносного кода, а переносы строк обрабатываются с помощью nl2br. Под каждой статьей загружаются комментарии, связанные с ней, с указанием имени пользователя, оставившего комментарий, и времени его публикации. Если комментариев нет, отображается соответствующее сообщение.

В конце страницы размещена ссылка на рейтинг букмекеров и стандартный футер с копирайтом. Страница поддерживает адаптивность и базовую логику прав доступа.

Страница имеет адаптивный дизайн благодаря мета-тегу viewport (<meta name="viewport" content="width=device-width, initial-scale=1.0">), что делает ее удобной для просмотра на мобильных устройствах.

Вид главной страницы для не авторизированного (рис. 4) и авторизированного (рис. 5) пользователя представлены ниже.



Рисунок 4 - Главная страница (вид для не авторизированного пользователя)

Рисунок 5 - Вид главной страницы (вид для администратора)

## 2.3 Организация авторизации и регистрации пользователей

Для реализации простой формы авторизации используется код на PHP (файл login.php). Он подключается к базе данных через файл db.php, проверяет учетные данные пользователя (имя и пароль) через запрос к таблице users, используя PDO. Пароль сверяется с хешем, хранящимся в базе данных, с помощью password\_verify.

При успешной аутентификации данные пользователя сохраняются в сессии, и он перенаправляется на страницу main.php. Если авторизация не удалась, выводится сообщение об ошибке. Код также включает HTML-форму с полями ввода для имени пользователя и пароля, а также ссылки на страницы регистрации и главной страницы (рисунок 6).

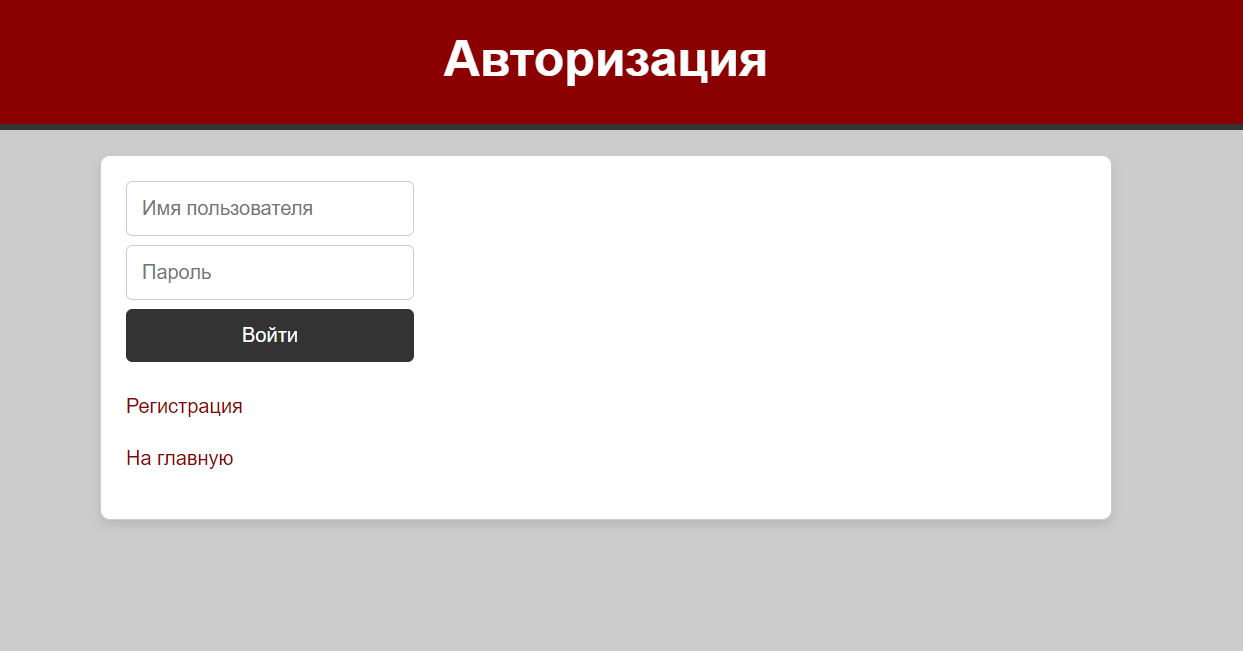


Рисунок 6 - Страница авторизации

Для реализации страницы регистрации пользователя используется код на PHP (register.php). При отправке формы пользовательские данные (имя, email, пароль) обрабатываются, пароль хэшируется с использованием PASSWORD\_BCRYPT, и затем данные сохраняются в таблице users базы данных через PDO.

После успешного добавления пользователь автоматически аутентифицируется: в сессию сохраняются его ID, имя и роль “user”. Затем происходит перенаправление на страницу main.php.

HTML-страница (рисунок 7) содержит форму для ввода имени, email и пароля, а также ссылки на страницы входа и главной страницы.

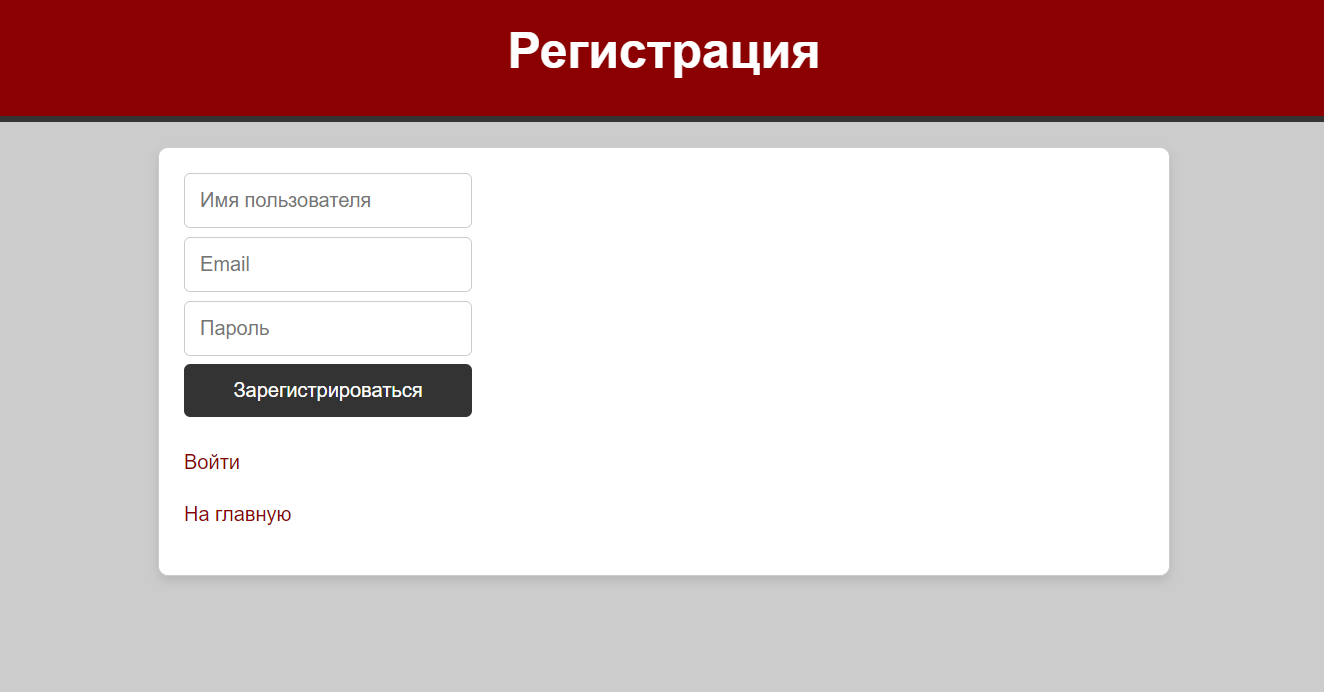


Рисунок 7 - Страница регистрации

Для выхода c сайта (завершения текущей пользовательской сессии) используется PHP-скрипт (logout.php) с помощью функции session\_destroy(). Происходит перенаправление пользователя на страницу “main.php” с помощью функции header(). Завершение сессии очищает все данные, связанные с ней, а команда exit гарантирует завершение выполнения скрипта сразу после перенаправления.

Код представлен на рисунке 8.

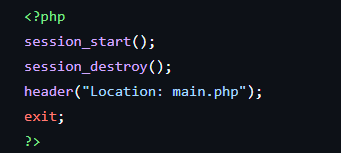


Рисунок 8 - Код из файла logout.php

## 2.4 Создание и редактирование личного кабинета пользователем

PHP-скрипт из файла profile.php реализует страницу профиля пользователя с проверкой авторизации и отображением данных из базы данных. Сначала запускается сессия, и проверяется наличие идентификатора пользователя в $\_SESSION. Если пользователь не авторизован, происходит перенаправление на страницу входа (login.php). Идентификатор пользователя используется для выполнения подготовленного SQL-запроса, который извлекает данные из таблицы users, включая имя, email, роль и дату регистрации. Если данные не найдены, выводится сообщение об ошибке.

На странице (рисунок 9) отображается информация о пользователе (имя пользователя, email, роль и дата регистрации) с использованием фильтрации через htmlspecialchars() для защиты от XSS-атак. Также предоставляются кнопка для перехода на страницу редактирования профиля (profile\_edit.php), ссылки для выхода из профиля (logout.php) и перехода на главную страницу (main.php). Для визуального оформления подключен файл стилей style.css. Таким образом, скрипт обеспечивает безопасное отображение данных пользователя и предоставляет базовые функции управления профилем.

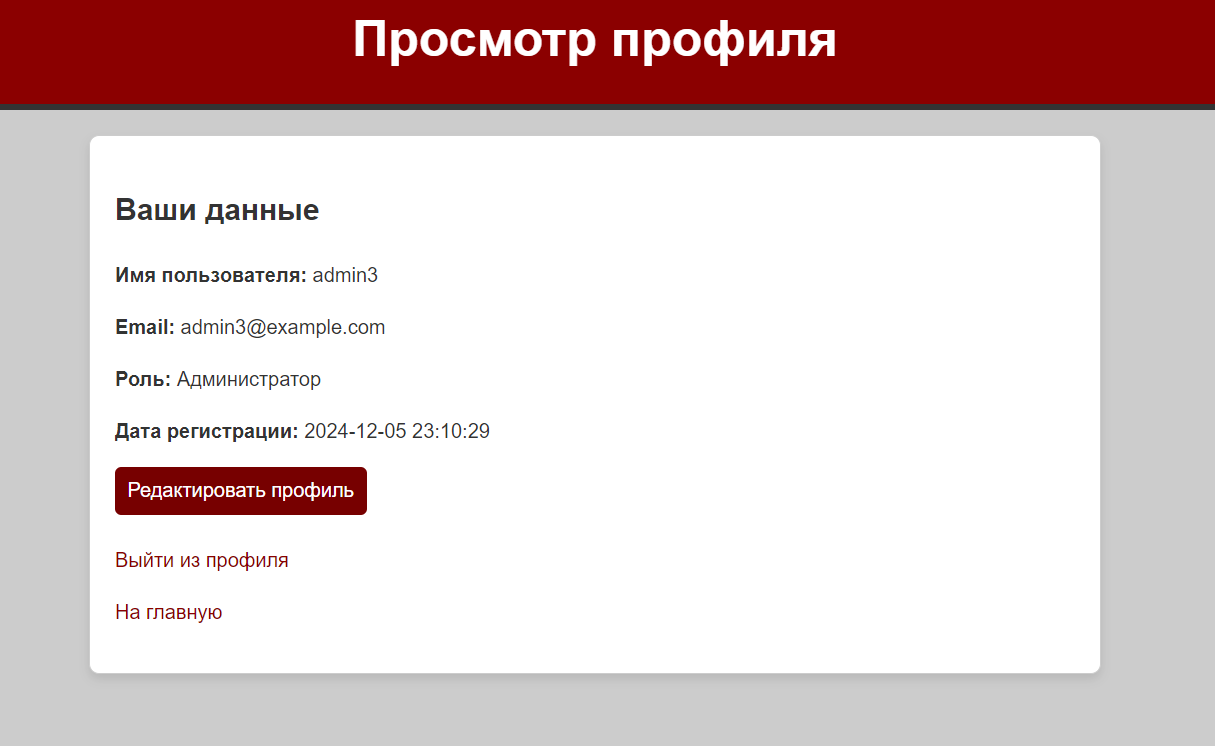


Рисунок 9 - Личный кабинет пользователя

PHP-скрипт из файла profile\_edit.php реализует страницу для изменения данных пользователя. После проверки авторизации через сессию данные текущего пользователя (имя и email) загружаются из базы данных. Пользователь может обновить имя, email и, при необходимости, изменить пароль. Скрипт проверяет, совпадают ли введенные пароли, и, в случае их совпадения, сохраняет изменения в базе данных. Пароль при этом хэшируется с использованием password\_hash.

На странице отображаются сообщения об успехе или ошибке, в зависимости от результата обработки данных. После обновления данных имя пользователя обновляется в сессии. Страница (рисунок 10) содержит удобную форму для ввода данных и ссылки для возврата в профиль или на главную страницу. Все данные фильтруются через htmlspecialchars() для защиты от XSS-атак.

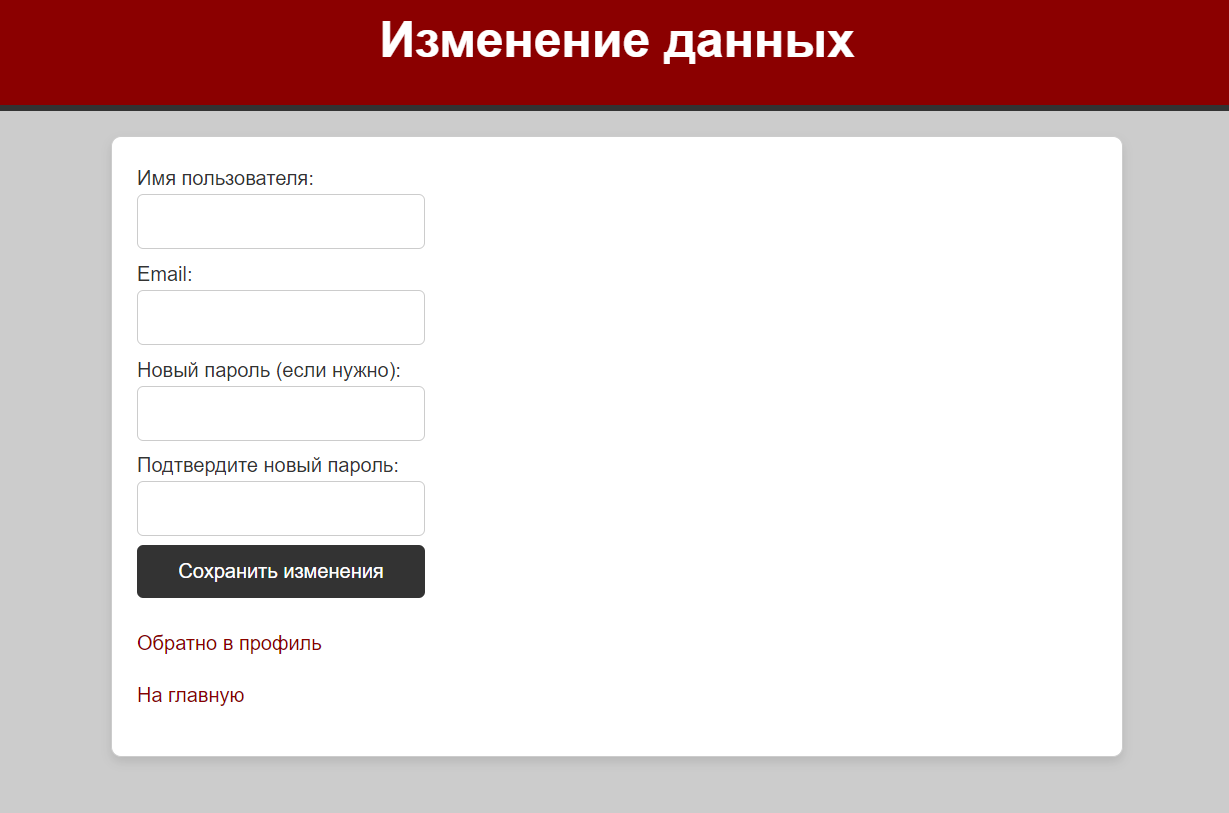


Рисунок 10 - Страница редактирования личных данных

## 2.5 Создание, добавление, редактирование и удаление статей

PHP-скрипт из файла personal\_articles.php предназначен для отображения страницы со списком статей, написанных текущим авторизованным пользователем. В начале сессия проверяется на наличие идентификатора пользователя ($\_SESSION['user\_id]), чтобы убедиться в его авторизации. Если пользователь не вошел в систему, он перенаправляется на страницу входа (login.php). Далее из базы данных с использованием PDO-запроса извлекаются все статьи, принадлежащие авторизованному пользователю, отсортированные по дате создания в порядке убывания.

На странице каждая статья отображается с заголовком, содержимым, изображением, датой публикации, а также кнопками для редактирования и удаления. Кнопка редактирования отправляет идентификатор статьи методом GET на страницу edit\_article.php, а кнопка удаления отправляет запрос методом POST на delete\_article.php с подтверждением перед удалением.

Если у пользователя нет статей, отображается соответствующее сообщение. Также предусмотрены ссылки для добавления новой статьи (create\_article.php) и перехода на главную страницу (main.php). Код защищен от XSS-атак с помощью htmlspecialchars и от SQL-инъекций за счет использования подготовленных выражений.

Пример вида страницы представлен на рисунке 11.

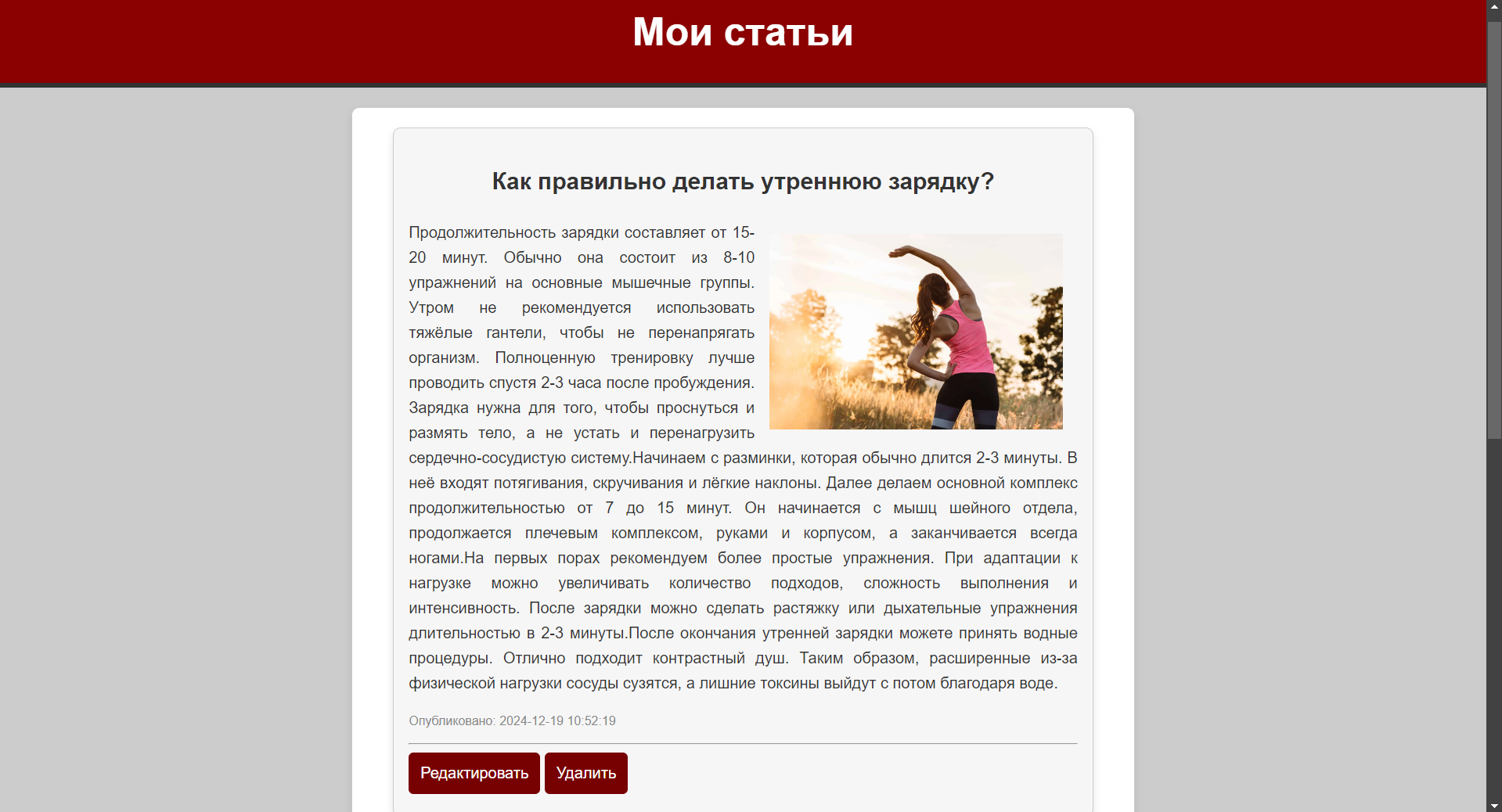


Рисунок 11 - Вид страницы «Мои статьи»

PHP-скрипт (файл edit\_article.php) предоставляет функциональность для редактирования статьи текущим авторизованным пользователем. Сначала выполняется проверка авторизации через сессию. Если пользователь не авторизован, он перенаправляется на страницу входа. Затем из параметра запроса id извлекается идентификатор статьи. Если идентификатор отсутствует или некорректен, пользователя перенаправляют на страницу со списком его статей с сообщением об ошибке.

Далее с использованием подготовленного запроса выполняется выборка статьи из базы данных, принадлежащей текущему пользователю. Если статья не найдена или доступ к ней запрещен, пользователь также перенаправляется на страницу со статьями с соответствующим сообщением.

Если запрос к странице выполнен методом POST, проверяется, заполнены ли поля заголовка и содержимого. Если поля не пусты, статья обновляется в базе данных с помощью подготовленного запроса, и пользователя перенаправляют на страницу со списком статей с сообщением об успешном обновлении. Если поля пусты, выводится сообщение об ошибке.

На странице (рисунок 12) отображается форма для редактирования статьи, в которой уже заполнены текущие данные статьи. Для защиты от XSS используется htmlspecialchars. После редактирования пользователь может сохранить изменения или вернуться к списку своих статей.

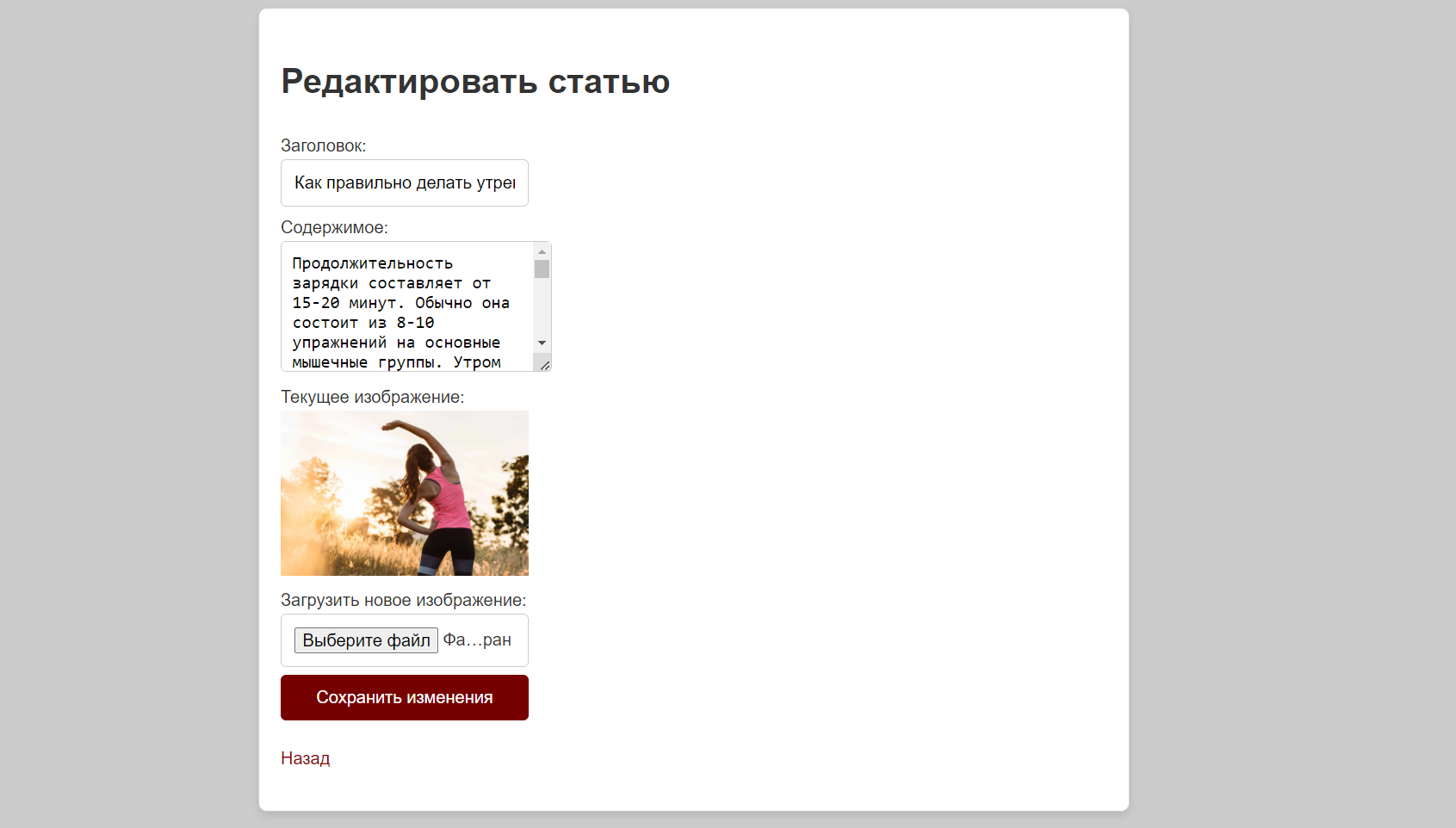


Рисунок 12 - Вид страницы для редактирования статьи

Для реализации добавления новых статей для авторизованных пользователей используется PHP-скрипт из файла create\_article.php. Сначала выполняется проверка авторизации: если пользователь не вошел в систему, он перенаправляется на страницу входа. После этого, если форма отправлена методом POST, из данных формы извлекаются и очищаются заголовок и содержимое статьи. Затем проверяется, что оба поля заполнены.

Если данные корректны, выполняется SQL-запрос для вставки новой статьи в базу данных. Каждая статья связывается с текущим пользователем через идентификатор автора, хранящийся в сессии. После успешного добавления статьи пользователь перенаправляется на главную страницу с сообщением об успехе. Если поля не заполнены, отображается сообщение об ошибке.

На странице (рисунок 13) выводится форма с полями для ввода заголовка и содержания статьи. Все данные, введенные пользователем, защищены от XSS-атак с помощью функции htmlspecialchars. Также предусмотрены ссылки для возвращения к списку своих статей или на главную страницу.

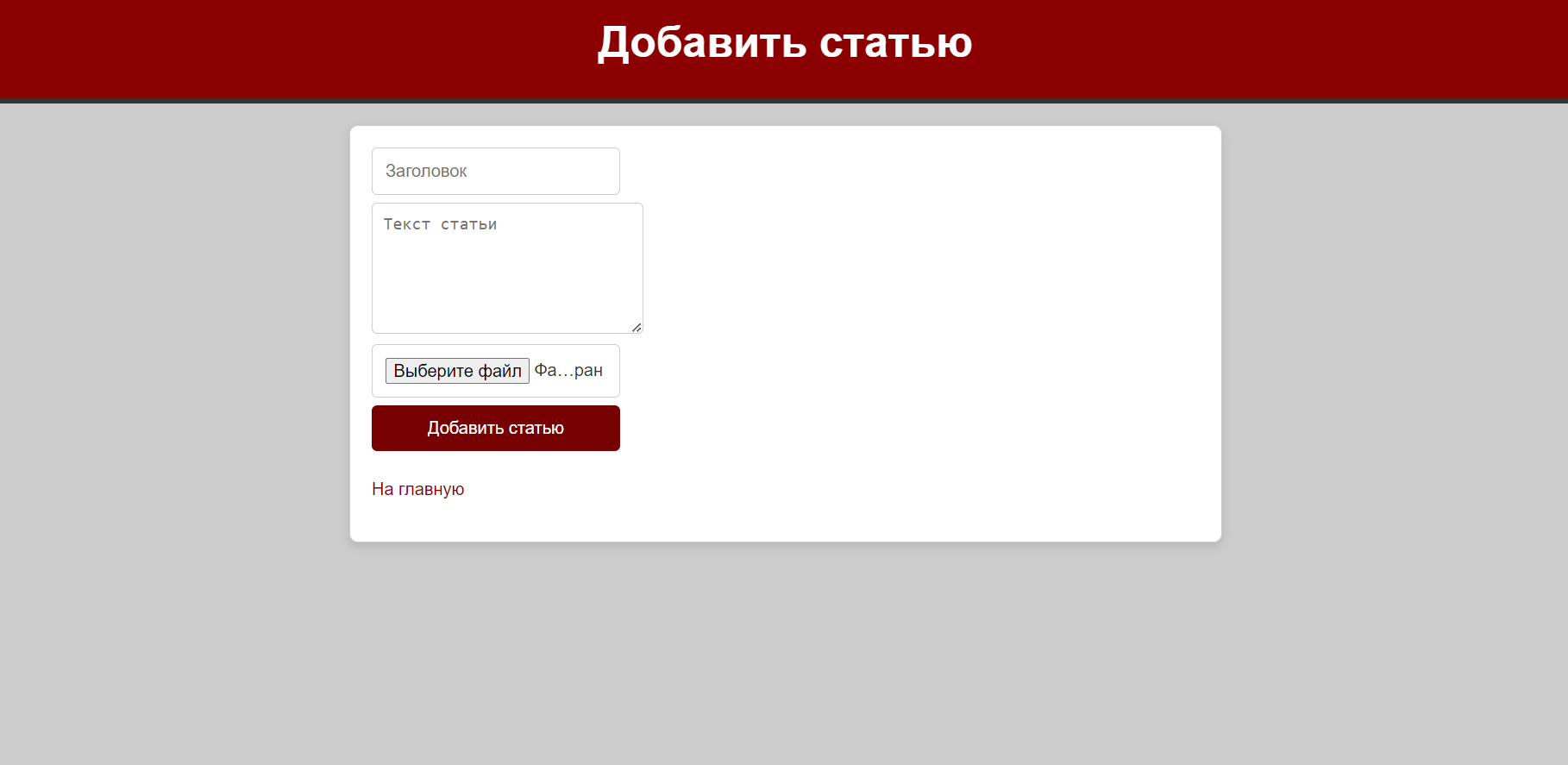


Рисунок 13 - Вид страницы для добавления статьи

PHP-скрипт из файла delete\_article.php отвечает за удаление статей, принадлежащих текущему авторизованному пользователю. В начале выполняется проверка авторизации: если пользователь не вошел в систему, он перенаправляется на страницу входа. Затем проверяется, был ли запрос отправлен методом POST и содержит ли он идентификатор статьи (article\_id).

Если данные корректны, выполняется подготовленный SQL-запрос, который удаляет статью с указанным идентификатором, принадлежащую текущему пользователю (идентификатор автора берется из сессии). После успешного удаления пользователь перенаправляется на страницу со списком его статей с сообщением об успешной операции.

Если запрос некорректен (например, отсутствует идентификатор статьи), пользователь также перенаправляется на страницу со списком статей, но с сообщением об ошибке.

Код представлен на рисунке 14.



Рисунок 14 - Код файла delete\_article.php

## 2.6 Организация админ-панели: просмотр, изменение, добавление и удаление пользователей

PHP-скрипт файла admin.php реализует административную панель для управления пользователями на сайте. Доступ к панели ограничен: проверяется, что пользователь авторизован и его роль равна admin. Если это не так, он перенаправляется на страницу входа.

После проверки авторизации выполняется запрос к базе данных для получения списка всех пользователей. Эти данные отображаются в таблице с такими полями, как ID, имя пользователя, email, роль и дата регистрации. Администратор может редактировать информацию о пользователе через ссылку на страницу редактирования или удалить пользователя, нажав соответствующую ссылку. При удалении выполняется подготовленный запрос для удаления пользователя по его идентификатору, и после успешного выполнения администратор возвращается на ту же страницу.

На странице (рисунок 15) также есть ссылка для добавления нового пользователя, ведущая на отдельный скрипт. Кроме того, предусмотрена ссылка для возврата на главную страницу сайта. Код защищает данные от XSS-атак с помощью htmlspecialchars и требует подтверждения перед удалением пользователя.

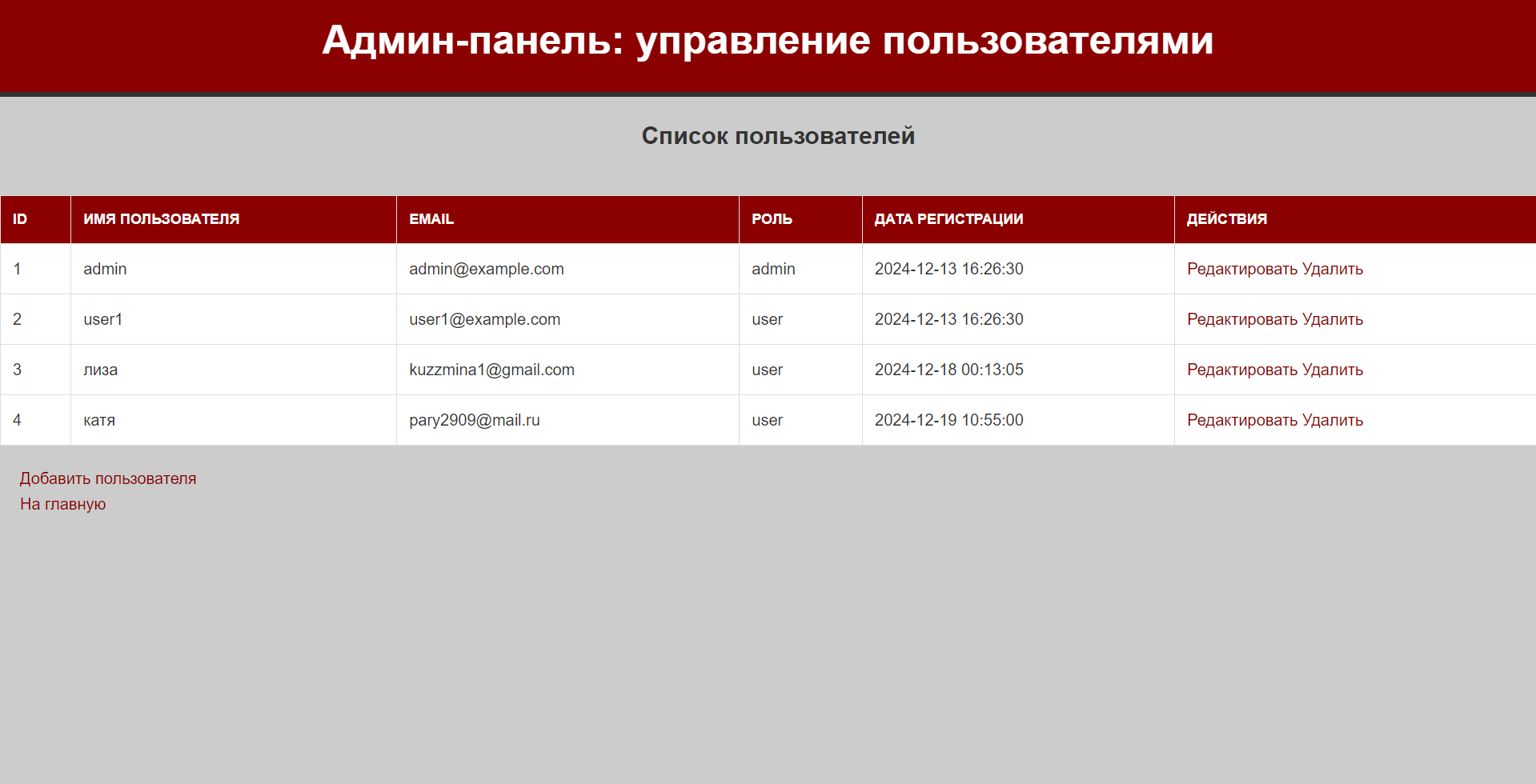


Рисунок 15 - Вид админ-панели

Для реализации редактирования данных пользователя в административной панели используется код на PHP (файл admin\_edit\_user.php). Доступ к странице ограничен: проверяется, что пользователь авторизован и его роль равна admin. Если проверка не пройдена, осуществляется перенаправление на страницу входа.

После авторизации из параметра запроса id извлекается идентификатор редактируемого пользователя. Если идентификатор отсутствует или пользователь с таким ID не найден в базе данных, администратор перенаправляется обратно на главную страницу панели администрирования.

На странице (рисунок 16) выводится форма с текущими данными пользователя, включая имя, email и роль. Если администратор вносит изменения и отправляет форму методом POST, данные обновляются в базе данных с использованием подготовленного запроса. После успешного обновления администратор перенаправляется на страницу со списком пользователей.

Форма защищена от XSS-атак с помощью htmlspecialchars для вывода данных, а поля с выбором роли (user или admin) заполняются на основе текущей роли пользователя. Также на странице есть ссылка для возврата в административную панель.

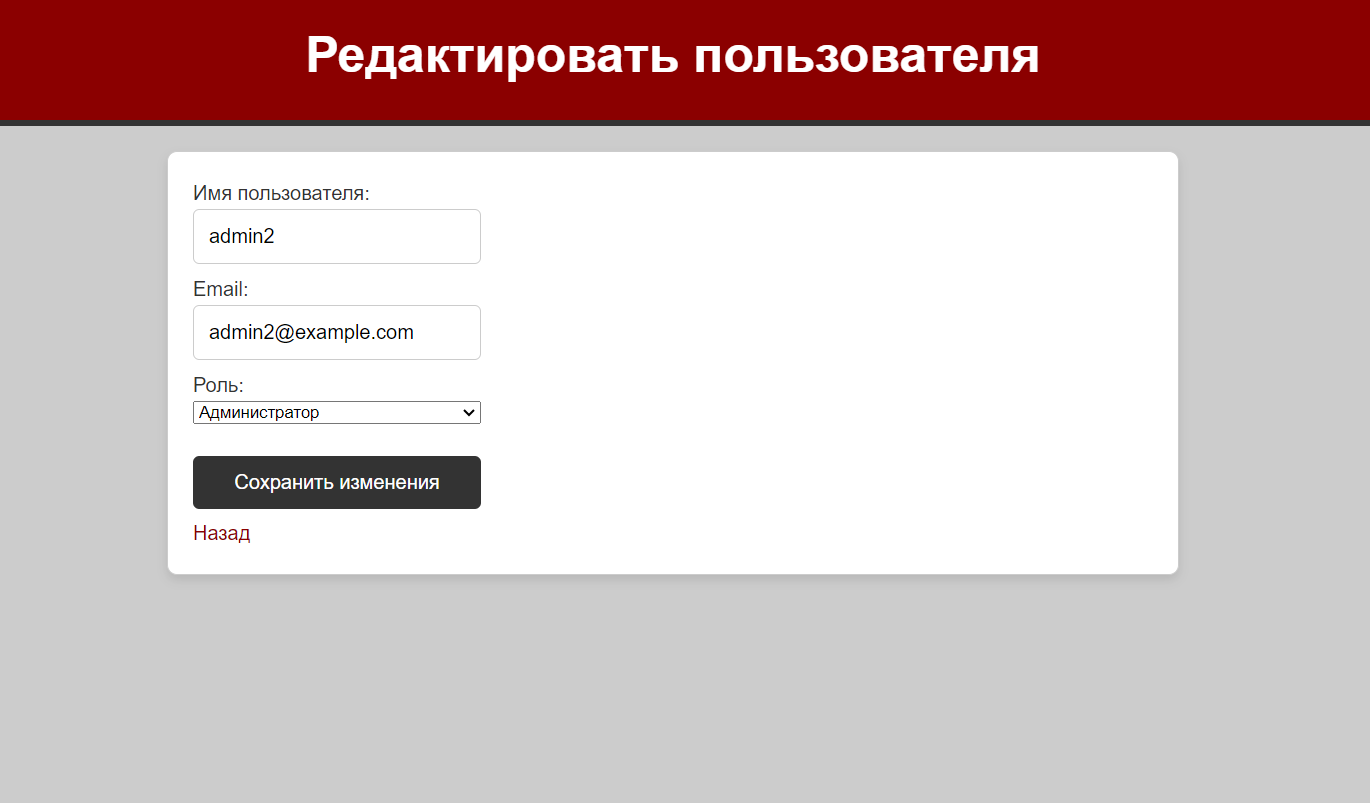


Рисунок 16 - Вид страницы редактирования пользователя

PHP-скрипт admin\_add\_user.php предоставляет административному пользователю возможность добавлять новых пользователей в систему. Доступ к странице ограничен: проверяется, что текущий пользователь авторизован и его роль равна admin. Если проверка не пройдена, происходит перенаправление на страницу входа.

На странице (рисунок 17) представлена форма, которая позволяет администратору указать имя пользователя, email, пароль и роль (user или admin). При отправке формы методом POST данные валидируются, после чего пароль хэшируется с использованием password\_hash для обеспечения безопасности. Затем выполняется SQL-запрос на добавление нового пользователя в базу данных.

После успешного выполнения операции администратор перенаправляется обратно на главную страницу панели администрирования. Форма защищена от XSS-атак за счет встроенных в HTML механизмов обработки, а также включает обязательные поля для предотвращения отправки пустых данных. На странице предусмотрена ссылка для возврата в административную панель.

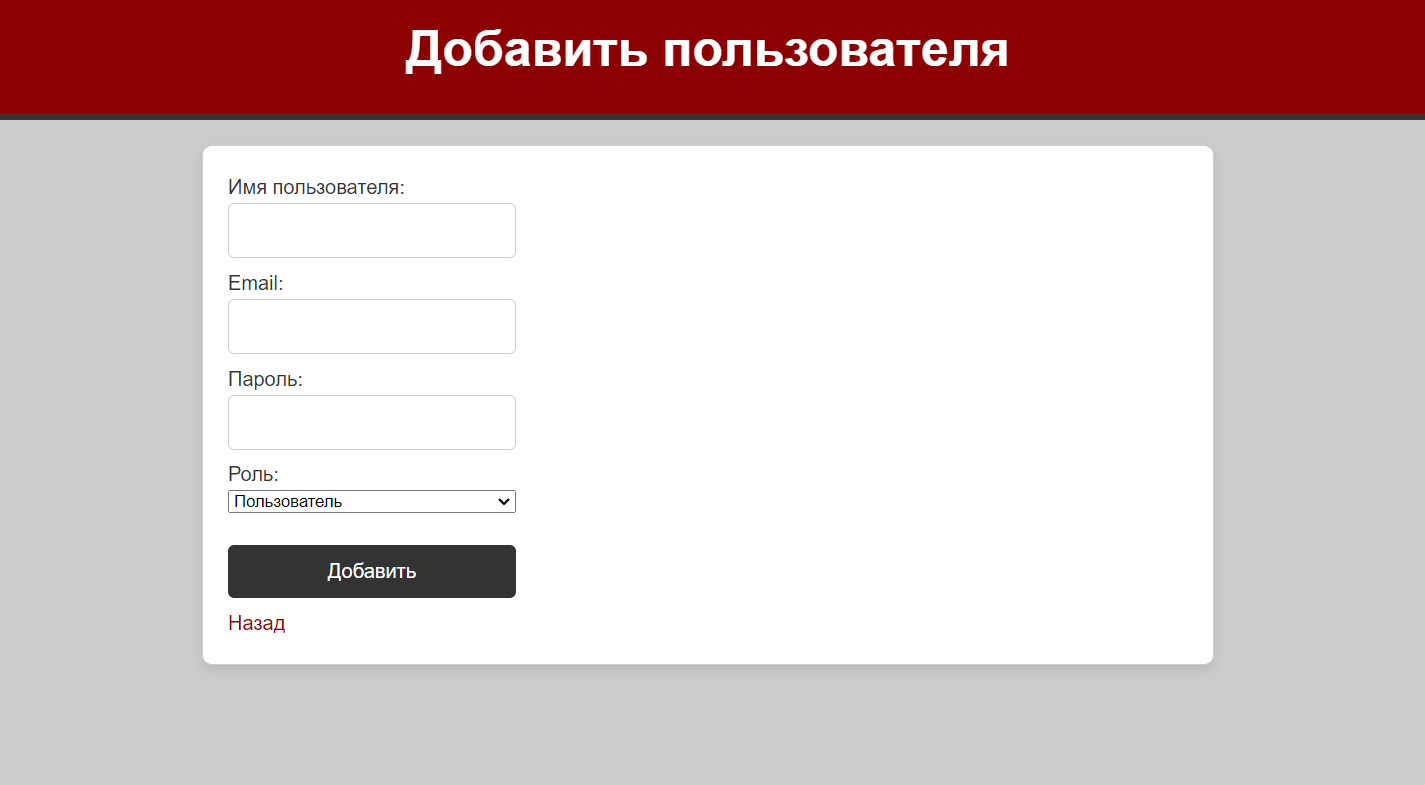


Рисунок 17 - Вид страницы добавления пользователя

**2.7 Комментарии к статье**

Фрагмент кода add\_comment.php представленный на рисунке 18 на PHP отвечает за обработку добавления комментариев к статьям. В начале кода подключается файл базы данных db.php и запускается сессия. Затем осуществляется проверка, авторизован ли пользователь: если идентификатор пользователя не задан в сессии, происходит перенаправление на страницу входа login.php.

Если метод запроса — POST, то осуществляется обработка данных из формы. Сначала извлекается идентификатор статьи, а затем содержимое комментария, введенное пользователем. Идентификатор пользователя берется из сессии. Если текст комментария не пустой, формируется SQL-запрос для вставки нового комментария в таблицу comments, в который передаются идентификатор статьи, идентификатор пользователя и текст комментария.

После успешного выполнения вставки выполняется перенаправление на страницу main.php, что завершает процесс добавления комментария.

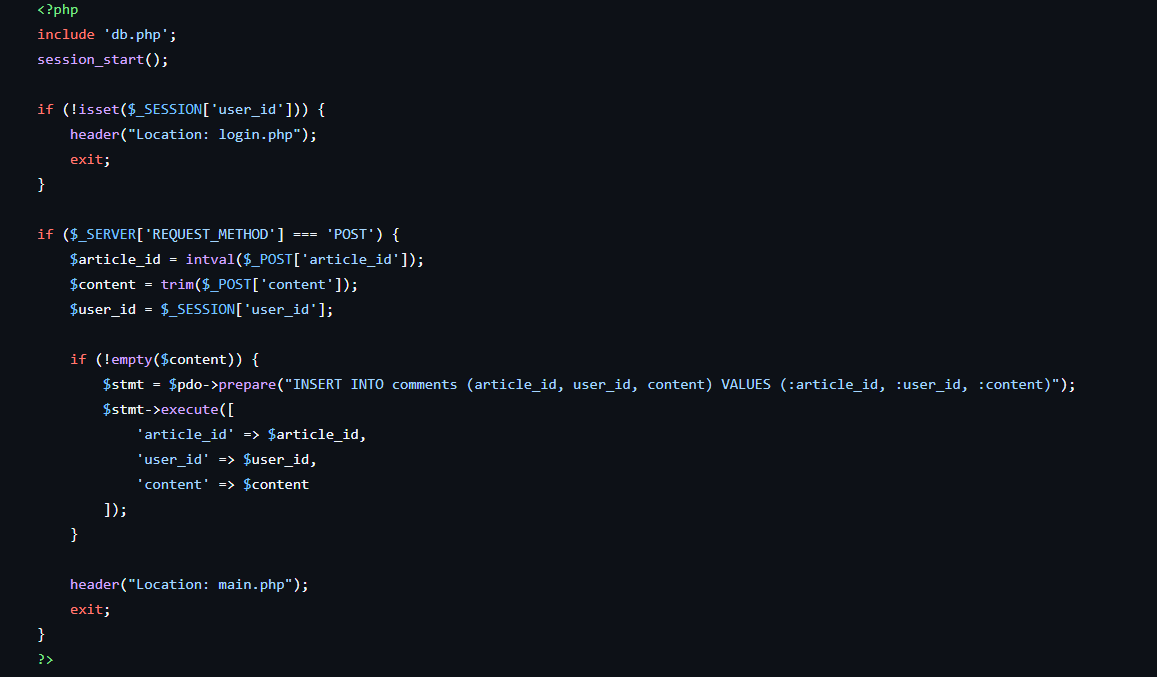


Рисунок 18 - Код из файла add\_comment.php

За реализацию добавления и отображения комментариев отвечает код из файла art.php. Он подключает базу данных и начинает сессию для отслеживания авторизованного пользователя.

Код получает ID статьи из URL и выполняет SQL-запрос для выборки статьи. Если статья не найдена, происходит перенаправление на главную страницу. Затем извлекаются комментарии к статье с именами пользователей через объединение таблиц.

При наличии POST-запроса и авторизации пользователя проверяется заполненность поля комментария. Если оно не пустое, комментарий добавляется в базу данных, иначе выводится сообщение с просьбой оставить комментарий.

Код также позволяет удалять комментарии: проверяется ID комментария и принадлежность к текущему пользователю. Если все условия выполнены, комментарий удаляется, иначе отображается ошибка.

В конце начинается HTML-разметка (результат на рисунке 19), где отображается заголовок и содержание статьи. Предоставляется форма для добавления комментариев только для авторизованных пользователей. Неавторизованным предлагается ссылка на страницу входа.

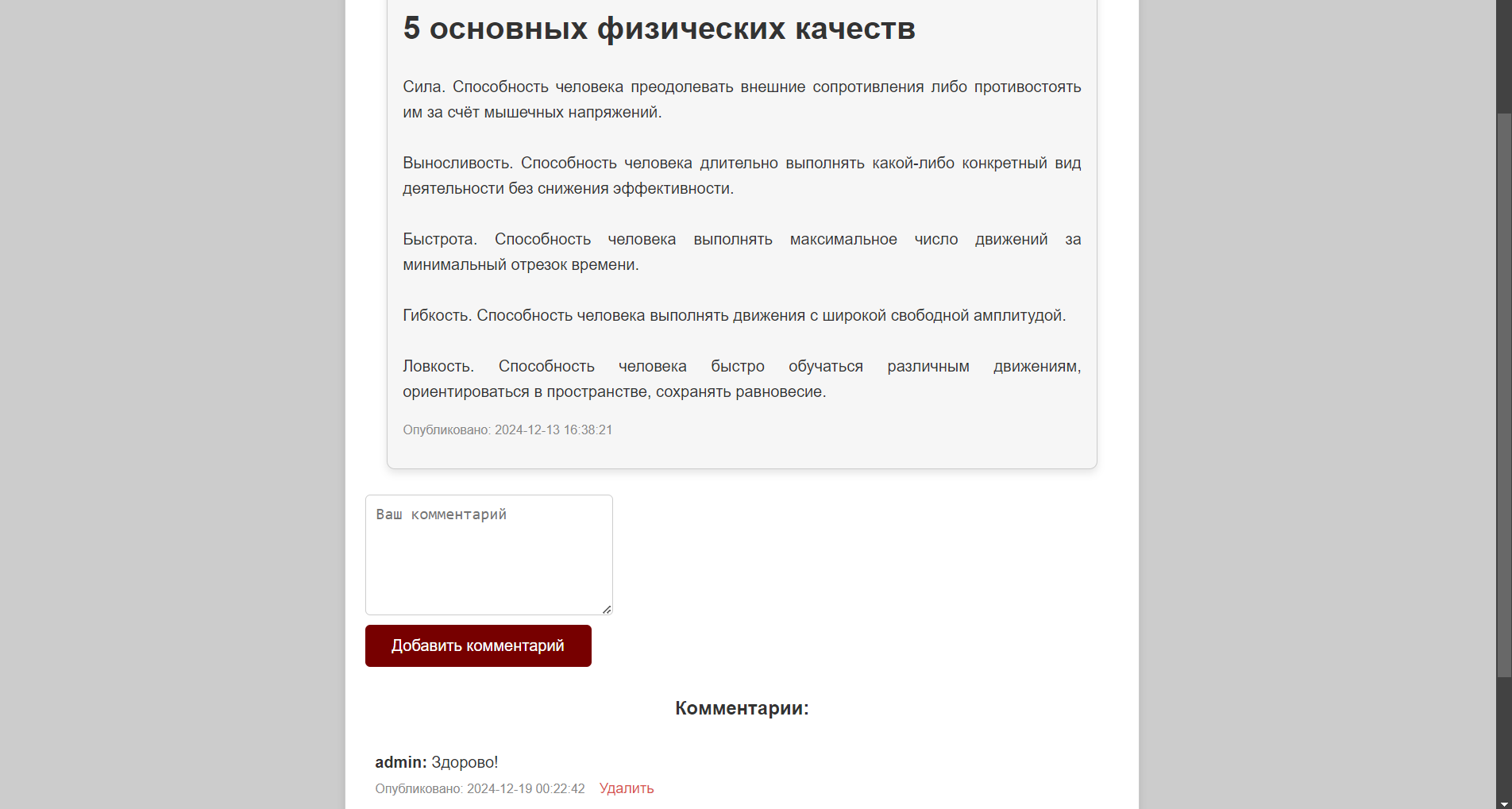


Рисунок 19 - Вид страницы конкретной статьи

## 2.8 Оформление сайта

Подключенный к проекту CSS- файл style.css добавляет сайту эстетичность и адаптивность. В том числе:

1. Базовые стили:

• Определяет шрифт Arial, фон #CCCCCC и стиль текста.

• Использует flexbox для выравнивания и адаптивности страницы.

2. Заголовок (header):

• Красный фон с белым текстом, центрирование и нижняя граница для акцента.

3. Контейнеры (.container, .container2):

• Центральное выравнивание, белый фон, тени, скруглённые углы.

4. Форма (form):

• Элементы формы с анимацией и изменением цвета/размера при фокусе или наведении.

5. Сообщения (.message):

• Стили для сообщений об успехе и ошибке с соответствующими цветами и обводкой.

6. Таблицы (table):

• Адаптивный дизайн: таблицы превращаются в блочные элементы на маленьких экранах с псевдоэлементами для обозначения заголовков.

7. Адаптивность:

• Для экранов шириной до 768px и 480px:

• Уменьшение шрифтов, отступов и размеров элементов.

• Преобразование таблиц в карточки.

8. Статьи (.article):

• Централизованный контент с тенью, обводкой и выравниванием текста.

9. Ссылки и кнопки:

• Анимация цвета и масштаба для улучшения UX при наведении.

Результаты оформления были представлены на рисунках раннее.

1. Меню:

* Создается горизонтальное меню с темным фоном, где элементы равномерно распределены по обе стороны.
* Элементы меню при наведении меняют цвет фона.

1. Комментарии:

* Комментарии оформлены в виде списка без маркировки, с отступами и границами между элементами.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была поставлена цель — разработать современное веб-приложение на тему «Спорт» с использованием технологий HTML, CSS, PHP и MySQL. Для достижения данной цели были выполнены все задачи, сформулированные на начальном этапе.

В результате разработки удалось создать функциональную и адаптивную платформу, которая предоставляет пользователям возможность:

• Регистрации и авторизации.

• Просмотра, добавления и комментирования статей спортивной тематики.

• Управления контентом (добавление, редактирование и удаление статей).

• Разделения пользователей на роли: администраторы и авторизованные пользователи.

Для реализации веб-приложения была спроектирована и создана база данных, которая обеспечивает хранение и управление информацией о пользователях, статьях и другой важной информации. Приложение также адаптировано для корректного отображения на устройствах с разными разрешениями экрана, что делает его доступным как для пользователей с компьютеров, так и для владельцев мобильных устройств.

Особое внимание было уделено безопасности, в частности, хэшированию паролей пользователей, предотвращению SQL-инъекций и других распространённых уязвимостей.

Практическая значимость проекта заключается в том, что созданное приложение может использоваться в качестве базовой платформы для спортивных сайтов или других тематических веб-ресурсов.

Таким образом, выполнение данной курсовой работы позволило не только углубить знания в области веб-разработки, но и приобрести практический опыт применения современных технологий, что является важным этапом в профессиональном становлении разработчика.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Веб-приложение phpMydmin // URL: <http://localhost/phpmyadmin/>
2. Что такое HTML и зачем он нужен // URL: <https://ru.hexlet.io/blog/posts/chto-takoe-html-i-zachem-on-nuzhen>
3. Что такое MySQL - как работает СУБД // URL: <https://selectel.ru/blog/mysql/>
4. Что такое phpMyAdmin простыми словами // URL: https://mchost.ru/articles/chto-takoe-phpmyadmin/
5. CSS: что это такое, основы языка разметки стилей // URL: <https://blog.skillfactory.ru/glossary/css/>
6. PHP: что это за язык программирования и зачем он нужен // URL: <https://blog.skillfactory.ru/glossary/php/>
7. JavaScript (JS): что это за язык программирования и зачем он нужен // URL: <https://blog.skillfactory.ru/glossary/javascript/>