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 «Психология».

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

подпись, дата

Студент УБ23-08Б, 452327077 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.И.Маклашевский подпись, дата

Красноярск 2024

**Содержание**

[**1 Теоретическая часть 5**](#_ih06gqd9fm2f)

[1.1 HTML 5](#_k5r1ksuf0r0i)

[1.2 CSS 6](#_fwbnelr7emxf)

[1.3 JavaScript 8](#_ibainj2ejpgo)

[1.4 PHP 10](#_8qj0gwa26szd)

[1.5 MySQL 12](#_1rdsxrbtrojt)

[**2 Практическая часть 15**](#_8o986d5moce)

[2.1 Требования к выполнению работы 15](#_dw10xf4gk72n)

[2.2 Создание формы регистрации и личного кабинета 17](#_pub345foilui)

**Введение**

В современном мире информационных технологий веб-приложения играют ключевую роль в предоставлении услуг и информации пользователям. В условиях глобальной цифровизации особое внимание уделяется разработке удобных, функциональных и эстетически привлекательных веб-сервисов, которые способны решать задачи пользователей в различных областях, включая психологическую поддержку.

Психология — это важная сфера, где доступность информации и профессиональных услуг может значительно улучшить качество жизни людей. Создание веб-приложения на эту тему позволяет соединить современные веб-технологии с потребностями аудитории, предоставляя возможность доступа к профессиональной информации, консультациям, а также другим сопутствующим услугам.

Цель данной курсовой работы — разработать веб-приложение, которое предоставляет пользователям функционал для взаимодействия с услугами в области психологии. Приложение должно быть интуитивно понятным, адаптивным, а также соответствовать современным стандартам веб-разработки.

В ходе работы реализовано приложение, включающее:

Личный кабинет пользователя, позволяющий управлять данными;

Функционал для администратора с возможностью управления услугами и пользователями;

Отображение списка услуг с детальным описанием и возможностью добавления новых;

Использование современных технологий веб-разработки, включая адаптивную верстку, анимации и удобный пользовательский интерфейс.

Объект : Веб-приложение на тему психологии, включающее функционал для взаимодействия пользователей с услугами, предоставляющими психологическую поддержку, и администрирования данного ресурса.

Предмет : Процесс разработки веб-приложения с использованием современных технологий веб-разработки, таких как HTML, CSS, PHP, JavaScript и базы данных MySQL.

## 1 Теоретическая часть

### 1.1 HTML

HTML (HyperText Markup Language) — это язык гипертекстовой разметки, который используется для создания структуры и содержания веб-страниц. HTML определяет, как должны отображаться тексты, изображения, ссылки, таблицы, формы и другие элементы на веб-странице.

Основой HTML являются элементы, которые описываются с помощью тегов. Например, тег <p> используется для создания абзаца, <a> — для гиперссылки, <img> — для изображений.

HTML не является языком программирования, так как он не позволяет выполнять вычисления или логику, а служит исключительно для описания структуры документа [3].

Плюсы HTML

1. Простота изучения и использования
   * HTML имеет простой синтаксис, который легко освоить даже начинающим разработчикам.
   * Не требует сложных инструментов для написания — достаточно текстового редактора.
2. Универсальность
   * HTML поддерживается всеми современными браузерами, что делает его универсальным стандартом для создания веб-страниц.
3. Гибкость при разработке
   * Легко интегрируется с CSS и JavaScript, что позволяет расширять функциональность и дизайн страниц.
   * Может использоваться совместно с другими технологиями, такими как PHP, Python или React.
4. Совместимость с SEO (поисковой оптимизацией)
   * HTML позволяет добавлять мета-теги, заголовки, семантические элементы и другие инструменты для улучшения индексации сайта в поисковых системах.
5. Бесплатность и открытость
   * HTML не требует лицензий или специальных программ для использования, так как это открытый стандарт, поддерживаемый консорциумом W3C (World Wide Web Consortium).

Недостатки HTML

1. Ограниченность функционала
   * HTML не позволяет создавать интерактивные элементы, такие как кнопки или анимации, без использования дополнительных технологий (CSS, JavaScript).
   * Отсутствует возможность выполнять сложную логику и обработку данных на сервере.
2. Необходимость использования дополнительных технологий
   * Для создания стильного и интерактивного веб-сайта требуется знание CSS для стилизации и JavaScript для динамики.
   * Работа с базами данных и серверной логикой невозможна только на HTML.
3. Сложность поддержки больших проектов
   * HTML сам по себе не обеспечивает модульности кода, что может затруднить поддержку и масштабирование крупных проектов.
4. Чувствительность к ошибкам
   * Неправильное использование тегов или атрибутов может привести к некорректному отображению страницы.

### 1.2 CSS

CSS (Cascading Style Sheets) — это язык каскадных таблиц стилей, используемый для описания внешнего вида и оформления веб-страниц, созданных на основе HTML. CSS управляет такими аспектами, как цвета, шрифты, отступы, размеры, размещение элементов, а также позволяет добавлять анимацию и адаптивность.

CSS действует как "слой стилей", отделяя визуальное оформление от структуры страницы. Это делает разработку более удобной и позволяет применять изменения стилей ко всему сайту, редактируя один файл [1].

Плюсы CSS

1. Разделение контента и стилей
   * CSS позволяет отделить оформление страницы от её структуры, что упрощает разработку и поддержку сайта.
2. Улучшение производительности
   * Один файл CSS может быть применён к нескольким страницам, что уменьшает объем кода и ускоряет загрузку сайта.
3. Удобство стилизации
   * CSS предоставляет мощные инструменты для настройки шрифтов, цветов, размеров, выравнивания и других визуальных аспектов.
4. Поддержка адаптивности
   * С помощью CSS можно создавать адаптивные веб-дизайны, которые автоматически подстраиваются под различные устройства и размеры экранов.
5. Поддержка анимаций и переходов
   * CSS позволяет создавать простые анимации и плавные переходы, улучшая пользовательский опыт без использования JavaScript.
6. Широкая поддержка браузерами
   * CSS поддерживается всеми основными веб-браузерами, что делает его универсальным инструментом.

Недостатки CSS

1. Сложность при масштабировании
   * В крупных проектах стили могут стать сложными для поддержки, если код плохо организован или стили написаны неструктурированно.
2. Ограниченные возможности
   * CSS не предназначен для обработки логики или работы с данными, поэтому он не может заменить функционал JavaScript или серверных технологий.
3. Браузерная несовместимость
   * Несмотря на широкую поддержку, некоторые свойства CSS могут отображаться по-разному в разных браузерах, особенно в устаревших версиях.
4. Зависимость от структуры HTML
   * Эффективное использование CSS зависит от качественно написанной HTML-разметки. Плохая структура HTML может затруднить стилизацию.
5. Чувствительность к порядку и каскадности
   * Из-за каскадной природы CSS неосторожное использование селекторов может привести к конфликтам между стилями.

### 1.3 JavaScript

JavaScript (JS) — это язык программирования, который используется для добавления интерактивности и динамического поведения веб-страницам. Он работает на стороне клиента (в веб-браузере) и позволяет создавать интерактивные элементы, такие как всплывающие окна, валидация форм, анимации, динамическая загрузка данных и многое другое.

JavaScript тесно интегрирован с HTML и CSS, что делает его основным инструментом для создания современных веб-приложений. С помощью JavaScript можно управлять DOM (Document Object Model) — структурой HTML-документа, что позволяет изменять элементы страницы без её перезагрузки [2].

Плюсы JavaScript

1. Интерактивность
   * Позволяет создавать интерактивные элементы, такие как выпадающие меню, слайдеры, формы с проверкой ввода.
2. Работа на стороне клиента
   * JavaScript выполняется в браузере пользователя, что снижает нагрузку на сервер и ускоряет работу приложения.
3. Гибкость и мощность
   * JavaScript может работать как с простыми веб-страницами, так и с крупными веб-приложениями, включая одностраничные приложения (SPA).
4. Широкая экосистема
   * Существуют многочисленные библиотеки и фреймворки, такие как React, Angular, Vue.js, которые упрощают разработку и ускоряют реализацию сложных проектов.
5. Анимации и динамика
   * С помощью JavaScript можно создавать сложные анимации, обновлять контент на странице в реальном времени (без перезагрузки страницы) и работать с API.
6. Кроссплатформенность
   * JavaScript поддерживается всеми современными веб-браузерами, что делает его доступным для любой аудитории.

Недостатки JavaScript

1. Безопасность
   * Код JavaScript выполняется на стороне клиента, поэтому его легко увидеть и изменить. Это может привести к потенциальным уязвимостям, если не предпринимаются меры безопасности.
2. Зависимость от браузера
   * Некоторые функции JavaScript могут работать по-разному в разных браузерах, особенно в устаревших версиях.
3. Сложность в больших проектах
   * При увеличении объёма кода JavaScript становится сложным для поддержки, если не использовать современные подходы (например, модули или фреймворки).
4. Отсутствие строгой типизации
   * До появления TypeScript (надстройки над JavaScript) работа с типами данных могла приводить к неожиданным ошибкам.
5. Перегрузка клиента
   * Сложные скрипты или плохо оптимизированный код могут замедлить работу браузера, особенно на устройствах с низкой производительностью.

### 1.4 PHP

PHP (Hypertext Preprocessor) — это серверный язык программирования, предназначенный для создания динамических веб-страниц и веб-приложений. PHP работает на стороне сервера, генерируя HTML-код, который отправляется в браузер пользователя.

PHP широко используется для управления базами данных, обработки форм, работы с файлами и выполнения других серверных задач. Он поддерживает интеграцию с различными базами данных (например, MySQL, PostgreSQL), что делает его идеальным выбором для разработки интерактивных веб-сайтов [5].

Плюсы PHP

1. Простота и доступность
   * PHP легко изучить, что делает его хорошим выбором для начинающих разработчиков.
   * PHP бесплатно распространяется, что позволяет создавать проекты без лишних затрат.
2. Широкая поддержка баз данных
   * PHP поддерживает множество баз данных, включая MySQL, PostgreSQL, SQLite, что упрощает создание динамических приложений.
3. Кроссплатформенность
   * PHP работает на различных операционных системах, включая Windows, Linux, macOS.
4. Широкая экосистема и поддержка
   * PHP имеет огромное количество готовых решений: фреймворки (например, Laravel, Symfony), системы управления контентом (CMS), такие как WordPress, Drupal, Joomla.
   * Обширная документация и активное сообщество разработчиков.
5. Динамическое создание страниц
   * PHP позволяет генерировать HTML-код на основе входных данных, например, показывать разные страницы для разных пользователей.
6. Интеграция с HTML, CSS и JavaScript
   * PHP легко интегрируется с другими веб-технологиями, что делает его универсальным инструментом для создания современных сайтов.

Недостатки PHP

1. Безопасность
   * Плохое написание кода может привести к уязвимостям, например, SQL-инъекциям, межсайтовым скриптам (XSS). Это требует дополнительных усилий по защите данных.
2. Проблемы с производительностью
   * PHP может быть менее производительным по сравнению с другими серверными языками, такими как Python или Node.js, особенно при обработке большого количества данных.
3. Сложность масштабирования
   * PHP менее эффективен для крупных проектов, если не использовать современные фреймворки или микросервисную архитектуру.
4. Исторические ограничения
   * Из-за долгой истории языка некоторые устаревшие функции и подходы (вроде глобальных переменных) всё ещё могут встречаться в коде, что усложняет поддержку.
5. Конкуренция с другими языками
   * Современные альтернативы, такие как Python, Ruby или JavaScript (Node.js), часто воспринимаются как более удобные или гибкие в определённых задачах.

### 1.5 MySQL

MySQL — это система управления реляционными базами данных (RDBMS), которая используется для хранения, управления и работы с данными. Она основана на языке SQL (Structured Query Language) и является одной из самых популярных СУБД благодаря своей скорости, надёжности и простоте использования.

MySQL чаще всего используется в веб-разработке для хранения данных, таких как информация о пользователях, публикациях, заказах или любых других объектах приложения. Она хорошо интегрируется с языками программирования, такими как PHP, Python, Java и другими [4].

Плюсы MySQL

1. Простота использования
   * MySQL имеет интуитивно понятный синтаксис SQL, который легко изучить и использовать.
2. Высокая производительность
   * MySQL оптимизирована для быстрого выполнения запросов, что делает её подходящей для обработки больших объёмов данных.
3. Кроссплатформенность
   * Работает на различных операционных системах, включая Windows, macOS и Linux.
4. Бесплатность
   * MySQL доступна в версии с открытым исходным кодом, что делает её бесплатной для большинства проектов.
5. Масштабируемость
   * Подходит как для небольших приложений, так и для крупных проектов с миллионами записей.
6. Широкая поддержка
   * MySQL интегрируется с различными языками программирования, такими как PHP, Python, Java, а также с популярными веб-фреймворками и CMS (например, WordPress, Joomla).
7. Безопасность
   * MySQL предоставляет механизмы аутентификации пользователей, шифрования данных и управления правами доступа.

Недостатки MySQL

1. Ограниченные функции в бесплатной версии
   * В версии Community отсутствуют некоторые функции, доступные в коммерческих редакциях (например, репликация на уровне предприятия).
2. Меньшая гибкость в сравнении с другими СУБД
   * Некоторые конкуренты, такие как PostgreSQL, предлагают больше функций (например, работа с массивами, расширенные возможности JSON).
3. Трудности при масштабировании
   * Хотя MySQL хорошо работает с крупными базами данных, для горизонтального масштабирования могут потребоваться дополнительные инструменты и настройки.
4. Сложности с большими данными
   * При обработке огромных объемов данных производительность MySQL может снизиться без оптимизации запросов и индексации.
5. Ограниченные возможности работы с NoSQL
   * Хотя MySQL поддерживает JSON, это не делает её полной альтернативой NoSQL-базам данных, таким как MongoDB или Cassandra.

## 2 Практическая часть

### 2.1 Требования к выполнению работы

Перед тем, как приступить к разработке web-приложения, необходимо разобраться с тем, что от нас требуется. К данной работе предъявлено 8 требований:

1) Должна быть предусмотрена авторизация и регистрация пользователей.

2) Зарегистрированный пользователь имеет право редактировать свои данные и просматривать свои услуги.

3) Администраторы имеют право редактировать пользователей и добавлять услуги.

4) Неавторизированные пользователи имеют право просматривать список услуг.

5) Сайт должен быть выдержан в определенном стиле и корректно открываться на любом разрешении.

6) Общий функционал может быть продуман самостоятельно.

7) Важно продемонстрировать не большой объем функционала, а минимальные требования к проекту. Проект может максимально простым, но при этом содержать все необходимые функции.

8) Отдельные бонусы предусмотрены за доп. функционал и красивый дизайн.

После ознакомления с требованиями к работе, необходимо настроить базовую конфигурацию для корректной работы приложения. Для этого воспользуемся сборкой локального веб-сервера XAMPP. После скачивания файла с официального сайта и его установки, все необходимые программы будут загружены. После установки мы открываем приложение XAMPP и запускаем модули “Apache” и “MySQL”. Далее необходимо открыть локальную папку с файлами и работать с программой по дистрибутиву xampp/htdocs.

Далее необходимо создать базу данных на платформе phpmyadmin. Для этого создадим файл database(db).sql, в которой кратко распишем, что будет находится в базе данных.

| CREATE DATABASE user\_management;   USE user\_management;  CREATE TABLE users ( id INT AUTO\_INCREMENT PRIMARY KEY, username VARCHAR(50) NOT NULL UNIQUE,  password VARCHAR(255) NOT NULL,  role ENUM('admin', 'user') DEFAULT 'user',  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP );  CREATE TABLE services (  id INT AUTO\_INCREMENT PRIMARY KEY,  title VARCHAR(100) NOT NULL,  description TEXT NOT NULL,  price DECIMAL(10, 2) NOT NULL,  image VARCHAR(255),  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ); |
| --- |

После импортируем данную базу данных в phpmyadmin. На рисунке 1 видно две таблицы: users и servises.

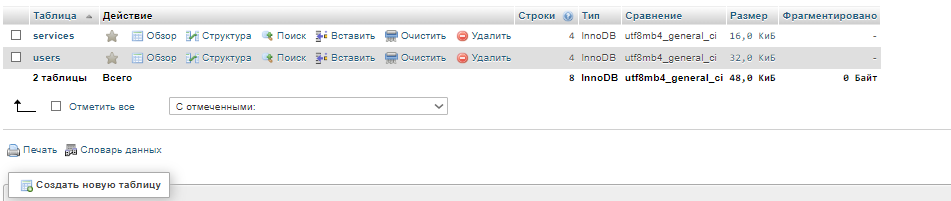


Рисунок 1 - Таблицы в базе данных phpmyadmin.

Все требования проекта были изучены и была создана база данных. Теперь можно приступить к созданию сайта.

### 2.2 Создание формы регистрации и личного кабинета

Цель раздела: Создание страницы регистрации, где новые пользователи могут создать учетную запись, а их данные будут сохранены в базе данных.

Процесс разработки:

* Backend: Использовался язык PHP для обработки данных формы и их последующего занесения в базу данных MySQL. При отправке формы проверяется уникальность имени пользователя. Пароль шифруется с использованием функции password\_hash для обеспечения безопасности.
* Frontend: Для визуализации формы использовался HTML и подключены стили из файла main.css. Визуальное оформление формы включает поля для ввода имени пользователя и пароля, а также кнопку отправки данных.

Листинг кода PHP для обработки формы:

| <?php include 'db.php';  if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  $username = $\_POST['username'];  $password = password\_hash($\_POST['password'], PASSWORD\_BCRYPT);   $stmt = $pdo->prepare("INSERT INTO users (username, password) VALUES (:username, :password)");  $stmt->execute(['username' => $username, 'password' => $password]);   header('Location: login.php');  exit; } ?> |
| --- |

Этот код обеспечивает добавление новых записей в таблицу users, которая включает столбцы для имени пользователя и зашифрованного пароля.

Frontend форма регистрации: Форма регистрации визуализируется следующим HTML-кодом, ее внешний вид изображен на рисунке 2:

| <form method="post">  <label for="username">Логин</label>  <input type="text" name="username" id="username" required>    <label for="password">Пароль</label>  <input type="password" name="password" id="password" required>    <input type="submit" value="Зарегистрироваться"> </form> |
| --- |

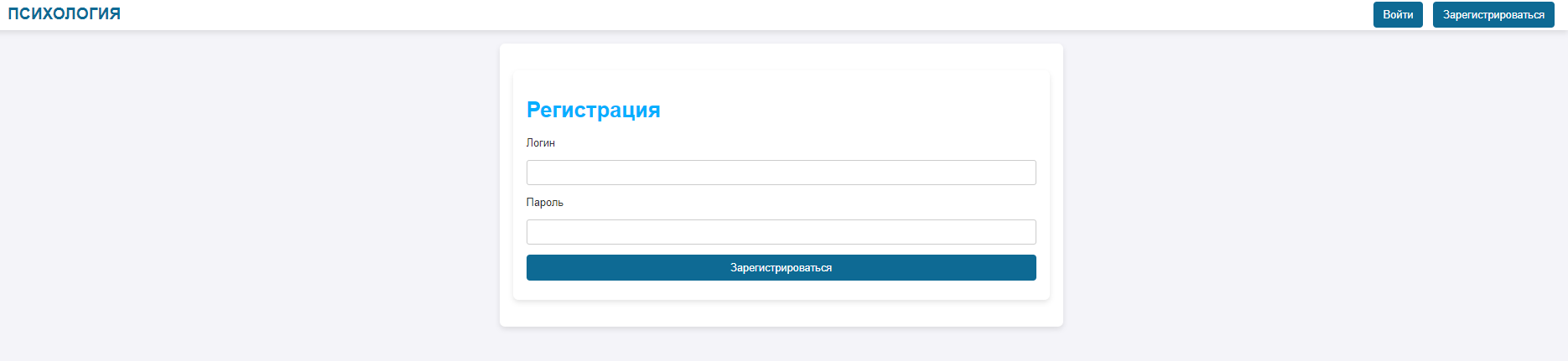


Рисунок 2 - Страница регистрации

Результаты выполнения:

* Успешная регистрация пользователя вызывает редирект на страницу входа.
* В случае дублирования имени пользователя скрипт возвращает сообщение об ошибке (добавление функционала планируется в следующих разделах).

Функционал входа в систему

Форма входа предоставляет пользователям возможность ввести свои данные (логин и пароль) для доступа к личному кабинету.

Описание работы скрипта входа

1. Подключение к базе данных:  
   В начале скрипта происходит подключение к базе данных через файл db.php, обеспечивающий доступ к таблице пользователей.

Обработка данных формы:  
Когда пользователь отправляет форму, сервер проверяет метод запроса (POST) и данные, переданные из формы:

| if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  $action = $\_POST['action'];  $username = $\_POST['username'];  $password = $\_POST['password']; } |
| --- |

1. Переменная $action определяет, выполняется ли вход.

Проверка данных пользователя:  
С помощью подготовленного SQL-запроса система проверяет наличие логина в базе данных:

| $stmt = $pdo->prepare("SELECT \* FROM users WHERE username = :username"); $stmt->execute(['username' => $username]); $user = $stmt->fetch(); |
| --- |

* + Если пользователь найден, то введённый пароль проверяется функцией password\_verify, чтобы убедиться, что он совпадает с зашифрованным паролем в базе данных.

Успешная авторизация:  
При успешной проверке пользователь сохраняется в сессии, и выполняется перенаправление в личный кабинет:

| $\_SESSION['user'] = $user; header('Location: dashboard.php'); exit; |
| --- |

* + Если данные неверны, пользователю возвращается сообщение об ошибке.

HTML-форма предоставляет интерфейс для ввода логина и пароля, ее внешний вид изображен на рисунке 3:

| <form method="post">  <input type="hidden" name="action" value="login">  <label for="login-username">Логин</label>  <input type="text" name="username" id="login-username" required>   <label for="login-password">Пароль</label>  <input type="password" name="password" id="login-password" required>   <input type="submit" value="Войти"> </form> |
| --- |

Результаты выполнения:

* Успешный вход перенаправляет пользователя на страницу личного кабинета.
* В случае ошибки отображается сообщение об ошибке.

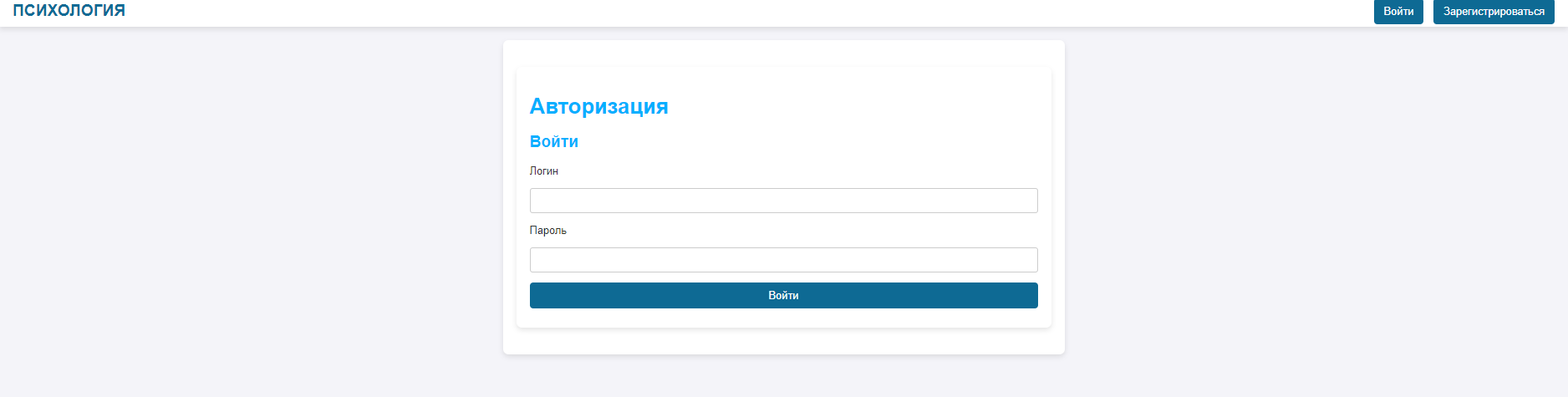


Рисунок 3 - Страница входа в аккаунт

Личный кабинет.

Личный кабинет предоставляет доступ к информации о пользователе, включая его имя и роль.

Описание работы скрипта личного кабинета:

Проверка авторизации:  
Если пользователь не вошёл в систему, он перенаправляется на страницу авторизации:

| if (!isset($\_SESSION['user'])) {  header('Location: login.php');  exit; } |
| --- |

Отображение данных пользователя:  
После успешной авторизации в личном кабинете выводится имя пользователя и его роль:

| <p>Добро пожаловать, <strong><?= htmlspecialchars($user['username']) ?></strong></p> <p>Роль: <?= htmlspecialchars($user['role']) ?></p> |
| --- |

Переход в админ-панель (для администраторов):  
Если роль пользователя — admin, в личном кабинете появляется ссылка для перехода в панель администратора:

| <?php if ($user['role'] === 'admin'): ?>  <p><a href="admin.php">Перейти в админ-панель</a></p> <?php endif; ?> |
| --- |

Кнопки для действий:  
Пользователь может редактировать свой профиль или выйти из системы:

| <a href="edit.php">Редактировать профиль</a> | <a href="logout.php">Выйти</a> |
| --- |

Пример интерфейса личного кабинета изображен на рисунке 4.

HTML-структура личного кабинета:

| <div class="container">  <h1>Личный кабинет</h1>  <p>Добро пожаловать, <strong><?= htmlspecialchars($user['username']) ?></strong></p>  <p>Роль: <?= htmlspecialchars($user['role']) ?></p>   <?php if ($user['role'] === 'admin'): ?>  <p><a href="admin.php">Перейти в админ-панель</a></p>  <?php endif; ?>   <a href="edit.php">Редактировать профиль</a> |  <a href="logout.php">Выйти</a> </div> |
| --- |

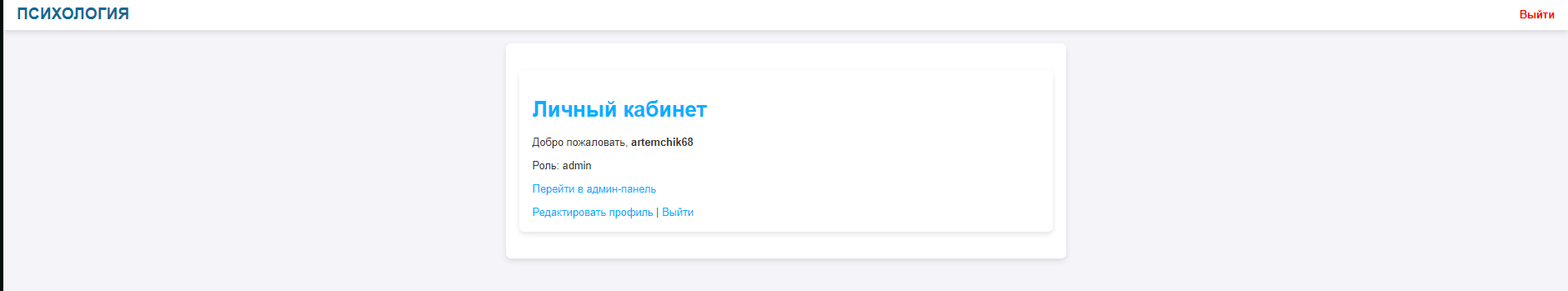


Рисунок 4 - Личный кабинет

### 2.3 Главная страница

Основные функции кода

Подключение к базе данных

| include 'db.php'; |
| --- |

Код подключает файл db.php, который обеспечивает соединение с базой данных через PDO.

Далее выполняется SQL-запрос для получения всех услуг из таблицы services:

| $stmt = $pdo->query("SELECT \* FROM services"); $services = $stmt->fetchAll(); |
| --- |

* + Результаты запроса сохраняются в массив $services.

1. Отображение приветствия

Если пользователь авторизован (переменная $\_SESSION['user'] установлена), выводится приветствие с именем пользователя и ссылка на личный кабинет:

| <?php if (isset($\_SESSION['user'])): ?>  <h1>Добро пожаловать, <?= htmlspecialchars($\_SESSION['user']['username']) ?></h1>  <a href="dashboard.php" class="button">Перейти в личный кабинет</a> <?php else: ?>  <h1>Добро пожаловать на сайт</h1>  <p>Войдите или зарегистрируйтесь, чтобы получить доступ к личному кабинету.</p> <?php endif; ?> |
| --- |

1. Вывод услуг

Все услуги из базы данных отображаются в формате карточек:

| <?php foreach ($services as $service): ?>  <div class="service">  <?php if ($service['image']): ?>  <img src="<?= htmlspecialchars($service['image']) ?>" alt="<?= htmlspecialchars($service['title']) ?>" class="service-image">  <?php endif; ?>  <h3><?= htmlspecialchars($service['title']) ?></h3>  <p><?= htmlspecialchars($service['description']) ?></p>  <p><strong>Цена:</strong> <?= htmlspecialchars($service['price']) ?> руб.</p>  </div> <?php endforeach; ?> |
| --- |

* + Каждый элемент услуги представлен заголовком, описанием, ценой и изображением (если оно указано).

1. Структура и стилизация

Секция услуг обёрнута в div с классом services-grid для размещения карточек в сетке, что обеспечивает аккуратное и современное отображение:

| <div class="services-grid">  ... </div> |
| --- |

1. Защита данных
   * Все данные, выводимые на страницу, проходят обработку функцией htmlspecialchars, чтобы предотвратить XSS-атаки.

Данный код реализует функциональность и внешний вид главной страницы веб-приложения. На ней отображаются услуги, добавленные в базу данных, а также предоставляется приветствие пользователю или приглашение войти/зарегистрироваться, если он не авторизован.

Главная страница предоставляет пользователям основную информацию о сайте и доступ к его функциям. Реализация отображения услуг и приветствия делает сайт удобным и информативным для посетителей Её внешний вид изображен на рисунке 5.

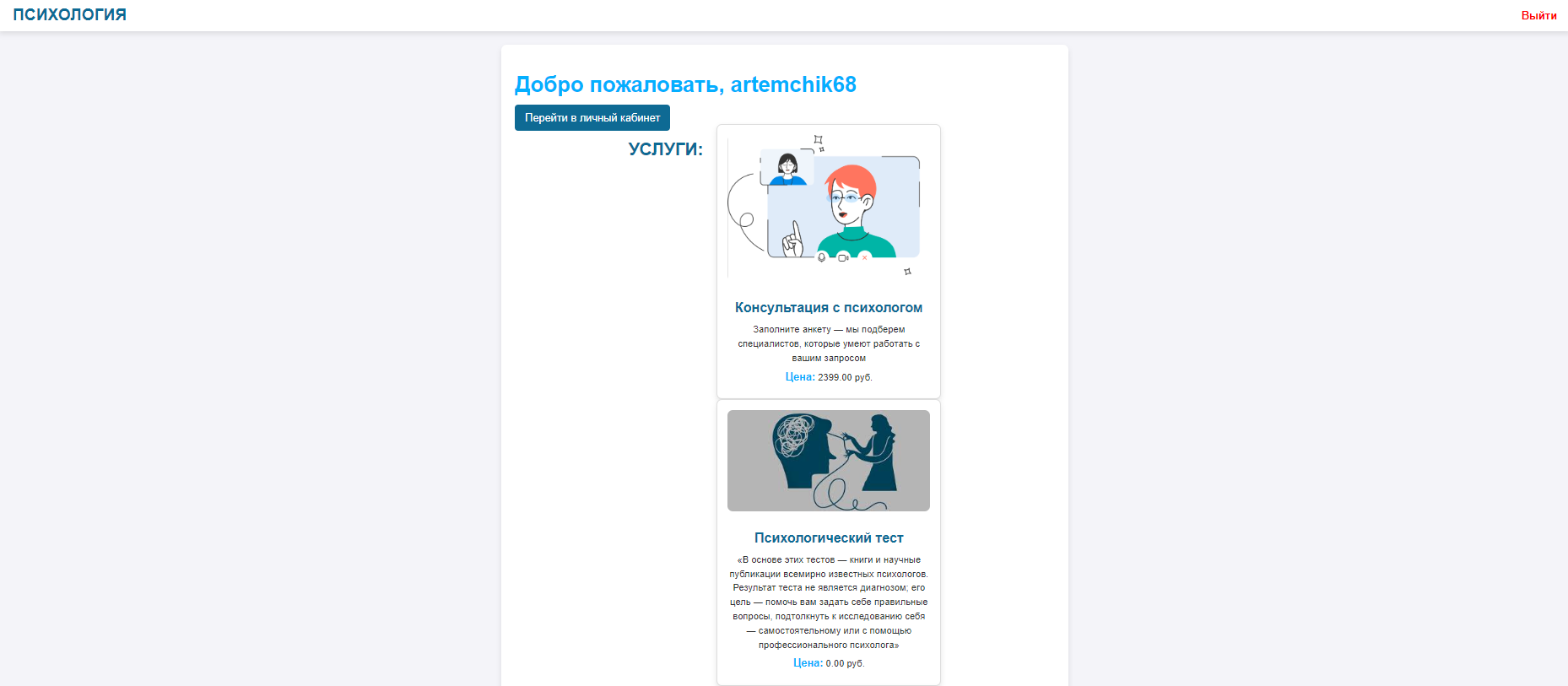


Рисунок 5 - Главная страница

### 2.4 Функции администратора

Администратор на данном сайте имеет доступ к админ-панели, где он может редактировать данные пользователей, добавлять их, удалять и менять им права пользования. Также админ способен создавать, редактировать и удалять услуги, которые есть на сайте.

1. Административная панель

Описание кода admin.php:

Проверяется, что текущий пользователь авторизован и имеет роль администратора:

| if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'admin') {  die("Доступ запрещён!"); } |
| --- |

1. Получение данных из базы:

Загружаются списки пользователей и услуг:

| $stmt = $pdo->query("SELECT \* FROM users"); $users = $stmt->fetchAll();  $stmt = $pdo->query("SELECT \* FROM services"); $services = $stmt->fetchAll(); |
| --- |

1. Отображение данных:

Список пользователей и услуг представлен в виде таблиц. Для каждого элемента добавлены ссылки на редактирование и удаление.

1. Основные функции:

Добавление нового пользователя:

| <a href="adduser.php" class="back-to-home">Добавить нового пользователя</a> |
| --- |

Добавление новой услуги:

| <a href="add\_service.php" class="back-to-home">Добавить новую услугу</a> |
| --- |

Пример отображения пользователей в таблице:

| <?php foreach ($users as $user): ?>  <tr>  <td><?= htmlspecialchars($user['username']) ?></td>  <td><?= htmlspecialchars($user['role']) ?></td>  <td>  <a href="edit.php?id=<?= $user['id'] ?>">Редактировать</a> |  <a href="delete.php?id=<?= $user['id'] ?>">Удалить</a>  </td>  </tr> <?php endforeach; ?> |
| --- |

2. Добавление пользователя

Описание кода adduser.php:

1. Авторизация администратора:  
   Проверяется, что доступ к странице имеют только администраторы.
2. Обработка данных формы:

Ввод логина, пароля и роли через HTML-форму:

| <form method="post">  <label for="username">Логин:</label>  <input type="text" name="username" id="username" required>    <label for="password">Пароль:</label>  <input type="password" name="password" id="password" required>    <label for="role">Роль:</label>  <select name="role" id="role">  <option value="user">User</option>  <option value="admin">Admin</option>  </select>    <button type="submit">Добавить пользователя</button> </form> |
| --- |

Добавление в базу данных:  
После отправки формы данные проверяются на уникальность, затем сохраняются в таблицу users:

| $stmt = $pdo->prepare("INSERT INTO users (username, password, role) VALUES (:username, :password, :role)"); $stmt->execute(['username' => $username, 'password' => $password, 'role' => $role]); |
| --- |

Визуализация добавления пользователя изображена на рисунке 6.

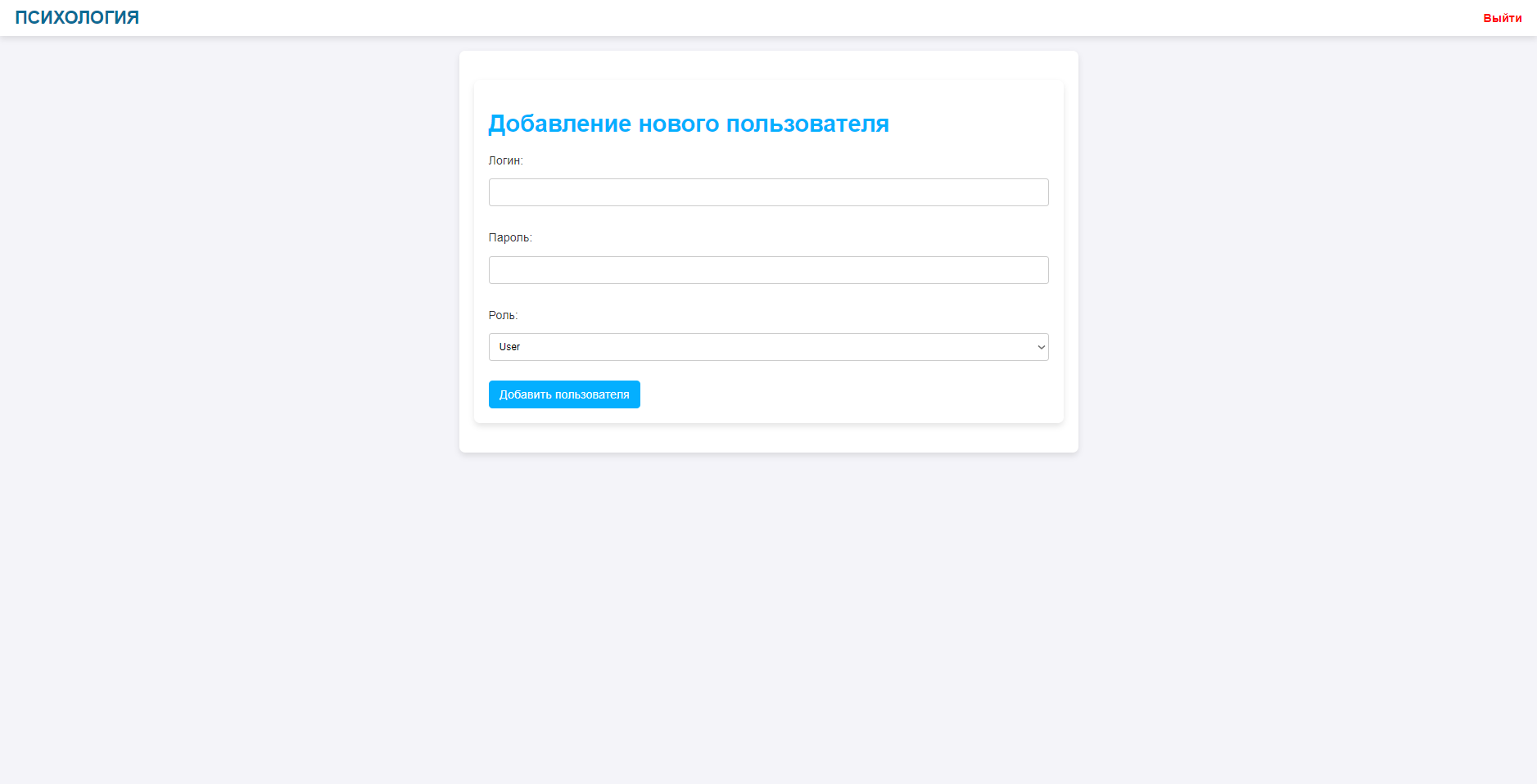


Рисунок 6 - Добавление нового пользователя.

3. Редактирование пользователя

Описание кода edit.php:

Получение данных пользователя:  
Данные текущего пользователя загружаются по его ID:

| $stmt = $pdo->prepare("SELECT \* FROM users WHERE id = :id"); $stmt->execute(['id' => $id]); $user = $stmt->fetch(); |
| --- |

Форма редактирования:  
HTML-форма позволяет изменить логин, пароль и роль пользователя:

| <form method="post">  <label for="username">Логин:</label>  <input type="text" name="username" value="<?= htmlspecialchars($user['username']) ?>" required>    <label for="password">Пароль (оставьте пустым, если не меняете):</label>  <input type="password" name="password">    <label for="role">Роль:</label>  <select name="role">  <option value="user" <?= $user['role'] === 'user' ? 'selected' : '' ?>>Пользователь</option>  <option value="admin" <?= $user['role'] === 'admin' ? 'selected' : '' ?>>Администратор</option>  </select>    <button type="submit">Сохранить изменения</button> </form> |
| --- |

Обновлённые данные записываются в базу:

| $stmt = $pdo->prepare("UPDATE users SET username = :username, password = :password, role = :role WHERE id = :id"); $stmt->execute(['username' => $username, 'password' => $password, 'role' => $role, 'id' => $id]); |
| --- |

Визуализация добавления пользователя изображена на рисунке 7.

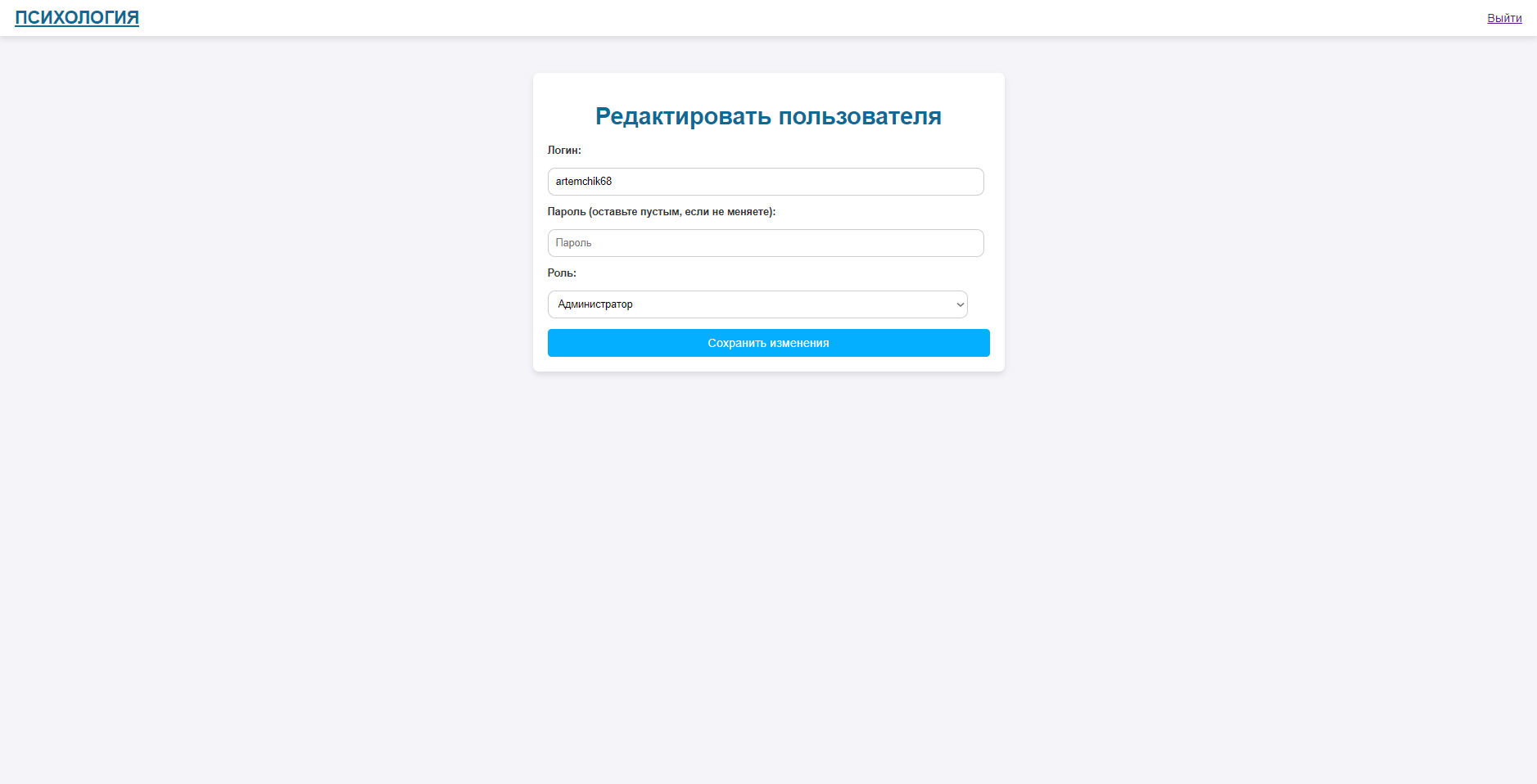


Рисунок 7 - Редактирование пользователя

4. Список услуг в админ-панели

Отображение услуг:

* + Каждая услуга отображается в таблице с колонками для названия, описания, цены и изображения.
  + Добавлены кнопки для редактирования и удаления услуг.

Пример HTML для списка услуг:

| <tr>  <td><?= htmlspecialchars($service['title']) ?></td>  <td><?= htmlspecialchars($service['description']) ?></td>  <td><?= htmlspecialchars($service['price']) ?> руб.</td>  <td>  <?php if ($service['image']): ?>  <img src="<?= htmlspecialchars($service['image']) ?>" alt="Изображение услуги" class="service-image">  <?php endif; ?>  </td>  <td>  <a href="edit\_service.php?id=<?= $service['id'] ?>">Редактировать</a> |  <a href="delete\_service.php?id=<?= $service['id'] ?>">Удалить</a>  </td> </tr> |
| --- |

Также для удаления реализован код delete.php

| <?php include 'db.php'; session\_start();  if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'admin') {  die("Доступ запрещён!"); }  if (isset($\_GET['id'])) {  $id = (int) $\_GET['id'];   $stmt = $pdo->prepare("DELETE FROM users WHERE id = :id");  $stmt->execute(['id' => $id]);   echo "Пользователь удалён!";  header('Location: admin.php');  exit; } else {  echo "Некорректный запрос!"; } ?> |
| --- |

Эти страницы реализуют управление услугами: их добавление и редактирование. Они доступны только администраторам и включают в себя обработку изображений и текстовых данных.

1. Добавление услуги

Авторизация администратора:  
Код проверяет, что текущий пользователь авторизован и имеет роль администратора:

| if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'admin') {  die("Доступ запрещён!"); } |
| --- |

1. Обработка данных формы:

Заголовок, описание, цена и изображение услуги принимаются из формы:

| $title = $\_POST['title']; $description = $\_POST['description']; $price = $\_POST['price']; |
| --- |

Изображение загружается в папку uploads/, а путь к файлу сохраняется в базе данных:

| if (isset($\_FILES['image']) && $\_FILES['image']['error'] === UPLOAD\_ERR\_OK) {  $imageTmpName = $\_FILES['image']['tmp\_name'];  $imageName = basename($\_FILES['image']['name']);  $imagePath = 'uploads/' . $imageName;   if (move\_uploaded\_file($imageTmpName, $imagePath)) {  $image = $imagePath;  } } |
| --- |

1. Сохранение данных в базу:

После валидации данные сохраняются в таблицу services:

| $stmt = $pdo->prepare("INSERT INTO services (title, description, price, image) VALUES (:title, :description, :price, :image)"); $stmt->execute(['title' => $title, 'description' => $description, 'price' => $price, 'image' => $image]); |
| --- |

HTML-форма добавления услуги:

Форма собирает данные, включая изображение:

| <form method="post" enctype="multipart/form-data">  <label for="title">Название услуги:</label>  <input type="text" name="title" id="title" required>   <label for="description">Описание услуги:</label>  <textarea name="description" id="description" required></textarea>   <label for="price">Цена:</label>  <input type="number" name="price" id="price" step="0.01" required>   <label for="image">Изображение услуги:</label>  <input type="file" name="image" id="image" accept="image/\*" required>   <button type="submit">Добавить услугу</button> </form> |
| --- |

2. Редактирование услуги

Загрузка текущих данных:  
Данные услуги загружаются из базы по её ID:

| $stmt = $pdo->prepare("SELECT \* FROM services WHERE id = :id"); $stmt->execute(['id' => $id]); $service = $stmt->fetch(); |
| --- |

Обновление данных:

Если новое изображение загружается, оно заменяет старое.

Все изменения сохраняются в таблице services:

| $stmt = $pdo->prepare("UPDATE services SET title = :title, description = :description, price = :price, image = :image WHERE id = :id"); $stmt->execute(['title' => $title, 'description' => $description, 'price' => $price, 'image' => $image, 'id' => $id]); |
| --- |

Обработка изображения:  
 Код поддерживает как добавление нового изображения, так и сохранение текущего:

| if (isset($\_FILES['image']) && $\_FILES['image']['error'] === UPLOAD\_ERR\_OK) {  $imageTmpName = $\_FILES['image']['tmp\_name'];  $imageName = basename($\_FILES['image']['name']);  $imagePath = 'uploads/' . $imageName;   if (move\_uploaded\_file($imageTmpName, $imagePath)) {  $image = $imagePath;  } } |
| --- |

HTML-форма редактирования услуги:

Форма позволяет редактировать данные услуги и загружать новое изображение. Визуализация редактирования услуги изображена на рисунке 8:

| <form method="post" enctype="multipart/form-data">  <label for="title">Название услуги:</label>  <input type="text" name="title" id="title" value="<?= htmlspecialchars($service['title']) ?>" required>   <label for="description">Описание услуги:</label>  <textarea name="description" id="description" required><?= htmlspecialchars($service['description']) ?></textarea>   <label for="price">Цена:</label>  <input type="number" name="price" id="price" value="<?= htmlspecialchars($service['price']) ?>" step="0.01" required>   <label for="image">Изображение услуги:</label>  <input type="file" name="image" id="image" accept="image/\*">    <?php if ($service['image']): ?>  <p>Текущее изображение:</p>  <img src="<?= htmlspecialchars($service['image']) ?>" alt="Изображение услуги" class="service-image">  <?php endif; ?>   <button type="submit">Обновить услугу</button> </form> |
| --- |

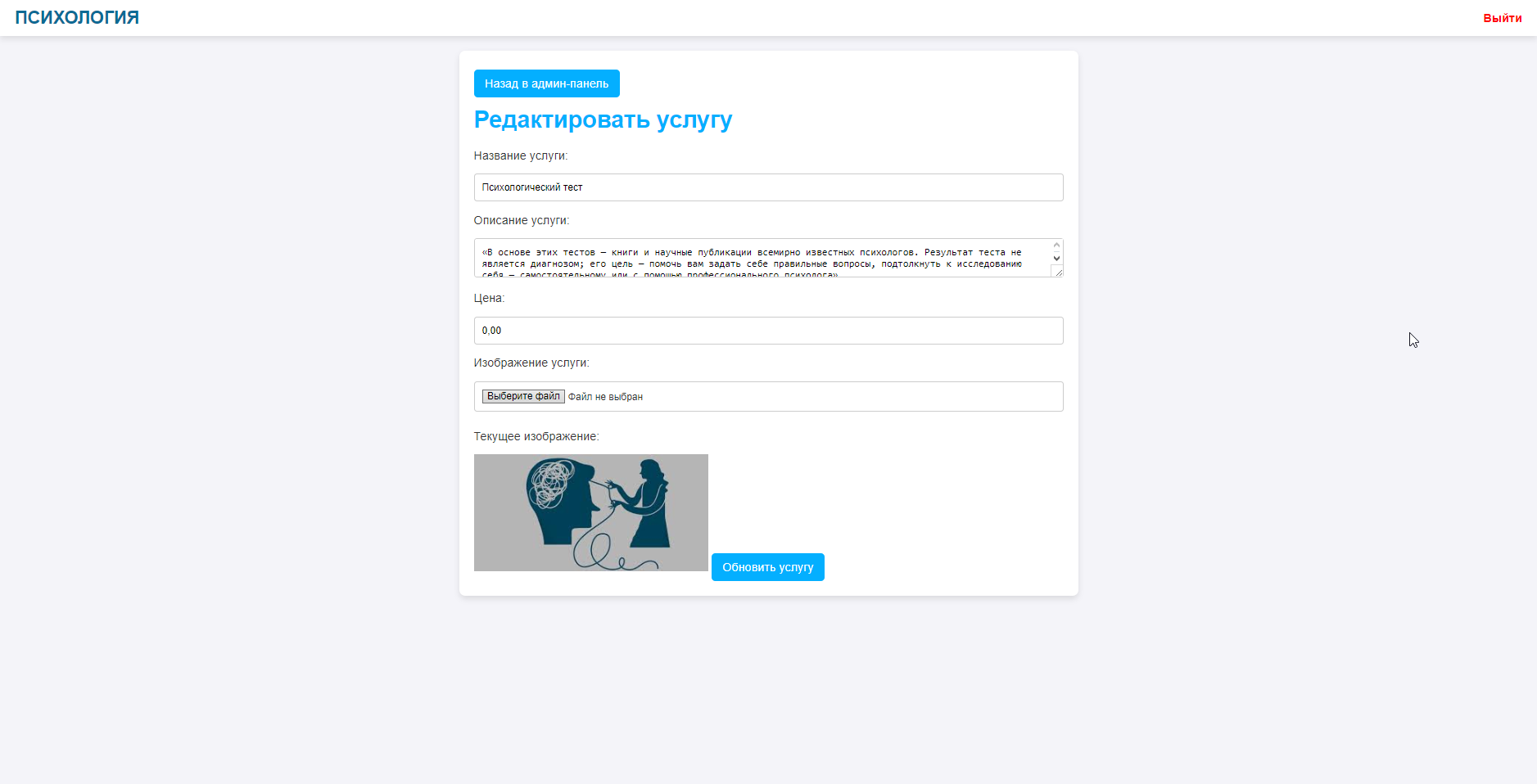


Рисунок 8 - Редактирование услуги

Также для удаления реализован код delete\_service.php

| <?php include 'db.php'; session\_start();  if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'admin') {  die("Доступ запрещён!"); }  if (isset($\_GET['id'])) {  $id = (int)$\_GET['id'];   $stmt = $pdo->prepare("DELETE FROM services WHERE id = :id");  $stmt->execute(['id' => $id]);   header('Location: admin.php');  exit; } else {   header('Location: admin.php');  exit; } |
| --- |

### 2.5 Стиль сайта

Созданный CSS-стиль обеспечивает современный, удобный и эстетически привлекательный интерфейс для веб-приложения. Основное внимание уделено организации элементов, адаптивности и интерактивным эффектам.

1. Базовая структура страницы

| body {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  background-color: #f4f4f9;  color: #333; } |
| --- |

* Шрифт: Arial, универсальный и хорошо читаемый.
* Цветовая палитра: нейтральный фон (#f4f4f9) и текст в тёмных тонах (#333), создающие контраст и удобство для чтения.
* Убранные отступы и поля создают чистую основу для верстки.

2. Шапка страницы

| .header {  display: flex;  justify-content: space-between;  align-items: center;  padding: 10px 20px;  background-color: #ffffff;  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); }  .header-left .logo {  color: #0e6a94;  font-size: 24px;  font-weight: bold;  text-transform: uppercase;  margin: 0; } |
| --- |

* Шапка (header) использует flexbox для распределения контента (логотипа и кнопок).
* Логотип выделяется синим цветом (#0e6a94) и заглавными буквами.
* Тень добавляет трёхмерный эффект, делая шапку визуально "приподнятой".

3. Карточки услуг

| .service {  background-color: #ffffff;  border: 1px solid #ddd;  border-radius: 8px;  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  width: 300px;  padding: 15px;  text-align: center;  transition: transform 0.2s ease, box-shadow 0.2s ease; }  .service:hover {  transform: translateY(-5px);  box-shadow: 0 6px 12px rgba(0, 0, 0, 0.2); }  .service img {  width: 100%;  height: auto;  border-radius: 8px;  margin-bottom: 15px; } |
| --- |

* Карточки услуг (service) визуально отделены белым фоном, закруглёнными углами и тенями.
* Эффекты при наведении добавляют интерактивность: карточка слегка поднимается.
* Изображения автоматически подстраиваются под размер контейнера, сохраняя пропорции.

4. Кнопки

| button, input[type="submit"], .button {  background-color: #0e6a94;  color: white;  border: none;  padding: 10px 15px;  cursor: pointer;  border-radius: 5px;  font-size: 16px;  transition: background-color 0.3s ease, transform 0.2s ease; }  button:hover, input[type="submit"]:hover, .button:hover {  background-color: #04afff;  transform: translateY(-2px);  }  button:active, input[type="submit"]:active, .button:active {  transform: scale(0.98);  } |
| --- |

* Основные кнопки выполнены в фирменном синем цвете (#0e6a94) с плавным изменением цвета при наведении.
* При нажатии кнопка слегка уменьшается, создавая тактильный отклик.

5. Таблицы

| table {  width: 100%;  border-collapse: collapse;  margin-top: 20px; }  table th, table td {  border: 1px solid #ddd;  padding: 8px;  text-align: left; }  table th {  background-color: #f2f2f2;  color: #333; } |
| --- |

* Таблицы используют минималистичный стиль: тонкие серые границы и светлый фон для заголовков.
* Простое оформление позволяет легко читать данные.

6. Сообщения об ошибке и успехе

| .error {  color: red;  margin-top: 10px; }  .success {  color: rgb(0, 238, 255);  margin-top: 10px; } |
| --- |

* Сообщения об ошибках выделяются красным, а об успехе — ярким голубым цветом, создавая визуальный акцент.

7. Адаптивность

| @media (max-width: 768px) {  .service {  width: 90%;  margin: 15px auto;  } } |
| --- |

* Для устройств с шириной экрана меньше 768px карточки услуг занимают всю ширину экрана с небольшими отступами.

## Заключение

В ходе выполнения данной курсовой работы была разработана и реализована современная веб-приложение с использованием технологий HTML, CSS, PHP и MySQL по тематике «Психология». Главной целью проекта было создание функционального, безопасного и удобного инструмента для управления услугами, предоставления информации пользователям и обеспечения работы администраторов.

Основные результаты работы:

1. Создан пользовательский интерфейс, который позволяет посетителям просматривать информацию об услугах, а зарегистрированным пользователям — управлять своими данными через личный кабинет.
2. Реализованы ключевые функции веб-приложения:
   * Регистрация и авторизация пользователей с использованием современных методов шифрования паролей.
   * Добавление, редактирование и удаление услуг с возможностью загрузки изображений.
   * Административная панель для управления пользователями и услугами, включая гибкое распределение ролей.
3. Дизайн интерфейса выполнен в минималистичном стиле, с акцентом на удобство использования и адаптивность под мобильные устройства.

## Список использованных источников

1. mdn web docs\_ / [Электронный курс] // CSS: каскадные таблицы стилей / [Сайт]. - URL: https://developer.mozilla.org/ru/docs/Web/CSS (дата обращения: 20.12.2024).
2. mdn web docs\_ / [Электронный курс] // Введение в JavaScript / [Сайт]. - URL: https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Introduction (дата обращения: 20.12.2024).
3. mdn web docs\_ / [Электронный курс] // Основы HTML / [Сайт]. - URL: <https://developer.mozilla.org/ru/docs/Learn_web_development/Getting_started/Your_first_website/Creating_the_content> (дата обращения: 20.12.2024).
4. METANIT.COM / [Электронный курс] // Руководство по MySQL / [Сайт]. - URL: <https://metanit.com/sql/mysql/> (дата обращения: 20.12.2024).
5. php / [Электронный курс] // Руководство по PHP / [Сайт]. - URL: https://www.php.net/manual/ru/index.php (дата обращения: 20.12.2024).