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 заметок, в нем пользователь может создавать свои заметки и по мере выполнения, отмечать как выполненные.

Основные функции:

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

Технологический стек:

Фронтенд: HTML, CSS, JavaScript, jQuery, Bootstrap, AJAX.

Бэкенд: PHP 8.1, MySQL.

Сервер: Apache\_2.4-PHP\_8.0 - 8.1

1. Архитектура приложения
   1. Общая схема

Todo List использует классическую архитектуру клиент-сервер.

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

* 1. Взаимодействие компонентов

Пользователь взаимодействует с интерфейсом через браузер обращаясь к php скриптам.

Серверная часть обрабатывает запросы и взаимодействует с MySQL для получения или записи данных и возвращает ответ в формате HTML.

* 1. Основные модули данных

connect.php - модуль для подключения к базе данных MySQL

signup-db.php - обработка запросов регистрации.

signin-db.php - обработка запросов авторизации

delete-task.php - обработка запросов для удаления заметки

status.php - управление статусом заметки

edit-db.php - обработка запросов для редактирования

* 1. Взаимодействие через AJAX

Взаимодействие через ajax происходит при удалении заметки.

1. Разработка фронтенда

3.1 Установка окружения

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

<!--bootstrap -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"integrity="sha384QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

<!-- jQuery -->

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

* 1. Структура проекта

TodoList/

index.php страница вывода карточек

signup.php страница для регистрации

signin.php страница для входа

logout.php страница для выхода

header.php страница с шапкой сайта

database/

connect.php страница подключения к бд

signup-db.php страница для обработки запросов регистрации.

signin-db.php страница для обработки запросов авторизации

delete-task.php страница для обработки запросов для удаления заметки

status.php страница для управления статусом заметки

edit-db.php страница для обработки запросов для редактирования

* 1. Взаимодействие с сервером через AJAX

Пример отправки AJAX-запроса на сервер для удаления заметки:

`javascript

//index.php

<script>

$(document).ready(function() {

$('.delete-task').on('click', function(e) {

e.preventDefault(); // Предотвращаем переход по ссылке

var taskId = $(this).data('id'); // Получаем ID задачи

$.ajax({

url: '/database/delete-task.php',

type: 'POST',

data: { id: taskId },

success: function(response) {

console.log(response); // Для отладки

if (response.trim() === 'success') {

alert('Заметка успешно удалена');

$('a.delete-task[data-id="' + taskId + '"]').closest('.card').fadeOut(300, function() {

$(this).remove();

});

} else {

alert('Ошибка при удалении заметки: ' + response);

}

},

error: function() {

alert('Ошибка при выполнении запроса');

}

});

});

});

</script>

* 1. Стилизация

Стилизация компонентов реализована с помощью Bootstrap и файла style.css.

css/style.css

1. .container{

display: flex;

justify-content: center;

font-family: Kanit;

font-size: 26px;

}

.search-bar {

position: relative;

width: 300px;

border-radius: 20px;

}…

Разработка бэкенда

4.1 Подключение к базе данных(connect.php)

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

<?php

$con = mysqli\_connect("localhost","root","","Todo list");

4.2 Обработка данных на сервере

Пример обработки запроса на добавление заметки в `db.php`:

<?php

require\_once "connect.php";

session\_start();

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

$\_SESSION["message"] = "Пользователь не авторизован!";

header('Location: ../index.php');

exit();

}

$title = isset($\_POST["title"]) ? $\_POST["title"] : false;

$description = isset($\_POST["description"]) ? $\_POST["description"] : false;

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

if ($title && $description) {

$stmt = $conn->prepare("INSERT INTO tasks (`user\_id`, `title`, `description`) VALUES ('$user\_id', '$title', '$description')");

if ($stmt->execute()) {

$\_SESSION["message"] = "Вы успешно добавили заметку!";

header('Location: ../index.php');

} else {

$\_SESSION["message"] = "Произошла ошибка, попробуйте снова!";

header('Location: ../index.php');

}

$stmt->close();

} else {

$\_SESSION["message"] = "Заполните все поля!";

header('Location: ../index.php');

}

?>

4.3 Аутентификация (signin-db.php)

При аутентификации происходит рехеширование пароля:

<?php

require\_once "connect.php";

session\_start();

$login = isset($\_POST["login"]) ? $\_POST["login"] : false;

$pass = isset($\_POST["pass"]) ? $\_POST["pass"] : false;

if ($login and $pass) {

    $sql = "SELECT \* FROM users WHERE username= '$login'";

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

    if (mysqli\_num\_rows($result) != 0) {

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

        if (password\_verify($pass, $user["password\_hash"])) {

            $\_SESSION["user\_id"] = $user["id"];

            $\_SESSION["message"] = "Добро пожаловать!";

            header('Location: ../index.php');

        } else {

            $\_SESSION["message"] = "Неверный пароль!";

            header('Location: ../signin.php');

        }

    } else {

        $\_SESSION["message"] = "Неверный логин!";

        header('Location: ../signin.php');

    }

} else {

    $\_SESSION["message"] = "Заполните все поля!";

    header('Location: ../signin.php');

}

?>

4.4 Структура базы данных

Пример таблиц базы данных:

CREATE TABLE User(

Id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(60)

password\_hash VARCHAR(40)

)

CREATE TABLE tasks(

Id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id: INT

title VARCHAR(100)

Description VARCHAR(150)

is\_completed TINYINT

created\_at TIMESTAMP

updated\_at TIMESTAMP

)

1. Тестирование
   1. Тестирование на клиенте
2. проверьте корректность удаления через AJAX;
3. убедитесь, что сервер корректно возвращает HTML ответы, которые обрабатываются на клиенте.
   1. Тестирование на сервере
      1. проверьте работу скриптов PHP с помощью тестовых данных;
      2. убедитесь, что обработка ошибок на сервере работает корректно.
4. Развертывание
   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.
   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.
5. Безопасность
   1. Защита пароля происходит с помощью хеширования (password\_hash()):

 $hashed\_password = password\_hash($pass, PASSWORD\_DEFAULT);

    $sql = "INSERT INTO users (username, password\_hash) VALUES ('$login', '$hashed\_password')";