Руководство программиста для веб-приложения «Список задач»

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.

Основные функции:

- регистрация и аутентификация пользователей;

- вывод задач с использованием фильтрации и сортировки;

- добавление, редактирование и удаление задач;

- асинхронная обработка запросов через ajax;

- изменение статуса задач.

Технологический стек:

- фронтенд: HTML, CSS, JavaScript, jQuery;

- бэкенд: PHP, MySQL;

- Сервер: Apache.

2 Архитектура приложения

2.1 Общая схема

Клиент отправляет AJAX-запросы из браузера на серверные скрипты PHP для обработки данных, таких как аутентификация, работа с базой данных и управление контентом.

2.2 Взаимодействие компонентов

1 Пользователь взаимодействует с веб-интерфейсом, отправляя запросы.

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

- Connect.php: модуль подключение к БД.

Методы Connect.php:

- \_ \_construct()- подключение к БД.

- Tasks.php: модуль для работы с базой данных, таблица tasks;

Методы Tasks.php:

- all\_tasks($user\_id, $search, $select, $newset, $passed)- вывод списка задач с поиском, фильтром и сортировкой;

- one\_task($id)- получение информации об одной записи по идентификатору;

- checked($id)- отметка задачи как выполненной по идентификатору;

- unchecked($id)- отметка задачи как невыполненной по идентификатору;

- create\_task($user, $title, $content)- создание задачи с получением id пользователя, названием задачи и описанием;

- edit\_task($id\_task, $title, $desc)- редактирование задачи с получением id пользователя, названием задачи и описанием;

- delete\_task($id)- удаление задачи по идентификатору.

- User.php: модуль работы с базой данных, таблица users;

Методы User.php:

- registration($login, $pass)- регистрация по полученным логину и паролю;

- authorization($login, $pass)- авторизация по полученным логину и паролю.

- redirection.php: обработка запросов пользователя.

3 Разработка фронтенда

3.1 Установка окружения

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

```html

<!-- jQuery -->

<script src="js\jquery-3.7.1.min.js"></script> ```

3.2 Структура проекта

```plaintext

/toDoListAjax

/css

style.css- файл со стилями;

/database

Connect.php- модуль подключения к БД;

Tasks.php- модуль обработки запросов к таблице tasks;

Users.php- модуль обработки запросов к таблице users;

/js

btns.js- обработка событий наведения мыши;

jquery-3.7.1.min.js- файл библиотеки jQuery;

/user

redirection.php- переадресация регистрации, авторизации и выхода;

/index.php- файл с шапкой и запросами AJAX;

/regAuth.php- файл с формами регистрации и авторизации;

/taskAct.php- файл, к которому обращается AJAX для обработки действий над задачами;

/tasks.php- файл, к которому обращается AJAX для обработки запросов и вывода информации на страницу.

```

3.3 Взаимодействие с сервером через AJAX

Пример отправки AJAX-запроса на сервер для выборки записей с фильтрацией по выполненности:

```javascript

// index.php

function select(){

let data = {

search: $('#searchHeader').val(),

select: $("#selectHeader").val(),

newest: $("#selectHeaderNewest").val(),

passed: $("#selectHeaderPassed").val(),

}

$.ajax({

method : "POST",

url: "tasks.php",

data: data,

success: function sele(response){

$('#toDos').html(response);

}

})

}

```

3.4 Стилизация

Для стилизации используется Bootstrap. Пример формы добавления поста:

```css

/toDoListAjax/css/style.css

#mainIndex{

display: flex;

flex-direction: row;

align-items: center;

justify-content: center;

width: 98vmax;

height: 48vmax;

gap: 3vmax;

}

```

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{

protected $host = 'localhost';

protected $user = 'root';

protected $pass ='';

protected $db = 'toDo';

protected $conn = null;

public function \_\_construct(){

$this->conn= mysqli\_connect($this->host, $this->user,$this->pass, $this->db);

return $this->conn;

}

}

?>

```

4.2 Обработка данных на сервере

Пример обработки запроса на добавление поста в `Tasks.php`:

```php

public function create\_task($user, $title, $content){

$query = mysqli\_query($this->conn, "INSERT INTO `tasks` (`user\_id`, `title`, `description`, `is\_completed`, `created\_at`, `updated\_at`) VALUES ($user, '$title', '$content', '0', NOW(), NOW());");

}

```

4.3 Аутентификация (auth.php)

Пример обработки логина пользователя через AJAX-запрос:

```php

<?php

session\_start();

print\_r($\_POST);

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

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

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

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

if(isset($\_GET['act'])){

if($\_GET['act'] == 'logout'){

unset($\_SESSION['login']);

unset($\_SESSION['user\_id']);

$\_SESSION['mess'] = "Вы вышли из аккаунта!";

header('Location: /toDoListAjax/regAuth.php');

}

}

else{

if($pass && $login){

if($\_POST['auth'] == 'false'){

$newUser = new User();

$newUser->registration($login, $pass);

}

else{

$user = new User();

$user->authorization($login, $pass);

}

}

else{

$\_SESSION['mess'] = 'Заполните все поля';

if($\_POST['auth'] == true){

header('Location: ../regAuth.php?auth=true');

}

else{

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

}

}

}

?>

/database

public function authorization($login, $pass){

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

if(mysqli\_num\_rows(mysqli\_query($this->conn, $check\_exist))>0){

$hashed\_password = mysqli\_fetch\_array(mysqli\_query($this->conn, $check\_exist))[2];

if(password\_verify($pass, $hashed\_password)){

$\_SESSION['user\_id'] = mysqli\_fetch\_array(mysqli\_query($this->conn, $check\_exist))[0];

$\_SESSION['mess'] = 'Вы успешно вошли в аккаунт!';

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

}

else{

$\_SESSION['mess'] = "Пароль введен не верно. Попробуйте снова!";

header('Location: ../regAuth.php?auth=true');

}

}

else{

$\_SESSION['mess'] = "Такого пользователя нет! Загеристрируйтесь!";

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

}

}

```

4.4 Структура базы данных

Пример таблиц базы данных:

```sql

CREATE TABLE `tasks` (

`id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`title` varchar(200) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`description` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

`is\_completed` enum('0','1') COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '0',

`exist` enum('0','1') COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '1',

`created\_at` timestamp NOT NULL,

`updated\_at` timestamp NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE `users` (

`id` int(11) NOT NULL,

`username` varchar(150) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`password\_hash` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

```

5 Тестирование

5.1 Тестирование на клиенте

1 Проверьте корректность работы форм и их отправку через AJAX.

2 Убедитесь, что сервер корректно возвращает 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 и MySQL.

2 Настройте виртуальный хост в Apache для работы с проектом.

3 Установите права на запись для директорий, где требуется запись данных.

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.