

5. РАЗРАБОТКА ПЛАТФОРМЫ

5.1 Функции общей работы

Данная подсистема должна выполнять следующие функции:

- возможность входа пользователя в аккаунт
- возможность регистрации пользователя
- возможность просмотра постов
- возможность просмотра категорий
- возможность просмотра избранного
- возможность просмотра популярных постов
- возможность поиска
- возможность просмотра страницы "О нас"
- возможность скачивание документов "Условия пользования",

"Пользовательское соглашение"

5.1.1 Функция главной страницы

5.1.1.1 Описание

Позволяет пользователям использовать функционал сайта.

Приоритет: высокий.

5.1.1.1.1 Функциональные требования

При нажатии на кнопку «Онлайн-галерея» происходит переход на главную страницу сайта. При нажатии на «Избранное» происходит переход на страницу «Избранное» (пункт 4.2.1.3). При нажатии на «Топ-10 публикаций» происходит переход на страницу «Топ-10 публикаций» (пункт 4.2.1.4). При нажатии на «Категории» происходит переход на страницу «Категории» (пункт 4.2.1.5). Посетитель также может найти что-то на сайте, используя строку поиска, она принимает следующие данные: цифры, буквы, специальные символы, эмодзи (пункт 4.2.1.6). Также в header-е есть ссылки, которые ведут на Регистрацию и Вход, они находятся справа с краю (пункты 4.2.1.8, 4.2.1.7 соответственно). В footer-е сайта находится ссылка на страницу "О нас" (пункт 4.2.1.9) и ссылки на скачивание документов: "Условия пользования", "Политика конфиденциальности" (пункт 4.2.1.10).

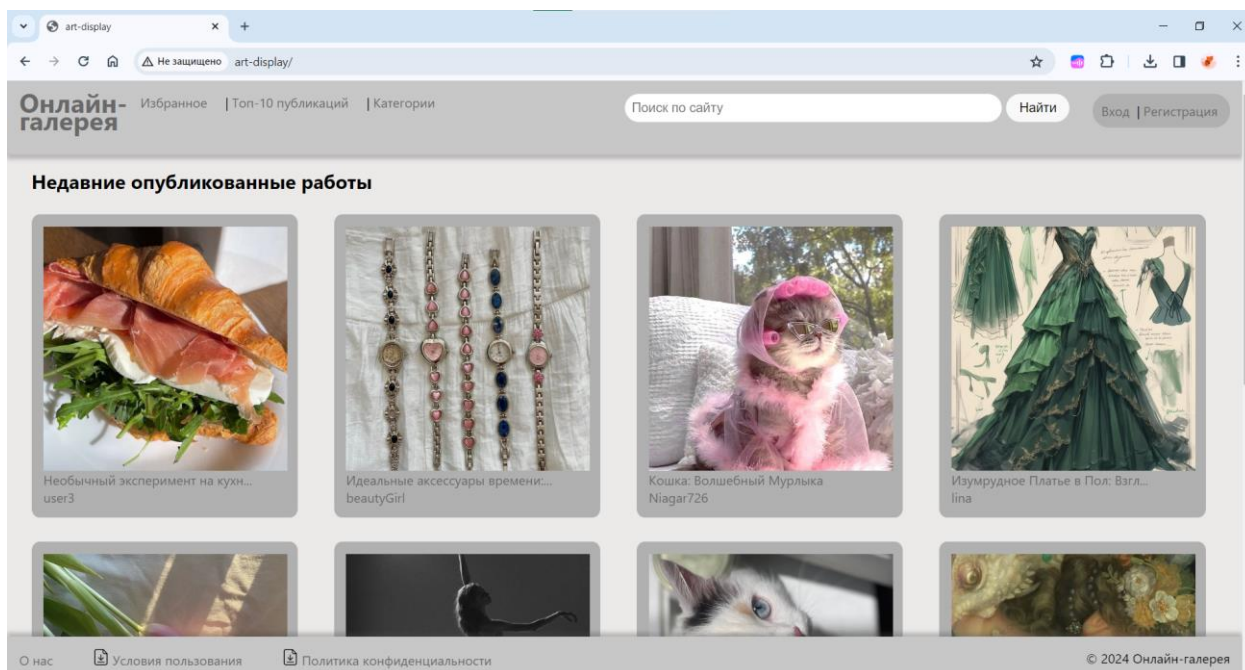


Рисунок 1 – Главная страница посетителя.

Главная страница администратора/пользователя выглядит аналогично, но вместо ссылок на вход или регистрацию отображается никнейм пользователя. При нажатии на который, происходит переход на страницу пользователя (пункты 4.2.2.1, 4.2.2.2). Если пользователь является администратором – то при наведении на никнейм отобразится выпадающий список: Админ. панель и Выход, если пользователь не администратор – то только Выход.

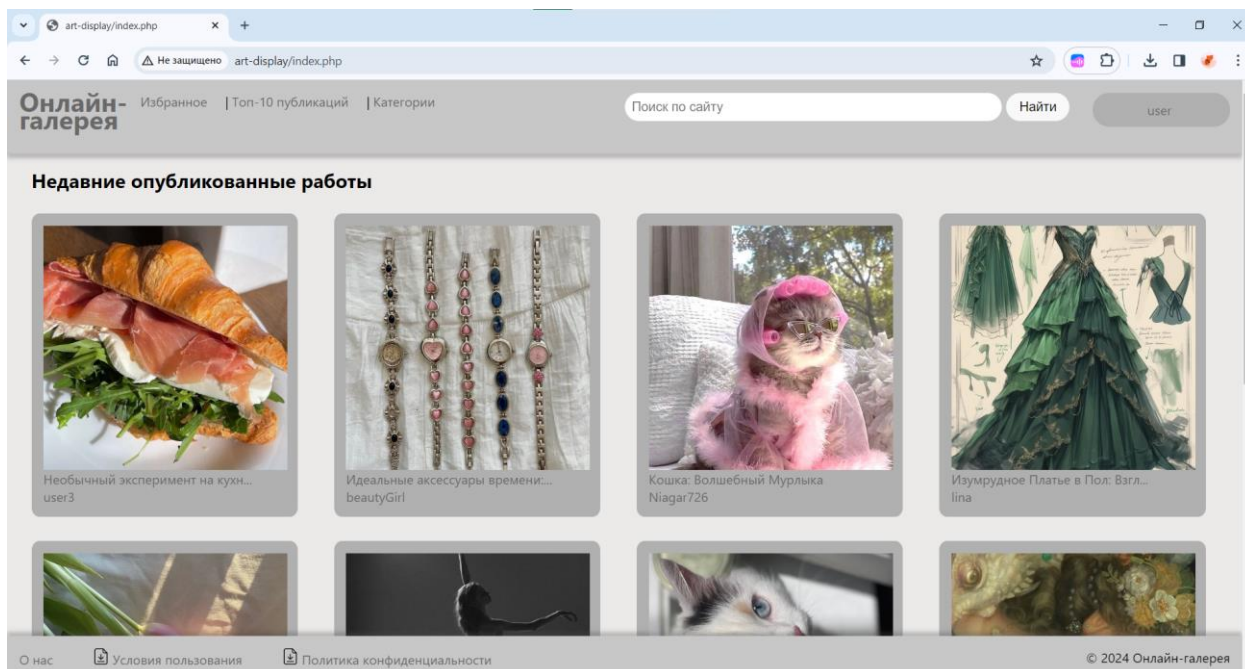


Рисунок 2.1 - Главная страница администратора/пользователя.

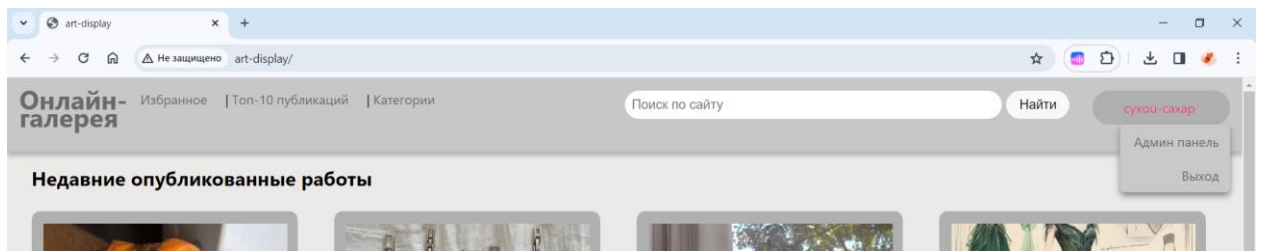


Рисунок 2.2 - Главная страница администратора.

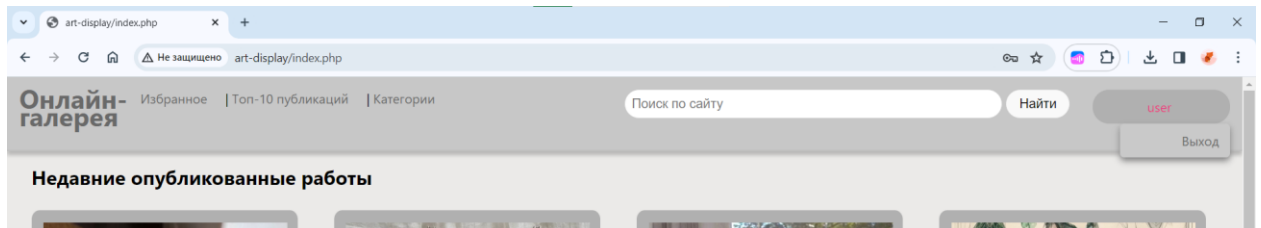


Рисунок 2.3 - Главная страница пользователя.

Реализация функции

1. Логотип и заголовок сайта:

```
<div id="logo">
  <a href="/" title="Перейти на главную"><h1><?= $title ?></h1></a>
</div>
```

Этот блок содержит логотип сайта, представленный в виде заголовка `<h1>`, который динамически отображает значение переменной `$title`. При нажатии на логотип, пользователь перейдет на главную страницу сайта.

2. Главное меню:

```
<div id="menuHead">
  <nav>
    <ul>
      <a href="../favorites.php">Избранное</a> |
      <a href="../index.php?best_posts=1">Топ-10 публикаций</a> |
      <a href="../../category/index-categories.php">Категории</a>
      <a href="#"></a>
    </ul>
  </nav>
</div>
```

Этот блок представляет собой меню навигации, включающее ссылки на страницы "Избранное", "Топ-10 публикаций" и "Категории". Пункты меню расположены в виде неупорядоченного списка ``.

3. Поисковая форма:

```
<div class="search">
```

```

        <form action="/search.php" method="post" onsubmit="return
validateSearch()">
            <input id="searchInput" name="search" type="text" placeholder="Поиск
по сайту"
                style="border: 1px solid #ccc; padding: 8px; border-radius: 20px; font-
size: 16px;" required>
            <button type="submit"
                style="background-color: #f9f9f9; border: 1px solid #ccc; padding:
8px 16px; border-radius: 20px; font-size: 16px; cursor: pointer;">
                Найти
            </button>
        </form>
    </div>

```

Эта часть кода создает форму для поиска по сайту. Пользователь вводит запрос в текстовое поле и нажимает кнопку "Найти". Форма отправляет данные методом POST на search.php. Валидация формы происходит с помощью функции validateSearch().

4. Блок авторизации:

```

<div id="regAuth" onmouseover="showDropdown()"
onmouseout="hideDropdown()">
    <div class="user-panel">
        <?php if (isset($_SESSION['id'])) : ?>
            <div class="user-link-container">
                <a href="../userPage/userPage.php?userId=<?= $_SESSION['id']; ?>"
class="user-link">
                    <?php
                    echo mb_substr($_SESSION['login'], 0, 15, 'UTF-8');
                    $title_length = mb_strlen($_SESSION['login'], 'UTF-8');
                    if ($title_length > 16) {
                        echo '...';
                    }
                    ?>
                </a>
                <ul class="user-dropdown" id="userDropdown"
onmouseover="showDropdown()" onmouseout="hideDropdown()">
                    <?php if ($_SESSION['role'] === "0") : ?>
                        <li><a href="../admin/posts/index.php">Админ панель</a></li>
                    <?php endif; ?>
                    <li><a href="../logout.php">Выход</a></li>
                </ul>
            </div>
        <?php else : ?>

```

```
<a href="https://art-display/authorization.php">Вход</a> |  
<a href="https://art-display/registration.php">Регистрация</a>  
<?php endif; ?>  
</div>  
</div>
```

Этот блок отображает панель пользователя. Если пользователь авторизован (проверяется с помощью `$_SESSION['id']`), то показывается ссылка на страницу пользователя с его логином. Логин отображается с ограничением до 15 символов, и если длина логина больше, добавляется троеточие. Также в выпадающем меню могут быть ссылки на "Админ панель" (если роль пользователя равна 0) и "Выход". Если пользователь не авторизован, отображаются ссылки на страницы входа и регистрации.

5.1.1.2 Функция просмотра поста

5.1.1.2.1 Описание

Позволяет просматривать пост с целью ознакомления с работой, автором, категорией.

Приоритет: высокий.

5.1.1.2.2 Функциональные требования

Страница поста содержит изображение, название, описание. Под названием отображается никнейм автора и дата публикации. Никнейм является кликабельным, после нажатия на него происходит переход на страницу пользователя (пункты 4.2.2.1, 4.2.2.2). Под описанием располагаются категория, просмотры и комментарии. На название категории можно нажать, тогда будет осуществлен переход на страницу с категорией и постами, которые относятся к ней (пункт 4.2.1.5). Просмотры считаются по одному от каждого зарегистрированного пользователя. Поле с комментарием доступно только для авторизованных пользователей, если комментариев под постом нет – то посетитель сайт ничего не видит после строки просмотров. Если вы администратор или автор комментария, то вам доступно обновление или удаление данного комментария (пункт 4.2.2.13). В комментариях также никнейм пользователя является кликабельным, справа отображается дата и

время создания. Справа от просмотров находится иконка в виде сердца, она отображается только для зарегистрированных пользователей. Если иконка белая – пост не в избранном, если красным – пост добавлен в избранное. На иконку можно нажать и выполнить операцию добавления или удаления (пункт 4.2.2.14).

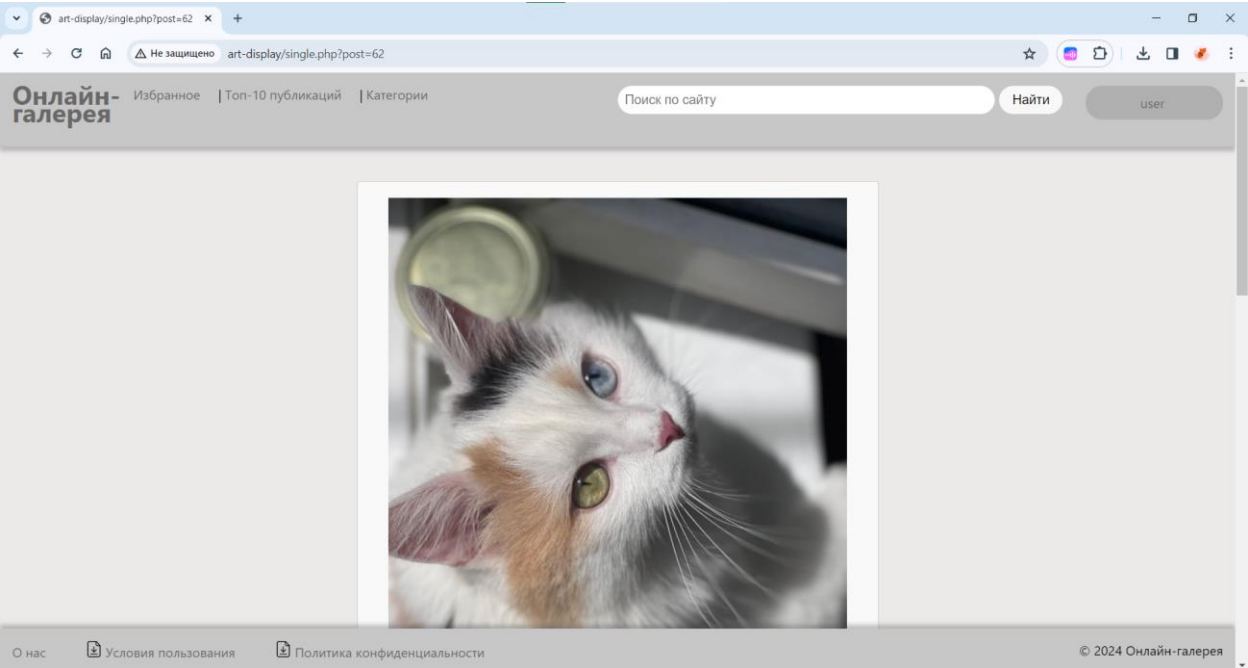


Рисунок 3.1 – Пост

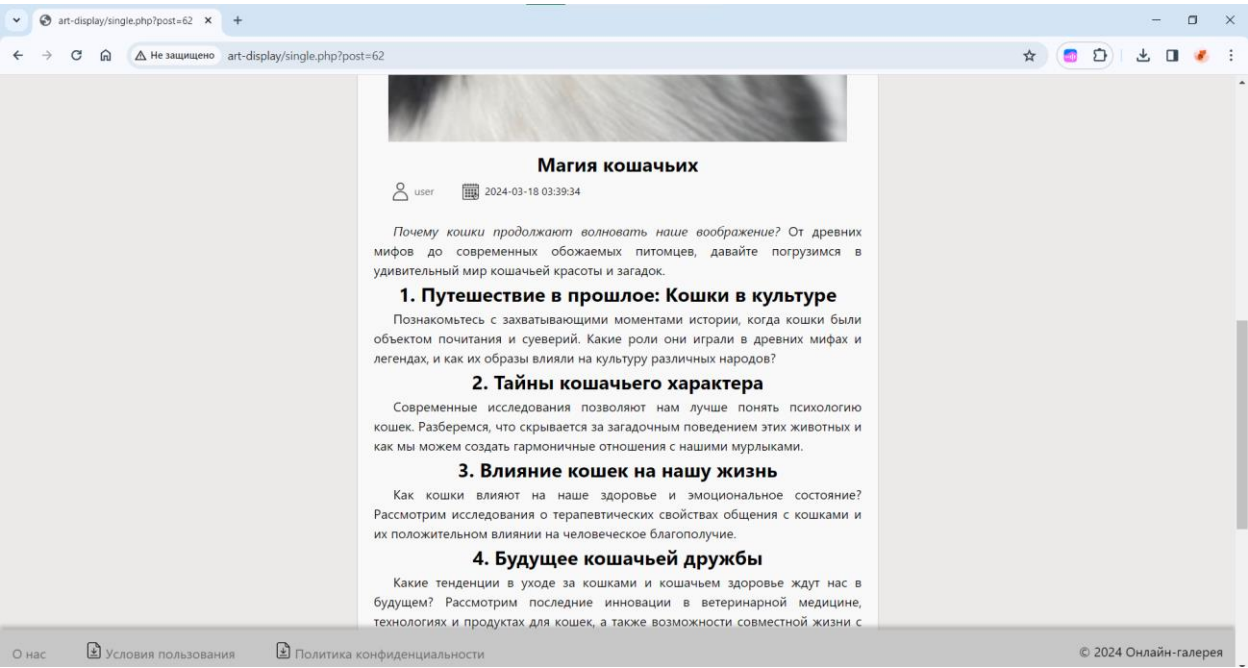


Рисунок 3.2 – Пост

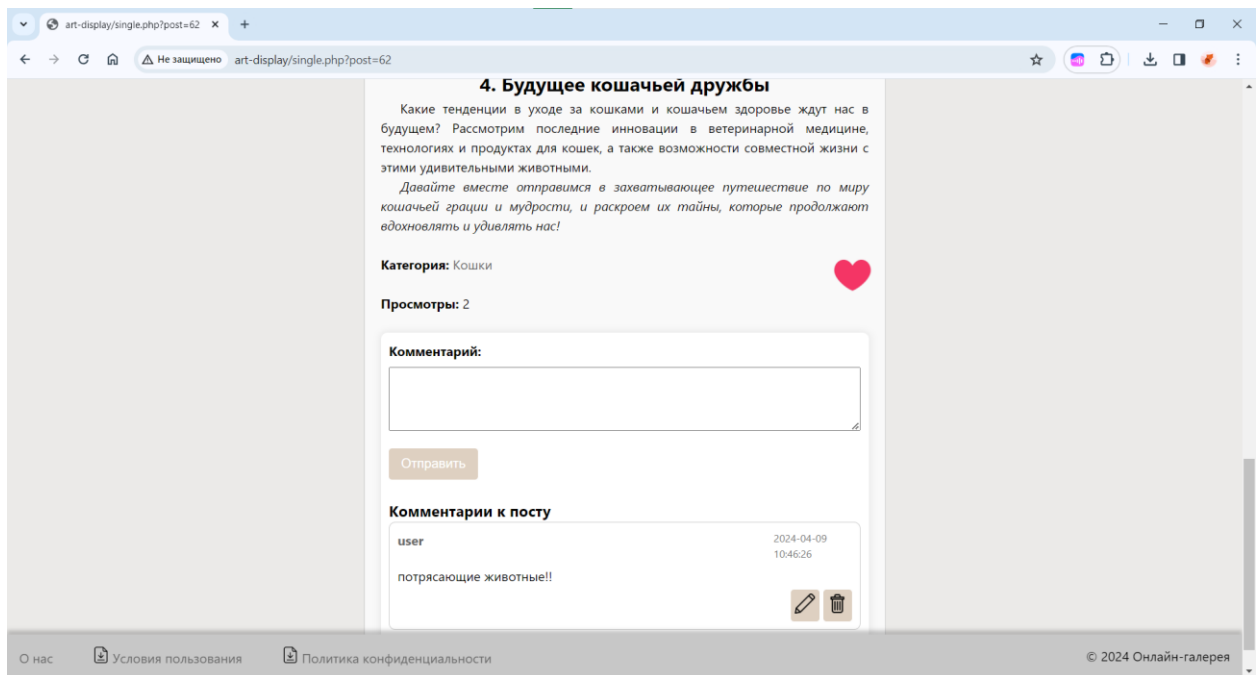


Рисунок 3.3 – Пост

Реализация функции

1. Подключение к базе данных и получение ID поста из URL:

```
// Подключение к базе данных
require_once('function/db.php');
// Получение ID поста из URL
if (isset($_GET['post']) && !empty($_GET['post'])) {
    $post_id = $_GET['post'];
```

В начале кода подключается файл с функциями для работы с базой данных и проверяется, передан ли ID поста через URL. Если ID поста передан, он сохраняется в переменной `$post_id`.

2. Запрос к базе данных для получения данных о посте:

```
// Запрос к базе данных
$sql = "SELECT * FROM posts WHERE id = '$post_id' AND status = 1";
$result = $conn->query($sql);
```

Выполняется SQL-запрос для получения данных о посте, где `id` соответствует переданному `$post_id`, а `status` равен 1 (означает, что пост опубликован).

3. Обработка результатов запроса и отображение данных:

```
// Обработка результатов запроса
if ($result->num_rows > 0) {
    // Вывод данных
    $row = $result->fetch_assoc();
```

Если запрос возвращает результаты, данные поста извлекаются и сохраняются в переменной \$row.

4. Обработка посещений пользователя:

```
if ($_SESSION) {  
    $userId = $_SESSION['id'];  
    // Увеличение счетчика посещений  
    $sql = "SELECT * FROM visits WHERE post_id = '$post_id' AND user_id =  
    '$userId'";  
    $result = $conn->query($sql);  
    if (!$result->num_rows > 0) {  
        $sql = "INSERT INTO `visits` (post_id,user_id) VALUES  
    ('$post_id','$userId')";  
        $result = $conn->query($sql);  
    }  
}
```

Если пользователь авторизован, проверяется, было ли уже зарегистрировано посещение этого поста данным пользователем. Если нет, в таблицу visits добавляется новая запись.

5. Отображение изображения, заголовка и информации об авторе:

```
$show_img = base64_encode($row['img']);  
?>  
<div class="image">  
      
</div>  
<h2><?php echo $row['title']; ?></h2>  
<div class="author-info">  
      
    <?php  
        $idUser = $row['user_id'];  
        $sqlAuthor = "SELECT * FROM `registeruser` WHERE id = '$idUser'";  
        $resultAuthor = $conn->query($sqlAuthor);  
        $resultAuthor = $resultAuthor->fetch_assoc();  
    ?>  
    <a href="http://art-display/userPage/userPage.php?userId=<?=  
$resultAuthor['id']; ?>"><?php echo $resultAuthor['login']; ?></a>  
      
    <p><?php echo $row['created_date']; ?></p>  
</div>  
<br>  
<div class="content">  
    <p><?php echo nl2br($row['content']); ?></p>
```


</div>

<div class="clear">
</div>

Код извлекает изображение поста, заголовок, имя автора и дату создания, и отображает их на странице.

6. Обработка лайков:

```
<?php if ($_SESSION) :  
    $user_id = $_SESSION['id'];  
    // Запрос к базе данных для получения значения like  
    $sqlLike = "SELECT `like` FROM visits WHERE post_id = '$post_id' AND  
user_id = '$user_id'";  
    $resultLike = $conn->query($sqlLike);  
    $likeValue = 0; // По умолчанию, если запись не найдена  
  
    if ($resultLike->num_rows > 0) {  
        $likeData = $resultLike->fetch_assoc();  
        $likeValue = $likeData['like'];  
    }  
?>  
<form id="likeForm" action="addToFavorites.php" method="POST">  
    <input type="hidden" name="post_id" value="<?php echo $post_id; ?>">  
    <input type="hidden" name="user_id" value="<?php echo $user_id ?>">  
    <?php if ($likeValue == 1) { ?>  
        <button name="addLike" type="submit" class="liked" title="Убрать из  
избранного"></button>  
        <?php } else { ?>  
            <button name="addLike" type="submit" class="like-icon"  
title="Добавить в избранное"></button>  
            <?php } ?>  
    </form>  
<?php endif; ?>
```

Если пользователь авторизован, проверяется, поставил ли он лайк этому посту. В зависимости от результата, отображается кнопка для добавления или удаления лайка.

7. Отображение категории поста и количества просмотров:

```
<p><strong>Категория:</strong> <?php  
    $categoryId = $row['category_id'];  
    $sql = "SELECT * FROM categories where id='$categoryId'";  
    $result = $conn->query($sql);  
    $categoryName = $result->fetch_assoc();  
?>
```

```

<a href="http://art-display/category/category.php?id=?= $categoryName['id'];
?>"><?php echo $categoryName['name']; ?></a> </p>
<div class="clear"><br></div>
<p><strong>Просмотры:</strong> <?php
    $sql = "SELECT * FROM visits WHERE post_id = '$post_id'";
    $result = $conn->query($sql);
    $visitsNum = $result->num_rows;
    $sql = "UPDATE `posts` SET visits='$visitsNum' WHERE id='$post_id'";
    $result = $conn->query($sql);
    echo $visitsNum;
?>
</p>

```

Отображается категория поста и количество просмотров. Категория определяется на основе category_id из таблицы categories, а количество просмотров подсчитывается по записям в таблице visits.

8. Загрузка комментариев:

```

<?php
require_once "blocks/comments.php";
?>
    Включается файл с комментариями к посту.

9. Обработка ошибок и закрытие соединения с базой данных:

    } else {
        echo "<p>Нет данных о посте.</p>";
    }
    } else {
        echo "<p>Неверный ID поста.</p>";
    }
    // Закрытие подключения к базе данных
    $conn->close();

```

Если ID поста не передан или данные о посте не найдены, выводится соответствующее сообщение. В конце закрывается соединение с базой данных.

5.1.1.3 Функция просмотра избранного

5.1.1.3.1 Описание

Позволяет пользователю просматривать избранное.

Приоритет: высокий.

5.1.1.3.2 Функциональные требования

Если посетитель не зашел в свой профиль, то на странице избранного отобразится надпись "Данная страница доступна только для авторизованных пользователей". Если пользователь зарегистрирован, то произойдет вывод постов, добавленных им в избранное, если таких постов нет, то будет выведена надпись "Нет понравившихся записей".

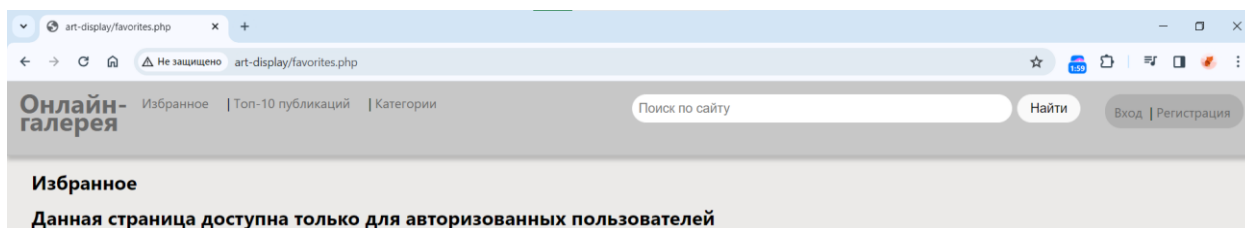


Рисунок 4.1 – Страница избранного

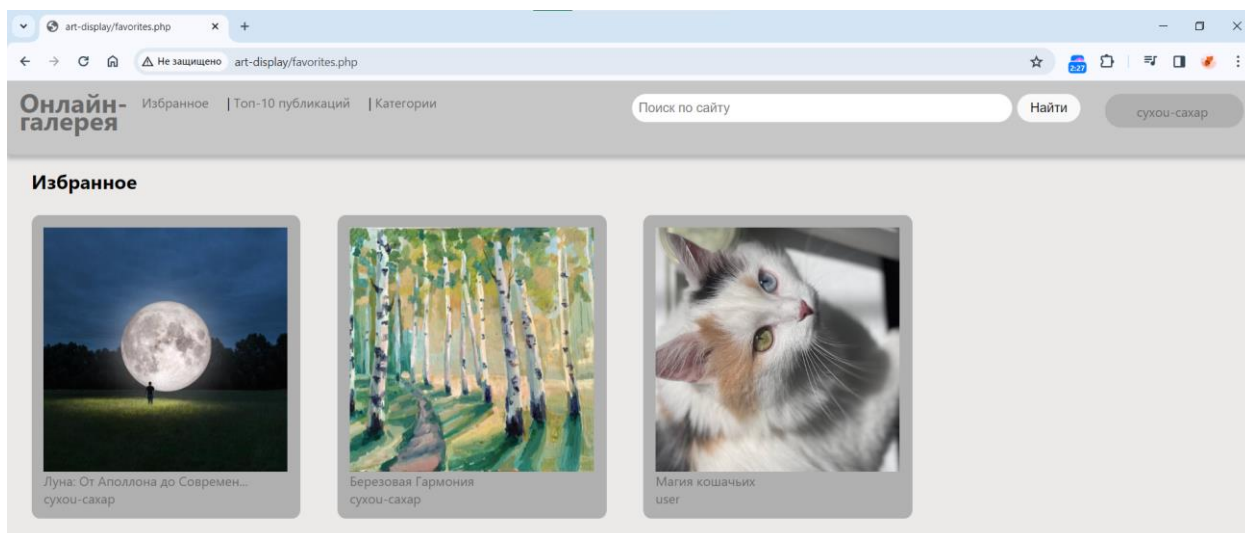


Рисунок 4.2 – Страница избранного

Реализация функции

1. Отображение заголовка и подключение файла с шапкой сайта:

```
<body>
  <header>
    <?php
      $title = "Онлайн-галерея";
      require_once "blocks/header.php";
    ?>
  </header>
  <div class="clear"><br></div>
  <div class="article">
    <h2>Избранное</h2>
  </div>
```

В начале файла задается заголовок страницы "Онлайн-галерея" и подключается внешний файл header.php, который содержит общую шапку сайта. После этого выводится заголовок раздела "Избранное".

2. Проверка авторизации пользователя:

```
<?php
if (!$_SESSION){
    ?>
    <div class="article">
        <br>
        <h2>Данная страница доступна только для авторизованных
пользователей</h2>
    </div>
    <?php
    } else {
        ?>
```

Проверяется, авторизован ли пользователь. Если нет, выводится сообщение о том, что доступ к странице ограничен только для авторизованных пользователей.

3. Отображение избранных постов для авторизованных пользователей:

```
<div id="posts">
    <?php
    // Подключение к базе данных
    require_once('function/db.php');
    $userId = $_SESSION['id'];
    $sql = "SELECT post_id FROM visits WHERE user_id = '$userId' and
`like`=1";
    $resultPostId = $conn->query($sql);
    if ($resultPostId->num_rows > 0) {
        while ($rowId = $resultPostId->fetch_assoc()) {
            $postId = $rowId['post_id'];
            $sql = "SELECT * FROM posts WHERE id = '$postId'";
            $resultPost = $conn->query($sql);
            while ($rowPost = $resultPost->fetch_assoc()) {
                if ($rowPost['status'] != 0) {
                    $show_img = base64_encode($rowPost['img']);
                    ?>
                    <div class="article">
                        <div class="like-icon"></div>
                        <div>
                            
```

```

        <a href="http://art-display/single.php?post=<?php echo
$rowPost['id']; ?>">
            <?php
            echo mb_substr($rowPost['title'], 0, 29, 'UTF-8');
            $title_length = mb_strlen($rowPost['title'], 'UTF-8');
            if ($title_length > 29) {
                echo '...';
            }
            ?>
        </a>
        <p><?php
        $idUser = $rowPost['user_id'];
        $sql = "SELECT * FROM `registeruser` WHERE id =
'idUser'";

        $resultName = $conn->query($sql);
        $resultName = $resultName->fetch_assoc();
        ?>
        <a href="http://art-
display/userPage/userPage.php?userId=<?php echo $resultName['id']; ?>"><?php echo
$resultName['login']; ?></a> </p>
    </div>
</div>
<?php
}
}
} else {
    ?> <h2>Нет понравившихся записей</h2> <?php
}
// Заккрытие подключения к базе данных
$conn->close();
}
?>
<div class="clear"><br></div>
</div>
<div class="clear"><br></div>
</body>

```

Если пользователь авторизован, выполняется подключение к базе данных. Затем выполняется SQL-запрос для получения ID всех постов, которые пользователь отметил как "понравившиеся". Для каждого такого поста выполняется еще один запрос для получения его данных.

4. Отображение каждого поста:

```

$show_img = base64_encode($rowPost['img']);
?>
<div class="article">
  <div class="like-icon"></div>
  <div>
    
    <a href="http://art-display/single.php?post=<?php echo $rowPost['id'];
?>">
      <?php
      echo mb_substr($rowPost['title'], 0, 29, 'UTF-8');
      $title_length = mb_strlen($rowPost['title'], 'UTF-8');
      if ($title_length > 29) {
        echo '...';
      }
      ?>
    </a>
    <p><?php
      $idUser = $rowPost['user_id'];
      $sql = "SELECT * FROM `registeruser` WHERE id = '$idUser'";
      $resultName = $conn->query($sql);
      $resultName = $resultName->fetch_assoc();
      ?>
      <a href="http://art-display/userPage/userPage.php?userId=<?=
$resultName['id']; ?>"><?php echo $resultName['login']; ?></a> </p>
    </div>
  </div>

```

Если пост опубликован (его статус не равен 0), он отображается на странице. Для каждого поста показывается его изображение, заголовок, автор и ссылка на полную версию поста. Заголовок обрезается до 29 символов с добавлением троеточия, если он длиннее.

5. Сообщение об отсутствии понравившихся записей:

```

    } else {
      ?> <h2>Нет понравившихся записей</h2> <?php
    }

    // Закрытие подключения к базе данных
    $conn->close();
    ?>

```

Если у пользователя нет понравившихся записей, выводится соответствующее сообщение. В конце закрывается подключение к базе данных.

5.1.1.4 Функция просмотра популярных публикаций

5.1.1.4.1 Описание

Позволяет пользователю просмотреть популярные публикации.

Приоритет: высокий.

5.1.1.4.2 Функциональные требования

На странице отображаются десять самых популярных постов. Популярность оценивается по количеству просмотров.

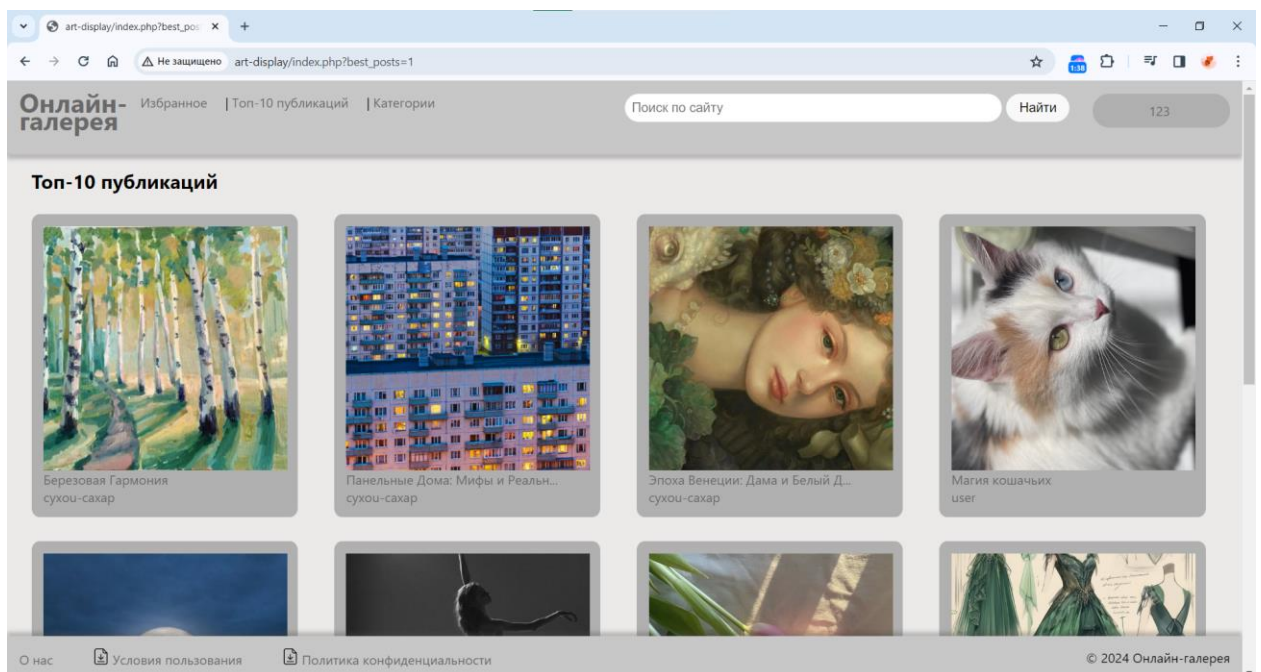


Рисунок 5 – Страница популярных публикаций

Реализация функции

Проверка параметра URL и условное отображение заголовка:

```
if (isset($_GET['best_posts'])) {  
    // Если нажата ссылка "Лучшее за неделю"  
    ?>  
    <div class="article">  
        <h2>Топ-10 публикаций</h2>  
    </div>  
    <?php  
} else {  
    ?>
```

```

<div class="article">
  <h2>Недавние опубликованные работы</h2>
</div>
<?php
}
?>

```

Этот блок кода проверяет, установлен ли параметр `best_posts` в URL.

Если параметр `best_posts` присутствует в URL (то есть ссылка "Лучшее за неделю" нажата), отображается заголовок "Топ-10 публикаций".

Если параметр `best_posts` отсутствует, отображается заголовок "Недавние опубликованные работы".

5.1.1.5 Функция просмотра категорий

5.1.1.5.1 Описание

Позволяет пользователю просматривать категории.

Приоритет: высокий.

5.1.1.5.2 Функциональные требования

Страница "Категории" содержит информацию о существующих категориях. В каждом блоке представлено название категории и описание, после нажатия на название категории происходит переход на страницу этой категории.

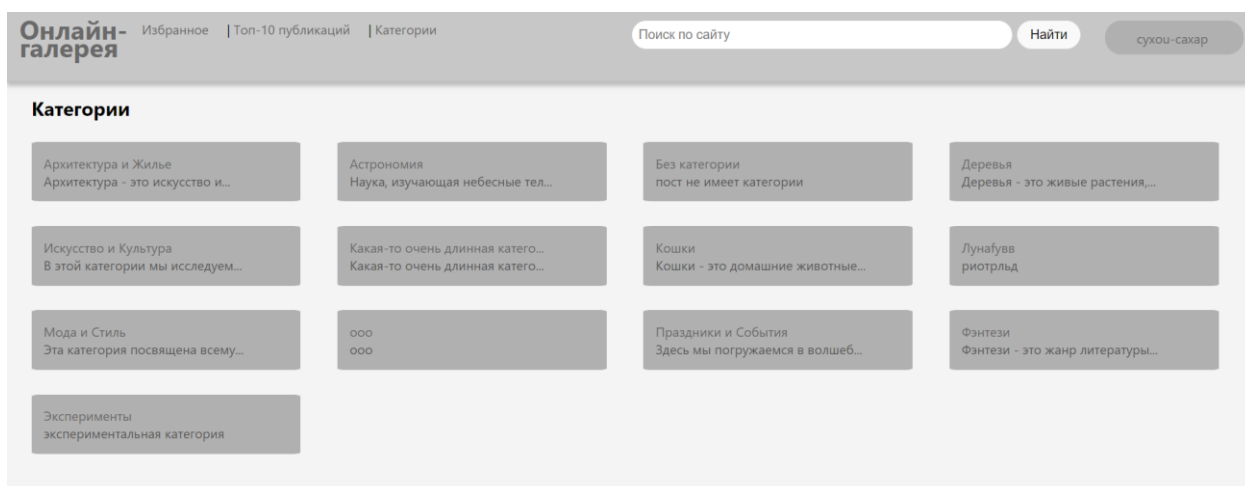


Рисунок 6.1 – Страница категорий

На странице выбранной категории представлено полное описание категории, а ниже посты, которые относятся к данной категории. Если таких нет – то будет выведено "Нет постов".

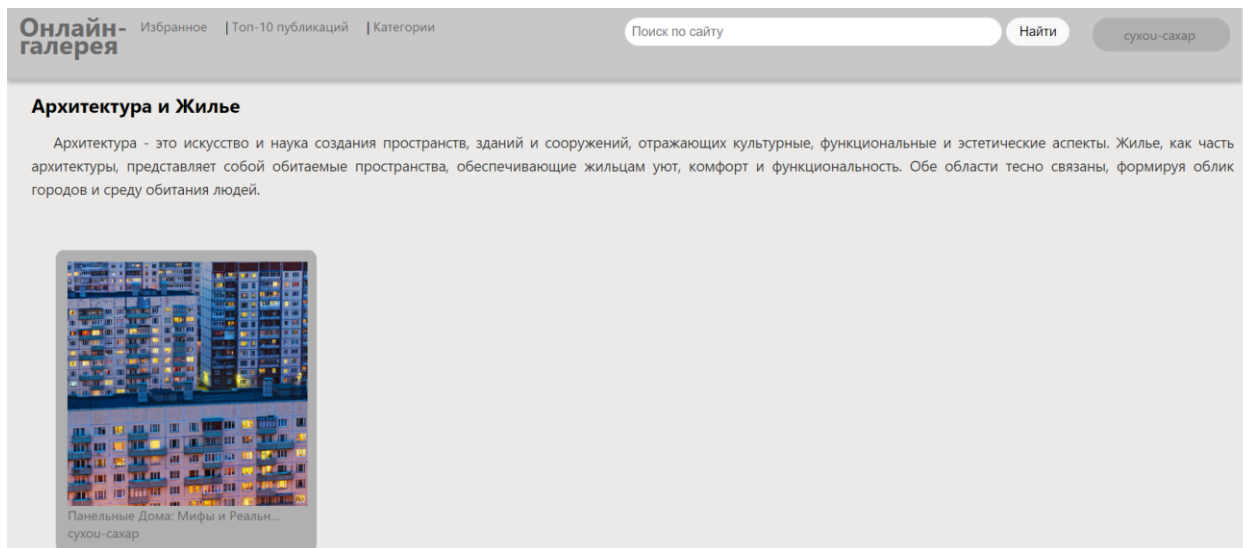


Рисунок 6.2 – Страница категории

Реализация функции

1. Заголовок раздела "Категории":

```
<div class="article">
  <h2>Категории</h2>
</div>
<div id="posts">
```

- Создается контейнер с классом article, внутри которого выводится заголовок "Категории".

- Далее начинается контейнер div с идентификатором posts, который будет содержать список категорий.

2. Подключение к базе данных и выполнение запроса:

```
<?php
// Подключение к базе данных
require_once('../function/db.php');
// Запрос к базе данных
$sql = "SELECT * FROM categories ORDER BY name ASC";
$result = $conn->query($sql);
// Обработка результатов запроса
```

- Подключается файл db.php, который устанавливает соединение с базой данных.

- Выполняется SQL-запрос для получения всех категорий, отсортированных по имени в алфавитном порядке.

- Результаты запроса сохраняются в переменной \$result.

3. Вывод категорий, если они найдены:

```

if ($result->num_rows > 0) {
// Вывод данных
while ($row = $result->fetch_assoc()) {
?>
<div class="article">
  <div>
    <a href="category.php?id=<?=$row['id']?>">
      <?php
        echo mb_substr($row['name'], 0, 29, 'UTF-8');
        $title_length = mb_strlen($row['name'], 'UTF-8');
        if ($title_length > 29) {
          echo '...';
        }
      ?>
    </a>
    <p>
      <?php
        echo mb_substr($row['description'], 0, 29, 'UTF-8');
        $title_length = mb_strlen($row['description'], 'UTF-8');
        if ($title_length > 29) {
          echo '...';
        }
      ?>
    </p>
  </div>
</div>
<?php
}
} else {?>
<div class="article">
  <h2>Нет категорий</h2>
</div>
<?php
}
// Закрытие подключения к базе данных
$conn->close();
?>

```

- Проверяется, если ли в результате запроса строки (т.е. есть ли категории в базе данных).

- Если категории найдены, выводится информация о каждой из них:

- Название категории: обрезается до 29 символов и добавляется троеточие, если название длиннее.

- Описание категории: также обрезается до 29 символов с добавлением троеточия при необходимости.

- Каждая категория выводится внутри отдельного контейнера div с классом article.

4. Сообщение об отсутствии категорий:

```
} else {?>
<div class="article">
    <h2>Нет категорий</h2>
</div>
<?php
}
```

- Если категории не найдены, выводится сообщение "Нет категорий".

5. Заккрытие подключения к базе данных:

```
// Заккрытие подключения к базе данных
$conn->close();
?>
```

Закрывается соединение с базой данных для освобождения ресурсов.

5.1.1.6 Функция поиска

5.1.1.6.1 Описание

Позволяет посетителю производить поиск по сайту.

Приоритет: высокий.

5.1.1.6.2 Функциональные требования

В строку поиска можно ввести цифры, буквы, спец. символы и эмодзи. При пустой строке поиск не будет осуществлён. После нажатия на кнопку "Найти" или enter, будет осуществлено перенаправление на страницу с результатом среди постов, категорий и пользователей. Если результатов нет, будет выведена надпись об отсутствии совпадений с искомыми данными.

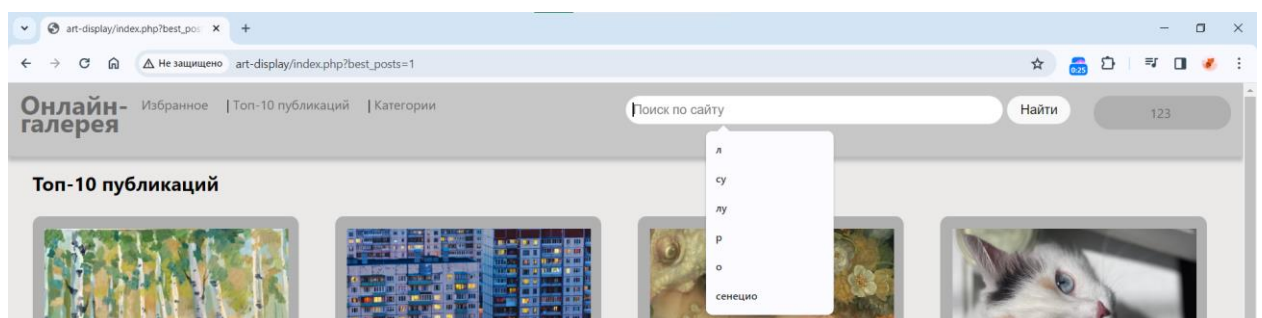


Рисунок 7.1 – Страница поиска

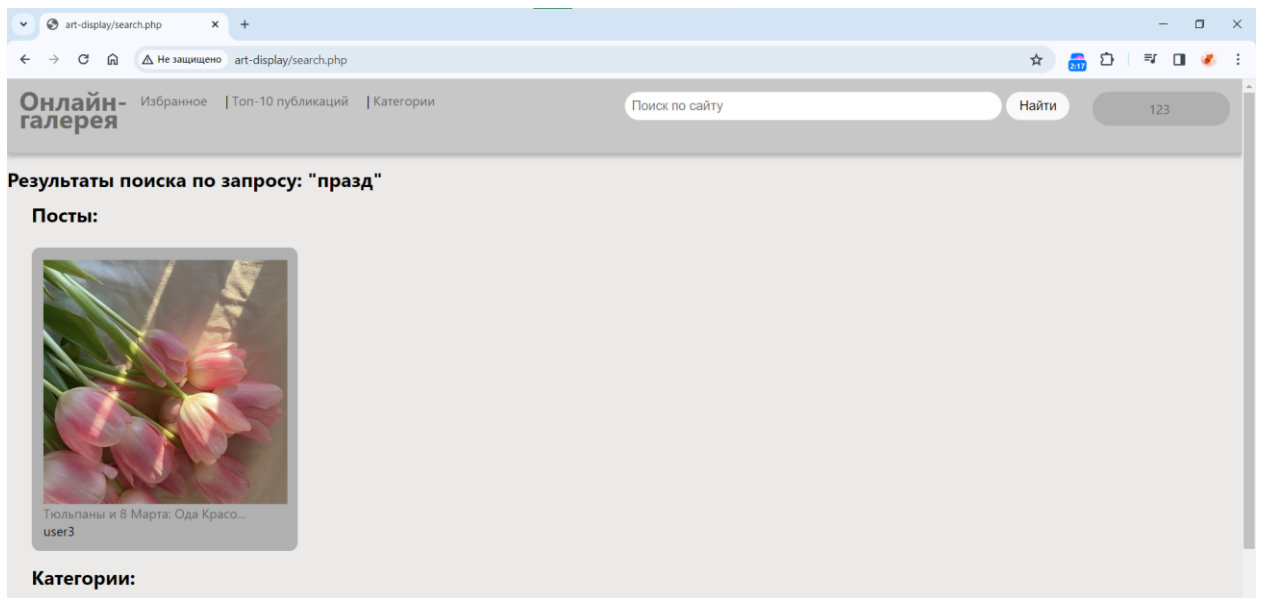


Рисунок 7.2 – Страница результата поиска

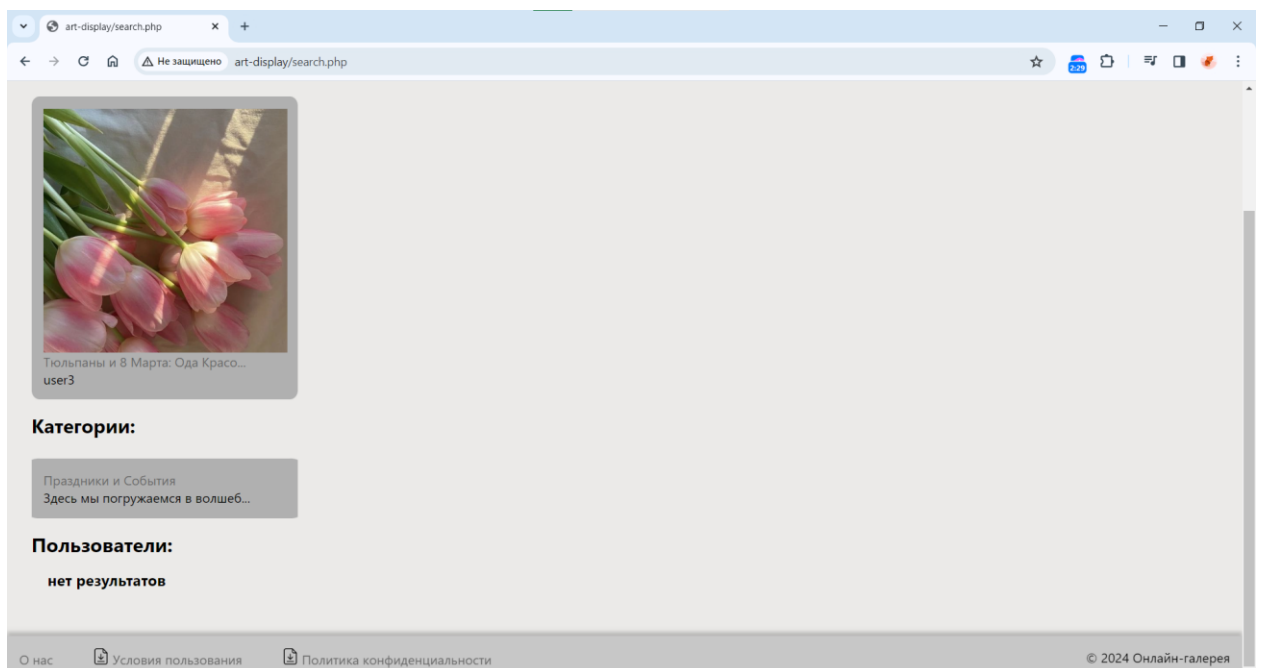


Рисунок 7.3 – Страница результата поиска

Реализация функции

1. Заголовок с запросом поиска:

```
<div class="clear"><br></div>
<h2 style="margin-left:10px;">Результаты поиска по запросу:
"<?=htmlspecialchars(trim($_POST['search']))?>"</h2>
<div id="posts">
    - Заголовок отображает текст запроса, введенный пользователем,
    очищенный от лишних пробелов и специальных символов.
```

2. Подключение к базе данных и выполнение поиска:

```

// Подключение к базе данных
require_once('function/db.php');

if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['search'])) {
    $text = trim($_POST['search']);
    if (!empty($text)){
        $sql = "SELECT * FROM posts AS p WHERE p.status = 1
                AND (p.title LIKE '%$text%' OR p.content LIKE '%$text%')";
        $resultPost = $conn->query($sql);
    } else {
        ?>
        <div class="clear"><br></div>
        <h3>строка поиска пуста</h3>
        <?php
        return;
    }
    - Подключается файл db.php для установления соединения с базой
    данных.

```

- Проверяется, была ли отправлена форма поиска методом POST и содержит ли она параметр search.

- Если строка поиска не пуста, выполняется SQL-запрос для поиска постов, в названии или содержимом которых присутствует введенный текст.

3. Вывод результатов поиска постов:

```

<div class="clear"><br></div>
<h2>Посты:</h2>
<?php
if ($resultPost->num_rows > 0) {
    while ($row = $resultPost->fetch_assoc()) {
        if ($row['status'] != 0) {
            $show_img = base64_encode($row['img']);
            ?>
            <div class="article">
                <div class="like-icon"></div>
                <div>
                    
                    <a href="http://art-display/single.php?post=<?php echo $row['id'];
?>">
                        <?php
                        echo mb_substr($row['title'], 0, 29, 'UTF-8');

```

```

        $title_length = mb_strlen($row['title'], 'UTF-8');
        if ($title_length > 29) {
            echo '...';
        }
        ?>
    </a>
    <p><?php
        $idUser = $row['user_id'];
        $sql = "SELECT login FROM `registeruser` WHERE id =
'idUser'";

        $resultName = $conn->query($sql);
        $resultName = $resultName->fetch_assoc()
        ?>
        <?php echo $resultName['login']; ?> </p>
    </div>
</div>
<?php
}
}
} else { ?>
    <div class="clear"><br></div>
    <h3>нет результатов</h3>
    <?php
}

```

- Если найдены посты, соответствующие запросу, они выводятся с изображением, заголовком и именем автора.

- Если постов нет, выводится сообщение "нет результатов".

4. Вывод результатов поиска категорий:

```

$sql = "SELECT * FROM categories WHERE (name LIKE '%$text%' OR
description LIKE '%$text%')";
$resultCategories = $conn->query($sql);

<div class="clear"><br></div>
<h2>Категории:</h2>
<?php
if ($resultCategories->num_rows > 0) {
    while ($row = $resultCategories->fetch_assoc()) {
        ?>
        <div class="article">
            <div class="title col-5">
                <a href="category/category.php?id=<?= $row['id'] ?>">
                    <?php
                        echo mb_substr($row['name'], 0, 29, 'UTF-8');

```

```

        $title_length = mb_strlen($row['name'], 'UTF-8');
        if ($title_length > 29) {
            echo '...';
        }
        ?>
    </a>
</div>
<div class="title col-5">
    <?php
        echo mb_substr($row['description'], 0, 29, 'UTF-8');
        $title_length = mb_strlen($row['description'], 'UTF-8');
        if ($title_length > 29) {
            echo '...';
        }
        ?>
    </div>
</div>
<?php
}
} else { ?>
    <div class="clear"><br></div>
    <h3>нет результатов</h3>
    <?php
}

```

- Выполняется SQL-запрос для поиска категорий по имени или описанию.

- Если категории найдены, они выводятся с названием и описанием.

- Если категорий нет, выводится сообщение "нет результатов".

5. Вывод результатов поиска пользователей:

```

$sql = "SELECT * FROM registeruser WHERE (login LIKE '%$text%')";
$resultUsers = $conn->query($sql);
<div class="clear"><br></div>
<h2>Пользователи:</h2>
<?php
if ($resultUsers->num_rows > 0) {
    while ($row = $resultUsers->fetch_assoc()) {
        ?>
        <div class="article">
            <div>
                <a href="http://art-display/userPage/userPage.php?userId=<?php echo
$row['id']; ?>"><?php
                    echo $row['login']; ?> </a><br>

```

```

        </div>
    </div>
    <?php
    }
} else { ?>
    <div class="clear"><br></div>
    <h3>нет результатов</h3>
    <?php
    }

```

- Выполняется SQL-запрос для поиска пользователей по логину.
- Если пользователи найдены, выводится их логин с ссылкой на страницу профиля.
- Если пользователей нет, выводится сообщение "нет результатов".

6. Закрытие соединения с базой данных:

```

// Закрытие подключения к базе данных
$conn->close();
?>

```

- Соединение с базой данных закрывается для освобождения ресурсов.

5.1.1.7 Функция входа пользователя в аккаунт

5.1.1.7.1 Описание

Позволяет войти пользователю в аккаунт.

Приоритет: высокий.

5.1.1.7.2 Функциональные требования

В поле «Email» должна быть введена почта пользователя, обязательно наличие символы "@". В данное поле записывается почта аккаунта.

В поле «Пароль» должны быть введены данные строкового типа. В данное поле записывается пароль аккаунта. Пароль вводится в состоянии маски (маска имеет формат *).

При нажатии кнопки «Войти» происходит проверка почты и пароля. В случае если аккаунт с данными почтой и паролем существует, происходит переход на главную страницу в зависимости от роли. Если роль администратор – на страницу админ. панели (пункт 4.2.3.1). Если должность пользователь – главную страницу сайта (пункт 4.2.1.1). Если такого аккаунта

нет, система выдаёт сообщение об ошибке. Также внизу есть ссылка для перехода на страницу регистрации пользователя (пункт 4.2.1.8).

Рисунок 8 – Страница входа в аккаунт

Реализация функции

```
// Проверяем, был ли пользователь заблокирован
if (isset($_SESSION['block_time']) && $_SESSION['block_time'] > time()) {
    $remaining_time = ceil(($_SESSION['block_time'] - time()) / 60); //
    Оставшееся время в минутах
    $errorMessage = "Попробуйте войти через {$remaining_time} мин.";
} elseif ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Получаем и обрабатываем введенные пользователем данные
    $email = htmlspecialchars(trim($_POST['email']));
    $pass = htmlspecialchars($_POST['pass']);
    $hashedPass = md5($pass);

    // Получаем текущее количество неверных попыток из сессии
    $attempts = isset($_SESSION['login_attempts']) ? $_SESSION['login_attempts']
: 0;

    // Проверяем наличие пользователя по email
    $sql = "SELECT * FROM `registeruser` WHERE email = '$email'";
    $resultLog = $conn->query($sql);
```

```

// Проверяем наличие пользователя по email и паролю
$sql = "SELECT * FROM `registeruser` WHERE email = '$email' AND pass = '$hashedPass'";
$resultPass = $conn->query($sql);

if ($resultPass->num_rows > 0) {
    // Сбрасываем счетчик неверных попыток при успешном входе
    unset($_SESSION['login_attempts']);
    while ($row = $resultPass->fetch_assoc()) {
        if ($resultPass) {
            $userId = $row['id'];
            $sql = "SELECT * FROM `registeruser` WHERE id = '$userId'";
            $result = $conn->query($sql);

            if ($result) {
                // Проверяем, есть ли данные
                if ($result->num_rows > 0) {
                    // Получаем данные пользователя
                    $userData = $result->fetch_assoc();
                    if (!($userData['status'] === '0')) {
                        // Устанавливаем сессионные переменные
                        $_SESSION['id'] = $userData['id'];
                        $_SESSION['login'] = $userData['login'];
                        $_SESSION['role'] = $userData['role'];
                        // Перенаправляем в зависимости от роли пользователя
                        if ($_SESSION['role'] === '0') {
                            header("location: http://art-display/admin/posts/index.php");
                        } else {
                            header("location: http://art-display/index.php");
                        }
                    } else {
                        $errorMessage = "Пользователь заблокирован!";
                    }
                }
            }
        }
    }
} elseif ($resultLog->num_rows > 0) {
    // Увеличиваем счетчик неверных попыток
    $_SESSION['login_attempts'] = ++$attempts;

    if ($attempts >= 5) {
        // Устанавливаем время блокировки на 5 минут
        $_SESSION['block_time'] = time() + 300; // 300 секунд = 5 минут
        $errorMessage = "Попробуйте войти через 5 минут";
    }
}

```

```
    } else {  
        $errorMessage = "Неверный пароль!";  
    }  
    } else {  
        $errorMessage = "Такого пользователя не существует";  
    }  
    } else {  
        $email = "";  
    }  
}
```

Основные этапы работы:

Проверка блокировки:

Если пользователь был ранее заблокирован (наличие переменной `block_time` в сессии и значение этой переменной больше текущего времени), вычисляется оставшееся время блокировки и формируется соответствующее сообщение об ошибке.

Обработка данных POST-запроса:

Получение и обработка введенных пользователем данных (`email` и пароль). Пароль хэшируется с помощью функции `md5`.

Проверка наличия учетной записи по `email`.

Проверка соответствия введенных данных (`email` и пароль).

Успешный вход:

При успешной валидации учетных данных сбрасывается счетчик неверных попыток.

Получение данных пользователя из базы данных и установка сессионных переменных.

Перенаправление пользователя в зависимости от его роли (администратор или обычный пользователь).

Неудачный вход:

Увеличение счетчика неверных попыток в случае ввода неверного пароля. При достижении 5 неверных попыток устанавливается блокировка на 5 минут.

Формирование сообщения об ошибке при неверном пароле или несуществующем пользователе.

Инициализация переменной:

Установка пустого значения для переменной \$email в случае, если запрос не является POST-запросом.

5.1.1.8 Функция регистрации пользователя

5.1.1.8.1 Описание

Позволяет посетителю зарегистрировать аккаунт на сайте.

Приоритет: высокий.

5.1.1.8.2 Функциональные требования

В поле «Логин» должны быть введены данные строкового типа. В данное поле записывает логин аккаунта. Если логин уже используется на сайте, будет выведено сообщение: "пользователь с никнеймом ... уже существует".

В поле «Пароль» и "Повтор ввода" должны быть введены данные строкового типа. В данное поле записывается пароль аккаунта. Пароль вводится в состоянии маски (маска имеет формат *). Если пароли не совпадают, будет выведена ошибка.

В поле «Email» должна быть введена почта пользователя, обязательно наличие символы "@". В данное поле записывается почта аккаунта.

При нажатии кнопки «Зарегистрироваться» происходит проверка почты и пароля. В случае если аккаунт с данной почтой и паролем существует, выводится сообщение о том, что пользователь уже есть в системе. Если все проверки пройдены, в зависимости от роли происходит переадресация. Если роль администратор – на страницу админ. панели (пункт 4.2.3.1). Если должность пользователь – главную страницу сайта (пункт 4.2.1.1). Также внизу есть ссылка для перехода на страницу входа для уже зарегистрированных пользователей (пункт 4.2.1.7).

Регистрация

Имя пользователя:

Пароль:

Повтор ввода:

Электронная почта:

Зарегистрироваться

У вас уже есть учетная запись?
[Войти](#)

Рисунок 9 – Страница регистрации посетителя

Реализация функции

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    // Получаем и обрабатываем введенные пользователем данные  
    $login = htmlspecialchars(trim($_POST['login']));  
    $pass = htmlspecialchars($_POST['pass']);  
    $repeatpass = htmlspecialchars($_POST['repeatpass']);  
    $email = htmlspecialchars(trim($_POST['email']));  
    $role = 1;  
  
    // Проверяем наличие пользователя с указанным логином  
    $sql = "SELECT * FROM `registeruser` WHERE login = '$login'";  
    $resultLog = $conn->query($sql);  
  
    // Проверяем наличие пользователя с указанным email  
    $sql = "SELECT * FROM `registeruser` WHERE email = '$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) {
    $errorMessage = "Логин пользователя должен быть до 25 символов!";
} elseif (mb_strlen($pass, 'UTF8') > 50) {
    $errorMessage = "Пароль пользователя должен быть до 50 символов!";
} elseif (mb_strlen($email, 'UTF8') > 50) {
    $errorMessage = "Email пользователя должен быть до 50 символов!";
} elseif (empty($login)) {
    $errorMessage = "Заполните корректно поле Логин!";
} else {
    // Хэшируем пароль и вставляем данные нового пользователя в базу
данных
    $hashedPass = md5($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 ($result) {
            // Проверяем, есть ли данные
            if ($result->num_rows > 0) {
                // Получаем данные пользователя
                $userData = $result->fetch_assoc();

                // Устанавливаем сессионные переменные
                $_SESSION['id'] = $userData['id'];
                $_SESSION['login'] = $userData['login'];
                $_SESSION['role'] = $userData['role'];
                if ($_SESSION['role'] === "0") {
                    header("location: http://art-display/admin/posts/index.php");
                } else {
                    header("location: http://art-display/index.php");
                }
            }
        }
    }
}

```

```

        // Вывод данных для отладки
        // echo "User ID: " . $_SESSION['id'] . "<br>";
        // echo "Login: " . $_SESSION['login'] . "<br>";
        // echo "Role: " . $_SESSION['role'] . "<br>";
    }
    $successMessage = "Пользователь успешно зарегистрирован!";
    // Очистка полей после успешной регистрации
    $login = "";
    $email = "";
} else {
    $errorMessage = "Ошибка: " . $conn->error;
}
}
}
} else {
    $login = "";
    $email = "";
}
}

```

Этот фрагмент кода отвечает за регистрацию нового пользователя. Включает валидацию введенных данных и добавление нового пользователя в базу данных. Основные этапы работы:

Обработка данных POST-запроса:

- Получение и обработка данных, введенных пользователем (логин, пароль, повтор пароля и email).
- Установка значения роли пользователя (1 - обычный пользователь).

Проверка существующих пользователей:

- Проверка наличия пользователя с указанным логином в базе данных.
- Проверка наличия пользователя с указанным email в базе данных.

Валидация данных:

- Проверка совпадения паролей.
- Проверка длины логина, пароля и email.
- Проверка на заполненность поля логина.

Добавление нового пользователя:

- Хэширование пароля с использованием функции md5.
- Вставка данных нового пользователя в таблицу registeruser.
- Получение данных зарегистрированного пользователя для установки сессионных переменных.

Установка сессионных переменных и перенаправление:

- Установка сессионных переменных (ID, логин и роль пользователя).
- Перенаправление пользователя в зависимости от его роли (администратор или обычный пользователь).

Очистка полей и обработка ошибок:

- Очистка полей формы после успешной регистрации.
- Обработка и вывод сообщений об ошибках.

5.1.1.9 Функция просмотр страницы "О нас"

5.1.1.9.1 Описание

Позволяет пользователю просматривать страницу "О нас".

Приоритет: высокий.

5.1.1.9.2 Функциональные требования

Страница "О нас" содержит информацию о сайте и его возможностях.

Перейти на нее можно нажав на ссылку "О нас" в footer-е сайта.

5.1.1.10 Функция скачивания документов

5.1.1.10.1 Описание

Позволяет пользователю скачать документы: Условия пользования, Политика конфиденциальности.

Приоритет: высокий.

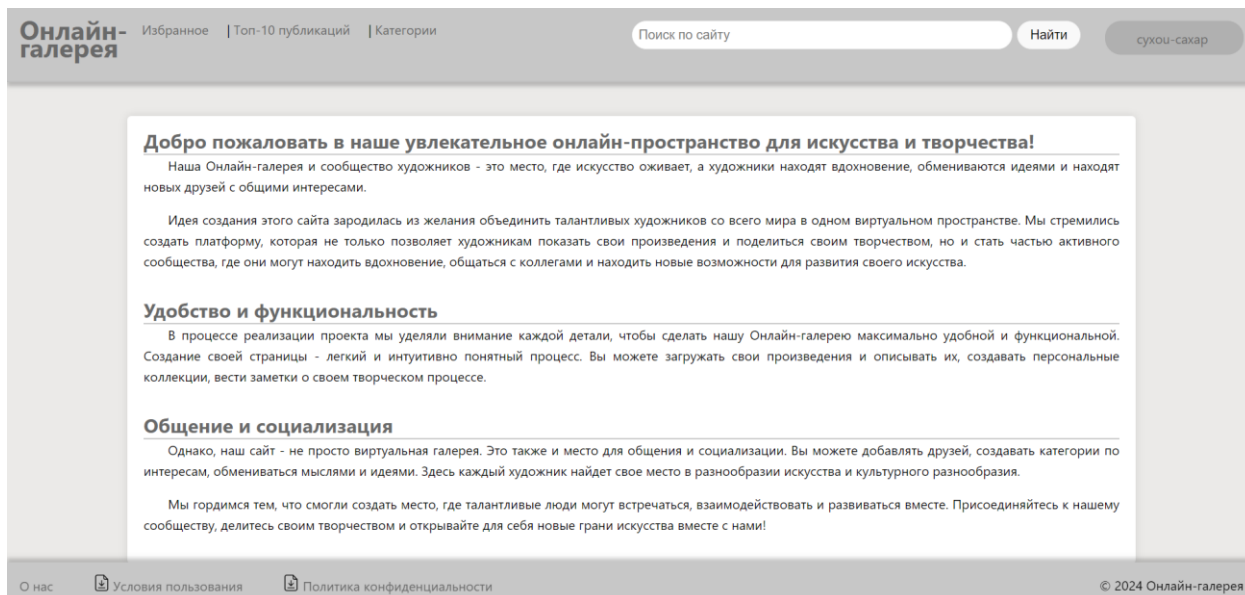


Рисунок 10 – Страница "О нас"

5.1.1.10.2 Функциональные требования

Файлы находятся в footer-е сайта. Если нажать на название файла, произойдет его скачивание. Файлы находятся в формате docx.



Рисунок 11 – Ссылки на документы

5.1.2 Функции подсистемы ведения пользователя от начала до конца сессии.

Данная подсистема должна выполнять следующие функции, позволяющие пользователям:

- просмотр своей страницы
- просмотр страниц других пользователей
- редактирование данных аккаунта
- просмотр портфолио
- управление (постами, категориями)
- просмотр подписчиков/подписок
- оформление подписки/отписки
- изменение обложки/аватара аккаунта
- комментирование постов
- добавление в избранное

4.2.2.1 Функция просмотра своей страницы

5.1.2.1.1 Описание

Позволяет пользователям просматривать собственную страницу.

Приоритет: высокий.

5.1.2.1.2 Функциональные требования

Если это страница пользователя, тогда ему будут доступны функции описанные далее. Доступны 8 кнопок, одна из которых в виде иконки. На странице аккаунта отображается обложка и аватар пользователя, а также возможность их редактирования (пункты 4.2.2.11, 4.2.2.12). При нажатии кнопки "Редактировать профиль" происходит переход на страницу редактирования (пункт 4.2.2.3). При нажатии на кнопку "Портфолио" в блок под кнопками отображаются посты, опубликованные пользователем, если постов нет, будет выведено сообщение об этом (пункт 4.2.2.4). После нажатия на кнопку "Управление постами" в блоке ниже будут отображены в таблице названия постов, их описание и кнопки для управления ими (пункт

4.2.2.5). После нажатия на кнопку "Подписки" и "Подписчики" в виде списка будут отображены пользователи (пункт 4.2.2.10). После нажатия на кнопку "Ваша статистика" будет произведен вывод статистики, которая содержит по оси x – количество работ, по оси y – даты их публикаций (пункт 4.2.4.1). Под никнеймом пользователя выводятся суммарные величины количества: просмотров, постов и подписчиков. Под кнопкой редактирования профиля отображается дата регистрации аккаунта.

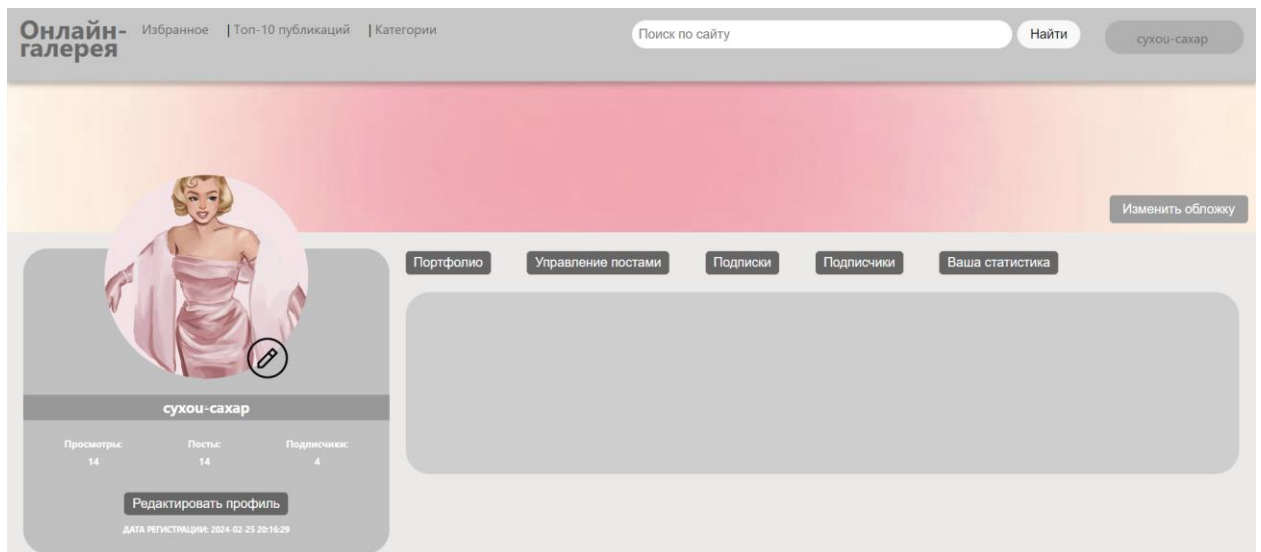


Рисунок 12 – Просмотр своей страницы

Реализация функции

JavaScript для обработки событий кнопок:

```
$(".buttons button").click(function(){
    var action = $(this).attr("class");
    $.ajax({
        url: "process.php",
        method: "POST",
        data: { action: action, user_id: <?=$user_id;?> },
        success: function(response){
            $(".content").html(response);
        }
    });
});
```

Этот фрагмент кода отслеживает клики по кнопкам и отправляет соответствующий запрос на сервер через AJAX. Полученные данные обновляют контент на странице без перезагрузки.

PHP для проверки авторизации пользователя и получения информации о нем:

```
session_start();
require_once('../function/users.php');
```

```

require_once('Z:/home/art-display/www/function/db.php');

$flag=false;
if (!($_SESSION)) {
    header("location: http://art-display/authorization.php");
    exit();
}

if (isset($_GET['userId']) && !($_GET['userId']==$_SESSION['id'])){
    $user_id = $_GET['userId'];
    $flag=true;
} else{
    $user_id = $_SESSION['id'];
}

$sql = "SELECT * FROM registeruser WHERE id = '$user_id'";
$result = $conn->query($sql);
$user = $result->fetch_assoc();

```

Этот участок кода начинает сессию, проверяет, авторизован ли пользователь. Если нет, он перенаправляет его на страницу входа. Затем он определяет идентификатор пользователя и извлекает его информацию из базы данных.

JavaScript для отображения статистики постов с использованием Chart.js:

```

$(".buttons button.statistic").click(function(){
    var action = $(this).attr("class");
    $.ajax({
        url: "process.php",
        method: "POST",
        data: { action: action, user_id: <?=$user_id;?> },
        success: function(response){
            var data = JSON.parse(response);
            // Создание графика с использованием данных из response
            // Добавление кнопки для сохранения статистики как изображения
        }
    });
});

```

Этот блок кода реагирует на клик по кнопке статистики пользователя и отправляет запрос на сервер для получения данных о статистике постов.

Полученные данные затем используются для создания графика с помощью библиотеки Chart.js, который отображается на странице пользователя.

PHP для получения информации о постах пользователя и их просмотрах:

```
$sql = "SELECT * FROM posts WHERE user_id = '$user_id'";
$result = $conn->query($sql);
$postSum = $result->num_rows;
$viewsSum=0;
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $postId = $row['id'];
        $sql = "SELECT * FROM visits WHERE post_id = '$postId'";
        $resultVis = $conn->query($sql);
        while ($vis = $resultVis->fetch_assoc()) {
            $viewsSum += 1;
        }
    }
}
```

Этот код выполняет запрос к базе данных для получения информации о постах пользователя и считает общее количество просмотров постов.

5.1.2.2 Функция просмотра страницы другого пользователя

5.1.2.2.1 Описание

Позволяет пользователям просматривать чужие аккаунты.

Приоритет: высокий.

5.1.2.2.2 Функциональные требования

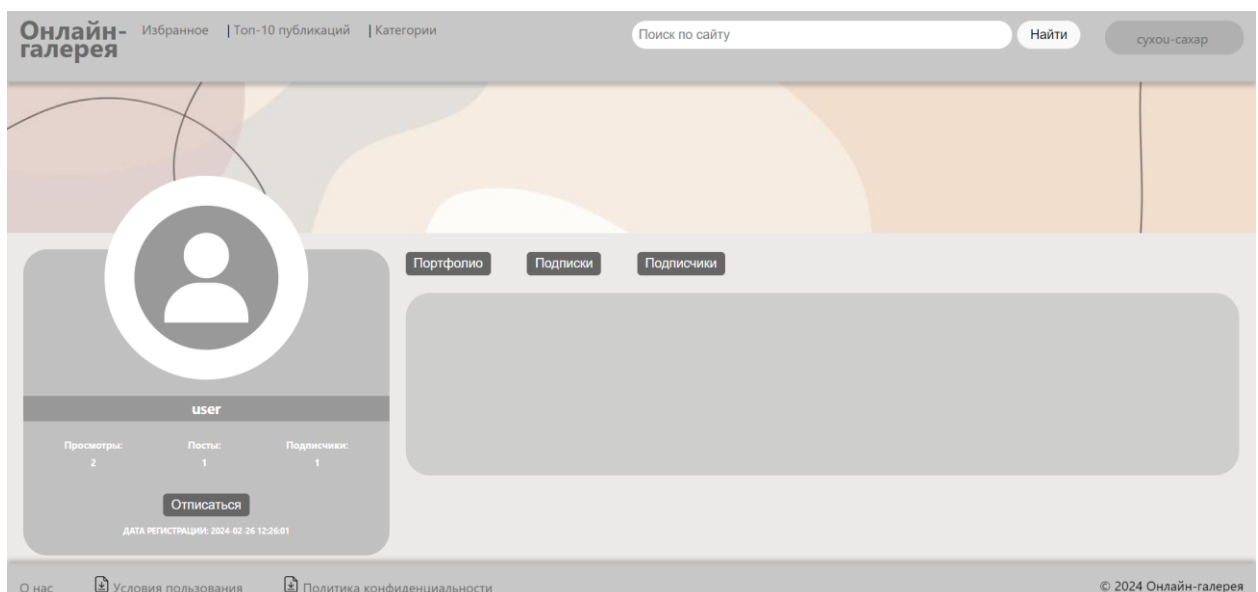


Рисунок 13 – Просмотр чужой страницы

При просмотре чужой страницы, пользователю доступна кнопка "Подписаться/Отписаться", при нажатии на нее происходит подписка или отписка соответственно. Можно просмотреть портфолио, подписки и подписчиков, аналогично, как на своей странице (пункты 4.2.2.4, 4.2.2.10).

Реализация функции

JavaScript для подписки пользователя:

```
$(".subscription").click(function(){
    var action = $(this).attr("class");
    $.ajax({
        url: "process.php",
        method: "POST",
        data: { action: action, user_id: <?=$user_id;?>, follower:
<?=$_SESSION['id'];?> },
        success: function(response){
            $(".content").html(response);
            location.reload();
        }
    });
});
```

Когда пользователь кликает на кнопку подписки, JavaScript обрабатывает событие и отправляет запрос на сервер (process.php) с указанием действия подписки, идентификатора пользователя и идентификатора текущего пользователя. После успешного выполнения запроса, содержимое блока с классом ".content" обновляется, и страница перезагружается.

JavaScript для отписки пользователя:

```
$(".unsubscribe").click(function(){
    var action = $(this).attr("class");
    $.ajax({
```

```

url: "process.php",
method: "POST",
data: { action: action, user_id: <?=$user_id;?>, follower:
<?=$_SESSION['id'];?> },
success: function(response){
    $(".content").html(response);
    location.reload();
}
});
});

```

При клике на кнопку отписки аналогичный процесс отправки AJAX запроса на сервер выполняется с указанием действия отписки, идентификатора пользователя и идентификатора текущего пользователя. После успешного выполнения запроса также обновляется содержимое блока ".content" и страница перезагружается, чтобы обновить данные о подписках.

5.1.2.3 Функция редактирования своих данных

5.1.2.3.1 Описание

Позволяет пользователю поменять данные своего аккаунта.

Приоритет: высокий.

5.1.2.3.2 Функциональные требования

Вся валидация аналогична (пункт 4.2.1.8). После нажатия на кнопку "Обновить", если все данные корректны и отличаются от старых, то произойдёт обновление.

**Обновление
пользователя**

Имя пользователя:

Новый пароль:

Электронная почта:

Обновить

Рисунок 14 – Редактирование своих данных

Реализация функции

Общий контейнер:

```
<div class="container">
```

Этот `<div>` ограничивает всю область контента, где размещается форма обновления пользователя.

Блок с постами:

```
<div class="posts col-10" style="margin: auto">
```

Это блок, который содержит форму. `col-10` указывает на то, что блок занимает 10 колонок в сетке, а `margin: auto` центрирует его по горизонтали.

Заголовок:

```
<h2>Обновление пользователя</h2>
```


Просто заголовок, который сообщает о том, что эта форма предназначена для обновления информации о пользователе.

Проверка на наличие сообщений об ошибках или успехе:

```
<?php if (!empty($errorMessage)): ?>
    <!-- Код для вывода сообщения об ошибке -->
<?php elseif (!empty($successMessage)): ?>
    <!-- Код для вывода сообщения об успехе -->
<?php endif; ?>
```

Здесь проверяется наличие сообщений об ошибках или успехе, и если таковые имеются, они выводятся соответственно.

Форма обновления пользователя:

```
<form method="post" enctype="multipart/form-data">
```

Форма, отправляющая данные методом POST и позволяющая загружать файлы.

Поля ввода:

```
<input value="<?=$user['id']?>" name="id" type="hidden">
```

Это скрытое поле для передачи id пользователя. Аналогично для других полей.

Элементы формы:

```
<label for="content" class="form-label">Имя пользователя: *</label>
<input type="text" class="form-control" value="<?=$user['login']?>"
placeholder="Логин" name="login" required>
```

Это поля для ввода имени пользователя, нового пароля и электронной почты. value="<?=\$user['login']?>" позволяет вставить текущее значение из переменной \$user.

Кнопка отправки формы:

```
<button name="user-edit" class="button-row" type="submit" class="btn-
add">Обновить</button>
```

Это кнопка отправки формы. При нажатии на нее данные формы отправляются на сервер для обработки.

5.1.2.4 Функция отображения портфолио

5.1.2.4.1 Описание

Позволяет просматривать опубликованные данным пользователем посты.

Приоритет: высокий.

5.1.2.4.2 Функциональные требования

Вывод постов аналогично, как на главной странице (пункт 4.2.1.1).
Ссылки в виде названия поста ведут на страницу с постом (пункт 4.2.1.2).

Используется код аналогичный коду вывода постов на главной странице сайта.

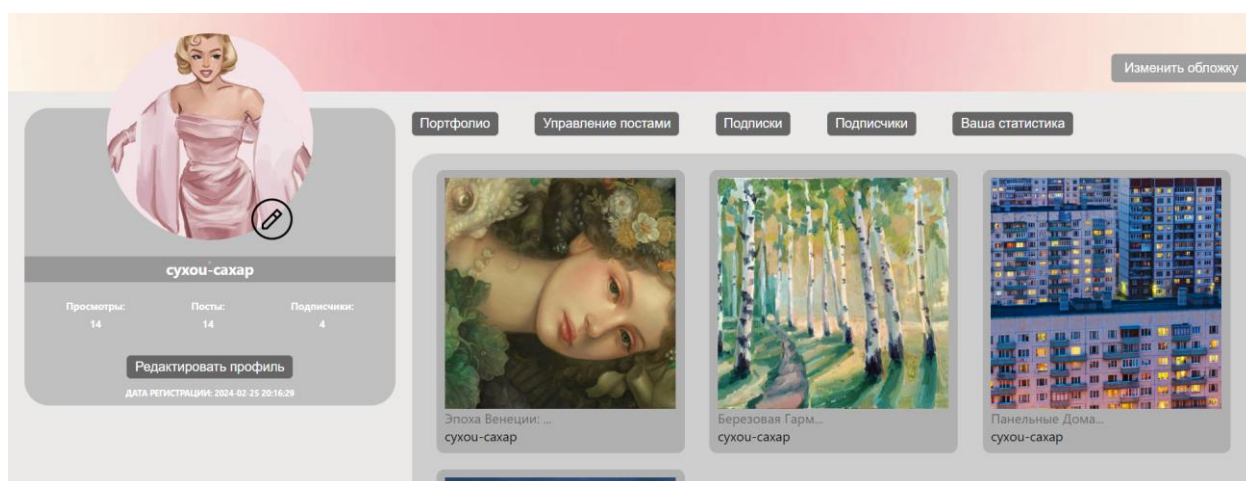


Рисунок 15 – Портфолио пользователя

5.1.2.5 Функция управления постами со страницы пользователя

5.1.2.5.1 Описание

Позволяет пользователю управлять своими постами, просматривать категории и добавлять их.

Приоритет: высокий.

5.1.2.5.2 Функциональные требования

При нажатии на кнопку "Вернуться в профиль" будет происходить переход обратно на страницу аккаунта (пункт 4.2.2.1).

После нажатия на кнопку "Посты" в блоке под кнопкой будут отображаться две кнопки "Добавление" и "Управление". При нажатии на кнопку "Добавление" под ней отобразится окно с добавлением поста (пункт 4.2.2.6). При нажатии на кнопку "Управление" под ней отобразится окно с таблицей. В таблицу выводятся данные из базы данных: ид, название, автор поста. А также кнопки управления: edit, delete. При нажатии на кнопку edit происходит переход на страницу редактирования (пункт 4.2.2.6). При нажатии кнопки delete происходит удаление поста.

После нажатия на кнопку "Категории" в блоке под кнопкой будут отображаться две кнопки "Добавление" и "Просмотр". При нажатии на кнопку "Добавление" под ней отобразится окно с добавлением категории (пункт 4.2.2.9). При нажатии на кнопку "Просмотр" под ней отобразится окно с таблицей, в которой находятся ид, название, описание категории (пункт 4.2.2.8).

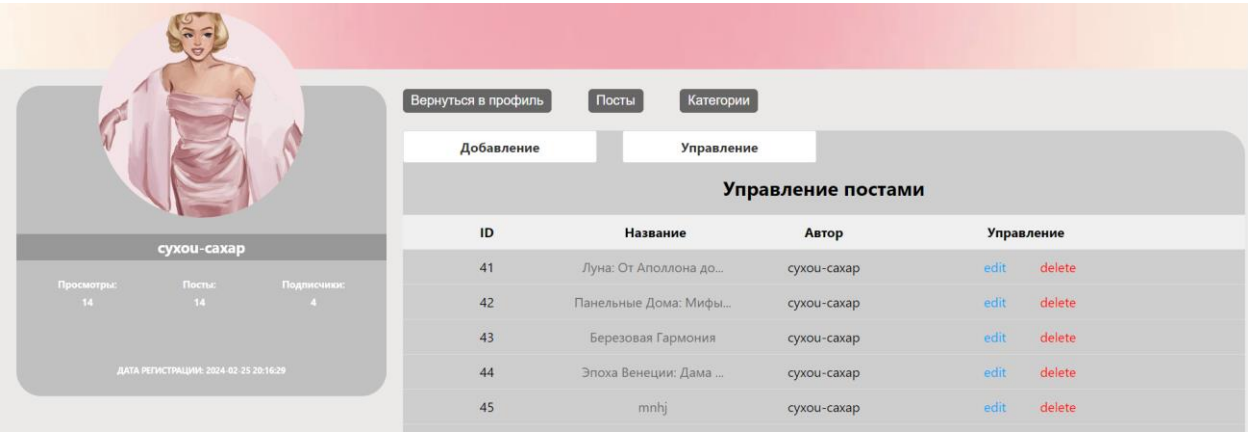


Рисунок 16 – Управление

5.1.2.6 Функция добавление поста

5.1.2.6.1 Описание

Позволяет пользователю добавить пост.

Приоритет: высокий.

5.1.2.6.2 Функциональные требования

В поле название поста можно ввести данные строкового типа. В описание можно применить стили для текста, добавить список, табуляцию, допускается ввод спец. символов и эмодзи. Изображение должно быть

определенного расширения: jpeg, jpg, png. Вес не должен превышать 2048 Кб. Категория представляет собой выпадающий список, из которого выбирается категория. При нажатии на кнопку «Добавить» происходит проверка на пустые строки и корректность введенных данных, если все проверки пройдены, отображается сообщение, что пост добавлен, иначе – ошибка с описанием проблемы.

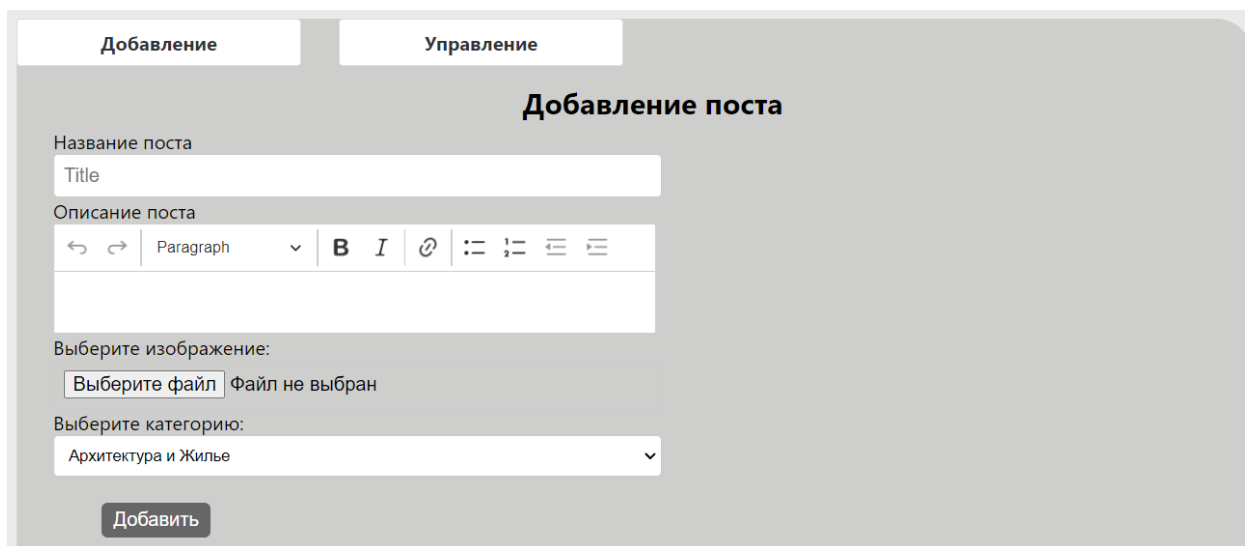


Рисунок 17 – Добавление поста

Реализация функции

Добавление поста:

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

    $title = htmlspecialchars(trim($_POST['title']));
    $content = trim($_POST['content']);
    $category=$_POST['category'];
    if (isset($_POST['id'])){
        $id=$_POST['id'];
    }

    if (empty($title) || empty($content) || empty($category) ||
empty($_FILES['img']['size'])) {
        $errorMessage= "Заполните все поля, изображение должно весить
меньше 2048 Кб";
    } else {
        if (isset($_FILES['img']) && !empty($_FILES['img']['name'])) {
            $img = addslashes(file_get_contents($_FILES['img']['tmp_name']));}
        else {
```

```

$errorMessage= "Изображение не было загружено.";
echo 'Error code: ';
}}
$sql="SELECT * FROM `categories` WHERE name = '$category'";
$resultUni = $conn->query($sql);
$row=$resultUni->fetch_assoc();
//var_dump($row);
//exit();
$idCategory=$row['id'];
$userId=$_SESSION['id'];

if ((mb_strlen($title,'UTF8')<3)||(mb_strlen($title,'UTF8')>255)){
    $errorMessage = "Название поста должно быть от 3-х до 255-и
СИМВОЛОВ!";
}
else {
    $sql = "INSERT INTO `posts` (title,content,user_id,img,category_id)
VALUES ('$title','$content', '$userId', '$img','$idCategory)";
    if ($conn->query($sql)) {
        $postId = $conn->insert_id;
        $sql = "SELECT * FROM `posts` WHERE id = '$postId'";
        $result = $conn->query($sql);
        if ($result) {
            $successMessage = "Пост добавлен!";

            $title="";
            $content="";
            $category="";
            $img="";

        } else {
            $errorMessage = "Ошибка: " . $conn->error;
        }
    }
}
}}else{
$title="";
$content="";
$category="";
$img="";
}

```

- Вначале проверяется метод запроса и наличие параметра **add_post**.

- Затем извлекаются данные из формы, введенные пользователем.

- Проверяются обязательные поля и размер загружаемого изображения.
- После этого проверяется длина заголовка поста.
- Если все проверки проходят успешно, данные добавляются в базу данных.

5.1.2.7 Функция обновления поста

5.1.2.7.1 Описание

Позволяет пользователю обновлять данные поста.

Приоритет: высокий.

5.1.2.7.2 Функциональные требования

Валидация та же, что при добавлении поста (пункт 4.2.2.6). После нажатия на кнопку "Обновить", если данные введены корректно, происходит переадресация на страницу управления, обновляются данные в базе данных. Если данные не прошли проверку, произойдет вывод сообщения об ошибке с описанием проблемы.

Рисунок 18 – Обновление поста

Реализация функции

Обновление поста:

```

if ($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['id'])) {
    $id=$_GET['id'];
    $sql="SELECT * FROM `posts` WHERE id = '$id'";
    $result = $conn->query($sql);
    $post = $result->fetch_assoc();
    $id=$post['id'];
    $name=$post['title'];
    $content=$post['content'];
    $category=$post['category_id'];
    $img=$post['img'];

    $sql="SELECT name FROM `categories` WHERE id = '$category'";
    $resultUni = $conn->query($sql);
    $row=$resultUni->fetch_assoc();
    $nameCategory=$row['name'];
}

// ...

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['posts-
edit'])) {
    $id = $_POST['id'];
    $name = htmlspecialchars(trim($_POST['title']));
    $content = trim($_POST['content']);
    $category = $_POST['category'];
    // var_dump($_POST);

    // Получение ID категории
    $sql = "SELECT id FROM `categories` WHERE name = '$category'";
    $resultUni = $conn->query($sql);
    $row = $resultUni->fetch_assoc();
    $idCategory = $row['id'];
    if (empty($name) || empty($content)){
        $errorMessage = "Название поста и его описание не должны быть
пусты!";
    }
    elseif (empty($_FILES['img']['size']) && !empty($_FILES['img']['name'])){
        $errorMessage= "Изображение должно весить меньше 2048 Кб";
        $post['img']=$_POST['imgPreview'];
    }
    elseif ((mb_strlen($name,'UTF8')<3)||(mb_strlen($name,'UTF8')>255)){
        $errorMessage = "Название поста должно быть от 3-х до 255-и
СИМВОЛОВ!";
    } else {

```

```

// Проверка наличия нового изображения
if (!empty($_FILES['img']['name'])) {
    $img = addslashes(file_get_contents($_FILES['img']['tmp_name']));

    // Обновление с изображением
    $sql = "UPDATE `posts` SET title='$name', content='$content',
category_id=$idCategory, img='$img' WHERE id='$id'";
} else {
    // Обновление без изображения
    $sql = "UPDATE `posts` SET title='$name', content='$content',
category_id=$idCategory WHERE id='$id'";
}

// Выполнение запроса
if ($conn->query($sql) === TRUE) {
    if (isset($_POST['us_id'])) {
        $id=$_POST['us_id'];
        header("location: http://art-display/userPage/index.php?us_id=$id");
        exit();
    }
    header("location: /admin/posts/index.php");
} else {
    $errorMessage = "Ошибка при обновлении поста: " . $conn->error;
}
}
}

```

- Если метод запроса - POST и существует параметр posts-edit, то это означает, что происходит редактирование поста.

- Извлекаются данные из формы.
- Проверяется наличие обязательных данных, а также проверяется длина заголовка.

- Если есть новое изображение, оно обрабатывается и обновляется в базе данных.

5.1.2.8 Функция просмотров категорий

5.1.2.8.1 Описание

Позволяет пользователю просматривать все категории.

Приоритет: высокий.

5.1.2.8.2 Функциональные требования

В таблице отображаются данные из базы: ид, название, описание. После нажатия на название категории происходит переход на страницу данной категории (пункт 4.2.1.5).

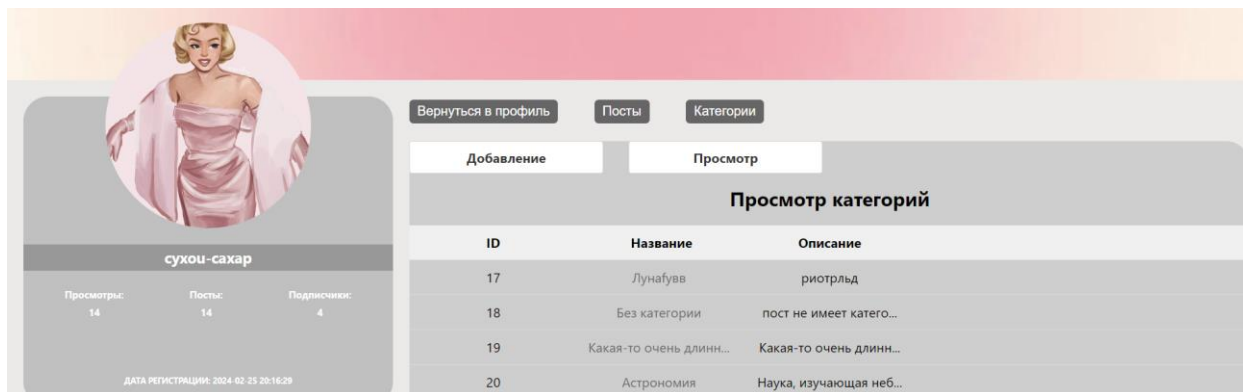


Рисунок 19 – Просмотр категорий

5.1.2.9 Функция добавления категории

5.1.2.9.1 Описание

Позволяет пользователю добавить категорию.

Приоритет: высокий.

5.1.2.9.2 Функциональные требования

В поле названия и описания можно ввести данные строкового типа. Если поля не пусты и прошли валидацию, после нажатия на кнопку "Добавить", категория будет добавлена в базу данных. Если данные не прошли проверку, произойдет вывод сообщения об ошибке с описанием проблемы.

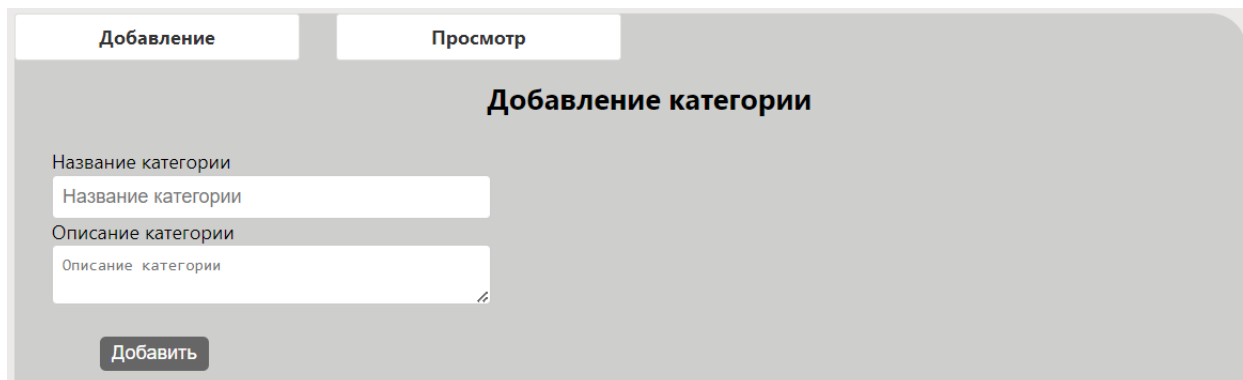


Рисунок 20 – Добавление категории

Реализация функции

Добавление категории:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['categories-create'])) {
//var_dump($_POST);
//exit();
$name = htmlspecialchars(trim($_POST['name']));
$description = htmlspecialchars(trim($_POST['description']));
if (!(empty($name)) && !(empty($description))) {
    $sql = "SELECT * FROM `categories` WHERE name = '$name'";
    $resultUni = $conn->query($sql);

    if ((mb_strlen($name, 'UTF8') < 3) || (mb_strlen($name, 'UTF8') > 60)) {
        $errorMessage = "Название категории должно быть от 3-х до 60-и
символов!";
    } elseif ($resultUni->num_rows > 0) {
        while ($row = $resultUni->fetch_assoc()) {
            $errorMessage = "Категория с названием: " . $row['name'] . " уже
существует!";
        }
    } elseif ((mb_strlen($description, 'UTF8') < 3)) {
        $errorMessage = "Описание категории должно быть от 3-х символов!";
    } else {
        $sql = "INSERT INTO `categories` (name, description) VALUES ('$name',
'$description')";
        $resultUni = $conn->query($sql);
        $successMessage = "Категория добавлена!";
        // Очистите поля после успешной регистрации
        $name = "";
        $description = "";
    }
} else {
    $errorMessage = "Название и описание не должны быть пустыми!";
}
} else {
    $name = "";
    $description = "";
}
```

Проверка метода запроса и наличия параметра categories-create:

Если метод запроса POST и существует параметр categories-create, то происходит создание категории.

Извлечение данных из формы:

Извлекаются значения имени и описания категории из формы.

Проверка наличия данных:

Проверяется, чтобы поля для имени и описания категории не были пустыми.

Проверка условий для валидации данных:

Проверяется длина имени категории (от 3 до 60 символов) и длина описания (минимум 3 символа).

Проверяется уникальность имени категории в базе данных.

Добавление категории:

Если все проверки пройдены успешно, то выполняется SQL-запрос на добавление категории в базу данных.

После успешного добавления категории выводится сообщение об успехе.

Поля для ввода очищаются для возможности создания новой категории.

Обработка ошибок:

Если какое-либо из условий проверки не выполнено, то формируется сообщение об ошибке, которое будет отображено пользователю.

5.1.2.10 Функция отображения подписчиков и подписок

5.1.2.10.1 Описание

Позволяет пользователю просматривать подписчиков и подписки.

Приоритет: высокий.

5.1.2.10.2 Функциональные требования

На странице отображаются в виде списка никнеймы пользователей. При нажатии на них происходит переход на их страницы (пункт 4.2.2.2).

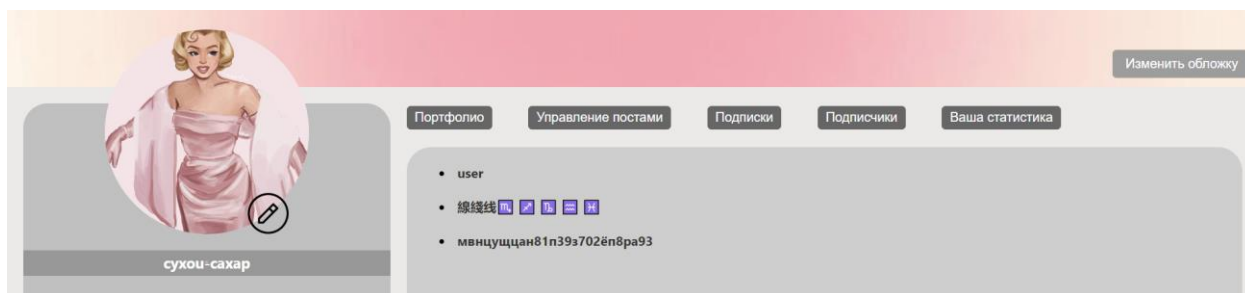


Рисунок 21– Подписки и подписчики

5.1.2.11 Функция обновления обложки

5.1.2.11.1 Описание

Позволяет пользователю изменить обложку страницы.

Приоритет: низкий.

5.1.2.11.2 Функциональные требования

После выбора изображения, оно отобразится для просмотра общего вида страницы. Если пользователю всё понравится, после нажатия на кнопку "Сохранить обложку" произойдет переход на его страницу с обновленной обложкой (пункт 4.2.2.1).



Рисунок 22 – Обновление обложки

5.1.2.12 Функция обновления аватара

5.1.2.12.1 Описание

Позволяет пользователю изменить аватар своего аккаунта.

Приоритет: низкий.

5.1.2.12.2 Функциональные требования

После выбора изображения, оно отобразится для просмотра общего вида страницы. Если пользователю всё понравится, после нажатия на кнопку

"Сохранить аватар" произойдет переход на его страницу с обновленным аватаром (пункт 4.2.2.1).

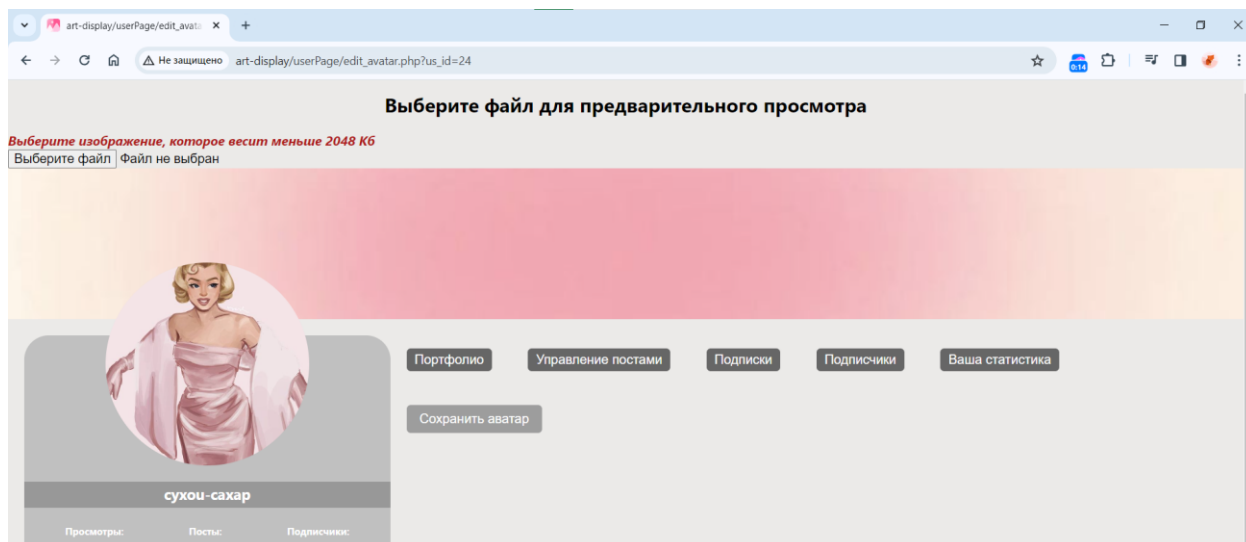


Рисунок 23 – Обновление аватара

5.1.2.13 Функция комментирования

5.1.2.13.1 Описание

Позволяет пользователю комментировать пост, управлять комментарием.

Приоритет: высокий.

5.1.2.13.2 Функциональные требования

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 (пункт 4.2.1.2). В комментариях никнейм пользователя является кликабельным (пункт 4.2.2.2), справа отображается дата и время создания.

Комментарий:

Отправить

Комментарии к посту

сухой-сахар

2024-03-20
10:22:02

хочется погулять в такой роще :)





Рисунок 24.1 – Комментарий

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

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 24.2 – Редактирование комментария

Комментарии к посту

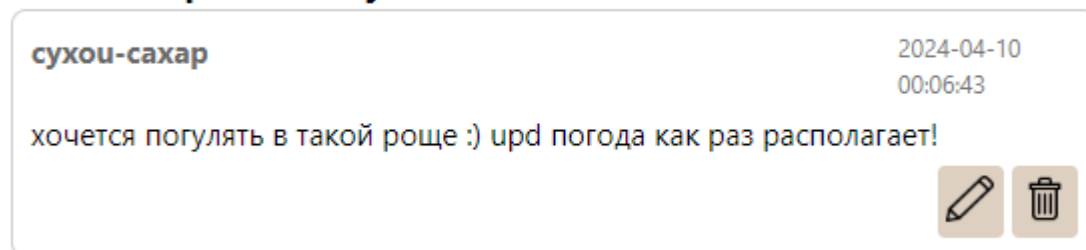


Рисунок 24.3 – Отредактированный комментарий

5.1.2.14 Функция добавления в избранное

5.1.2.14.1 Описание

Позволяет пользователю добавить пост в избранное или убрать его из него.

Приоритет: низкий.

5.1.2.14.2 Функциональные требования

Справа от просмотров находится иконка в виде сердца, она отображается только для зарегистрированных пользователей. Если иконка белая – пост не в избранном, если красным – пост добавлен в избранное. На иконку можно нажать и выполнить операцию добавления или удаления. После добавления поста в избранное, его можно наблюдать на странице "Избранное", если нажать на ссылку в header-е (пункт 4.2.1.3).



Панельные Дома: Мифы и Реальность



сухой-сахар



2024-03-12 23:39:41

Что скрывается за фасадом панельных зданий? Давайте разберемся в мифах и узнаем, что делает панельные дома актуальными и интересными.

1. Прошлое и Настоящее

Исследуйте историю панельных домов: от времен, когда они были символом советского строительства, до современных тенденций в их дизайне и обновлении.

2. Комфорт и Энергоэффективность

Омрачены многими стереотипами, панельные дома сегодня предлагают уют и экономию энергии. Узнайте, какие технологии делают их современными и комфортабельными.

3. Дизайн и Инновации

Покопаясь в новейших тенденциях дизайна панельных домов. Как архитектурные решения и инновации преобразуют стандартные фасады и интерьеры?

4. Будущее Жилья: Панели Reloaded

Какие тенденции ждут панельные дома в будущем? Рассмотрим перспективы обновления существующих зданий и строительства новых, отвечающих современным требованиям.

Откройте для себя мир панельных домов - с их историей, современными тенденциями и будущими возможностями!

Категория: Архитектура и Жилье



Просмотры: 2

Комментарий:

Отправить

Рисунок 25 – Добавление в избранное

Функции подсистемы контроля за сайтом

Данная подсистема должна выполнять следующие функции:

- возможность просмотра всех постов, пользователей, категорий
- возможность добавления/удаления постов, категорий и пользователей
- возможность блокировки постов и пользователей

Возможность пользования данной подсистемой имеет только администратор.

5.1.2.15 Функция панели администратора

5.1.2.15.1 Описание

Позволяет администратору управлять постами, пользователями и категориями.

Приоритет: высокий.

5.1.2.15.2 Функциональные требования

На странице отображается боковая панель, на которой отмечены вкладки: посты, пользователи, категории. При нажатии на "Посты" произойдет переход на страницу с управлением постами(пункт 4.2.2.5). При нажатии на "Пользователи" произойдет переход на страницу с управлением пользователями (пункт 4.2.3.2). При нажатии на "Категории" произойдет переход на страницу с управлением категориями (пункт 4.2.2.9). 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.

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
	43	Березовая Гармония	сухой-сахар	edit	delete	block
Категории	44	Эпоха Венеции: Дама ...	сухой-сахар	edit	delete	block
	45	mnhj	сухой-сахар	edit	delete	unblock

Рисунок 26.1 – Создание аккаунта

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

Рисунок 26.2 – Создание аккаунта

Посты

Пользователи

Категории

Добавление

Управление

Управление категориями

ID	Название	Описание	Управление	
17	Лунафувв	риотрльд	edit	delete
18	Без категории	пост не имеет катего...		
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

Рисунок 26.3 – Создание аккаунта

Реализация функции

```

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 = "";
}

```

Проверка метода запроса и наличия параметра user-create:

Если метод запроса POST и существует параметр user-create, то происходит создание нового пользователя.

Извлечение данных из формы:

Извлекаются значения логина, пароля, повторного ввода пароля и электронной почты из формы.

Определяется роль пользователя на основе наличия параметра isAdmin.

Проверка наличия данных и уникальности:

Проверяется, чтобы логин и электронная почта пользователя были уникальными.

Проверяется совпадение паролей.

Проверяется длина логина, пароля и электронной почты.

Хэширование пароля и добавление пользователя:

Если все проверки пройдены успешно, пароль хэшируется и пользователь добавляется в базу данных.

После успешного добавления пользователя выводится сообщение об успехе.

Поля для ввода логина и электронной почты очищаются для возможности создания нового пользователя.

Обработка ошибок:

Если какое-либо из условий проверки не выполнено, формируется сообщение об ошибке, которое будет отображено пользователю.

5.1.2.16 Функция управления пользователями

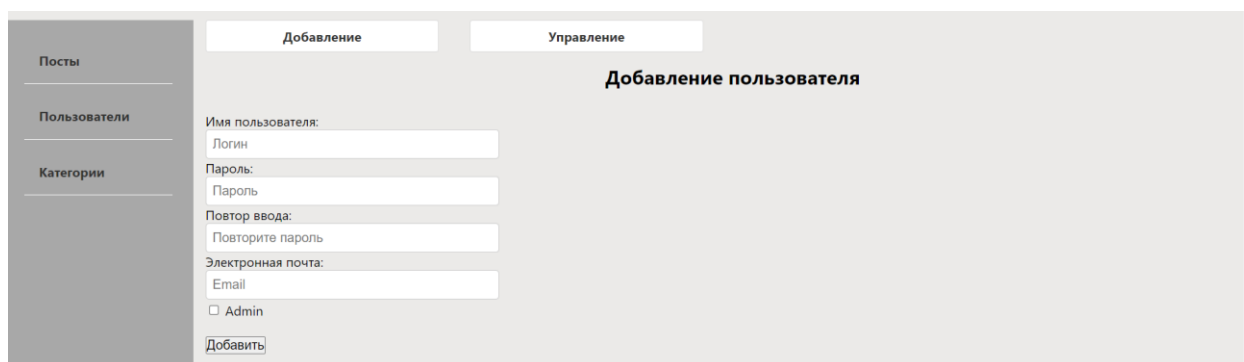
5.1.2.16.1 Описание

Позволяет администратору управлять пользователями.

Приоритет: высокий.

5.1.2.16.2 Функциональные требования

При нажатии на кнопку "Добавить" произойдет переход на страницу добавления пользователя. Вся валидация соответствует пункту 4.2.1.8. 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.

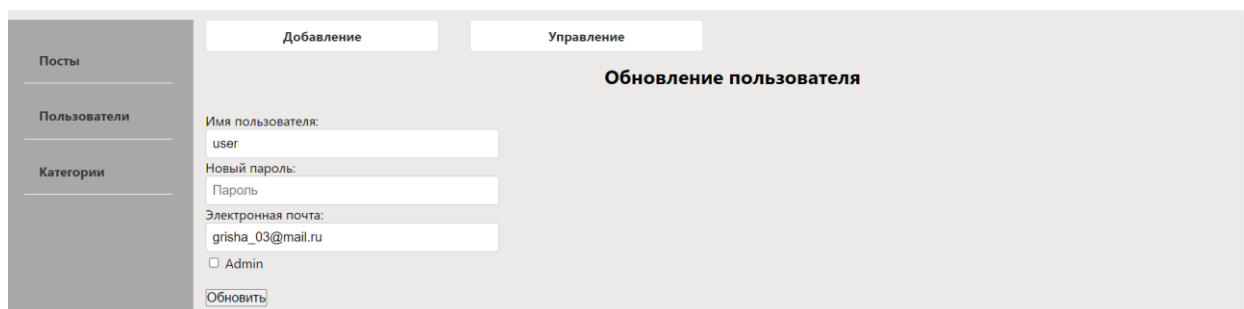


The screenshot shows a web application interface with a sidebar on the left containing links for 'Посты', 'Пользователи', and 'Категории'. The main content area has two tabs: 'Добавление' (selected) and 'Управление'. Under the 'Добавление' tab, the title 'Добавление пользователя' is displayed. The form includes the following fields: 'Имя пользователя:' (with a sub-label 'Логин'), 'Пароль:' (with a sub-label 'Пароль'), 'Повтор ввода:' (with a sub-label 'Повторите пароль'), and 'Электронная почта:' (with a sub-label 'Email'). There is an 'Admin' checkbox and a 'Добавить' button at the bottom.

Рисунок 27 – Добавление пользователя

При нажатии на кнопку "edit" произойдет переход на страницу редактирования пользователя, старые данные пользователя автоматически введутся в поля. Вся валидация соответствует пункту 4.2.1.8. 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.



The screenshot shows the same web application interface as Figure 27, but the 'Управление' tab is selected. The title 'Обновление пользователя' is displayed. The form includes the following fields: 'Имя пользователя:' (with a sub-label 'user'), 'Новый пароль:' (with a sub-label 'Пароль'), and 'Электронная почта:' (with a sub-label 'grisha_03@mail.ru'). There is an 'Admin' checkbox and an 'Обновить' button at the bottom.

Рисунок 28 – Обновление пользователя

5.1.3 Функции подсистемы генерации отчётности

Данная подсистема должна выполнять следующие функции:

- возможность генерации статистики публикации постов за определённый период времени

5.1.3.1 Функция просмотра статистики пользователя

5.1.3.1.1 Описание

Позволяет пользователю просмотреть и сохранить свою статистику.

Приоритет: высокий.

5.1.3.1.2 Функциональные требования

По оси x указано количество работ, по оси y – даты их публикаций. После нажатия на кнопку "Сохранить статистику", статистика будет сохранена в формате png.

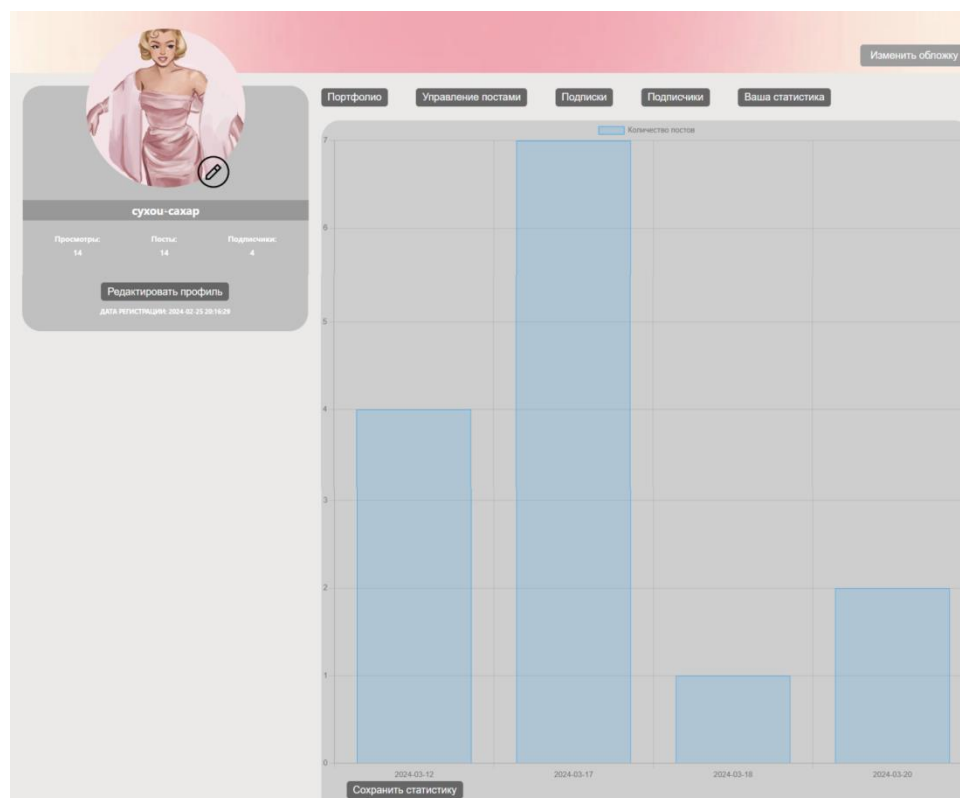


Рисунок 29 – Статистика на странице

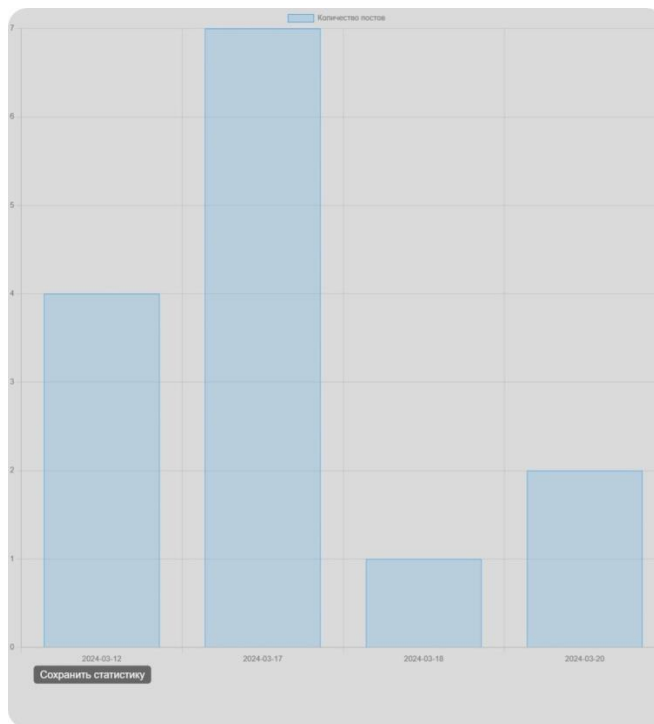


Рисунок 30 – Сохраненная статистика

Реализация функции

```
<script>
$(document).ready(function(){
    // Обработчик события click для всех кнопок
    $(".buttons button").click(function(){
        var action = $(this).attr("class");
        $.ajax({
            url: "process.php",
            method: "POST",
            data: { action: action, user_id: <?=$user_id;?> },
            success: function(response){
                $(".content").html(response);
            }
        });
    });

    $(".buttons button.statistic").click(function(){
        var action = $(this).attr("class");
        $.ajax({
            url: "process.php",
            method: "POST",
            data: { action: action, user_id: <?=$user_id;?> },
            success: function(response){
                var data = JSON.parse(response);
            }
        });
    });
});
</script>
```

```

// Получаем массивы для дат и количества постов из данных
var dates = data.labels;
var counts = data.posts;

// Создаем элемент canvas для отображения графика
var canvas = $('<canvas>').attr('id', 'postChart').attr('width',
'625').attr('height', '625');

// Добавляем созданный canvas внутрь div с классом content
$('.content').html(canvas);

// Получаем контекст canvas для отображения графика
var ctx = document.getElementById('postChart').getContext('2d');

// Создаем новый график с использованием Chart.js
var postChart = new Chart(ctx, {
  type: 'bar',
  data: {
    labels: dates, // Даты постов
    datasets: [{
      label: 'Количество постов',
      data: counts, // Количество постов
      backgroundColor: 'rgba(54, 162, 235, 0.2)',
      borderColor: 'rgba(54, 162, 235, 1)',
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      yAxes: [{
        ticks: {
          beginAtZero: true
        }
      }]
    }
  }
});

// Добавляем кнопку для сохранения статистики
var saveButton = $('<button class="saveButton">').text('Сохранить
статистику').addClass('save-statistic-button');
$('.content').append(saveButton);

// Обработчик события click для кнопки сохранения статистики

```



```

        $('.save-statistic-button').click(function(){
            // Используем html2canvas для создания скриншота элемента
с классом content
            html2canvas($('.content')[0]).then(function(canvas) {
                // Получаем данные изображения в формате base64
                var imageData = canvas.toDataURL('image/png');

                // Отправляем изображение на сервер для сохранения
                $.ajax({
                    url: "save_image.php",
                    method: "POST",
                    data: { image: imageData },
                    success: function(response){
                        console.log("Статистика успешно сохранена в виде
изображения!");
                    },
                    error: function(xhr, status, error) {
                        console.error("Произошла ошибка при сохранении
статистики как изображения: " + error);
                    }
                });
            });
        });
    });

```

1. Глобальные обработчики событий:
 - `$(document).ready(function(){...});`: Этот код выполняется, когда весь HTML-документ полностью загружен и готов к манипуляциям JavaScript.
2. Обработчики событий для кнопок:
 - `$(".buttons button").click(function(){...});`: Этот обработчик срабатывает при клике на любую кнопку в блоке с классом `.buttons`.
 - `$(".buttons button.statistic").click(function(){...});`: Этот обработчик срабатывает только при клике на кнопку с классом `.statistic` в блоке `.buttons`.
3. AJAX-запросы:

- Отправляются асинхронные запросы на сервер методом POST с помощью jQuery AJAX.

- Для каждого запроса указывается URL-адрес обработчика на сервере (process.php или save_image.php), метод запроса, данные для отправки и функция обратного вызова при успешном выполнении запроса.

4. Обновление контента на странице:

- В случае успешного выполнения запроса контент страницы обновляется с помощью jQuery: `$(".content").html(response);`.

5. Создание графика с использованием Chart.js:

- Полученные данные для графика обрабатываются и отображаются на странице в виде столбчатой диаграммы с помощью библиотеки Chart.js.

- Добавляется кнопка для сохранения статистики в виде изображения.

6. Сохранение статистики в виде изображения:

- При клике на кнопку "Сохранить статистику" используется библиотека html2canvas для создания скриншота области с классом .content.

- Скриншот преобразуется в формат изображения и отправляется на сервер для сохранения в виде файла.

ТЕСТИРОВАНИЕ

Сводный отчет об ошибках

ИДЕНТИФИКАТО Р ДЕФЕКТА	НАЗВАНИЕ МОДУЛЯ	ВЕРСИ Я	КРАТКОЕ ОПИСАНИЕ	ШАГИ ПО ВОСПРОИЗВЕДЕНИЮ	ОЖИДАЕМЫЙ РЕЗУЛЬТАТ	ФАКТИЧЕСКИЙ РЕЗУЛЬТАТ	СЕРЬЕЗНОС ТЬ ДЕФЕКТА	ПРИОРИТЕ Т ДЕФЕКТА	НАЗНАЧЕН	СТАТУС
1.2.4	<u>Авторизация</u>	1.0	Пользователь с введенным логином и паролем существует в системе, но заблокирован модерацией (страница заморожена)	Ввести данные заблокированного пользователя. Нажать кнопку "Войти"	Вывод сообщения о блокировке пользователя, авторизация не прошла	Вывод сообщения о блокировке пользователя, авторизация прошла	<u>Высокая</u>	<u>Высокий</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
1.3.1	<u>Валидация полей</u>	1.0	<u>Максимальная и минимальная длина</u>	Ввод данных, нажатие на кнопку отправки их на сервер	Если данные прошли <u>валидацию</u> , отправить данные, если нет – вывести сообщение об ошибке	Не везде есть проверка на максимальную длину, поэтому данные обрезаются в базе данных	<u>Высокая</u>	<u>Высокий</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
1.3.4	<u>Отображение астериска</u>	1.0	Убедитесь, что астериск (знак звездочки) отображается у всех обязательных полей	Зайти на страницы, где есть обязательные поля.	<u>Отображение астериска</u>	<u>Астериски отсутствуют</u>	<u>Средняя</u>	<u>Средний</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
1.4.0	<u>Форма обратной связи</u>	1.0	<u>Наличие формы обратной связи</u>	Зайти на страницу с формой обратной связи, убедиться в её корректности	Отображение формы, ввод данных, отправка	<u>Форма отсутствует</u>	<u>Высокий</u>	<u>Высокий</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
1.2.3	<u>Поиск пустой строки</u>	1.0	<u>Пустой поисковый запрос</u>	Ничего не вводить в строку поиска, нажать на кнопку "Поиск"	<u>Ничего не происходит</u>	Отображаются все данные, которые есть на сайте	<u>Высокая</u>	<u>Высокий</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
1.4.2	Отображаются подтверждающие сообщения для операций обновления и удаления	1.0	Подтверждающие сообщения отображаются для операций обновления и удаления.	Удалить или обновить пост/пользователя/категорию.	<u>Отсутствие всплывающих окон</u>	<u>Появление всплывающего окна</u>	<u>Средняя</u>	<u>Средний</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
1.7.0	favicon	1.0	<u>Наличие favicon</u>	Открыть сайт, рядом с названием увидеть иконку	<u>Наличие иконки</u>	<u>Иконка отсутствует</u>	<u>Низкая</u>	<u>Низкий</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
1.12.2	<u>Кеш, cookie и сессии</u>	1.0	Пользователь удалил куки, находясь на сайте	Очистить <u>куки</u> , находясь в сессии	При обновлении страницы происходит переадресация на <u>главную</u> , сессия завершается	При обновлении страницы сессия завершается, но пользователь остается на странице.	<u>Высокая</u>	<u>Высокий</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>
3.1.0	<u>Лимит авторизаций</u>	1.0	<u>Пользователь достиг лимита авторизаций</u>	Ввести данные неверно три раза	Сообщение о лимите авторизации, таймер на 10 минут	Пользователь может сколько угодно раз попытаться войти	<u>Высокая</u>	<u>Высокий</u>	<u>Гришкина П. В.</u>	<u>Открыт</u>

3.2.0	Страница авторизации и регистрации открываются через HTTPS	1.0	Регистрация и Вход через страницы на HTTPS	Открыть страницы с регистрацией или со входом	Страницы открываются через HTTPS	Страницы открываются через HTTP	Высокая	Высокий	Гришкина П. В.	Открыт
3.8.0	Cross-Site Scripting (XSS) уязвимости	1.0	Защита от Cross-Site Scripting	Ввести код в поле	Код не должен сработать	Код срабатывает	Высокая	Высокий	Гришкина П. В.	Открыт
3.9.0	HTML-инъекции	1.0	Защита от HTML-инъекций	Ввести код в поле	Код не должен сработать	Код срабатывает	Высокая	Высокий	Гришкина П. В.	Открыт
3.10.0	Cookie должны храниться в зашифрованном виде	1.0	Защита Cookie	Ввести код в поле	Код не должен сработать	Код срабатывает	Высокая	Высокий	Гришкина П. В.	Открыт
5.4.0	Подсказки	1.0	Подсказки существуют для всех полей	Открыть страницу с полями для ввода, нажать на них	Появляется подсказка по вводу	Подсказки есть не везде	Средняя	Средний	Гришкина П. В.	Открыт
5.11.0	Наличие плейсхолдеров в полях	1.0	Наличие плейсхолдеров в полях	Открыть страницу с полями для ввода и увидеть плейсхолдеры в полях	Отображение плейсхолдеров в полях	Не во всех полях есть плейсхолдеры	Средняя	Средний	Гришкина П. В.	Открыт

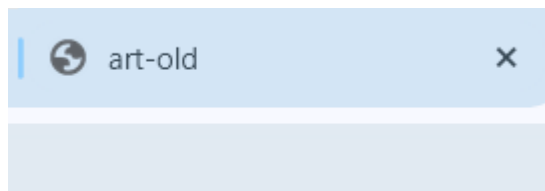


Рисунок 44 - Отсутствие favicon

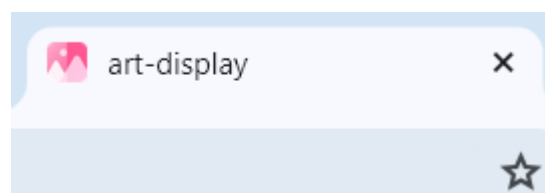


Рисунок 45 - Появилась favicon

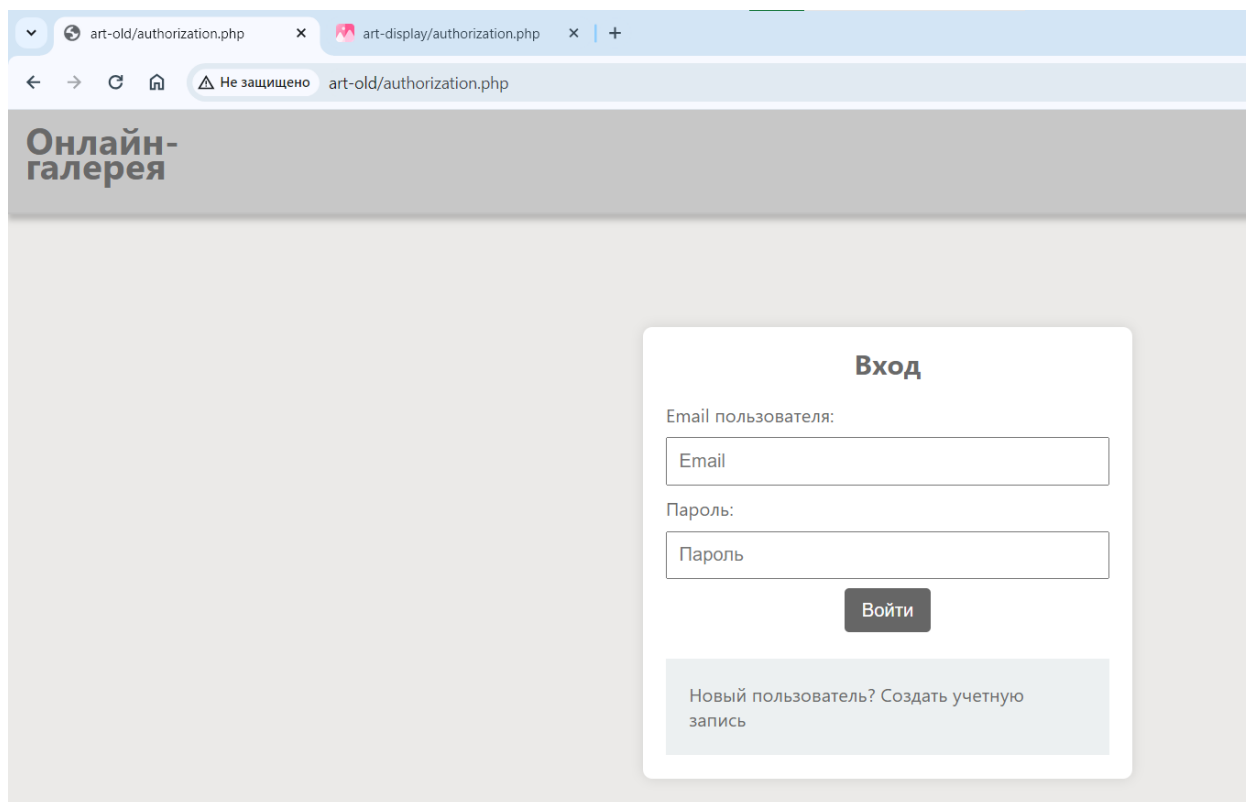


Рисунок 46 - Отсутствие астерисков и https

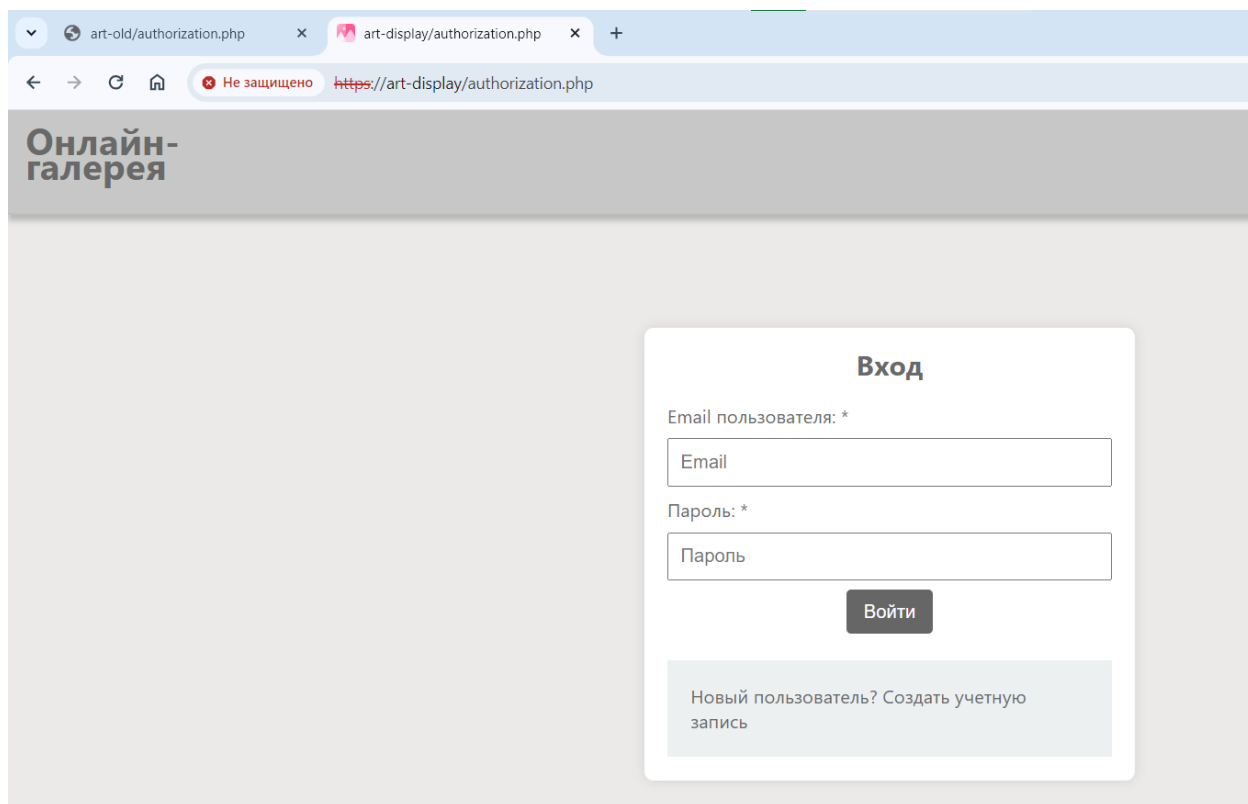


Рисунок 47 - Исправлена ошибка

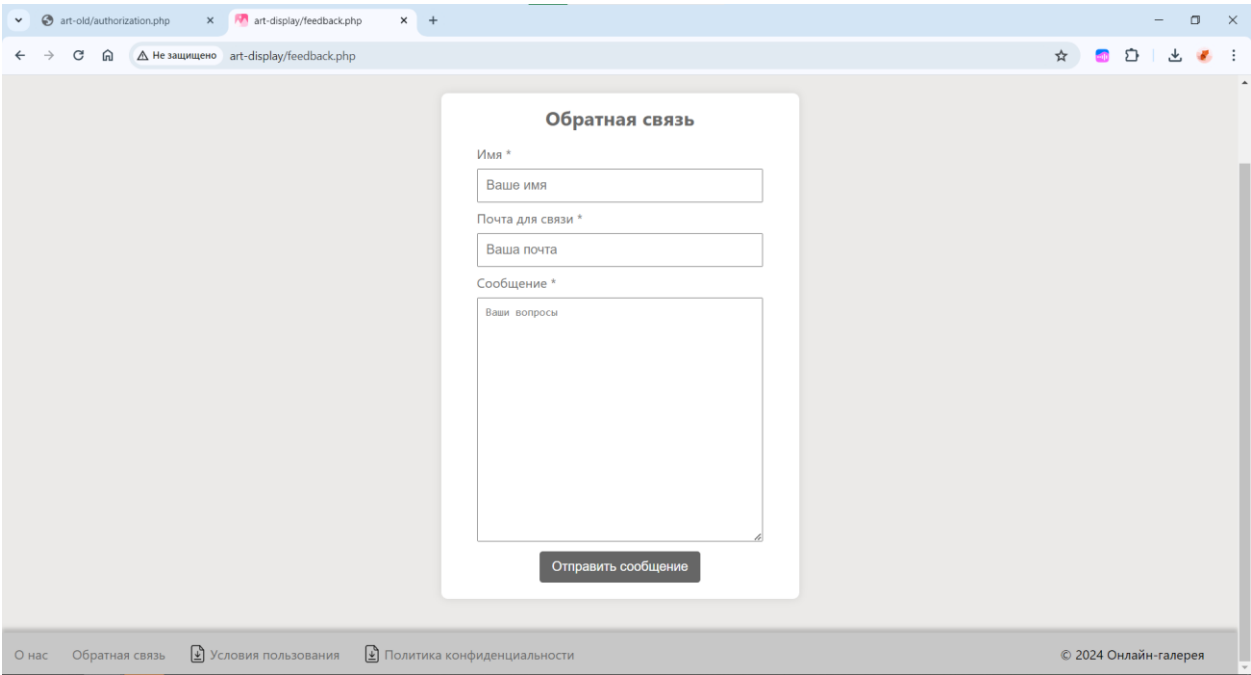


Рисунок 48 - Добавлена форма обратной связи

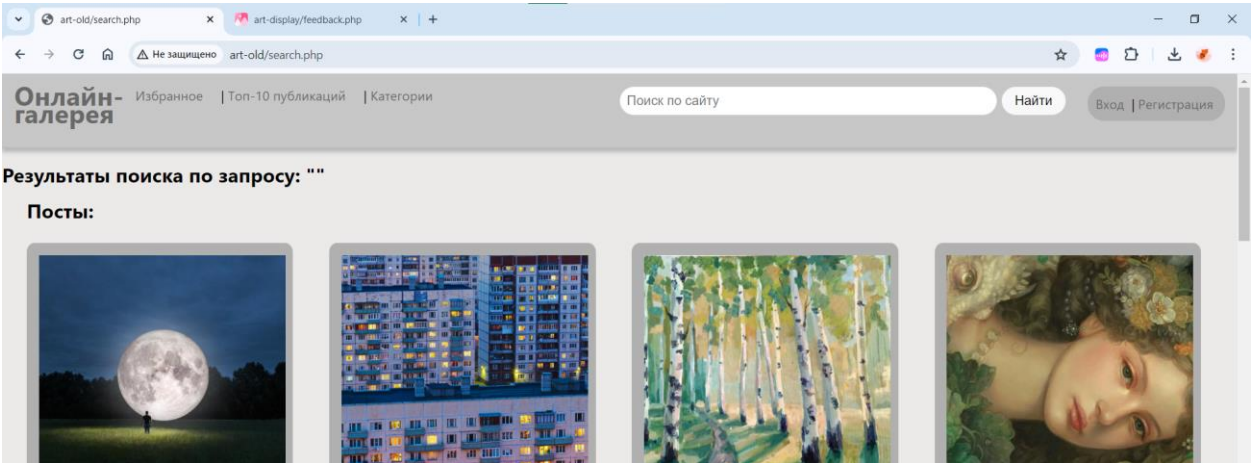


Рисунок 49 - Поиск пустой строки

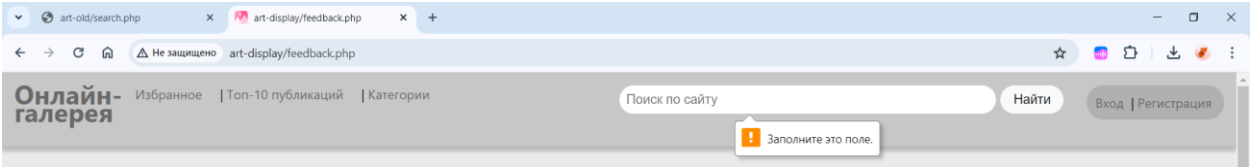


Рисунок 50 - Исправлена ошибка

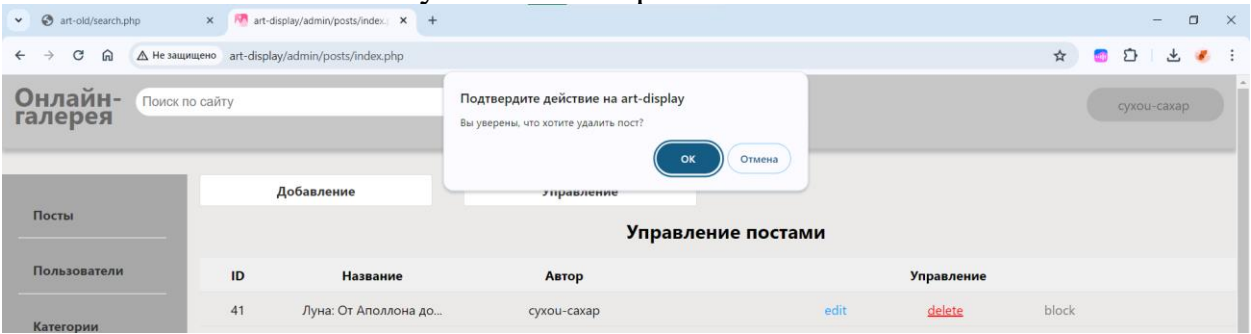
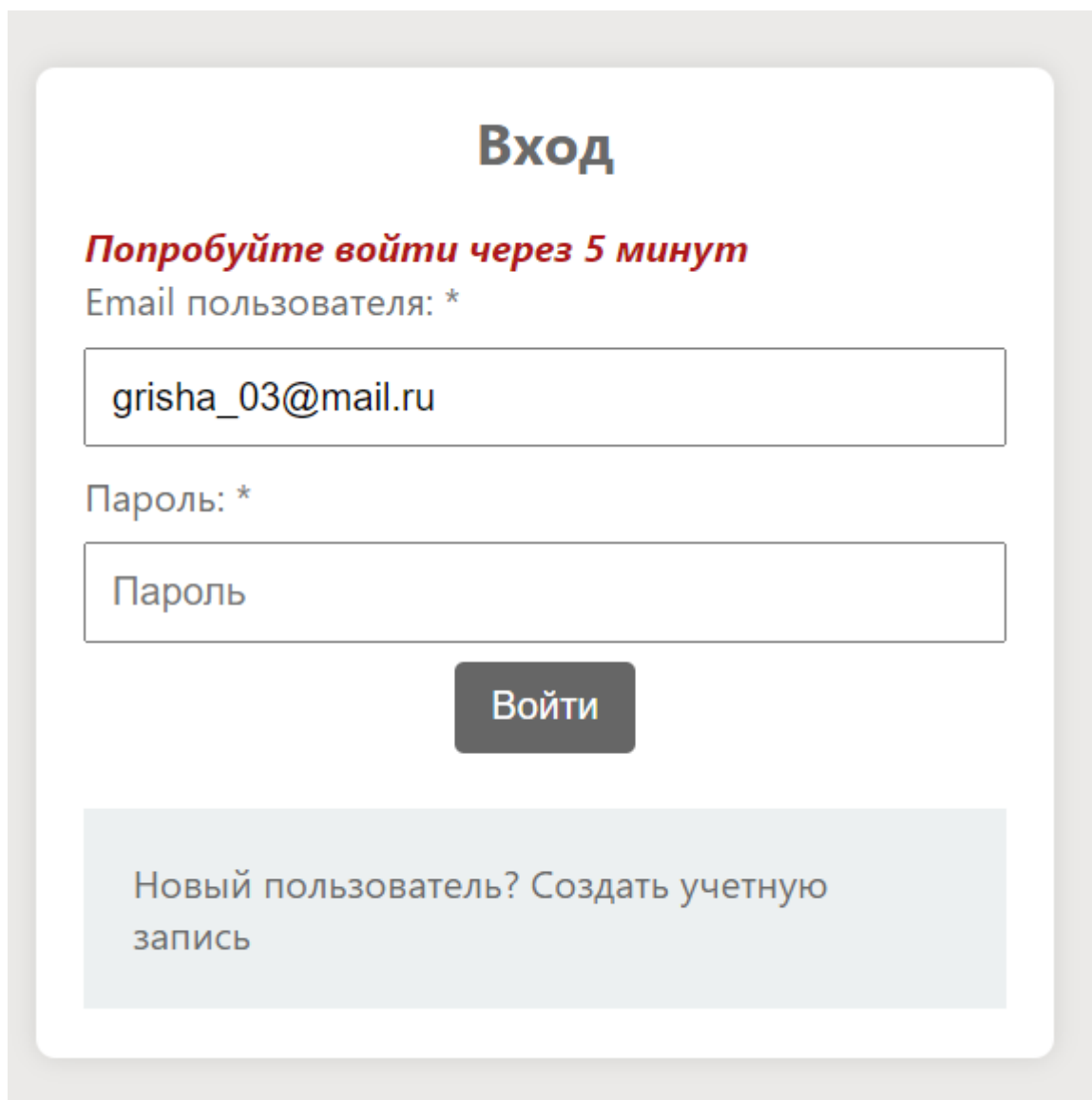


Рисунок 51 – Добавлены уведомления перед удалением/блокировкой постов/пользователей/категорий



The image shows a login form titled "Вход" (Login). At the top, there is a red message: "Попробуйте войти через 5 минут" (Try logging in again in 5 minutes). Below this, there are two input fields: "Email пользователя: *" (User email: *) and "Пароль: *" (Password: *). The email field contains the text "grisha_03@mail.ru". The password field is empty and has the placeholder text "Пароль". Below the password field is a dark button labeled "Войти" (Login). At the bottom of the form, there is a light blue box containing the text "Новый пользователь? Создать учетную запись" (New user? Create an account).

Вход

Попробуйте войти через 5 минут

Email пользователя: *

grisha_03@mail.ru

Пароль: *

Пароль

Войти

Новый пользователь? Создать учетную запись

Рисунок 52 – Добавлен лимит авторизации

Комментарий:

<i>этот шрифт не должен быть курсивным</i>


Отправить

Комментарии к посту

сухой-сахар

2024-05-15
17:46:51

этот шрифт не должен быть курсивным






Рисунок 53 – Отсутствует Защита от HTML-инъекций

Комментарий:

Ваш комментарий


Отправить

Комментарии к посту

сухой-сахар

2024-05-15
17:46:51

<i>этот шрифт не должен быть курсивным</i>






Рисунок 54 – Исправлена ошибка