Руководство пользователя для веб-приложения "Todo List"

Данная программа предназначена для отслеживания выполнения задач.

Подготовка системы к работе:

Для начала работы пользователя с системой нужно перейти по ссылке «Todo List».

Перейдя по ссылке, пользователь попадает на страницу авторизации. Зарегистрированный пользователь может авторизоваться по нажатию на кнопку «Войти» (Рисунок 1).



Рисунок 1 – Страница авторизации

Пользователь может перейти к регистрации по нажатию на ссылку “У меня нет аккаунта”. После нажатия откроется форма для регистрации, где по нажатию на “Зарегистрироваться” пользователь сможет зарегистрироваться или вернуться на авторизацию по нажатию на “У меня есть аккаунт” (Рисунок 2).

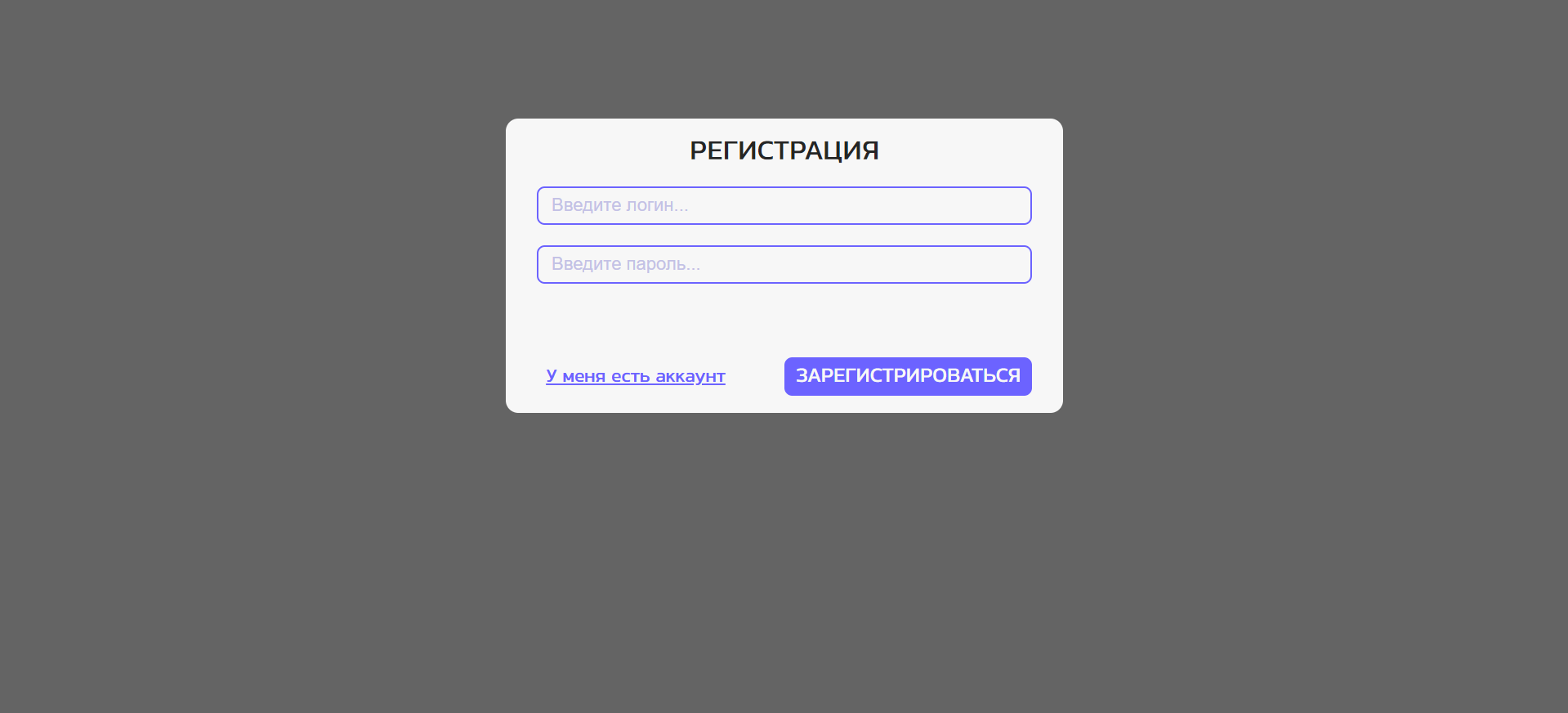


Рисунок 2 – Страница регистрации

После авторизации или регистрации пользователь попадает на страницу всех задач, где он может просмотреть список своих задач или выйти из аккаунта по нажатию на “Выйти” (Рисунок 3).

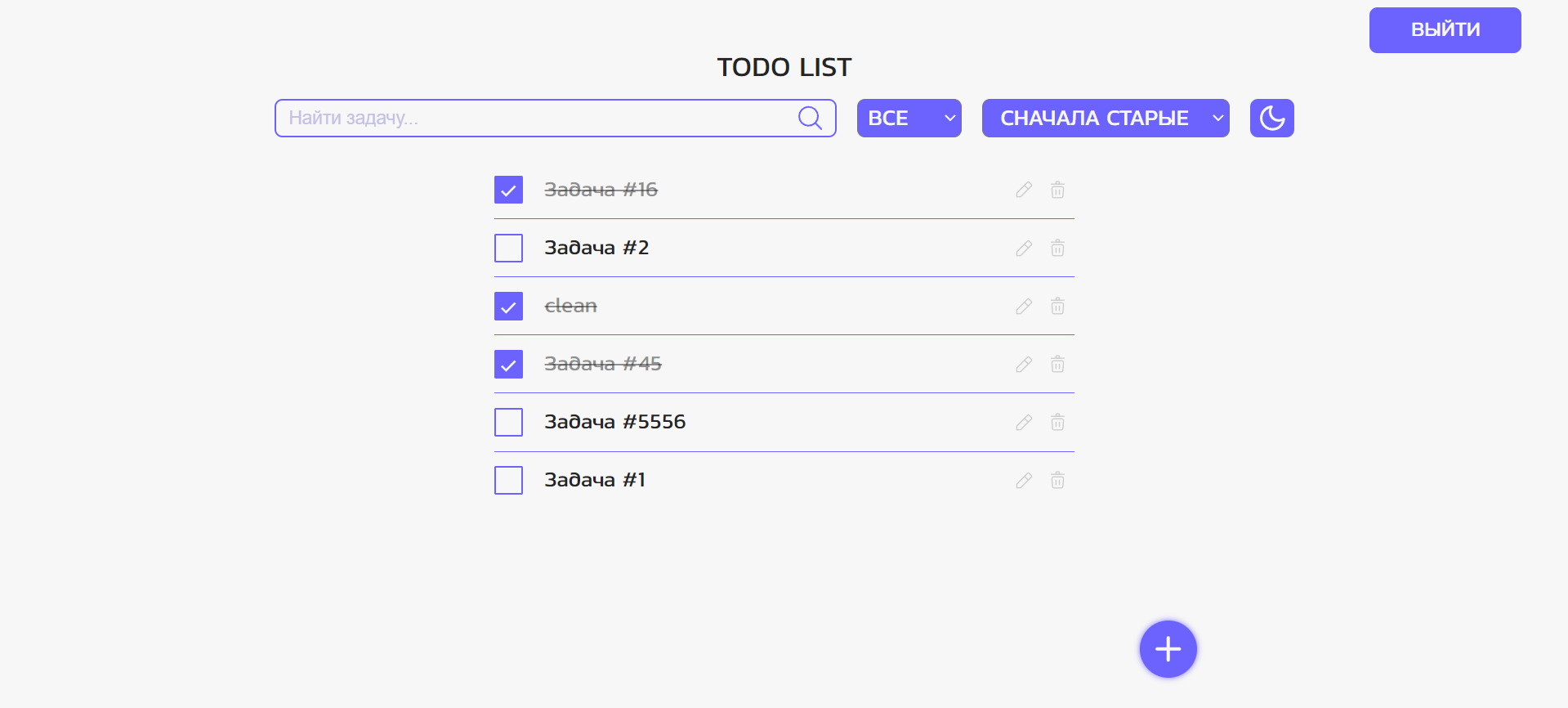


Рисунок 3 – Страница всех задач

Пользователь может воспользоваться поиском, фильтрацией и сортировкой для изменения отображения задач (Рисунок 4).

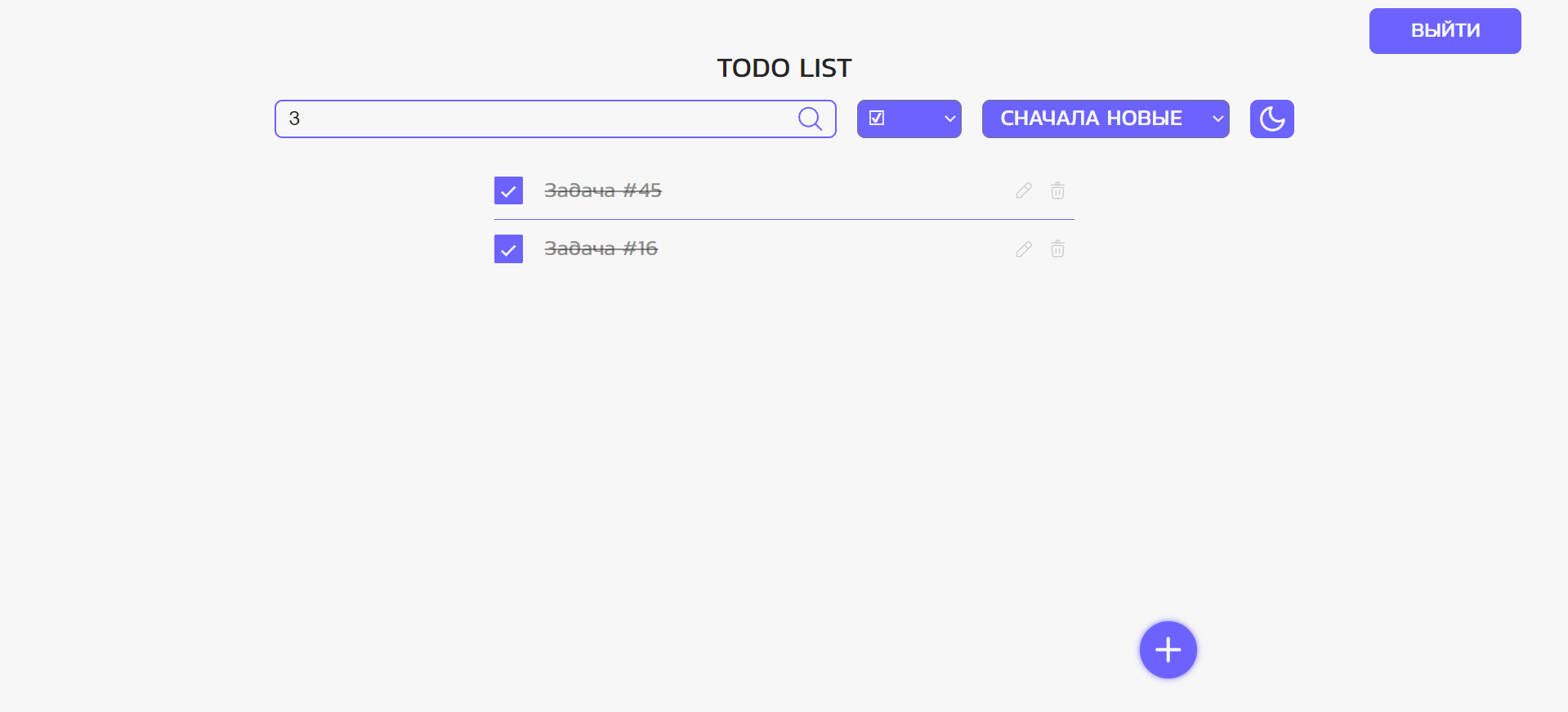


Рисунок 4 – Поиск, фильтрация и сортировка

Также пользователь может создать новую задачу по нажатию на “+”. После введения данных в форму и нажатия на “Сохранить” новая задача появится в списке задач вместе с соответствующим уведомлением (Рисунок 5).

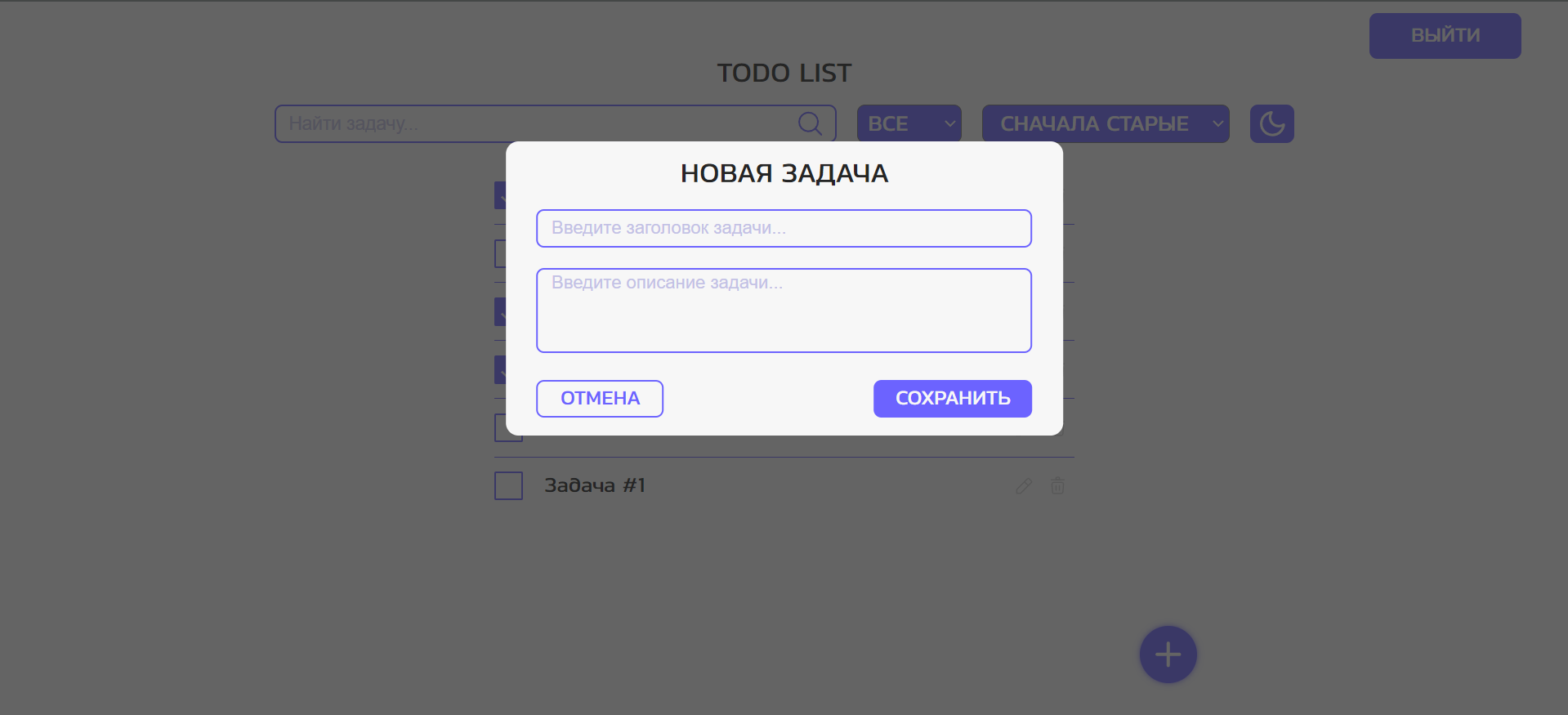


Рисунок 5 – Создание задачи

По нажатию на значок карандаша возле соответствующей задачи пользователь может отредактировать задачу, изменяя данные в форме. После нажатия на “Сохранить” будет выведено соответствующее уведомление (Рисунок 6).

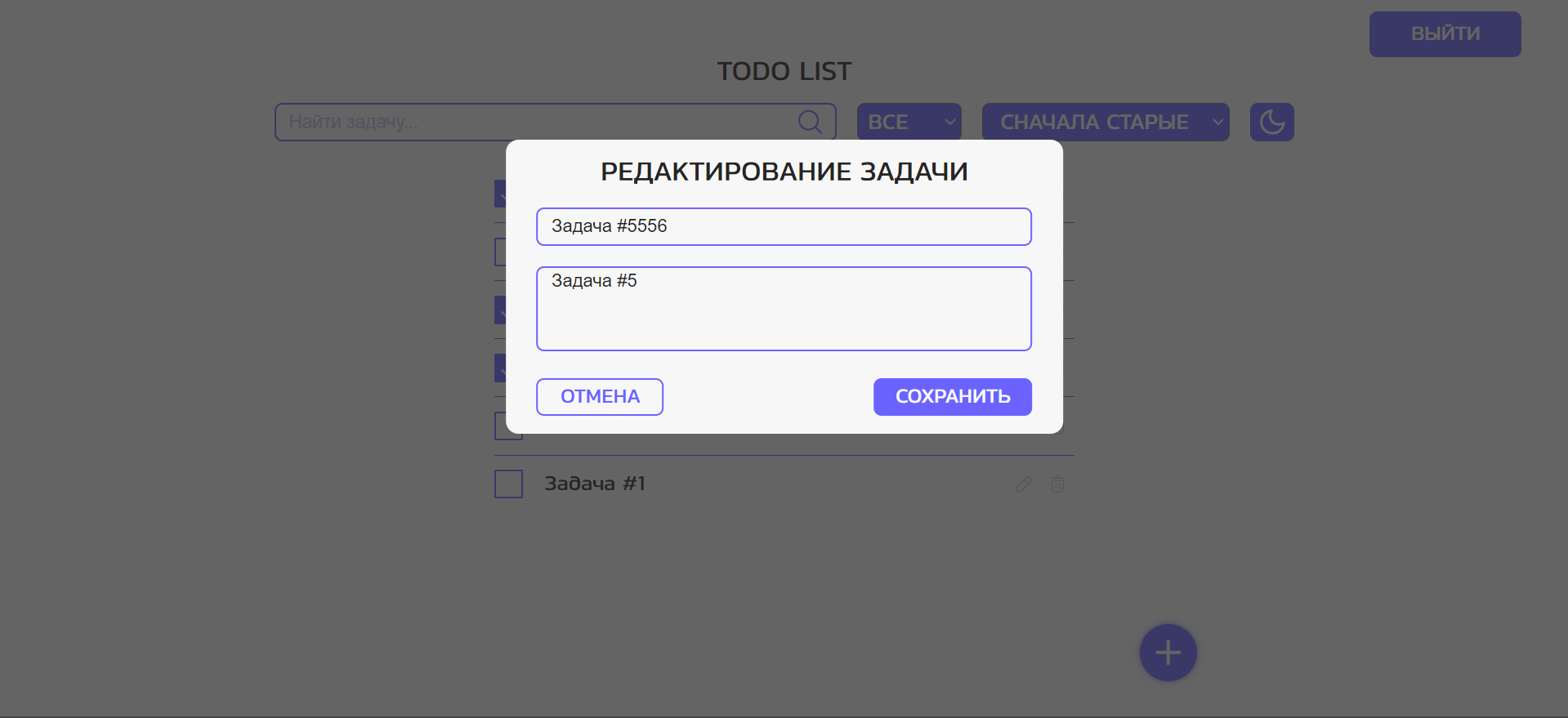


Рисунок 6 – Редактирование задачи

По нажатию на значок корзины возле соответствующей задачи пользователь может удалить задачу. После удаления будет выведено сообщение “Задача удалена” (Рисунок 7).

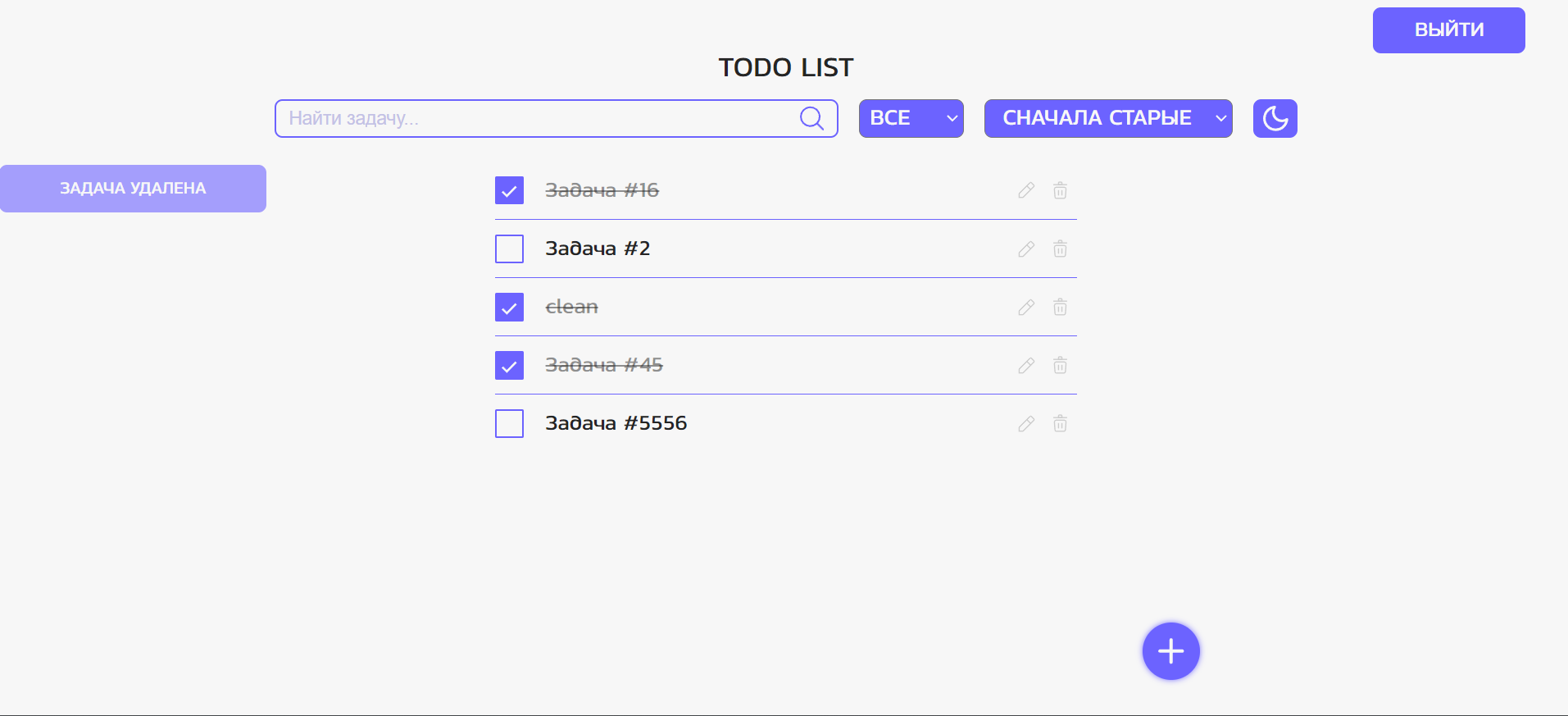


Рисунок 7 – Удаление задачи

Руководство программиста для веб-приложения "Todo List"

1. Введение

Todo List — это веб-приложение для отслеживания выполнения задач, построенное на основе клиент-серверной архитектуры, где серверная часть реализована с использованием PHP, а фронтенд использует JavaScript и jQuery для отправки AJAX-запросов.

Основные функции и возможности:

- регистрация и аутентификация пользователей;

- создание, редактирование и удаление задач;

- изменение статуса выполнения задач;

- асинхронная обработка запросов через AJAX;

- сортировка задач по новизне и по выполненности;

- фильтрация задач по выполненности;

- поиск задач по ключевым словам.

Технологический стек:

- Фронтенд: HTML, CSS, JavaScript, jQuery.

- Бэкенд: PHP, MySQL.

2. Архитектура приложения

2.1 Общая схема

Клиент отправляет AJAX-запросы из браузера на серверные скрипты PHP для обработки данных, таких как работа с базой данных и управление контентом. Ответы возвращаются в формате HTML для обработки на стороне клиента.

2.2 Взаимодействие компонентов

1. Пользователь взаимодействует с веб-интерфейсом, отправляя запросы (например, добавление задачи или фильтрация данных).

2. Запросы отправляются с помощью jQuery AJAX на серверные PHP-файлы.

3. PHP обрабатывает запросы и взаимодействует с базой данных MySQL.

4. Ответ возвращается в формате HTML и обрабатывается JavaScript для обновления интерфейса.

2.3 Основные модули

* add.php - обработка создания задачи;
* auth.php - обработка авторизации;
* change.php - обработка изменения статуса задачи;
* delete.php - обработка удаления задачи;
* edit.php - обработка редактирования задачи;
* exit.php - обработка выхода из аккаунта;
* get\_edit.php - файл для получения данных о редактируемой задаче;
* reg.php - обработка регистрации;
* search.php - обработка поиска, фильтрации и сортировки задач.

- Task.php - модуль для работы с таблицей tasks базы данных;

- User.php - модуль для работы с таблицей users базы данных;

- Connect.php - модуль для подключения к базе данных.

3. Разработка фронтенда

3.1 Установка окружения

Для разработки фронтенда вам потребуется запустить локальный сервер и подключить необходимые библиотеки:

```html

<!-- jQuery -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

```

3.2 Структура проекта

/todo

all\_notes.php - Страница всех записей

header.php - Шапка для всех страниц

index.php - Страница авторизации

reg.php - Страница регистрации

/css

style.css - Стили CSS

/database

Connect.php - Компонент для подключения к базе данных

Task.php - Компонент для выполнения действий над задачами

User.php - Компонент для выполнения действий с аккаунтами пользователей

/fonts - шрифты

/images - изображения

/js

for\_edit.js - Компонент для модального окна редактирования

mess.js - Компонент для отображения уведомлений

script.js - Компонент для выполнения ajax-запросов и отображения модальных окон

/user

add.php - Компонент обработки создания задачи

auth.php - Компонент обработки авторизации

change.php - Компонент обработки изменения статуса задачи

delete.php - Компонент обработки удаления задачи

edit.php - Компонент обработки редактирования задачи

exit.php - Компонент обработки выхода из аккаунта

get\_edit.php - Компонент для получения данных о редактируемой задаче

reg.php - Компонент обработки регистрации

search.php - Компонент обработки поиска, фильтрации и сортировки задач

3.3 Взаимодействие с сервером через AJAX

Пример отправки AJAX-запроса на сервер для удаления задачи:

```javascript

// script.js

function sendDelete(id) {

let request\_data = $("#search").val();

let request\_data2 = $("#filter").val();

let request\_data3 = $("#sort").val();

document.getElementById("filter\_val").value = $("#filter").val();

document.getElementById("search\_val").value = $("#search").val();

document.getElementById("sort\_val").value = $("#sort").val();

$.ajax({

type: "POST", // Метод отправки

url: "user/delete.php", // Путь до php файла отправителя

data: {

search: request\_data,

filter: request\_data2,

sort: request\_data3,

delete: id},

success: function (html) {

$("#main").html(html);

setInterval(func, 4000);

}

});

}

```

3.4 Стилизация

Стилизация компонентов реализована с помощью кастомных CSS-модулей:

```css

/ todo/css/style.css /

#body {

background-color: f7f7f7;

}

```

4. Разработка бэкенда

4.1 Подключение к базе данных (Connect.php)

Файл `Connect.php`, содержащий функции подключения к базе данных MySQL:

```php

<?php

class Connect { private $host = 'localhost';

private $user = 'root';

private $pass = '';

private $db = 'todo';

public $connection = null;

public function \_\_construct () {$this->connection = new mysqli ($this->host, $this->user , $this->pass , $this->db);} }

?>

```

4.2 Обработка данных на сервере

Пример обработки запроса на удаление задачи в `delete.php`:

```php

<?php

require "../database/Task.php";

$task\_mess = new Task;

$task\_mess = $task\_mess -> delete\_task($\_POST['delete']);

?>

<?php

$tasks = new Task;

if (isset($\_POST['filter']) and $\_POST['filter'] == 'all') {$\_POST['filter'] = NULL;}

if (!isset($\_POST['filter'])) {$\_POST['filter'] = NULL;}

if (!isset($\_POST['sort'])) {$\_POST['sort'] = NULL;}

if (!isset($\_POST['search'])) {$\_POST['search'] = NULL;}

$tasks = $tasks->get\_tasks($\_POST['search'], $\_POST['filter'], $\_POST['sort']);

$count = count($tasks);

$counter = 0;

if ($count !=0) {

foreach ($tasks as $task) {

$counter++;

if ($task[4] == 'false') {

?>

<div class="one\_note\_block">

<form class="not\_done">

<input type="hidden" value="<?=$task[0]?>" name="id">

<input type="checkbox" checked value="false" name="checkbox" style="display:none;">

</form>

<div></div>

<div class="not\_done\_note"><?=$task[2]?></div>

<svg class="edit" viewBox="0 0 15 15" fill="none" xmlns="http://www.w3.org/2000/svg" onclick="getEdit(<?=$task[0]?>)">

<path d="M7.67272 3.99106L1 10.6637V14H4.33636L11.0091 7.32736M7.67272 3.99106L10.0654 1.59837L10.0669 1.59695C10.3962 1.26759 10.5612 1.10261 10.7514 1.04082C10.9189 0.986392 11.0993 0.986392 11.2669 1.04082C11.4569 1.10257 11.6217 1.26735 11.9506 1.59625L13.4018 3.04738C13.7321 3.37769 13.8973 3.54292 13.9592 3.73337C14.0136 3.90088 14.0136 4.08133 13.9592 4.24885C13.8974 4.43916 13.7324 4.60414 13.4025 4.93398L13.4018 4.93468L11.0091 7.32736M7.67272 3.99106L11.0091 7.32736" stroke="#CDCDCD" stroke-linecap="round" stroke-linejoin="round"/>

</svg>

<svg class="trash" viewBox="0 0 18 18" fill="none" xmlns="http://www.w3.org/2000/svg" onclick="sendDelete(<?=$task[0]?>)">

<path d="M3.87414 7.61505C3.80712 6.74386 4.49595 6 5.36971 6H12.63C13.5039 6 14.1927 6.74385 14.1257 7.61505L13.6064 14.365C13.5463 15.1465 12.8946 15.75 12.1108 15.75H5.88894C5.10514 15.75 4.45348 15.1465 4.39336 14.365L3.87414 7.61505Z" stroke="#CDCDCD"/>

<path d="M14.625 3.75H3.375" stroke="#CDCDCD" stroke-linecap="round"/>

<path d="M7.5 2.25C7.5 1.83579 7.83577 1.5 8.25 1.5H9.75C10.1642 1.5 10.5 1.83579 10.5 2.25V3.75H7.5V2.25Z" stroke="#CDCDCD"/>

<path d="M10.5 9V12.75" stroke="#CDCDCD" stroke-linecap="round"/>

<path d="M7.5 9V12.75" stroke="#CDCDCD" stroke-linecap="round"/>

</svg>

</div>

<?php

} else {

?>

<div class="one\_note\_block">

<form class="done">

<input type="hidden" value="<?=$task[0]?>" name="id">

<input type="checkbox" checked value="true" name="checkbox" style="display:none;">

<svg viewBox="0 0 15 15" fill="none" xmlns="http://www.w3.org/2000/svg">

<mask id="path-1-inside-1\_18\_421" fill="white">

<path d="M4.9978 14.6488L1.72853e-05 9.74756L9.55927 2.22748e-06L14.5571 4.90124L4.9978 14.6488Z"/>

</mask>

<path d="M4.9978 14.6488L3.59745 16.0767L5.02539 17.4771L6.42574 16.0491L4.9978 14.6488ZM6.39816 13.2209L1.40037 8.31962L-1.40034 11.1755L3.59745 16.0767L6.39816 13.2209ZM13.1291 3.50089L3.56986 13.2484L6.42574 16.0491L15.985 6.30159L13.1291 3.50089Z" fill="#F7F7F7" mask="url(#path-1-inside-1\_18\_421)"/>

</svg>

</form>

<div></div>

<div class="done\_note"><?=$task[2]?></div>

<svg class="edit" viewBox="0 0 15 15" fill="none" xmlns="http://www.w3.org/2000/svg" onclick="getEdit(<?=$task[0]?>)">

<path d="M7.67272 3.99106L1 10.6637V14H4.33636L11.0091 7.32736M7.67272 3.99106L10.0654 1.59837L10.0669 1.59695C10.3962 1.26759 10.5612 1.10261 10.7514 1.04082C10.9189 0.986392 11.0993 0.986392 11.2669 1.04082C11.4569 1.10257 11.6217 1.26735 11.9506 1.59625L13.4018 3.04738C13.7321 3.37769 13.8973 3.54292 13.9592 3.73337C14.0136 3.90088 14.0136 4.08133 13.9592 4.24885C13.8974 4.43916 13.7324 4.60414 13.4025 4.93398L13.4018 4.93468L11.0091 7.32736M7.67272 3.99106L11.0091 7.32736" stroke="#CDCDCD" stroke-linecap="round" stroke-linejoin="round"/>

</svg>

<svg class="trash" viewBox="0 0 18 18" fill="none" xmlns="http://www.w3.org/2000/svg" onclick="sendDelete(<?=$task[0]?>)">

<path d="M3.87414 7.61505C3.80712 6.74386 4.49595 6 5.36971 6H12.63C13.5039 6 14.1927 6.74385 14.1257 7.61505L13.6064 14.365C13.5463 15.1465 12.8946 15.75 12.1108 15.75H5.88894C5.10514 15.75 4.45348 15.1465 4.39336 14.365L3.87414 7.61505Z" stroke="#CDCDCD"/>

<path d="M14.625 3.75H3.375" stroke="#CDCDCD" stroke-linecap="round"/>

<path d="M7.5 2.25C7.5 1.83579 7.83577 1.5 8.25 1.5H9.75C10.1642 1.5 10.5 1.83579 10.5 2.25V3.75H7.5V2.25Z" stroke="#CDCDCD"/>

<path d="M10.5 9V12.75" stroke="#CDCDCD" stroke-linecap="round"/>

<path d="M7.5 9V12.75" stroke="#CDCDCD" stroke-linecap="round"/>

</svg>

</div>

<?php

}

if ($counter != $count) {

?>

<div class="line"></div>

<?php

}

?>

<?php

} } else {

?>

<img src="images/Detective-check-footprint 1.svg" class="detective" id="detective">

<div class="vh2\_3" id="vh\_empty"></div>

<div class="empty" id="empty">Пусто...</div>

<?php

} ?>

<div id="mess" ><div><?=$task\_mess?></div></div>

<script src="js/mess.js"></script>

<script src="js/script.js"></script>

```

4.3 Аутентификация (auth.php)

Пример обработки аутентификации пользователя в файле auth.php:

```php

<?php

require "../database/User.php";

$user = new User;

$user -> auth($\_POST['login'], $\_POST['pass']);

?>

```

Пример обработки аутентификации пользователя в файле User.php:

```php

public function auth($login, $pass) {

$this->checkEmpty(trim($login), 'login-auth', 'Не введен логин');

$this->checkEmpty(trim($pass), 'pass-auth', 'Не введен пароль');

if (!isset($this->error\_valid\_text['login'])) {

$login\_check = $this->connection->query("SELECT \* FROM users WHERE username='$login'");

if ($login\_check->num\_rows == 0) {

$this->error\_valid = true;

$this->error\_valid\_text["login-auth"] = 'Неверный логин';

$\_SESSION['mess'] = $this->error\_valid\_text;

header("Location: ../index.php"); }

else if (!isset($this->error\_valid\_text['pass'])) {

if ($login\_check->num\_rows != 0) {

$login\_check = $login\_check->fetch\_array();

if (password\_verify($pass, $login\_check['password\_hash'])) {

$\_SESSION['user'] = $login\_check['id'];

$this->error\_valid = true;

$this->error\_valid\_text["done"] = "Вы успешно авторизовались";

$\_SESSION['mess'] = $this->error\_valid\_text;

header("Location: ../all\_notes.php"); } else {

$this->error\_valid = true;

$this->error\_valid\_text["pass-auth"] = 'Неверный пароль';

$\_SESSION['mess'] = $this->error\_valid\_text;

header("Location: ../index.php"); } }} else {

$\_SESSION['mess'] = $this->error\_valid\_text;

header("Location: ../index.php"); } } else {

$\_SESSION['mess'] = $this->error\_valid\_text;

header("Location: ../index.php"); } }

```

4.4 Структура базы данных

Пример таблиц базы данных:

```sql

CREATE TABLE `tasks` (

`id` int NOT NULL,

`user\_id` int NOT NULL,

`title` varchar(40) NOT NULL,

`description` varchar(600) DEFAULT NULL,

`is\_completed` enum('true','false') CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL DEFAULT 'false',

`created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

`updated\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `users` (

`id` int NOT NULL,

`username` varchar(50) NOT NULL,

`password\_hash` varchar(500) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

```

5. Тестирование

5.1 Тестирование на клиенте

1. Проверьте корректность работы форм и их отправку через AJAX.

2. Убедитесь, что сервер корректно возвращает HTML-ответы, которые обрабатываются на клиенте.

5.2 Тестирование на сервере

1. Проверьте работу скриптов PHP с помощью тестовых данных.