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

1. Введение

Приложение разработано на основе клиент-серверной архитектуры с использованием PHP на сервере, JavaScript и jQuery на клиенте для взаимодействия с сервером и Bootstrap для стилизации интерфейса.

Todolist – Веб-приложение, которое позволяет создавать списки задач, добавлять заметки к ним и отмечать выполненные задачи.

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

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

Connect.php модуль для подключения к базе данных MySQL

signup-db.php - обработка запросов регистрации;

signin-db.php - обработка запросов авторизации;

delete.php - обработка запросов для удаления заметки;

edit.php - обработка запросов для редактирования.

1. Разработка фронтенда
   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">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"

integrity="sha384YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"crossorigin="anonymous"></script>

<!-- jQuery -->

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

* 1. Структура проекта

TODOLIST/

index.php страница для входа;

signup.php страница для регистрации;

user.php Страница вывода карточек;

exit.php страница для выхода;

header.php страница с шапкой сайта;

delete.php страница для обработки запросов для удаления заметки;

edit.php страница для обработки запросов для редактирования.

database/

Connect.php страница подключения к бд;

signup-db.php страница для обработки запросов регистрации;

signin-db.php страница для обработки запросов авторизации.

* 1. Взаимодействие с сервером через AJAX

Пример отправки AJAX-запроса на сервер для удаления заметки:

`javascript

//user.php

$(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: 'delete.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 index.css.

TODOLIST/css/index.css

@font-face {

font-family: "Kanit-ExtraBold";

src: url("/fonts/Kanit-ExtraBold.ttf");

}

\* {

margin: 0;

padding: 0;

}

input {

background-color: rgb(255, 255, 255);

color: #000;

border: 1px solid rgb(46, 82, 172);

border-radius: 5px;

padding: 5px;

}

.tema button {

display: flex;

background-image: url(images/sun.png);

background-size: cover;

}…

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", "", "TodoList");

4.2 Обработка данных на сервере

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

<?php

require\_once "database/Connect.php";

session\_start();

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

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

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

if ($title and $description) {

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

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

if ($result) {

$\_SESSION["message"] = "Успех!";

header("Location: /user.php");

} else {

$\_SESSION["message"] = "Ошибка создания заметки!";

header("Location: /user.php");

}

} else {

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

header("Location: /user.php");

}

4.3 Аутентификация (signin-db.php)

При аутентификации происходит расхеширование пароля:

<?php

require\_once "../header.php";

require\_once "Connect.php";

session\_start();

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

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

if ($login and $password) {

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

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

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

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

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

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

$\_SESSION["message"] = "Успех!";

header("Location: /user.php");

} else {

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

header("Location: /");

}

} else {

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

header("Location: /");

}

} else {

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

header("Location: /");

}

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

)

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()):

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

$sql = mysqli\_query($con, "INSERT INTO users (username, password\_hash) VALUES ('$login', '$passHash')");