

ГУАП
КАФЕДРА № 43

ОНЛАЙН-ГАЛЕРЕЯ И СООБЩЕСТВО ХУДОЖНИКОВ ОБЩЕГО ТИПА
Руководство программиста

Санкт-Петербург 2024

АННОТАЦИЯ

Настоящий документ представляет собой руководство программиста (далее Руководство) онлайн-галереи и сообщества художников общего типа (далее Платформа).

Руководство определяет порядок развертывания системы и работы со списком пользователей, постов, категорий, комментариев.

Перед работой программиста с Платформой рекомендуется внимательно ознакомиться с настоящим руководством.

Документ подготовлен в соответствии с РД 50-34.698-90.

Содержание

АННОТАЦИЯ	2
1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ	4
1.1 НАЗНАЧЕНИЕ ПРОГРАММЫ.....	4
1.2 ФУНКЦИИ, ВЫПОЛНЯЕМЫЕ ПРОГРАММОЙ.....	4
1.3 УСЛОВИЯ, НЕОБХОДИМЫЕ ДЛЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ	5
1.3.1. ОБЪЕМ ОПЕРАТИВНОЙ ПАМЯТИ.....	5
1.3.2. ТРЕБОВАНИЯ К СОСТАВУ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ.....	5
1.3.3. ТРЕБОВАНИЯ К ПАРАМЕТРАМ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ.....	5
1.3.4. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ.....	5
1.3.5. ТРЕБОВАНИЯ К ПЕРСОНАЛУ (ПРОГРАММИСТУ).....	5
2. ХАРАКТЕРИСТИКА ПРОГРАММЫ.....	6
2.1 СОСТАВ И СОДЕРЖАНИЕ ДИСТРИБУТИВНОГО НОСИТЕЛЯ ДАННЫХ.....	6
2.2 ПОРЯДОК ПРОВЕРКИ РАБОТОСПОСОБНОСТИ.....	6
3. ОБРАЩЕНИЕ К ПРОГРАММЕ	6
3.1 ПАНЕЛЬ АДМИНИСТРАТОРА	7
3.2 УПРАВЛЕНИЯ ПОЛЬЗОВАТЕЛЯМИ	8
3.3 ФУНКЦИЯ КОММЕНТИРОВАНИЯ	17
4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ	24
4.1. ОРГАНИЗАЦИЯ ИСПОЛЬЗУЕМОЙ ВХОДНОЙ ИНФОРМАЦИИ	24
4.2. ОРГАНИЗАЦИЯ ИСПОЛЬЗУЕМОЙ ВЫХОДНОЙ ИНФОРМАЦИИ.....	24
СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ.....	25

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

1.1 Назначение программы

Платформа разработана для обеспечения информационной поддержки художников и любителей искусства. Она позволяет:

- Создавать персональные страницы для художников и размещать их работы.
- Формировать и управлять коллекциями работ.
- Вести личные заметки и записи о творческом процессе.
- Взаимодействовать с другими участниками сообщества через личные сообщения и группы по интересам.
- Обмениваться мнениями и идеями, участвовать в обсуждениях и оценивать работы других участников.

1.2 Функции, выполняемые программой

Платформа обеспечивает выполнение следующих основных функций для администратора:

- Создание, изменение и удаление, блокировка/разблокировка пользователей
- Создание, изменение и удаление, блокировка/разблокировка постов
- Создание, изменение и удаление категорий
- Создание, изменение и удаление комментариев

Платформа предоставляет широкий спектр возможностей для удобной работы пользователей:

- Создание, редактирование и удаление профиля художника, включая информацию о себе, своем творчестве и контактных данных.
- Загрузка и публикация художественных работ в галерею с указанием названия, описания и характеристик.
- Формирование и управление категориями.
- Ведение личного блога или журнала, где художник может делиться своими мыслями, идеями и творческим процессом.

- Возможность добавления других пользователей в друзья и участие в обсуждениях.

Эти функции позволяют художникам эффективно представлять свое творчество, взаимодействовать с другими участниками сообщества и находить вдохновение для новых проектов.

1.3 Условия, необходимые для выполнения программы

Платформа может эксплуатироваться и выполнять заданные функции при соблюдении требований, предъявляемых к техническому, системному и прикладному программному обеспечению.

1.3.1. Объем оперативной памяти

Рекомендуемый объем оперативной памяти 2 Гб или выше.

1.3.2. Требования к составу периферийных устройств

Особые требования к составу периферийных устройств не предъявляются.

1.3.3. Требования к параметрам периферийных устройств

Подключаемые периферийные устройства должны быть IBM-совместимыми.

1.3.4. Требования к программному обеспечению

Системные программные средства, используемые Платформой, должны быть представлены локализованной версией операционной системы Windows 7 и выше.

1.3.5. Требования к персоналу (программисту)

Программист должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы, должен быть аттестован минимум на II квалификационную группу по электробезопасности, должен иметь квалификацию «Пользователь ЭВМ».

2. ХАРАКТЕРИСТИКА ПРОГРАММЫ

2.1 Состав и содержание дистрибутивного носителя данных

Для запуска Платформы используется Denwer, хостинг не применяется. Необходимые для этого компоненты (СУБД MySQL, phpMyAdmin, PHP) не входят в комплект поставки. У программиста должно быть установлено IDE для работы с html, css, js и php.

В комплект поставки Платформы входит Руководство пользователя и Руководство программиста.

2.2 Порядок проверки работоспособности

Проверка работоспособности Платформы осуществляется путем выполнения операций, описанных в разделе 4 Руководства пользователя.

3. ОБРАЩЕНИЕ К ПРОГРАММЕ

В данном разделе приводится описание всех операций, существующих в Платформе для администратора.

Таблица 1. Роли и права доступа к данным и операциям

Роль	Доступные разделы	Доступные действия
Администратор	Пользователи	Просмотр, создание, изменение, удаление, блокировка/разблокировка
	Посты	Просмотр, создание, изменение, удаление, блокировка/разблокировка
	Комментарии	Просмотр, создание, изменение, удаление
	Категории	Просмотр, создание, изменение, удаление

Работа с разделами Пользователи, Посты и Комментарии, Категории, кроме блокировки/разблокировки и изменения Пользователей и Категорий, описаны в Руководстве пользователя.

3.1 Панель администратора

Посты	Добавление		Управление		
	Управление постами				
	ID	Название	Автор	Управление	
	41	Луна: От Аполлона до...	сухой-сахар	edit	delete block
	42	Панельные Дома: Мифы...	сухой-сахар	edit	delete block
Пользователи	43	Березовая Гармония	сухой-сахар	edit	delete block
	44	Эпоха Венеции: Дама ...	сухой-сахар	edit	delete block
	45	mnhj	сухой-сахар	edit	delete unblock
Категории					

Рисунок 1 – Создание аккаунта

Посты	Добавление		Управление	
	Управление пользователями			
	ID	Логин	Роль	Управление
	24	сухой-сахар	Admin	edit delete block
	25	user	User	edit delete block
Категории	31	user2	User	edit delete block
	32	polina	User	edit delete block

Рисунок 2 – Создание аккаунта

Посты	Добавление		Управление	
	Управление категориями			
	ID	Название	Описание	Управление
	17	Лунафувв	риотрльд	edit delete
	18	Без категории	пост не имеет катего...	
Пользователи	19	Какая-то очень длинн...	Какая-то очень длинн...	edit delete
	20	Астрономия	Наука, изучающая неб...	edit delete
	21	Архитектура и Жилье	Архитектура - это ис...	edit delete
Категории				

Рисунок 3 – Создание аккаунта

На странице отображается боковая панель, на которой отмечены вкладки: посты, пользователи, категории. При нажатии на "Посты" произойдет переход на страницу с управлением постами. При нажатии на "Пользователи" произойдет переход на страницу с управлением пользователями. При нажатии на "Категории" произойдет переход на страницу с управлением категориями. На страницах управления представлены кнопки "Добавление" и "Управление". После нажатия на "Управление", происходит вывод данных из базы в таблицу с колонками в зависимости от выбранной вкладки. Если это посты - ид, название, автор, управление; если пользователи – ид, логин, роль, управление; если категории – ид, название, описание, управление. В колонке управление указываются следующие кнопки: edit – редактирование, delete – удаление, block/unblock –

блокировка/разблокировка. Категорию "Без названия" нельзя удалить, она является шаблонной.

3.2 Управления пользователями

Посты

Пользователи

Категории

Добавление

Управление

Добавление пользователя

Имя пользователя:
Логин

Пароль:
Пароль

Повтор ввода:
Повторите пароль

Электронная почта:
Email

☐ Admin

Добавить

Рисунок 4 – Добавление пользователя

При нажатии на кнопку "Добавить" произойдет переход на страницу добавления пользователя. Администратор может выбрать в чекбоксе будет ли пользователь администратором, если галочка не проставлена – то это обычный пользователь. После нажатия на кнопку "Добавить", если все данные прошли проверку, пользователь будет добавлен, если нет – отобразится ошибка с описанием проблемы.

Реализация функции на php:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['user-create'])) {
    //var_dump($_POST);
    //exit();
    $login = $_POST['login'];
    $pass = $_POST['pass'];
    $repeatpass = $_POST['repeatpass'];
    $email = $_POST['email'];
    if (isset($_POST['isAdmin'])) {
        $role = 0;
    } else {
        $role = 1;
    }
}
```



```

$sql = "SELECT * FROM `registeruser` WHERE login = '$login'";
$resultLog = $conn->query($sql);

$sql = "SELECT * FROM `registeruser` WHERE email = '$email'";
$resultEmail = $conn->query($sql);

if ($resultLog->num_rows > 0) {
    while ($row = $resultLog->fetch_assoc()) {
        $errorMessage = "Пользователь " . $row['login'] . " уже
зарегистрирован!";
    }
} elseif ($resultEmail->num_rows > 0) {
    while ($row = $resultEmail->fetch_assoc()) {
        $errorMessage = "Пользователь с email: " . $row['email'] . " уже
зарегистрирован!";
    }
} elseif ($pass != $repeatpass) {
    $errorMessage = "Пароли не совпадают";
} elseif ((mb_strlen($login, 'UTF8') > 25)) {
    $errorMessage = "Логин пользователя должен быть до 25-и
символов!";
} elseif ((mb_strlen($pass, 'UTF8') > 50)) {
    $errorMessage = "Пароль пользователя должен быть до 50-и
символов!";
} elseif ((mb_strlen($email, 'UTF8') > 50)) {
    $errorMessage = "Email пользователя должен быть до 50-и
символов!";
} else {
    $hashedPass = md5($pass);

```

```

        $sql = "INSERT INTO `registeruser` (role, login, pass, email)
VALUES ('$role', '$login','$hashedPass','$email')";

        if ($conn->query($sql)) {

            $userId = $conn->insert_id;

            $sql = "SELECT * FROM `registeruser` WHERE id = '$userId'";

            $result = $conn->query($sql);

            if ($result) {

                $successMessage = "Пользователь успешно зарегистрирован!";

                // Очистите поля после успешной регистрации

                $login = "";

                $email = "";

            } else {

                $errorMessage = "Ошибка: " . $conn->error;

            }

        }

    } else {

        $login = "";

        $email = "";

    }
}

```

1. Проверка метода запроса и наличия ключа **user-create** в массиве **\$_POST**:

```

if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['user-create'])) { // Код обработки данных } else { //
Устанавливаются значения по умолчанию для переменных $login и $email }

```

Это условие проверяет, что запрос отправлен методом POST и в нем присутствует ключ **user-create**.

2. Получение данных из массива **\$_POST**:

```
$login = $_POST['login']; $pass = $_POST['pass']; $repeatpass =  
$_POST['repeatpass']; $email = $_POST['email'];
```

Эти строки извлекают значения, отправленные формой, в соответствующие переменные.

3. Определение роли пользователя:

```
if (isset($_POST['isAdmin'])) { $role = 0; // Администратор } else { $role  
= 1; // Обычный пользователь }
```

Если в форме присутствует чекбокс **isAdmin**, то пользователь будет создан с ролью администратора, иначе - с ролью обычного пользователя.

4. Проверка наличия логина и email в базе данных:

```
$sql = "SELECT * FROM `registeruser` WHERE login = '$login'";  
$resultLog = $conn->query($sql); $sql = "SELECT * FROM `registeruser`  
WHERE email = '$email'"; $resultEmail = $conn->query($sql);
```

Эти запросы проверяют, есть ли уже пользователь с таким логином или email в базе данных.

5. Проверка условий валидации:

```
if ($resultLog->num_rows > 0) { // Пользователь с таким логином уже  
существует } elseif ($resultEmail->num_rows > 0) { // Пользователь с таким  
email уже существует } elseif ($pass != $repeatpass) { // Пароли не совпадают }  
elseif ((mb_strlen($login, 'UTF8') > 25)) { // Логин пользователя должен быть  
до 25 символов } elseif ((mb_strlen($pass, 'UTF8') > 50)) { // Пароль  
пользователя должен быть до 50 символов } elseif ((mb_strlen($email, 'UTF8')  
> 50)) { // Email пользователя должен быть до 50 символов } else { // Все  
данные валидны, можно провести регистрацию }
```

Этот блок проверяет различные условия, такие как уникальность логина и email, совпадение паролей и длину данных.

6. Регистрация пользователя:

```
$hashedPass = md5($pass); $sql = "INSERT INTO `registeruser` (role,  
login, pass, email) VALUES ('$role', '$login', '$hashedPass', '$email')";
```

Здесь пароль хэшируется с помощью функции **md5()** и затем данные пользователя вставляются в базу данных.

7. Очистка полей после успешной регистрации:

```
$login = ""; $email = "";
```

Поля **\$login** и **\$email** очищаются для предотвращения повторной отправки данных при следующем запросе.

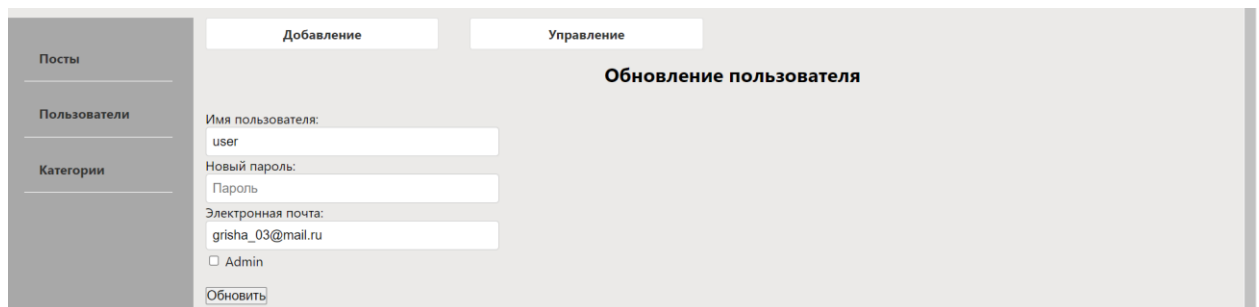


Рисунок 5 – Обновление пользователя

При нажатии на кнопку "edit" произойдет переход на страницу редактирования пользователя, старые данные пользователя автоматически введутся в поля. Администратор может выбрать в чекбоксе будет ли пользователь администратором, если галочка не проставлена – то это обычный пользователь. После нажатия на кнопку "Обновить", если все данные прошли проверку, пользователь будет обновлен, если нет – отобразится ошибка с описанием проблемы.

При нажатии на кнопку "delete" произойдет удаление пользователя, все его работы будут заблокированы.

Реализация функции на php:

```
if ($_SERVER['REQUEST_METHOD'] === 'GET' && (isset($_GET['id'])  
|| (isset($_GET['us_id'])))) {  
    if (isset($_GET['id'])) {  
        $id = $_GET['id'];  
    } else {  
        $id = $_GET['us_id'];  
    }  
}
```

```

$sql = "SELECT * FROM `registeruser` WHERE id = '$id'";
$result = $conn->query($sql);
$post = $result->fetch_assoc();
$login = $post['login'];
$email = $post['email'];
$role = $post['role'];
}

```

//редактирование пользователя

```

if      ($_SERVER['REQUEST_METHOD']      ===      'POST'      &&
isset($_POST['user-edit'])) {
    //var_dump($_POST);
    //exit();
    $id = $_POST['id'];
    $login = $_POST['login'];
    $email = $_POST['email'];
    $pass = $_POST['pass'];
    if (isset($_POST['isAdmin'])) {
        $role = 0;
    } else {
        $role = 1;
    }
    $sql = "SELECT * FROM `registeruser` WHERE login = '$login'";
    $resultLog = $conn->query($sql);

    $sql = "SELECT * FROM `registeruser` WHERE email = '$email'";
    $resultEmail = $conn->query($sql);
    $err = 0;

```

```

if ($resultLog->num_rows > 0) {
    while ($row = $resultLog->fetch_assoc()) {
        if ($row['id'] != $id) {
            $errorMessage = "Пользователь " . $row['login'] . " уже
зарегистрирован!";
            $err = 1;
        }
    }
}

if ($resultEmail->num_rows > 0) {
    while ($row = $resultEmail->fetch_assoc()) {
        if ($row['id'] != $id) {
            $errorMessage = "Пользователь с email: " . $row['email'] . " уже
зарегистрирован!";
            $err = 1;
        }
        elseif ((mb_strlen($login,'UTF8')>25)){
            $errorMessage = "Логин пользователя должен быть до 25-и
СИМВОЛОВ!";
            $err = 1;
        }
        elseif ((mb_strlen($pass,'UTF8')>50)){
            $errorMessage = "Пароль пользователя должен быть до 50-и
СИМВОЛОВ!";
            $err = 1;
        }
        elseif ((mb_strlen($email,'UTF8')>50)){
            $errorMessage = "Email пользователя должен быть до 50-и
СИМВОЛОВ!";
            $err = 1;
        }
    }
}

```

```

        }
    }
}
if (!$err) {
    if (!empty($pass)) {
        $hashedPass = md5($pass);
        if (isset($_POST['us_id'])) {
            $sql = "UPDATE `registeruser` SET login='$login',
pass='$hashedPass', email='$email' WHERE id = '$id'";
        } else {
            $sql = "UPDATE `registeruser` SET login='$login',
pass='$hashedPass', role='$role', email='$email' WHERE id = '$id'";
        }
    } else {
        if (isset($_POST['us_id'])) {
            $sql = "UPDATE `registeruser` SET login='$login', email='$email'
WHERE id = '$id'";
        } else {
            $sql = "UPDATE `registeruser` SET login='$login', role='$role',
email='$email' WHERE id = '$id'";
        }
    }
    $successMessage = "Данные пользователя обновлены!";
    if ($conn->query($sql) === TRUE && !(isset($_POST['us_id']))) {
        header("location: /admin/users/index.php");
    } elseif (isset($_POST['us_id'])) {
        $successMessage = "Данные пользователя обновлены!";
        $_SESSION['login'] = $login;
    } else {

```

```

        $errorMessage = "Ошибка при обновлении пользователя: " .
$conn->error;
    }
}
}

```

1. Проверка метода запроса и наличия параметров **id** или **us_id** в массиве **\$_GET**:

```

if ($_SERVER['REQUEST_METHOD'] === 'GET' && (isset($_GET['id'])
|| isset($_GET['us_id']))) { // Код для обработки данных }

```

Это условие проверяет, что запрос отправлен методом GET и содержит либо параметр **id**, либо **us_id**.

2. Получение данных пользователя для редактирования:

```

if (isset($_GET['id'])) { $id = $_GET['id']; } else { $id = $_GET['us_id']; }
$sql = "SELECT * FROM `registeruser` WHERE id = '$id'"; $result = $conn-
>query($sql); $post = $result->fetch_assoc(); $login = $post['login']; $email =
$post['email']; $role = $post['role'];

```

В этой части скрипта данные пользователя извлекаются из базы данных на основе переданного **id** или **us_id**.

3. Обработка редактирования пользователя:

```

if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['user-edit'])) { // Код для обработки редактирования }

```

Это условие проверяет, что запрос отправлен методом POST и содержит параметр **user-edit**, что указывает на то, что отправлена форма для редактирования данных пользователя.

4. Проверка наличия пользователя с таким логином и email:

```

if ($resultLog->num_rows > 0) { // Пользователь с таким логином уже
существует } elseif ($resultEmail->num_rows > 0) { // Пользователь с таким
email уже существует } elseif ((mb_strlen($login,'UTF8')>25)){ // Логин
пользователя должен быть до 25 символов } elseif
((mb_strlen($pass,'UTF8')>50)){ // Пароль пользователя должен быть до 50

```



```
символов } elseif ((mb_strlen($email,'UTF8')>50)){ // Email пользователя  
должен быть до 50 символов }
```

Этот блок проверяет уникальность логина и email, а также их длину.

5. Обновление данных пользователя:

```
$sql = "UPDATE `registeruser` SET login='$login', pass='$hashedPass',  
role='$role', email='$email' WHERE id = '$id'";
```

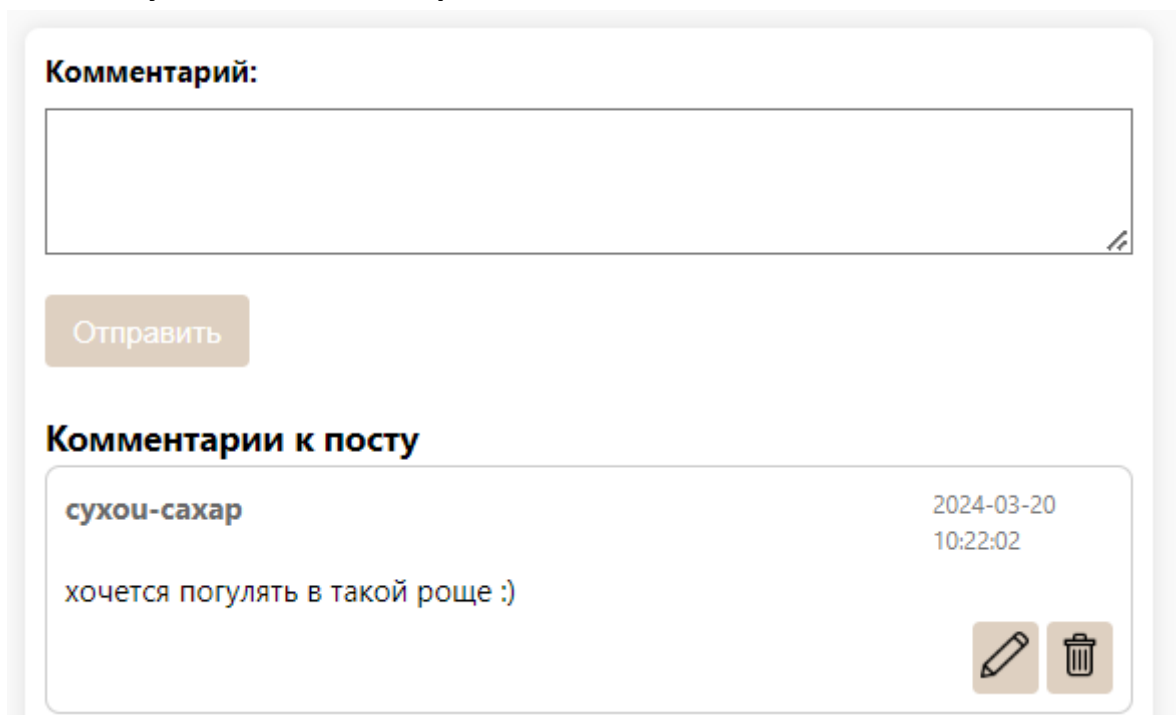
Этот SQL запрос обновляет данные пользователя в базе данных на основе переданных параметров.

6. Перенаправление пользователя после успешного обновления:

```
header("location: /admin/users/index.php");
```

Этот код перенаправляет пользователя на страницу **/admin/users/index.php** после успешного обновления данных, если не был указан параметр **us_id**.

3.3 Функция комментирования



The screenshot shows a web interface for comments. At the top, there is a section titled "Комментарий:" with a large text input field and a small "Отправить" (Send) button. Below this is a section titled "Комментарии к посту" (Comments to the post). It displays a single comment from a user named "сухой-сахар" (dry-sugar) posted on "2024-03-20 10:22:02". The comment text is "хочется погулять в такой роще :)" (want to walk in such a forest :). To the right of the comment text are two icons: a pencil for editing and a trash can for deleting.

Рисунок 6 – Комментарий

Под постом находится поле с комментарием, оно доступно только для авторизованных пользователей, если комментариев под постом нет – то посетитель сайта ничего не видит после строки просмотров. В комментариях

никнейм пользователя является кликабельным, справа отображается дата и время создания.

Реализация функции на php:

```
if      ($_SERVER['REQUEST_METHOD']      ===      'POST'      &&
isset($_POST['goComment'])) {
    //var_dump($_SESSION);
    //exit();

    $userId = $_POST['userId'];
    $login = $_POST['login'];
    $comment = trim($_POST['comment']);
    if (!(empty($comment))) {
        if (mb_strlen($comment, 'UTF8') < 3) {
            $errorMessage = "Комментарий должен быть длиннее трёх
СИМВОЛОВ!";
        } else {
            $sql = "INSERT INTO `comments` (post_id, user_id, comment)
VALUES ('$postId','$userId', '$comment')";
            if ($conn->query($sql)) {
                $successMessage = "Комментарий добавлен!";
            } else {
                $errorMessage = "Ошибка: " . $conn->error;
            }
        }
    } else {
        $errorMessage = "Комментарий не должен быть пуст";
    }
} else {
    $userId = "";
    $login = "";
    $comment = "";
}
```

```
}
```

Проверка метода запроса и наличия параметра **goComment** в массиве **\$_POST**:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&  
isset($_POST['goComment'])) { // Код для обработки данных } else { //  
Установка значений по умолчанию для переменных }
```

Это условие проверяет, что запрос отправлен методом POST и содержит параметр **goComment**.

Получение данных из массива **\$_POST**:

```
$userId = $_POST['userId']; $login = $_POST['login']; $comment =  
trim($_POST['comment']);
```

Здесь извлекаются данные, отправленные формой, такие как ID пользователя, его логин и комментарий.

Проверка введенного комментария:

```
if (!(empty($comment))) { if (mb_strlen($comment, 'UTF8') < 3) {  
$errorMessage = "Комментарий должен быть длиннее трёх символов!"; } else  
{ // Код для добавления комментария в базу данных } } else { $errorMessage =  
"Комментарий не должен быть пустым"; }
```

Этот блок проверяет, не пуст ли комментарий, и имеет ли он минимальную длину в три символа.

Добавление комментария в базу данных:

```
$sql = "INSERT INTO `comments` (post_id, user_id, comment) VALUES  
('$postId','$userId', '$comment')"; if ($conn->query($sql)) { $successMessage =  
"Комментарий добавлен!"; } else { $errorMessage = "Ошибка: " . $conn->error;  
}
```

Если комментарий прошел проверку и введен корректно, он добавляется в базу данных.

Установка значений по умолчанию, если запрос не был отправлен:

```
} else { $userId = ""; $login = ""; $comment = ""; }
```

Если запрос не был отправлен методом POST, переменные устанавливаются в пустые значения.

Комментарии к посту

сухой-сахар

2024-03-20
10:22:02

хочется погулять в такой роще :)

хочется погулять в такой роще :) upd погода как раз располагает!

Сохранить изменения





Рисунок 7 – Редактирование комментария

Если вы администратор, то вам доступно обновление или удаление данного комментария. После обновления дата также будет обновлена.

Комментарии к посту

сухой-сахар

2024-04-10
00:06:43

хочется погулять в такой роще :) upd погода как раз располагает!





Рисунок 8 – Отредактированный комментарий

Реализация функции на php:

```
if ($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['edit'])) {  
  
    $id = $_GET['edit'];  
  
    $sql = "SELECT * FROM `comments` WHERE id = '$id'";  
  
    $result = $conn->query($sql);
```

```

$post = $result->fetch_assoc();

$id = $post['id'];

$post_id = $post['post_id'];

$user_id = $post['user_id'];

$comment = trim($post['comment']);

$img = $post['img'];
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit'])) {

    $id = $_POST['comment_id_to_edit'];

    $comment = trim($_POST['comment']);

    $post_id = $_POST['post_id'];

    $url = "http://art-display/single.php?post=" . $post_id;

    if (!(isset($comment))) {

        if (mb_strlen($comment, 'UTF8') < 3) {

            $errorMessage = "Комментарий должен быть длиннее трёх символов!";

        } else {

            $sql = "UPDATE `comments` SET comment='$comment' WHERE
id='$id'";

            $result = $conn->query($sql);

        }

    } else {

```

```

        $errorMessage = "Комментарий не должен быть пуст";
    }
}

```

1. Проверка метода запроса и наличия параметра **edit** в массиве **\$_GET**:

```

if ($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['edit'])) { //
Код для обработки данных }

```

Это условие проверяет, что запрос отправлен методом GET и содержит параметр **edit**.

2. Получение данных комментария для редактирования:

```

$id = $_GET['edit']; $sql = "SELECT * FROM `comments` WHERE id = '$id'";
$result = $conn->query($sql); $post = $result->fetch_assoc(); $id = $post['id'];
$post_id = $post['post_id']; $user_id = $post['user_id']; $comment =
trim($post['comment']); $img = $post['img'];

```

В этой части скрипта данные комментария извлекаются из базы данных на основе переданного **id**.

3. Обработка редактирования комментария:

```

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit'])) {
// Код для обработки редактирования }

```

Это условие проверяет, что запрос отправлен методом POST и содержит параметр **edit**, что указывает на то, что отправлена форма для редактирования комментария.

4. Проверка введенного комментария:

```

if (!(isset($comment))) { if (mb_strlen($comment, 'UTF8') < 3) { $errorMessage =
"Комментарий должен быть длиннее трёх символов!"; } else { // Код для

```

```
обновления комментария в базе данных } } else { $errorMessage =  
"Комментарий не должен быть пуст"; }
```

Этот блок проверяет, не пуст ли комментарий, и имеет ли он минимальную длину в три символа.

5. Обновление комментария в базе данных:

```
$sql = "UPDATE `comments` SET comment='$comment' WHERE id='$id';"  
$result = $conn->query($sql);
```

Если комментарий прошел проверку и введен корректно, он обновляется в базе данных.

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1. Организация используемой входной информации

Входная информация может быть представлена в виде:

- Ввода с клавиатуры.
- Использования оптического манипулятора типа «мышь».
- Получения изображений от устройств.

4.2. Организация используемой выходной информации

Выходная информация может быть реализована в виде:

- Вывод в виде комментария.
- График со статистикой пользователя.
- Вывод на сайт в виде поста.
- Передачи данных в базу данных.

СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

Платформа	– онлайн-галерея и сообщество художников общего типа
СУБД	– система управления базами данных