Руководство программиста для Веб-Приложения «Список дел» (Todo list).

1. Введение

Это веб-приложение построено на основе клиент-серверной архитектуры, где серверная часть реализована с использованием PHP, а фронтенд использует JavaScript и jQuery для отправки AJAX-запросов. Приложение также использует Bootstrap для стилизации интерфейса.

Todo list – это веб-приложение для создания задач, в нем пользователь может создавать свои задачи и по мере выполнения отмечать, как выполненные.

Основные функции:

* регистрация и авторизация пользователей;
* создание, редактирование и удаление заметок;
* Фильтрация и поиск на клиентской стороне
* отметки выполненных задач.

Технологический стек:

Фронтенд: HTML, CSS, JavaScript, AJAX.

Бэкенд: PHP 8.1, MySQL.

Сервер: Apache\_2.4-PHP\_8.0 - 8.1

2. Архитектура приложения

2.1 Общая схема

TodoList использует классическую архитектуру клиент-сервер.

MySQL— база данных для хранения информации о пользователях, постах и комментариях.

2.2 Взаимодействие компонентов

* Пользователь взаимодействует с интерфейсом через браузер, отправляя запросы к базе данных.
* Серверная часть обрабатывает запросы и взаимодействует с MySQL для получения или записи данных.
* Ответ возвращается в формате HTML

2.3 Основные модули данных

сonnect.php модуль для подключения к базе данных MySQL

register.php обработка запросов регистрации.

login.php обработка запросов авторизации

logout.php выход с аккаунта

delete-task.php обработка запросов для удаления задач

add-task.php обработка запросов для добавления задач

update-task.php управление статусом задачи

get-task.php получение задач с бд и логика фильтрации и поиска

edit\_task.php обработка запросов для редактирования задач

1. Разработка фронтенда
   1. Установка окружения

Для разработки фронтенда вам потребуется запустить локальный сервер

* 1. Структура проекта

Todo/

index.php страница для входа

reg.php страница для регистрации

todo\_list.php Страница вывода задач

connect.php страница подключения к бд

logout.php страница для выхода

database/

connect.php страница для подключения к базе данных

sign/

register.php обработка запросов регистрации.

login.php обработка запросов авторизации

logout.php выход с аккаунта

delete-task.php обработка запросов для удаления задач

add-task.php обработка запросов для добавления задач

update-task.php управление статусом задачи

get-task.php получение задач с бд и логика фильтрации и поиска

edit\_task.php обработка запросов для редактирования задач

* 1. Взаимодействие с сервером через AJAX

Пример отправки AJAX-запроса на сервер для удаления заметки:

`js

//script.js

function deleteTask(id) {

if (confirm('Вы уверены, что хотите удалить задачу?')) {

const xhr = new XMLHttpRequest();

xhr.open('POST', '../sign/delete\_task.php', true);

xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.onload = function() {

if (xhr.status !== 200) {

alert('Ошибка удаления задачи.');

} else {

alert('Задача успешно удалена.');

loadTasks(); // Перезагрузка списка задач

}

};

const data = `id=${id}`;

xhr.send(data);

}

}

* 1. Стилизация

Стилизация компонентов реализована с помощью файла style.css.

styles/style1.css

body {

font-family: 'Arial', sans-serif;

background-color: #F4F4F9;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

padding: 0;

}

.auth-container {

display: flex;

justify-content: center;

align-items: center;

width: 100%;

height: 100%;

}…

1. Разработка бэкенда

4.1 Подключение к базе данных(connect.php)

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

<?php

$conn = mysqli\_connect("localhost","root","","todo\_app");

4.2 Обработка данных на сервере

Пример обработки запроса на добавление заметки в `add-task.php`:

<?php

session\_start();

include '../database/connect.php';

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

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

echo json\_encode(['status' => 'error', 'message' => 'Пользователь не авторизован']);

exit();

}

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

$title = $\_POST['title'];

$note = $\_POST['note'];

$sql = "INSERT INTO tasks (user\_id, title, description) VALUES ('$userId', '$title', '$note')";

if (mysqli\_query($conn, $sql)) {

echo json\_encode(['status' => 'success', 'message' => 'Задача добавлена успешно!']);

} else {

echo json\_encode(['status' => 'error', 'message' => 'Ошибка при добавлении задачи.']);

}

exit();

}

?>  
4.3 Аутентификация (login.php)

При аутентификации происходит расхеширование пароля:

<?php

session\_start();

include '../database/connect.php';

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

$username = isset($\_POST['username']) ? $\_POST['username'] : '';

$password = isset($\_POST['password']) ? $\_POST['password'] : '';

if (!empty($username) && !empty($password)) {

$query = "SELECT id, password\_hash FROM users WHERE username = '$username'";

$result = mysqli\_query($conn, $query);

if (mysqli\_num\_rows($result) === 1) {

$row = mysqli\_fetch\_assoc($result);

$userId = $row['id'];

$hashedPassword = $row['password\_hash'];

if (password\_verify($password, $hashedPassword)) {

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

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

echo "<script>alert('Вы успешно вошли в систему!');

window.location.href='../todo\_list.php';</script>";

} else {

echo "<script>alert('Неверный логин или пароль.');

window.location.href='../index.php';</script>";

}

} else {

echo "<script>alert('Неверный логин или пароль.');

window.location.href='../index.php';</script>";

}

} else {

echo "<script>alert('Пожалуйста, заполните все поля.');

window.location.href='../index.php';</script>";

}

}

?>  
4.4 Структура базы данных

Пример таблиц базы данных:

CREATE TABLE users(

Id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50)

password\_hash VARCHAR(255)

)

CREATE TABLE tasks(

Id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id: INT

title VARCHAR(255)

Description TEXT

is\_completed TINYINT

created\_at TIMESTAMP

updated\_at TIMESTAMP

)

5. Тестирование

5.1 Тестирование на клиенте

1. Проверьте корректность удаления 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.

2. Убедитесь, что обработка ошибок на сервере работает корректно.

6. Развертывание

6.1 Настройка сервера

1. Убедитесь, что сервер поддерживает PHP 8.1 и MySQL.

2. Настройте виртуальный хост в Apache\_2.4-PHP\_8.0 - 8.1 для работы с проектом.

6.2 Развертывание на хостинге

1. Скопируйте файлы проекта на сервер.

2. Настройте файл конфигурации базы данных (`connect.php`) с корректными данными подключения.

3. Импортируйте структуру базы данных (SQL-файлы) на сервер.

7. Безопасность

7.1 Защита пароля происходит с помощью хеширования password\_hash():

$passwordHash = password\_hash($password, PASSWORD\_BCRYPT);