Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт управления бизнес-процессами

Кафедра «Бизнес информатика и моделирование бизнес-процессов»

**КУРСОВАЯ РАБОТА**

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

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

подпись, дата

Студент УБ22-08Б, 432217552 \_\_\_\_\_\_\_\_ А.Э. Бикмеева

подпись, дата

Красноярск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc156099039)

[1 Теоретическая часть 5](#_Toc156099040)

[1.1 HTML и CSS 5](#_Toc156099041)

[1.2 JavaScript 6](#_Toc156099042)

[1.3 PHP 8](#_Toc156099043)

[1.4 MySQL 10](#_Toc156099044)

[2 Практическая часть 13](#_Toc156099045)

[2.1 Требования и описание реализации 13](#_Toc156099046)

[2.2 Общая структура 14](#_Toc156099047)

[2.3 Создание базы данных 14](#_Toc156099048)

[2.4 Регистрация в web-приложении 16](#_Toc156099049)

[2.5 Авторизация в web-приложении 24](#_Toc156099050)

[2.6 Личный кабинет в web-приложении 27](#_Toc156099051)

[2.6.1 Функционал обычного пользователя 30](#_Toc156099052)

[2.6.2 Функционал администратора 31](#_Toc156099053)

[ЗАКЛЮЧЕНИЕ 36](#_Toc156099054)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc156099055)

# ВВЕДЕНИЕ

Сегодня, в свете глобального распространения интернета, программирование выделяется в отдельную отрасль - веб-программирование. Возникает необходимость в автоматизации различных процессов. Разработка веб-приложений предлагает создание высокотехнологичного продукта с интересным функционалом для пользователя и администратора.

Создание MySQL-базы данных для веб-приложения является важным шагом для повышения эффективности приложения. Ниже перечислены несколько ключевых аспектов, обосновывающих важность такой базы данных:

- База данных позволяет эффективно управлять информацией. Это упрощает процесс поиска и обработки данных, способствуя более эффективной работе.

- Важно обеспечить надежное хранение больших объемов информации о клиентах, столиках, меню, событиях и других связанных данных. База данных MySQL обеспечивает эту возможность и позволяет эффективно управлять информацией, даже при большом объеме данных.

В целом, создание MySQL-базы данных для веб-приложения является ключевым аспектом, способствующим эффективной работе приложения, улучшению управления данными, обеспечению безопасности и защите информации, а также позволяет взаимодействие с другими системами для более гибкого управления процессом.

Целью курсовой работы является разработка сайта, используя технологии HTML, CSS, Javascript, PHP, MySQL, согласно теме «Психологии», а также разработка базы данных для управления пользователями, используя СУБД MySQL.

Объектом исследования является web-приложение с применением различных web-технологий.

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

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

# 1 Теоретическая часть

## HTML и CSS

HTML — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.

CSS (Cascading Style Sheets, каскадные таблицы стилей) — 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.

HTML и CSS являются основными языками для создания веб-страниц. Вот некоторые особенности каждого из них:

HTML (Hypertext Markup Language):

1. Язык разметки: HTML используется для структурирования и организации содержимого веб-страниц.

2. Элементы и теги: HTML использует различные элементы и теги для обозначения различных частей содержимого, таких как заголовки, абзацы, списки, изображения и ссылки.

3. Семантика: HTML тесно связан с семантикой, что означает, что различные теги используются для обозначения значимости и назначения содержимого.

CSS (Cascading Style Sheets):

1. Язык стилей: CSS используется для определения внешнего вида и оформления веб-страниц, включая цвета, шрифты, отступы, размеры и макеты.

2. Каскадирование и наследование: CSS позволяет наследовать стили от одного элемента к другому и использовать каскадирование, что позволяет определять приоритетность стилей.

3. Разделение содержимого и оформления: CSS отделяет структуру и содержимое веб-страницы от ее визуального представления. Вместе HTML и CSS обеспечивают возможность создания красивых и структурированных веб-страниц, что делает их ключевыми инструментами для веб-разработчиков.

## JavaScript

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений.

Ниже приведены основные функции JavaScript:

1. Манипуляция DOM: С помощью JavaScript можно изменять содержимое и структуру веб-страницы, обеспечивая интерактивное взаимодействие с пользователем.

2. Обработка событий: JavaScript позволяет отслеживать различные события, такие как щелчки мыши, ввод с клавиатуры, загрузка страницы и другие, и реагировать на них соответствующим образом.

3. Асинхронные запросы: С использованием технологии AJAX, JavaScript позволяет отправлять и получать данные с сервера без перезагрузки всей страницы, что делает веб-приложения более отзывчивыми.

4. Создание анимаций: JavaScript может использоваться для создания анимаций и эффектов на веб-страницах, что делает пользовательский опыт более привлекательным.

5. Валидация форм: JavaScript может использоваться для проверки ввода данных в формы перед их отправкой на сервер, обеспечивая более точное и безопасное взаимодействие с пользователями.

6. Управление куками и локальным хранилищем: С помощью JavaScript можно управлять файлами cookie и локальным хранилищем браузера для сохранения данных между сеансами.

Это только некоторые из многих функций, которые предлагает JavaScript. Язык постоянно развивается, открывая новые возможности для создания более динамичных и привлекательных веб-приложений.

Важность JavaScript для web-разработки состоит в его способности добавлять динамичность и интерактивность на веб-страницы. Он позволяет создавать более привлекательные визуальные эффекты, улучшать пользовательский опыт, улучшать функциональность и сокращать время загрузки страницы. Большинство современных web-сайтов и приложений используют JavaScript, поэтому знание этого языка программирования является важным навыком для веб-разработчиков.

JavaScript имеет множество преимуществ, включая:

1. Универсальность: JavaScript является одним из самых распространенных языков программирования, который поддерживается практически всеми браузерами. Он также может быть использован для создания серверных приложений с помощью платформы Node.js.

2. Интерактивность: JavaScript позволяет создавать динамические и интерактивные веб-приложения, обеспечивая более привлекательный пользовательский опыт.

3. Возможность асинхронных запросов: JavaScript обеспечивает возможность отправки запросов на сервер без перезагрузки всей страницы, что делает веб-приложения более отзывчивыми и быстрыми.

4. Обширное сообщество и ресурсы: JavaScript имеет огромное сообщество разработчиков и обширную базу библиотек, фреймворков и ресурсов, которые облегчают разработку.

5. Многофункциональность: JavaScript может использоваться для создания веб-приложений, игр, мобильных приложений, а также для разработки скриптов на стороне клиента и сервера.

6. Быстрая скорость выполнения: JavaScript выполняется на стороне клиента и может обеспечивать быструю реакцию на действия пользователя без необходимости отправки запросов на сервер.

Это лишь несколько из множества преимуществ JavaScript, которые делают его популярным и востребованным языком программирования для веб-разработки.

## 1.3 PHP

PHP — C-подобный скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Основные особенности PHP включают в себя:

1. Встраивание в HTML: PHP код может быть встроен в HTML код, что обеспечивает гибкость при создании веб-страниц и обработке форм.

2. Поддержка большого количества баз данных: PHP поддерживает множество баз данных, включая MySQL, PostgreSQL, Oracle, и другие, что делает его идеальным для взаимодействия с базами данных веб-сайтов.

3. Широкое коммьюнити: PHP имеет огромное сообщество разработчиков, которое поддерживает язык и создает множество библиотек и расширений.

4. Кроссплатформенность: PHP может выполняться на различных операционных системах, таких как Windows, Linux, macOS, что обеспечивает его универсальность.

5. Простота в изучении: PHP относительно легко изучить, особенно для тех, кто уже знаком с HTML, что делает его привлекательным выбором для начинающих разработчиков.

6. Код выполняется на сервере: PHP является языком программирования, выполняемым на стороне сервера, что обеспечивает более безопасное взаимодействие с клиентами и защиту кода от общего доступа.

Одной из ключевых особенностей PHP является его специализация на разработке веб-приложений и удобство по встраиванию в HTML, что делает его мощным инструментом для создания динамических и интерактивных веб-сайтов.

PHP предоставляет широкий спектр функций для обработки данных, работы с базами данных, создания веб-приложений и т. д.

Вот несколько основных функций PHP:

1. Работа с формами: PHP обеспечивает возможности для обработки данных, отправленных через веб-формы методом POST или GET, и обеспечивает доступ к этим данным.

2. Работа с базами данных: PHP предоставляет функции для выполнения запросов к базам данных, включая MySQL, PostgreSQL, SQLite и другие, для извлечения, добавления, обновления и удаления данных.

3. Обработка файлов: PHP может выполнять чтение, запись и обработку файлов на сервере, включая загрузку файлов с клиентской стороны.

4. Работа с сессиями и куками: PHP обеспечивает возможность работы со сессиями пользователей и управления файлами cookie для сохранения информации между запросами.

5. Обработка ошибок и исключений: PHP предоставляет механизмы для обработки ошибок и исключений, что обеспечивает более надежное и безопасное выполнение кода.

6. Генерация динамического контента: PHP может быть использован для генерации динамического контента, такого как динамических страниц, изображений, PDF-файлов и других типов файлов.

Это лишь небольшой перечень из множества функций, предоставляемых PHP. Как серверный язык программирования, PHP позволяет создавать мощные веб-приложения и обеспечивает множество инструментов для работы с данными и обработки запросов со стороны сервера.

## MySQL

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB.

MySQL предоставляет разнообразные функции для работы с базами данных, включая манипулирование данными, выполнение запросов и управление структурой базы данных.

Вот несколько основных функций MySQL:

1. Создание таблиц: MySQL позволяет создавать новые таблицы в базе данных, определяя структуру и типы данных для каждого столбца.

2. Вставка, обновление и удаление данных: С помощью MySQL можно вставлять новые записи, обновлять существующие данные в таблицах и удалять ненужные записи.

3. Выполнение запросов: MySQL предоставляет множество операторов для выполнения запросов, таких как SELECT, INSERT, UPDATE, DELETE, а также агрегатные функции для обработки данных.

4. Управление пользователями и безопасность: MySQL позволяет администраторам управлять привилегиями пользователей, создавать учетные записи и устанавливать права доступа к базам данных и таблицам.

5. Транзакции и управление данными: MySQL поддерживает транзакции для обеспечения целостности данных и предоставляет возможности управления блокировками и индексами.

6. Управление структурой базы данных: MySQL обеспечивает возможности для создания и удаления индексов, изменения типов данных, добавления новых столбцов и других манипуляций со структурой базы данных.

Это лишь небольшой перечень из множества функций, предоставляемых MySQL. Как система управления реляционными базами данных, MySQL предоставляет мощные инструменты для работы с данными и управления базами данных.

Применение MySQL для сайта важно по ряду причин:

1. Управление данными: MySQL обеспечивает эффективное хранение и управление данными, позволяя сайту хранить большие объемы информации, такие как пользовательские данные, содержимое и другие типы информации.

2. Быстродействие: MySQL предлагает высокую производительность, что важно для сайтов с большим количеством трафика. Оптимизированные запросы к базе данных позволяют быстро получать и обрабатывать данные.

3. Надежность и безопасность: MySQL предлагает механизмы резервного копирования данных, обеспечивая сохранность информации в случае сбоев. Также существуют механизмы безопасности для защиты данных от несанкционированного доступа.

4. Масштабируемость: MySQL позволяет масштабировать базу данных в соответствии с ростом сайта и увеличением количества данных и пользователей.

5. Поддержка различных типов приложений: MySQL универсальна и может быть использована для различных видов сайтов, включая интернет-магазины, блоги, форумы, порталы и другие приложения.

6. Совместимость: MySQL совместима с различными языками программирования и платформами, что позволяет интегрировать ее в различные типы веб-приложений.

Использование MySQL для сайта обеспечивает надежное хранение, обработку и управление данными, что является ключевым аспектом для эффективной работы интернет-ресурса.

# 2 Практическая часть

## 2.1 Требования и описание реализации

Курсовой проект предполагает создание web-приложение на тему «Психология», используя технологии HTML, CSS, Javascript, PHP, MySQL. Требуется реализовать добавление, редактирование, просмотр и удаление пользователей. Также предусмотреть функции авторизации и аутентификации.

**Требования к проекту:**

1. Должна быть предусмотрена авторизация и регистрация пользователей.
2. Зарегистрированный пользователь имеет право редактировать свои данные и просматривать свои услуги.
3. Администраторы имеют право редактировать пользователей и добавлять услуги.
4. Неавторизированный пользователи имеют право просматривать список услуг
5. Сайт должен быть выдержан в определенном стиле и корректно открываться на любом разрешении.
6. Общий функционал может быть продуман самостоятельно.
7. Важно продемонстрировать не большой объем функционала, а минимальные требования к проекту. Проект может максимально простым, но при этом содержать все необходимые функции.
8. Отдельные бонусы предусмотрены за доп. функционал и красивый дизайн.

## 2.2 Общая структура

Установка open server позволяет воспроизводиться web-приложению локально. С помощью встроенного phpMyAdmin была создана база данных (Рисунок 1).

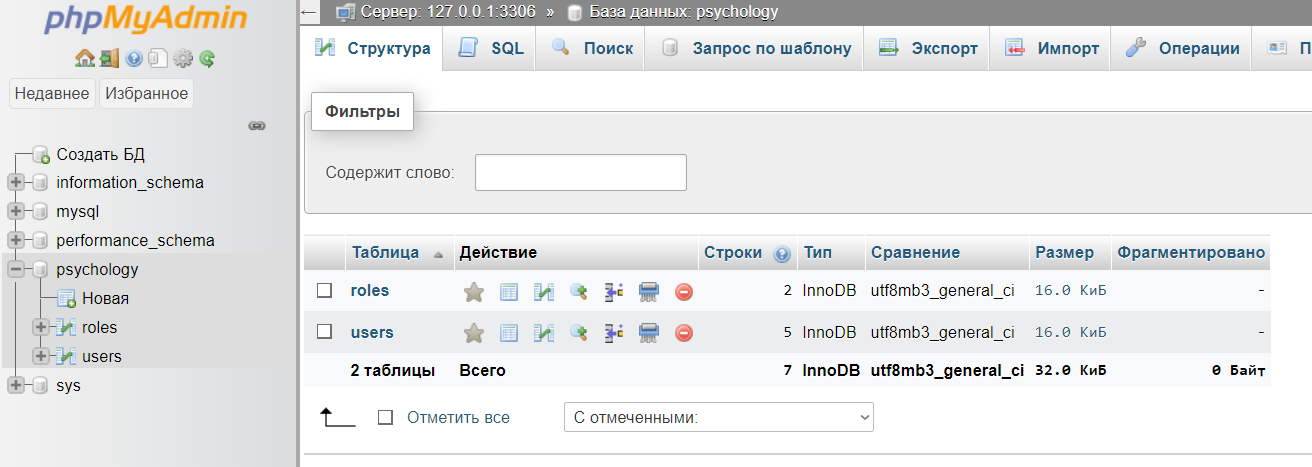


Рисунок 1 — База данных

## 2.3 Создание базы данных

Листинг создания базы данных:

--

-- База данных: `psychology`

--

-- --------------------------------------------------------

--

-- Структура таблицы `roles`

--

CREATE TABLE `roles` (

`role\_id` int NOT NULL,

`role\_name` enum('user','admin') NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

--

-- Дамп данных таблицы `roles`

--

INSERT INTO `roles` (`role\_id`, `role\_name`) VALUES

(1, 'admin'),

(2, 'user');

-- --------------------------------------------------------

--

-- Структура таблицы `users`

--

CREATE TABLE `users` (

`user\_id` int UNSIGNED NOT NULL,

`name` varchar(255) DEFAULT NULL,

`email` varchar(255) DEFAULT NULL,

`password` varchar(255) DEFAULT NULL,

`role\_id` int NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;

--

-- Дамп данных таблицы `users`

--

INSERT INTO `users` (`user\_id`, `name`, `email`, `password`, `role\_id`) VALUES

(1, 'алиса', 'bikmeeva.04@mail.ru', '$2y$10$Q1ncLcUBXv0Ctt5bEOBlX.MPquXzvGvPwnnRgoHqI9Dsm/XgUtAoG', 1),

(2, 'марина', 'marina@mail.ru', '$2y$10$5tjFIUGiZc2FODY7xtDOGeayINAmeqthvGHR1UzvK9ysk.7dKMUD2', 2),

(4, 'ярослава', 'mihyar@mail.ru', '$2y$10$nJRQUNxt1CNzJeOkl1rGTeoeAUjrJS4hMLx3M9TIFfb5OrNOmXoqq', 1),

(6, 'григорий', 'gribgoriy@mail.ru', '$2y$10$6hudFarmWh/x5S2vsGb93.WZyAUlQO7GEu9x1cboLybYLEzhpMkxq', 2),

(7, 'роман', 'roma@mail.ru', '$2y$10$94IhacfOcmi40s5bTE3dC.vW.nwkb87gWX2xzqlBg/QdEMa6K.sZO', 2);

COMMIT;

Название базы данных *psychology* с кодировкой *utf8mb3\_general\_ci*. Создаётся таблица ролей, которая имеет role\_id и role\_name (роль админа или пользователя). После создаём таблицу пользователей, которая содержит в себе user\_id, username, password, email, role\_id.

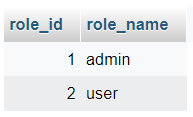


Рисунок 2 — База данных: роль пользователя и админа

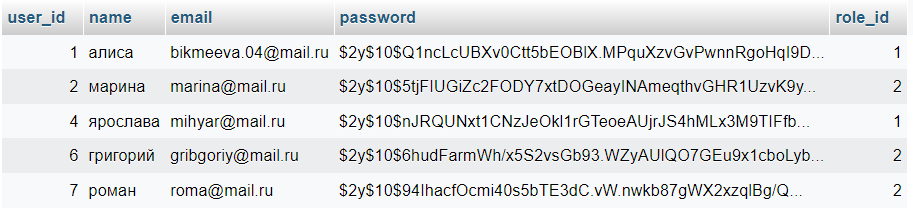


Рисунок 3 — База данных: пользователи

## 2.4 Регистрация в web-приложении

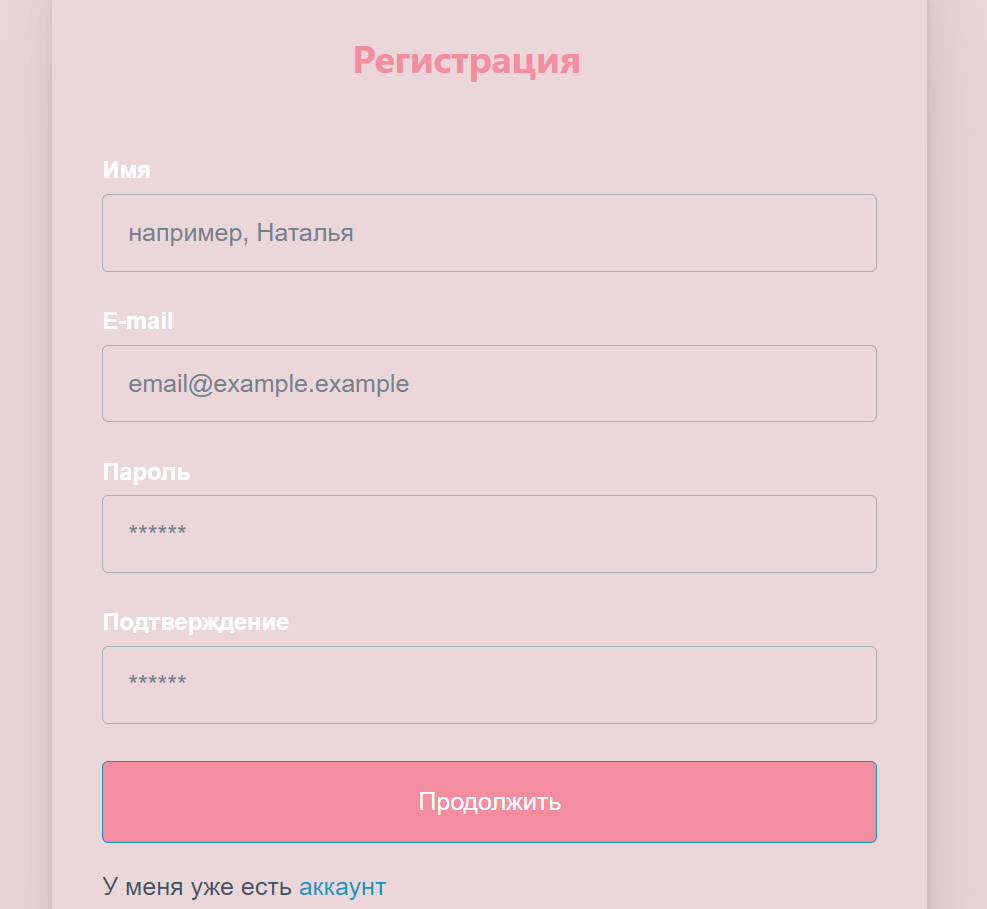


Рисунок 4 — Регистрация в web-приложении

Листинг HTML составляющей панели регистрации:

<!DOCTYPE html>

<html lang="ru" data-theme="light">

<?php include\_once \_\_DIR\_\_ . '/components/head.php'?>

<body>

<form class="card" action="src/actions/register.php" method="post" enctype="multipart/form-data">

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

<h2 class="reg">Регистрация</h2>

<label for="name">

<label class="reg2">

Имя

<input

type="text"

id="name"

name="name"

placeholder="например, Наталья"

value="<?php echo old('name') ?>"

<?php echo validationErrorAttr('name'); ?>

>

<?php if(hasValidationError('name')): ?>

<small><?php echo validationErrorMessage('name'); ?></small>

<?php endif; ?>

</label>

<label for="email">

<label class="reg2">

E-mail

<input

type="text"

id="email"

name="email"

placeholder="email@example.example"

value="<?php echo old('email') ?>"

<?php echo validationErrorAttr('email'); ?>

>

<?php if(hasValidationError('email')): ?>

<small><?php echo validationErrorMessage('email'); ?></small>

<?php endif; ?>

</label>

<div class="grid">

<label for="password">

<label class="reg2">

Пароль

<input

type="password"

id="password"

name="password"

placeholder="\*\*\*\*\*\*"

<?php echo validationErrorAttr('password'); ?>

>

<?php if(hasValidationError('password')): ?>

<small><?php echo validationErrorMessage('password'); ?></small>

<?php endif; ?>

</label>

<label for="password\_confirmation">

<label class="reg2">

Подтверждение

<input

type="password"

id="password\_confirmation"

name="password\_confirmation"

placeholder="\*\*\*\*\*\*"

>

</label>

</div>

<button class="btn1"

type="submit"

id="submit"

>Продолжить</button>

</form>

<p>У меня уже есть <a href="3-2.php">аккаунт</a></p>

<?php include\_once \_\_DIR\_\_ . '/components/scripts.php' ?>

</body>

</html>

Форма регистрации содержит в себе поля для ввода имени, электронной почты, пароля, строки подтверждения пароля, а также кнопки «зарегистрироваться». Отдельно хочется добавить удобное перемещение по сайту: если пользователь уже зарегистрирован, то его перебросит на страницу авторизации. «У меня уже есть аккаунт».

С помощью тега <link rel="stylesheet" href="app.css"> подключаем стили. Листинг стилей для панели регистрации и авторизации:

html[data-theme="light"] .card {

background: #ebd6d9;

}

body {

display: flex;

align-items: center;

justify-content: center;

flex-direction: column;

height: 140vh;

}

.card {

box-shadow: 0.0145rem 0.029rem 0.174rem rgba(0, 0, 0, 0.01698),

0.0335rem 0.067rem 0.402rem rgba(0, 0, 0, 0.024),

0.0625rem 0.125rem 0.75rem rgba(0, 0, 0, 0.03),

0.1125rem 0.225rem 1.35rem rgba(0, 0, 0, 0.036),

0.2085rem 0.417rem 2.502rem rgba(0, 0, 0, 0.04302),

0.5rem 1rem 6rem rgba(0, 0, 0, 0.06),

0 0 0 0.0625rem rgba(0, 0, 0, 0.015);

background: #ebd6d9;

border-radius: 0.25rem;

padding: 40px;

}

form {

min-width: 500px;

max-width: 700px;

}

#menu {margin-top: 50px;

background-color: #444;

color: #fff;

padding: 30px;

text-align: center;

width: 100%

}

.reg { color: #f48da0;

font-size: 30px;

font-weight: bold;

margin-left: 200px;

align-items: center;

}

.reg2 { color: white;

font-size: 19px;

font-weight: bold;

margin-left: 0px;

align-items: center;

}

.log { color: #f48da0;

font-size: 30px;

font-weight: bold;

margin-left: 250px;

align-items: center;

}

.btn1 {

background-color: #f48da0;

color: white;

text-align: center;

margin-left: 0px;

padding: 17px;

}

.btn2 {

background-color: #f48da0;

color: red;

text-align: center;

margin-left: 0px;

padding: 17px;

}

.home {

display: flex;

flex-direction: column;

align-items: center;

row-gap: 20px;

padding: 100px;

}

.notice {

padding: 10px;

border-radius: 6px;

margin: 20px 0;

}

.notice.error {

border: 2px solid red;

background: red;

color: white;

}

.notice.success {

border: 2px solid red;

background: pink;

color: white;

}

Далее при отправке формы активируется PHP код, устанавливается подключение к базе данных (подключение к локальному хосту). Указывается имя базы данных, имя пользователя, имеющего доступ и пароль. Листинг:

<?php

const DB\_HOST = 'localhost';

const DB\_PORT = '3306';

const DB\_NAME = 'psychology';

const DB\_USERNAME = 'root';

const DB\_PASSWORD = '';

<?php

require\_once \_\_DIR\_\_ . '/../helpers.php';

$name = $\_POST['name'] ?? null;

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

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

$passwordConfirmation = $\_POST['password\_confirmation'] ?? null;

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

header("Location: 3-3.php");

exit();

}

if (empty($name)) {

setValidationError('name', 'Неверное имя');

}

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

setValidationError('email', 'Указана неправильная почта');

}

if (empty($password)) {

setValidationError('password', 'Пароль пустой');

}

if ($password !== $passwordConfirmation) {

setValidationError('password', 'Пароли не совпадают');

}

if (!empty($\_SESSION['validation'])) {

setOldValue('name', $name);

setOldValue('email', $email);

redirect('/register.php');

}

$pdo = getPDO();

$query = "INSERT INTO users (name, email, password, role\_id) VALUES (:name, :email, :password, :role\_id)";

// Добавляем role\_id и устанавливаем его в 2 для новых пользователей (роль 'user')

$params = [

'name' => $name,

'email' => $email,

'password' => password\_hash($password, PASSWORD\_DEFAULT),

'role\_id' => 2, // Роль 'user'

];

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

try {

$stmt->execute($params);

// Получаем ID только что зарегистрированного пользователя

$userId = $pdo->lastInsertId();

// Получаем данные пользователя из базы данных

$user = findUserById($userId);

// Сохраняем пользователя в сессии

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

} catch (\Exception $e) {

die($e->getMessage());

}

redirect('/3-3.php');

При проверке пароля, если он неверный, система оповестит о том, что пароли не совпадают. Если же пароли совпали, запись отобразится в базе данных и пароль хешируется (Делается это для безопасности пользователей).

## 2.5 Авторизация в web-приложении

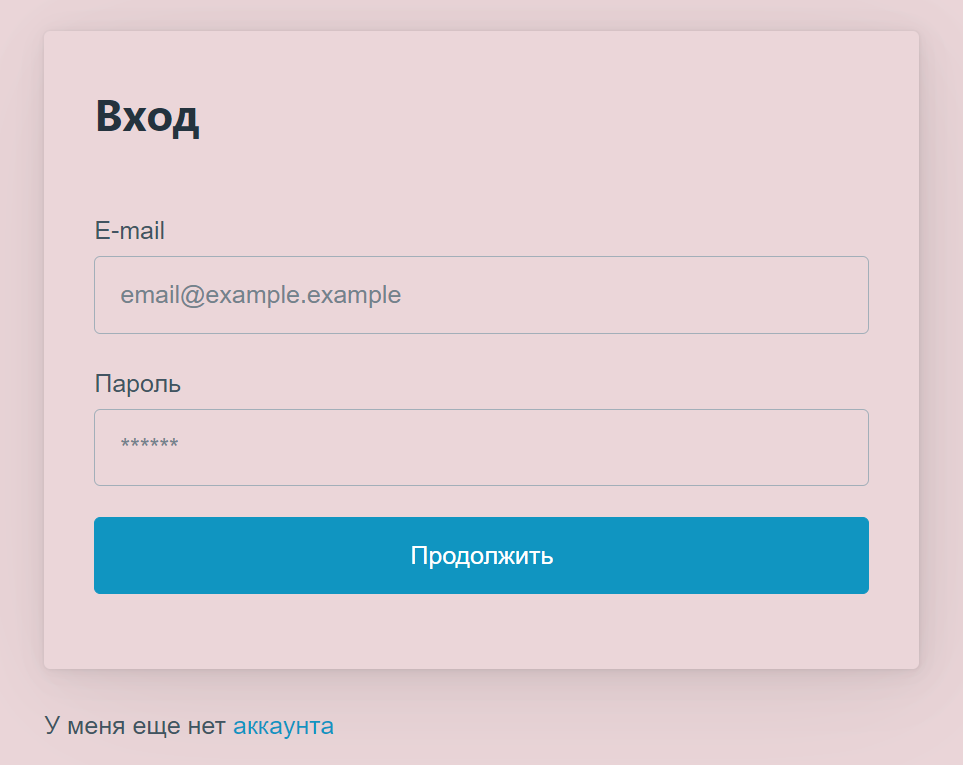


Рисунок 5 — Авторизация в web-приложении

Листинг:

<!DOCTYPE html>

<html lang="ru" data-theme="light">

<?php include\_once \_\_DIR\_\_ . '/components/head.php'?>

<body>

<form class="card" action="src/actions/login.php" method="post">

<h2>Вход</h2>

<?php if(hasMessage('error')): ?>

<div class="notice error"><?php echo getMessage('error') ?></div>

<?php endif; ?>

<label for="email">

E-mail

<input

type="text"

id="email"

name="email"

placeholder="email@example.example"

value="<?php echo old('email') ?>"

<?php echo validationErrorAttr('email'); ?>

>

<?php if(hasValidationError('email')): ?>

<small><?php echo validationErrorMessage('email'); ?></small>

<?php endif; ?>

</label>

<label for="password">

Пароль

<input

type="password"

id="password"

name="password"

placeholder="\*\*\*\*\*\*"

>

</label>

<button

type="submit"

id="submit"

>Продолжить</button>

</form>

<p>У меня еще нет <a href="/3.php">аккаунта</a></p>

<?php include\_once \_\_DIR\_\_ . '/components/scripts.php' ?>

</body>

</html>

Авторизация включает в себя форму ввода электронной почты, пароль и кнопку «войти».

Далее с помощью PHP кода обрабатывается форма авторизации, проверяется существование пользователя с соответствующими введёнными данными. После успешной регистрации пользователь будет перенаправлен в личный кабинет. Листинг:

<?php

require\_once \_\_DIR\_\_ . '/../helpers.php';

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

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

if (empty($email) || !filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

setOldValue('email', $email);

setValidationError('email', 'Неверный формат электронной почты');

setMessage('error', 'Ошибка валидации');

redirect('/3-2');

}

$user = findUser($email);

if (!$user) {

setMessage('error', "Пользователь $email не найден");

redirect('/3-2');

}

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

setMessage('error', 'Неверный пароль');

redirect('/3-2');

}

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

redirect('/3-3.php');

## 2.6 Личный кабинет в web-приложении

Далее было необходимо создать личный кабинет пользователя, который имеет определенные функции в зависимости от роли.

Листинг:

<?php

require\_once \_\_DIR\_\_ . '/src/actions/helpers.php';

checkAuth();

$user = currentUser();

?>

<div id="menu">

<nav class="nav nav-pills flex-column flex-sm-row">

<a class="flex-sm-fill text-sm-center nav-link" href="1.php">Статьи</a>

<a class="flex-sm-fill text-sm-center nav-link" href="4.php">Цитаты</a>

<a class="flex-sm-fill text-sm-center nav-link" href="5.php">Фильмы</a>

<a class="flex-sm-fill text-sm-center nav-link" href="2.php">Книги</a>

<?php if (isset($\_SESSION['user']['user\_id'])) : ?>

<a class="flex-sm-fill text-sm-center nav-link" href="3-3.php"><?= $\_SESSION['user']['name'] ?></a>

<?php else : ?>

<a class="flex-sm-fill text-sm-center nav-link" href="3.php">Войти/Зарегистрироваться</a>

<?php endif; ?>

</nav>

</div>

<body>

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

<div class="container mt-4">

<!DOCTYPE html>

<html lang="ru" data-theme="light">

<?php include\_once \_\_DIR\_\_ . '/components/head.php'?>

<body>

<div class="card home">

<h1>Добро пожаловать, <?php echo $user['name'] ?>!</h1>

<button id="editNameButton" role="button">Изменить имя</button>

<form id="editNameForm" action="src/actions/editName.php" method="post" style="display: none;">

<label for="newName">Новое имя:</label>

<input type="text" id="newName" name="newName">

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

</form>

<?php if ($user['role\_id'] == 1) : ?>

<button id="userListButton" role="button">Список пользователей</button>

<div id="userList" style="display: none;">

<h2>Список пользователей</h2>

<?php

$allUsers = getAllUsers();

echo '<ul>';

foreach ($allUsers as $userData) {

echo '<li><a href="/src/actions/userDetails.php?user\_id=' . $userData['user\_id'] . '">' . $userData['name'] . '</a></li>';

}

echo '</ul>';

?>

</div>

<?php endif; ?>

<form action="src/actions/logout.php" method="post">

<button role="button">Выйти из аккаунта</button>

</form>

</div>

<?php include\_once \_\_DIR\_\_ . '/components/scripts.php' ?>

<script>

document.getElementById('editNameButton').addEventListener('click', function() {

document.getElementById('editNameForm').style.display = 'block';

});

document.getElementById('userListButton').addEventListener('click', function() {

document.getElementById('userList').style.display = 'block';

});

</script>

</body>

</html>

Мы получаем данные об авторизованном пользователе (его имя) и создаём приветствие. При нажатии на имя, пользователя переносит в его личный кабинет.

Также имеется кнопка «выход». Реализованный код для разрыва пользователя с сайтом (кнопка «выход»).

Листинг:

<?php

require\_once \_\_DIR\_\_ . '/../helpers.php';

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

logout();

}

redirect('/3.php');

## 2.6.1 Функционал обычного пользователя

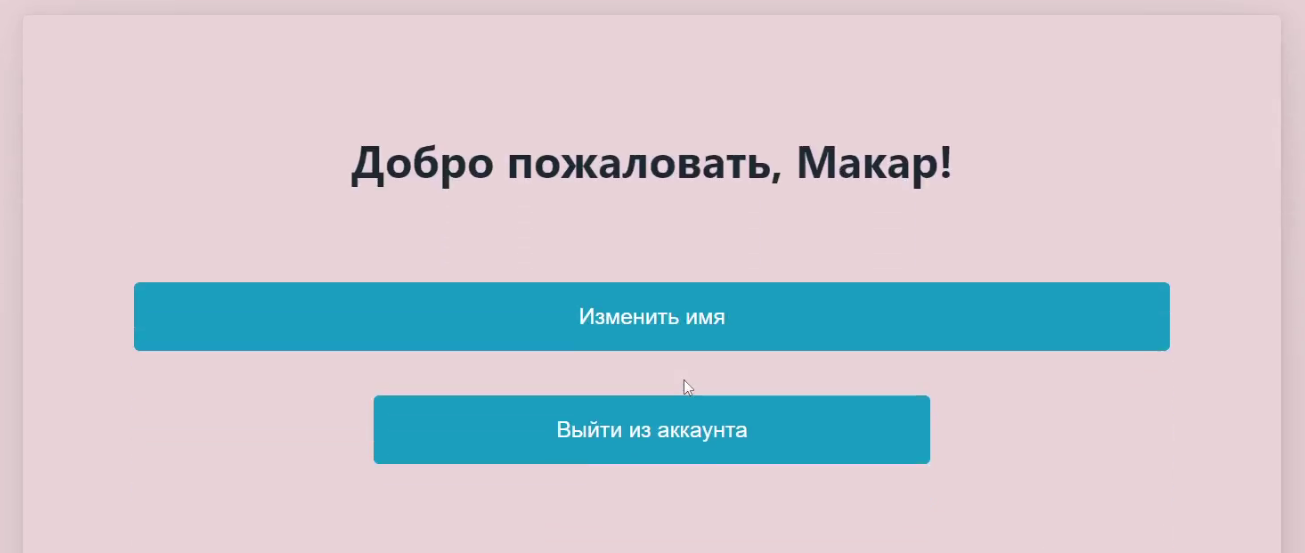


Рисунок 6 — Личный кабинет пользователя в web-приложении

Пользователь видит приветствие и обращение по имени.

На web-странице возле имени есть кнопка «изменить», при нажатии на которую запускается PHP код (+ кнопка «сохранить»). Листинг формы обновления имени пользователя:

<?php

session\_start();

require\_once \_\_DIR\_\_ . '/helpers.php';

checkAuth();

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

$newName = $\_POST['newName'] ?? '';

// Валидация нового имени

if (empty($newName)) {

$\_SESSION['validation']['newName'] = 'Новое имя не может быть пустым';

redirect('/3-3.php');

}

$pdo = getPDO();

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

try {

// Обновление имени в базе данных

$stmt = $pdo->prepare("UPDATE users SET name = :newName WHERE user\_id = :userId");

$stmt->execute(['newName' => $newName, 'userId' => $userId]);

// Обновление имени в сессии

$\_SESSION['user']['name'] = $newName;

// Редирект обратно на страницу 3-3.php

redirect('/3-3.php');

} catch (\Exception $e) {

die("Ошибка при обновлении имени: {$e->getMessage()}");

}

} else {

// Если запрос не POST, перенаправляем на главную страницу

redirect('/');

}

## 2.6.2 Функционал администратора

С помощью PHP кода проверяется, соответствует ли название пользователя роли администратора. При выполнении данного условия администратор получает список всех пользователей. Листинг:

<?php if ($user['role\_id'] == 1) : ?>

<button id="userListButton" role="button">Список пользователей</button>

<div id="userList" style="display: none;">

<h2>Список пользователей</h2>

<?php

$allUsers = getAllUsers();

echo '<ul>';

foreach ($allUsers as $userData) {

echo '<li><a href="/src/actions/userDetails.php?user\_id=' . $userData['user\_id'] . '">' . $userData['name'] . '</a></li>';

}

echo '</ul>';

?>

</div>

<?php endif; ?>

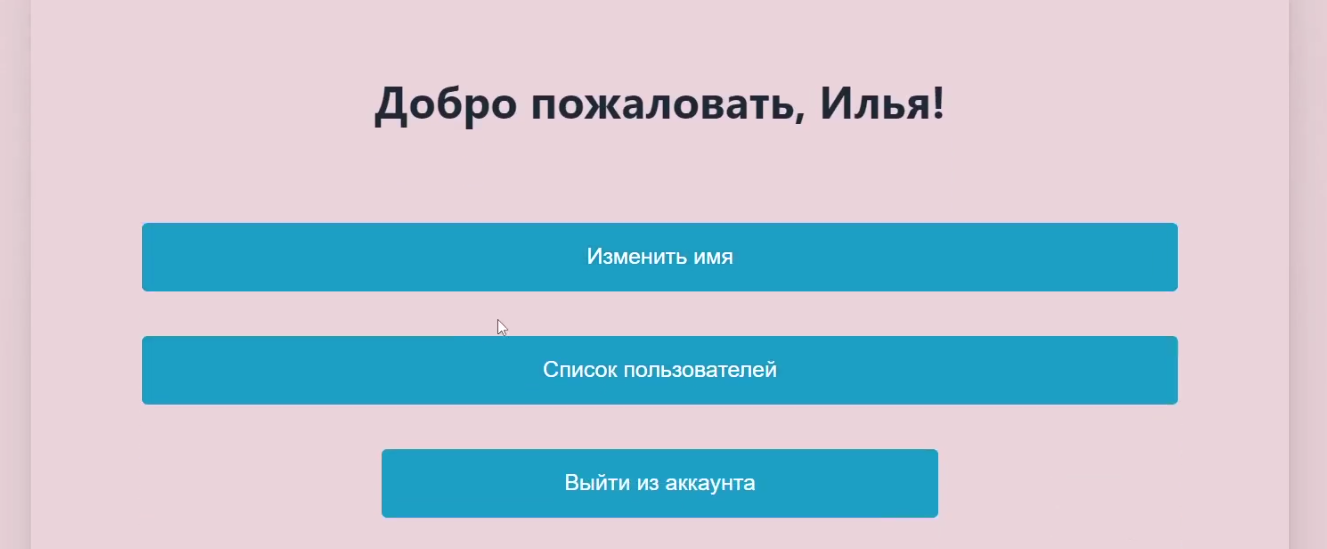
Администратор сайта может просматривать список юзеров на сайте и удалять их.

Рисунок 7 – Личный кабинет администратора.

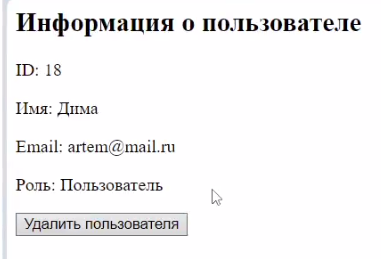


Рисунок 8 – Удаление пользователя администратором.

Листинг:

<?php

require\_once \_\_DIR\_\_ . '/../helpers.php';

checkAuth();

$userId = $\_GET['user\_id'] ?? null;

$user = currentUser();

if (!$userId) {

die("Отсутствует информация о пользователе");

exit;

}

$userData = findUserById($userId);

if (!$userData) {

die("Отсутствует информация о пользователе");

exit;

}

echo '<h2>Информация о пользователе</h2>';

echo '<p>ID: ' . $userData['user\_id'] . '</p>';

echo '<p>Имя: ' . $userData['name'] . '</p>';

echo '<p>Email: ' . $userData['email'] . '</p>';

echo '<p>Роль: ' . ($userData['role\_id'] == 1 ? 'Администратор' : 'Пользователь') . '</p>';

if ($user['role\_id'] == 1){

echo '<form action="/src/actions/deleteUser.php" method="post">';

echo '<input type="hidden" name="userId" value="' . $userData['user\_id'] . '">';

echo '<button type="submit" onclick="return confirm(\'Вы уверены, что хотите удалить пользователя?\')">Удалить пользователя</button>';

echo '</form>';

}

?>

# ЗАКЛЮЧЕНИЕ

В заключение необходимо подчеркнуть, что создание web-приложений играет важную роль в современном обществе, поскольку они обладают большим потенциалом для улучшения работы и удовлетворения потребностей в различных сферах.

В современном мире web-приложение, посвященное психологии, играет важную роль в области развития личности, улучшения психологического благополучия и распространения знаний о психологических аспектах жизни. Такое приложение может предоставлять доступ к информации о различных методах консультирования, психотерапии, самопомощи, а также разъяснять психологические концепции и теории.

Web-приложение о психологии может быть полезным для широкой аудитории, включая тех, кто ищет помощь в решении психологических проблем, а также для специалистов в области психологии, которые стремятся расширить свои знания и узнать о последних научных исследованиях.

Помимо этого, приложение о психологии может способствовать обмену опытом и информацией между специалистами, а также повышению общественной осведомленности о психическом здоровье и психологических аспектах жизни. Такой сайт может также привлекать спонсоров и рекламодателей, заинтересованных в продвижении своих продуктов и услуг в данной сфере.

В целом, web-приложение о психологии имеет потенциал стать важным источником информации, поддержки и образования в области психологии, способствуя деланию позитивного вклада в жизнь людей и общество в целом.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кузьмин, А.А. Профессиональное программирование. Web и PHP. Москва: Издательство "ДМК Пресс", 2018.

2. Лоупицкая, Н.А. Web-программирование и web-сайтостроение. Москва: Издательский центр "Адамант", 2011.

3. Макгроу, А. Web-программирование с использованием HTML, CSS, JavaScript, PHP, MySQL. Санкт-Петербург: БХВ-Петербург, 2017.

4. Мартинез, Д., Мельников, А. PHP. От новичка к профессионалу. Москва: Диалектика, 2008.

5. Поляков, В.В. Как научиться программировать web-сайты. Москва: Диалектика, 2014.

6. Ричардсон, В., Пиль, К., Вилсон, Э. RESTful Web сервисы: Построение сервисов следующего поколения. Москва: Издательство "Вильямс", 2017.

7. Сартбеков, Я.В. Web-программирование с использованием HTML, CSS, JavaScript, PHP и MySQL. Москва: ООО "Айти-вектор", 2016.

18. Фримен, Э., Робсон, Э. Изучаем PHP, MySQL и JavaScript. Москва: Издательство "Вильямс", 2017.