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 2.3.1).



Рисунок 2.3.1 – 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 2.3.2).

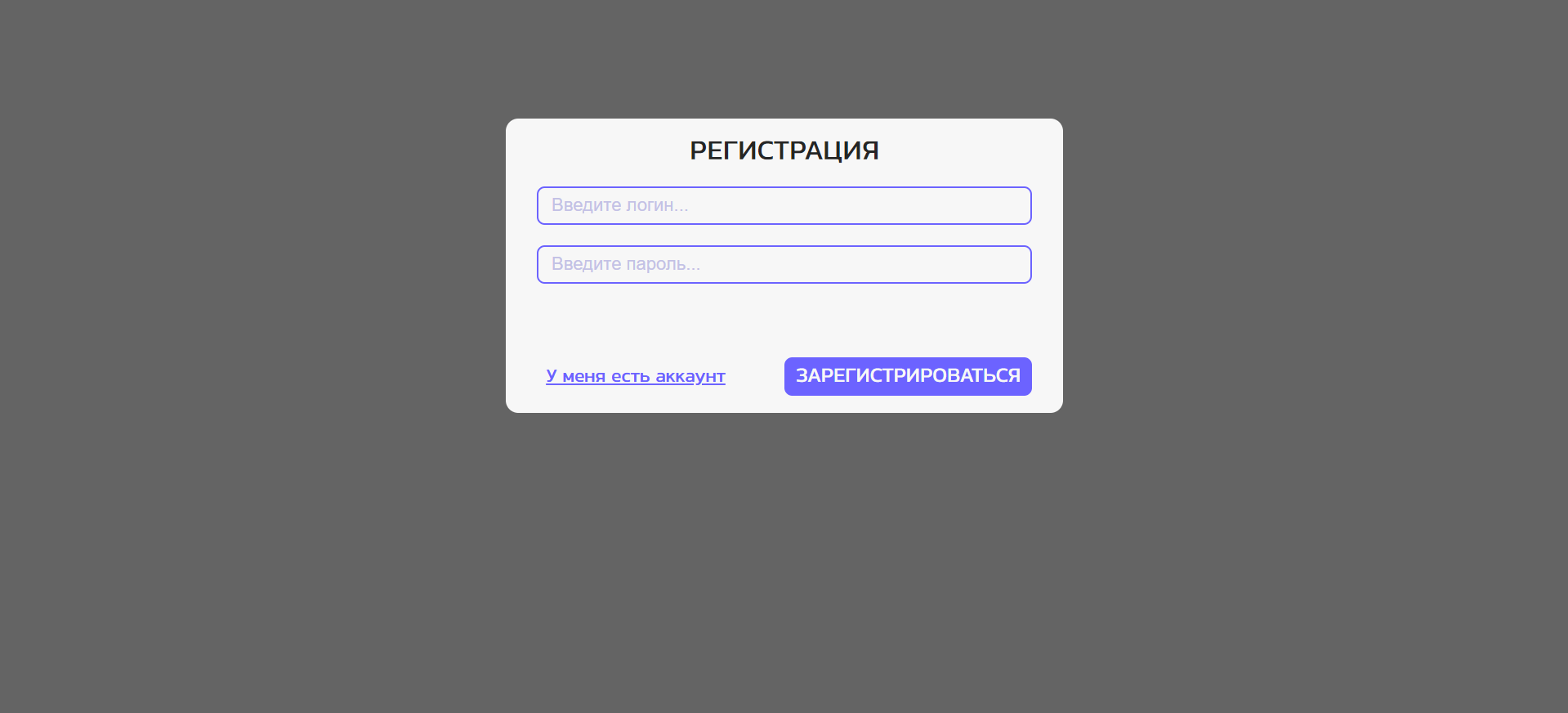


Рисунок 2.3.2 – Страница регистрации

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 2.3.3).

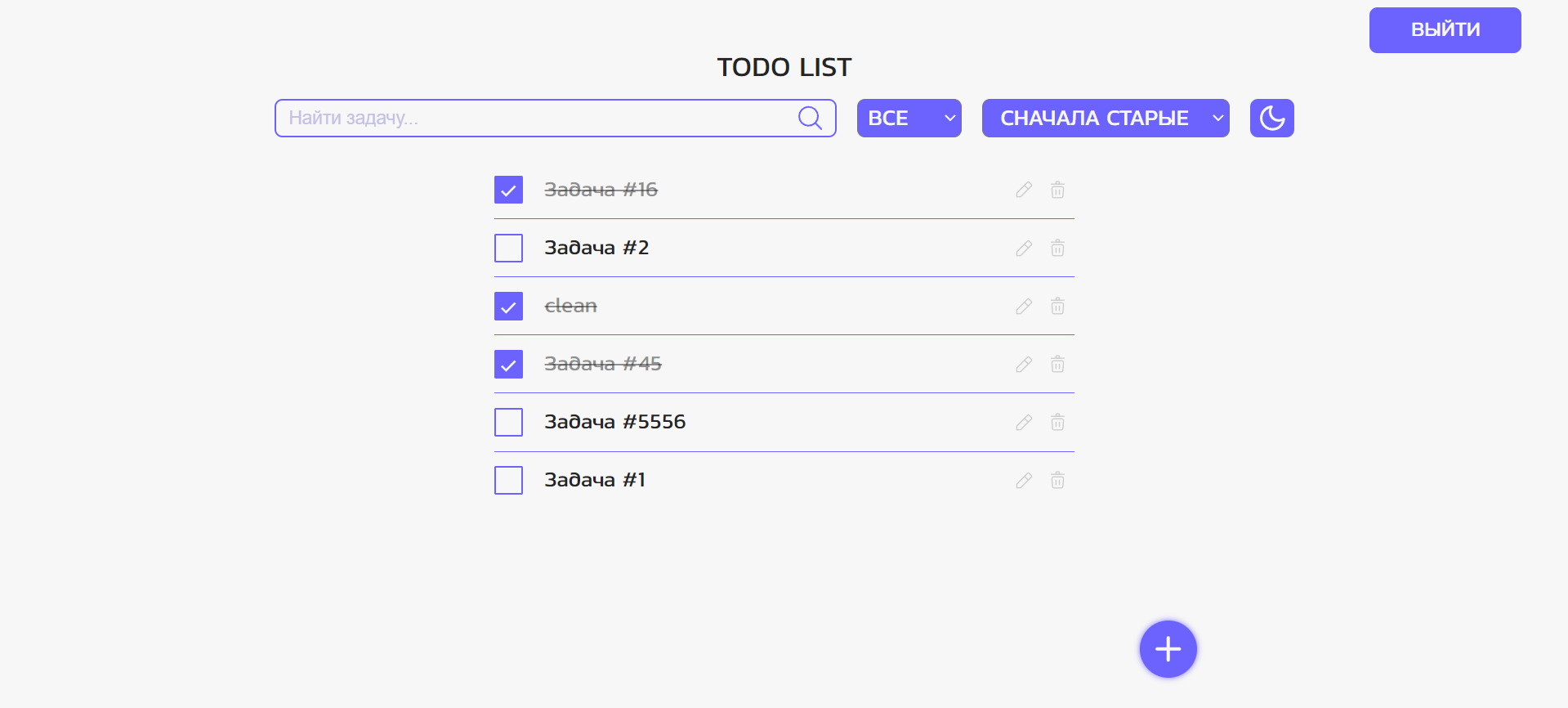


Рисунок 2.3.3 – Страница всех задач

Пользователь может воспользоваться поиском, фильтрацией и сортировкой для изменения отображения задач (Рисунок 2.3.4).

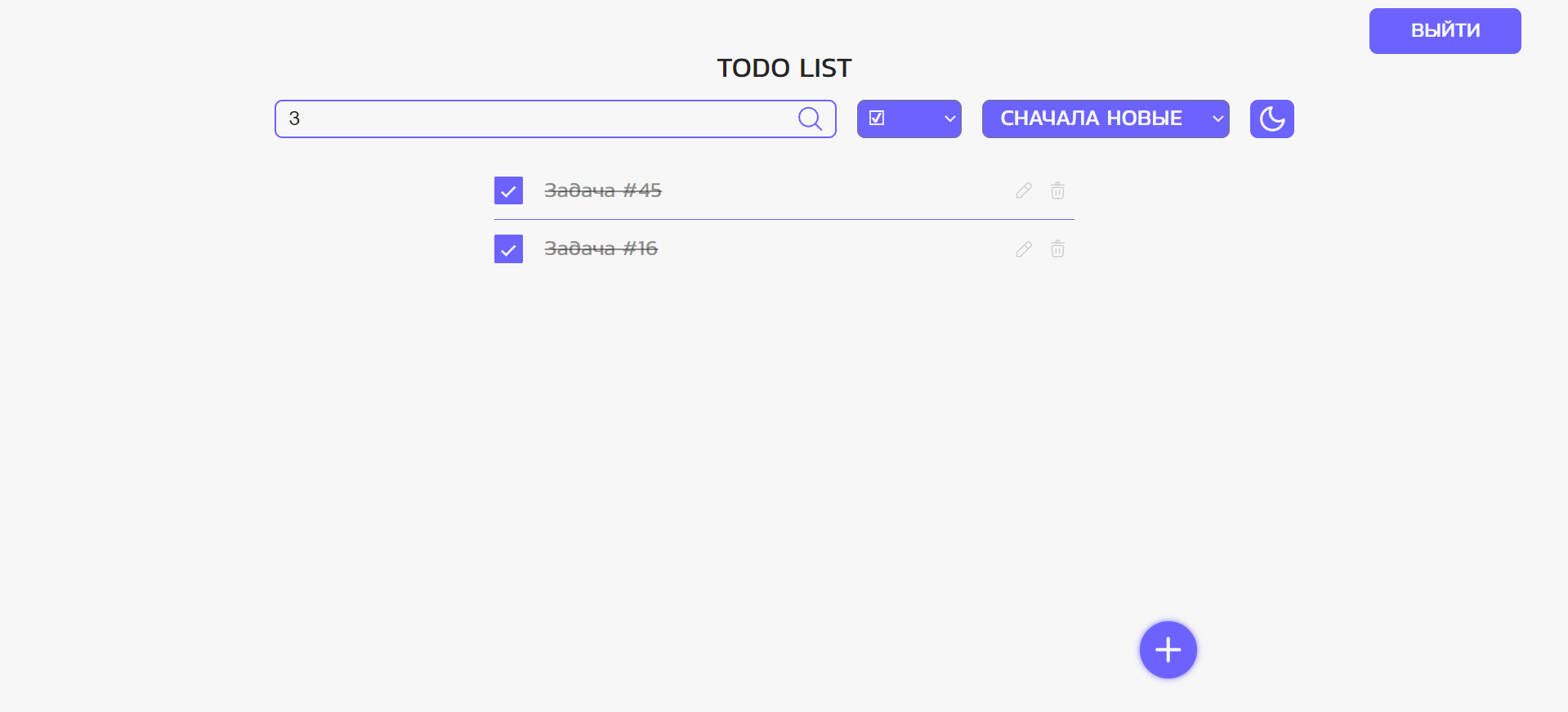


Рисунок 2.3.4 – Поиск, фильтрация и сортировка

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 2.3.5).

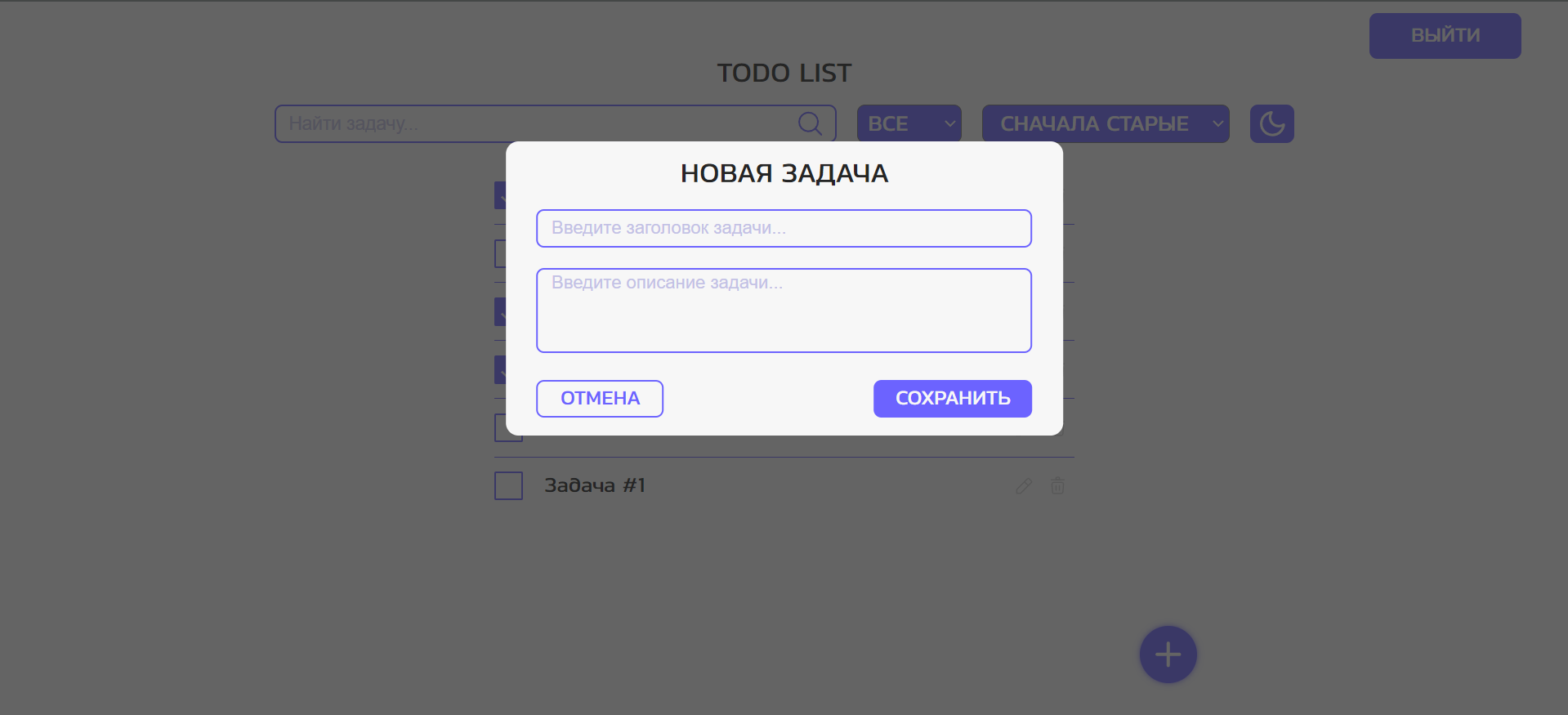


Рисунок 2.3.5 – Создание задачи

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 2.3.6).

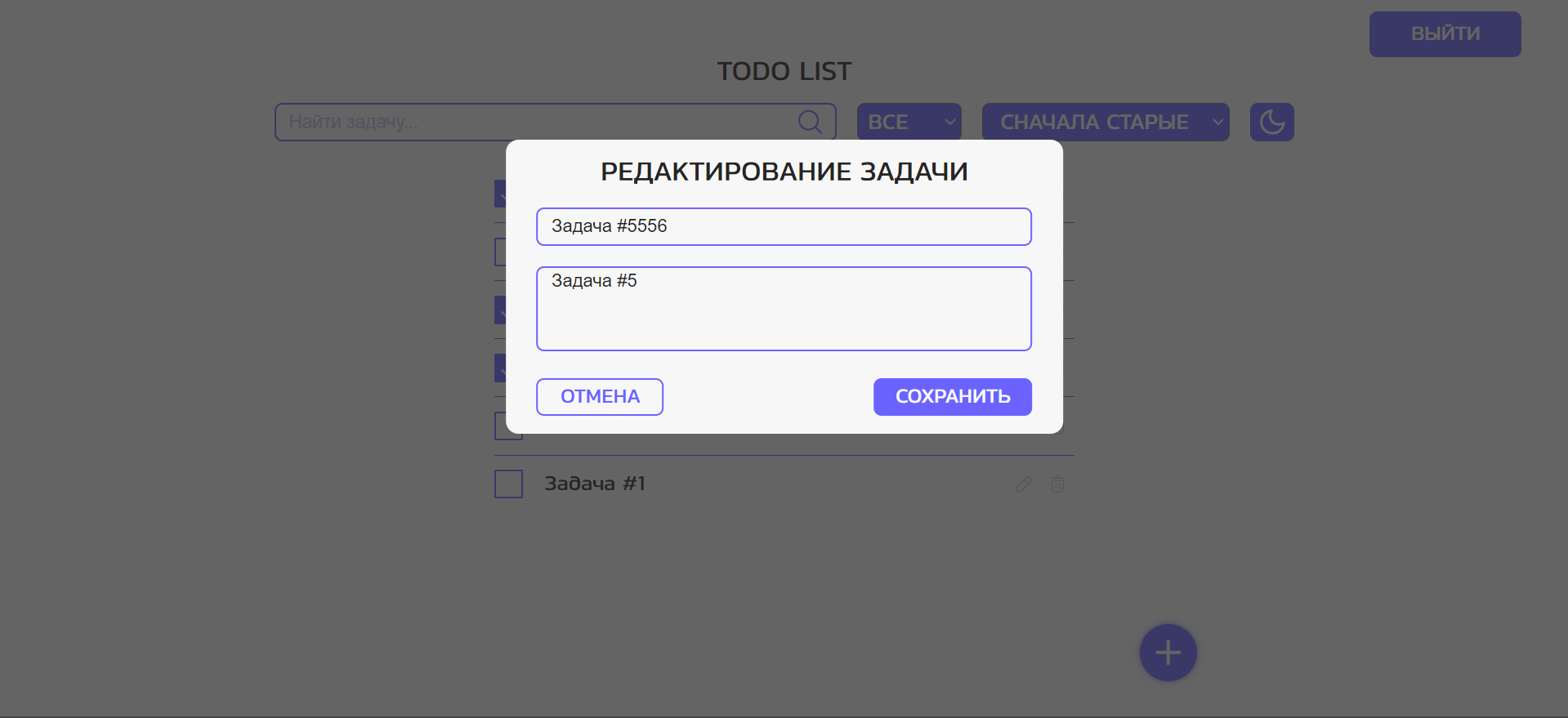


Рисунок 2.3.6 – Редактирование задачи

По нажатию на значок корзины возле соответствующей задачи пользователь может удалить задачу. После удаления будет выведено сообщение “Задача удалена” (Рисунок 2.3.7).

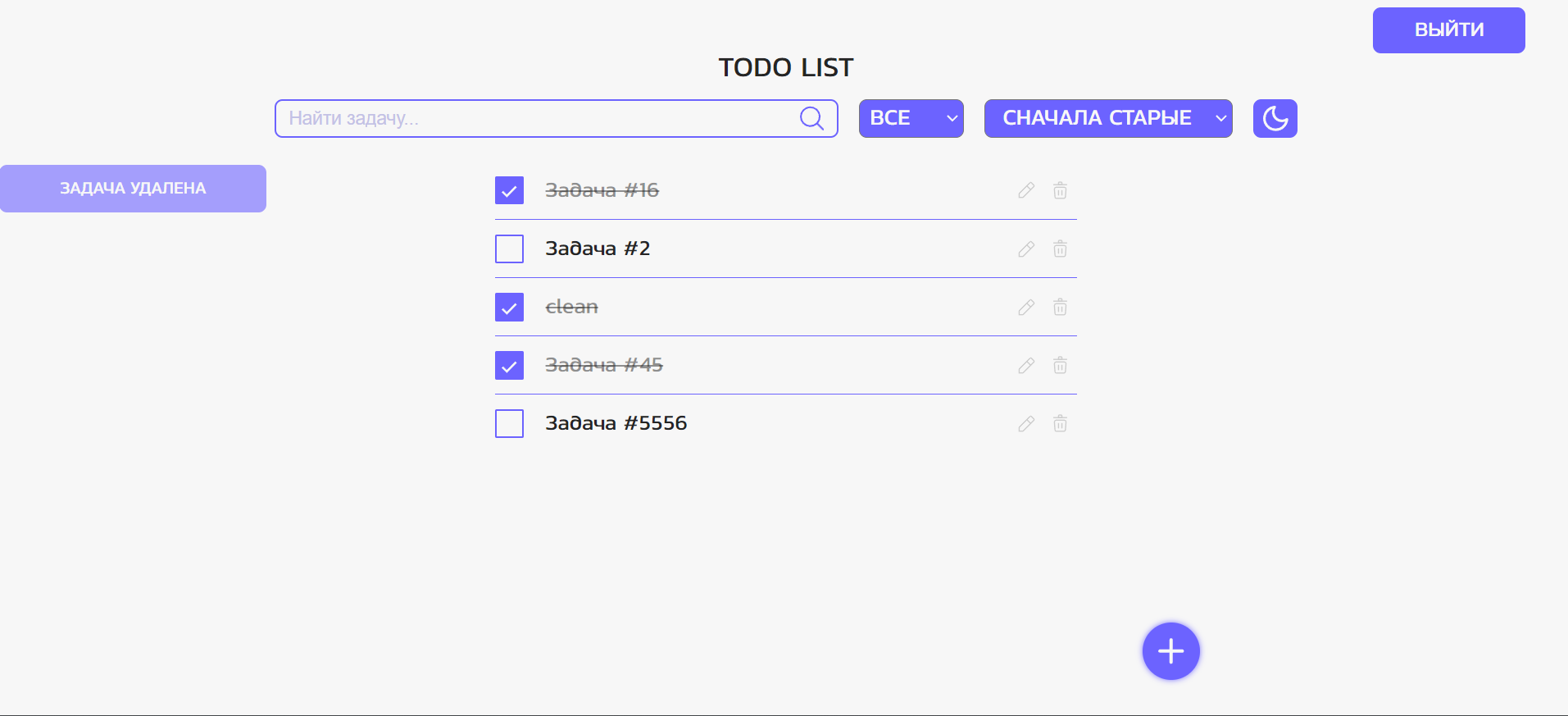


Рисунок 2.3.7 – Удаление задачи

Руководство программиста для веб-приложения "Todo List"

1. Введение

Todo List — это веб-приложение для отслеживания выполнения задач. Пользователи могут зарегистрироваться, авторизоваться, создавать задачи, редактировать их и удалять, а также отмечать их выполненными или невыполненными.

Основные функции и возможности:

- регистрация и аутентификация пользователей;

- создание, редактирование и удаление задач;

- изменение статуса выполнения задач;

- сортировка задач по новизне и по выполненности;

- фильтрация задач по выполненности;

- поиск задач по ключевым словам.

Технологический стек:

- Фронтенд: HTML, JavaScript, CSS.

- Бэкенд: PHP, JavaScript, JQuery.

- База данных: MySQL.

2. Архитектура приложения

2.1 Общая схема

Todo List использует классическую архитектуру клиент-сервер. Фронтенд — это SPA (Single Page Application), построенное на JQuery с использованием запросов ajax. Серверное приложение построено на PHP для обработки запросов. MySQL — база данных для хранения информации о пользователях и задачах.

2.2 Взаимодействие компонентов

* пользователь взаимодействует с интерфейсом через браузер, отправляя запросы к API;
* серверная часть обрабатывает запросы и взаимодействует с MySQL для получения или записи данных;
* результаты операций возвращаются в виде появляющихся уведомлений для пользователя.

2.3 Основные модели данных

users: представляет зарегистрированных пользователей.

- `id`: уникальный идентификатор пользователя;

- `username`: строка, имя пользователя, логин;

- `password\_hash`: строка, хешированный пароль пользователя.

tasks: представляет задачи пользователей.

- `id`: уникальный идентификатор задачи;

- `user\_id`: идентификатор пользователя, создавшего задачу;

- `title`: строка, заголовок задачи;

- `description`: строка, описание задачи;

- `is\_completed`: выполненность задачи;

- `created\_at`: дата создания задачи;

- `updated\_at`: дата последнего изменения задачи.

3. Разработка фронтенда

3.1 Установка окружения

Отсутствует.

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 API

Для взаимодействия с бэкендом используется JQuery. Пример ajax-запроса для удаления задачи:

```javascript

$.ajax({

type: "POST", // Метод отправки

url: "user/get\_edit.php", // Путь до php файла отправителя

data: {

search: request\_data,

filter: request\_data2,

sort: request\_data3,

get: id},

success: function (html) {

document.getElementById("shadow").style.display = "block";

if ( window.innerHeight > document.getElementById("body").scrollHeight) {

document.getElementById("shadow").style.height = `${window.innerHeight}px`;

}

else {

document.getElementById("shadow").style.height = `${document.getElementById("body").scrollHeight}px`;

}

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

document.getElementById("modal\_edit").style.display = "grid";

}

});

```

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 Установка окружения

Отсутствует.

4.2 Маршруты API

POST /user/auth.php

Авторизация пользователя. Принимает `username` и `password\_hash`, возвращает сообщение об успешной или неуспешной авторизации.

POST /user/reg.php

Регистрация пользователя. Принимает `username` и `password\_hash`, возвращает сообщение об успешной или неуспешной регистрации.

5. Тестирование

Отсутствует.