

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики

Разработка приложений баз данных

Лабораторная работа

Направление 01.03.02 Прикладная математика и информатика

Обучающийся \_\_\_\_\_

А.А. Кособуцкая 8 гр., 4 курс

Преподаватель \_\_\_\_\_

Ю.С. Левицкая

Воронеж 2021

## Содержание

1	Постановка задачи.....	3
2	Схема базы данных .....	5
3	Разработка приложения.....	7
3.1	Базовый интерфейс сайта, онлайн пользователей .....	7
3.2	Таблица стилей .....	10
3.3	Регистрация и авторизация .....	11
3.4	Профиль пользователя, его редактирование, выход .....	15
3.5	Реализация блокировок пользователя .....	29
3.6	Список тем на форуме.....	30
4	Обзор программы .....	54
5	Заключение .....	79
6	Список использованных источников.....	80

## 1 Постановка задачи

Главной задачей работы является разработка веб-форума с тематикой «программирование». Форум должен предоставлять следующий функционал:

- Возможность регистрации и авторизации пользователей;
- Доступная только авторизованным пользователям возможность размещения новых тем на форуме и их комментирования;
- Возможность редактирования пользователем размещенного им сообщения, если с момента публикации прошло не больше более суток;
- Возможность удаления пользователем размещенного им сообщения;
- Возможность редактирования пользователем своего аккаунта: смена аватара, логина, пароля;
- Возможность выхода из своего аккаунта;
- Система подтверждений. Если пользователь хочет совершить действие, так или иначе направленное на уничтожение данных (например, удаление темы на форуме или выход из своего аккаунта), то он должен подтвердить свои действия;
- Подсчет количества тем и комментариев, созданных пользователем;
- Присутствие модерации. Модератор – специальный статус пользователя. Модератор может отредактировать и удалить любое сообщение на форуме в любое время, а также отключить возможность комментирования у заданной темы;
- Возможность забанить пользователя на сайте (доступна исключительно модераторам). Бан подразумевает, что пользователь не сможет авторизоваться на сайте указанный срок;
- Просмотр истории банов пользователя;
- Отслеживание последней активности пользователей и присвоение им статуса «онлайн» или «офлайн»;
- Возможность просматривать список пользователей сайта, сортировать его и выполнять поиск;

- Возможность отправки личного сообщения пользователю;
- Возможность просмотра полученных и отправленных сообщений;
- Присутствие статуса «Прочитано» у сообщений. Прочитанные и непрочитанные сообщения будут помечены разными значками;
- Присутствие элементарной системы оповещения о новых сообщениях;
- Создание специального раздела на сайте, публикация новых сообщений в котором будет доступна только модераторам;
- Создание главной страницы сайта. Последние три сообщения, опубликованные в описанном выше разделе на форуме, должны выводиться на главную страницу с возможностью перехода в соответствующую тему.

Разумеется, не менее важной задачей является создание удобного пользовательского интерфейса и системы навигации.

Для проектирования базы данных использована программа ErWin, субд: MySQL. Frontend-часть веб-приложения была реализована с помощью технологий HTML, CSS, JavaScript. Backend-часть программы была реализована на языке программирования PHP. Приложение было написано в редакторе Visual Studio Code.

## 2 Схема базы данных

База данных была спроектирована в программе ErWin. Она имеет следующую структуру:

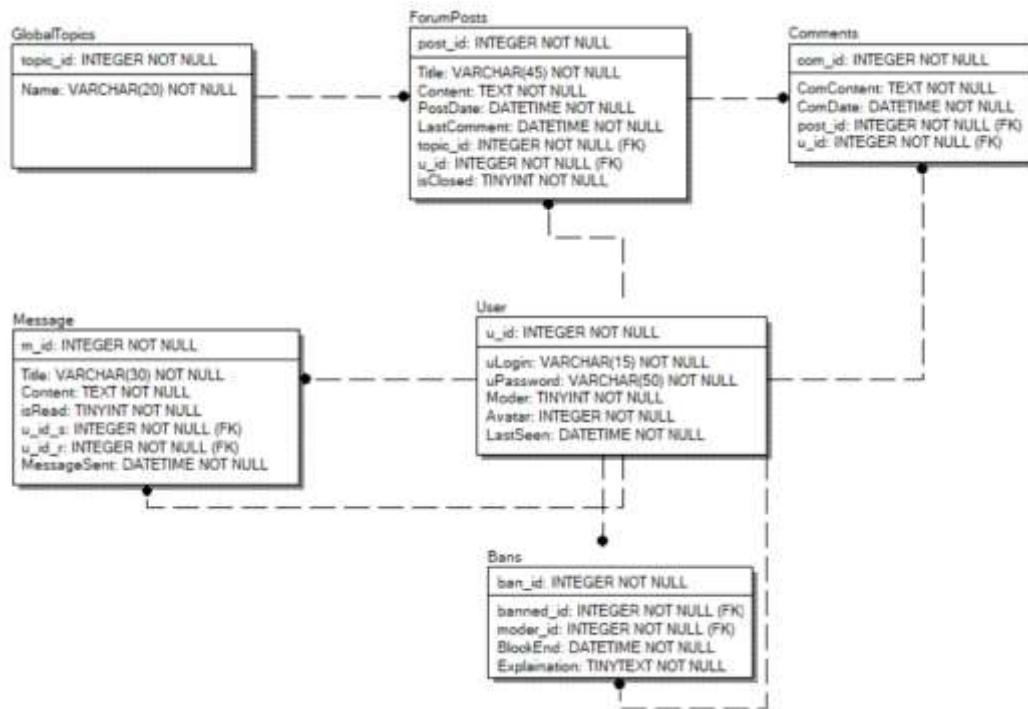


Рис. 2.1. Схема базы данных

Здесь:

- GlobalTopics – таблица, содержащая в себе названия разделов на форуме и их id. Название определяет тематику раздела;
- ForumPosts – таблица, содержащая в себе информацию о постах на форуме. Пост характеризуется следующей информацией: название, содержание, дата публикации, дата последнего комментария (требуется для сортировки), id раздела, в котором размещен пост, id пользователя, разместившего пост, и bool-переменная, определяющая, закрыт ли пост для комментирования;
- Comments – таблица, содержащая в себе все комментарии, оставленные на форуме. Комментарий характеризуется следующей информацией: содержание, дата публикации комментария, id поста, к которому был

добавлен комментарий, и id пользователя, который добавил комментарий;

- User – хранит информацию о пользователях. Содержит логин и пароль пользователя, его статус («пользователь» или «модератор»), номер выбранного аватара, дату и время, когда пользователь в последний раз проявил активность на сайте;
- Message – содержит в себе личные сообщения пользователей. Сообщения характеризуются следующей информацией: название, содержание, статус сообщения («прочитано» или «не прочитано» адресатом), id отправителя сообщения, id адресата, дата отправки сообщения;
- Bans – хранит информацию о банах. Бан характеризуется следующей информацией: id забаненного пользователя, id модератора, который забанил пользователя, дата окончания блокировки, и причина блокировки.

По данной схеме с помощью опции «Schema Generation» был сгенерирован код для создания соответствующей базы данных. База данных была создана для субд MySQL.

### 3 Разработка приложения

#### 3.1 Базовый интерфейс сайта, онлайн пользователей

В целях прощения работы с сайтом, а также обеспечения согласованности дизайна и сведения к минимуму ошибок при разработке сайта было принято решения перенести общие для всех веб-страниц элементы в отдельные файлы, которые в дальнейшем можно подключать к другим файлам. Тем самым написанный код будет общим для всех.

«Общими для всех страниц» считаются навигационные элементы сайта, а также элементы общего дизайна: баннер, навигационная панель и боковое меню. Кроме того, на каждой странице сайта должны выполняться определенные предписанные ранее скрипты, которые также вынесены в отдельный файл.

В файле «banner.php» реализован баннер сайта. Он представляет собой небольшую однотонную панель сверху, на которой представлена следующая информация: текущее время на сайте, а также число зарегистрированных пользователей. При нажатии на число зарегистрированных пользователей пользователь будет перенаправлен на страницу-поиск пользователей, где также представлен их полный список.

```

</script>
<div class="logo">
  <a href="/"><img src = "/img/logo_img.png" alt = "hobby world"></a>
</div>
<div class="infoarea" id="infoarea">
  <div class="infotext">
    <script language="javascript">
      showTime();
    </script>

    <?php
      $comand_u = 'SELECT COUNT(*) FROM `user`';
      $result = $forum->prepare($comand_u);
      $result->execute();
      $cntusers_res = $result->fetch(PDO::FETCH_ASSOC);

      echo '
      <div class="users-banner">
        <a href="/community/allusers.php" style="color: #00cc99;">Зарегистрировано '.current($cntusers_res).' пользователей</a>
      </div>';
    </?php>
  </div>

```

Рис. 3.1.1. Начало файла «banner.php»

То, как будет выглядеть остальная часть баннера, зависит от того, авторизован ли пользователь, или нет. Если да, то будут выведены три иконки.

```
<?php if(isset($_COOKIE["login"])){
    echo '<div class="user-controls">
    <a title="Мой профиль" href="/userpage.php?id='.$_COOKIE['id'].'"></a>
    <a title="Сообщения" style="margin-left: 32px;" href="/community/messages.php"></a>';
    $sql_getmsgs = 'SELECT COUNT(*) FROM `message` WHERE u_id_r = :cur_res AND isRead = 0';
    $query_getmsgs = $forum->prepare($sql_getmsgs);
    $query_getmsgs->execute(['cur_res' => $_COOKIE['id']]);
    $cntmsgs = $query_getmsgs->fetch(PDO::FETCH_ASSOC);

    echo '<a href="/community/messages.php"><div style="margin-left: 6px;";
    if(current($cntmsgs) > 0)
    echo ' font-weight: bold; color: #b90e0a;';
    else echo ' color: #e8cc9f;';
    echo '>'.current($cntmsgs).'

```

Рис. 3.1.2. Вывод иконок для авторизованного пользователя

При нажатии на первую будет открыт профиль пользователя.

Нажатие на вторую перенаправит пользователя в центр сообщений. Также в этой части кода производится запрос к базе данных (подробнее о запросах в п. 3.3) к таблице «сообщения» для получения числа непрочитанных сообщений, адресованных текущему пользователю. Считается их количество, а после полученное значение выводится. Если их больше нуля, то будет применен красный цвет, сигнализирующий о наличии непрочитанных сообщений.

При нажатии на третью иконку будет осуществлен выход из сайта. Также следует обратить внимание на то, что в самом начале файла указан скрипт, который будет выполнен при нажатии. Он опросит пользователя, уверен ли он, что хочет выйти. Если пользователь ответит «да», то будет осуществлен выход, если пользователь ответит «нет», то он останется на сайте, выход не будет осуществлен.

Если пользователь не авторизован, то ему будет выведен текст, при нажатии на который он попадет на страницу для регистрации или авторизации.

```
<div class="userinfo" style=<?php if(isset($_COOKIE["login"])) {echo "'left: 600px;";} else {echo "'left: 800px;";}>
<?php
if(isset($_COOKIE["login"])) {
    echo '<a href="/login.php" style="color: #e8cc9f;">Вход/Регистрация</a>';
}
>
```

Рис. 3.1.3. Перенаправление пользователя на страницу для регистрации



В файле «globalscripts.php» хранятся скрипты, которые должны быть выполнены при загрузке каждой страницы. Первый скрипт нужен для определения текущей даты и времени, а также вывода его в баннер (код баннера был представлен выше). Вторая часть кода выполняется, если пользователь авторизовался на сайте. Будет произведено обновление даты его последней активности на сайте, в качестве нового значения будет установлена текущая дата и время. Тем самым на сайте реализована функция онлайн.

```
<script language="JavaScript">
    function showTime(){
        var Now = new Date();
        var str="";
        var actualMounth = Now.getMonth() + 1;
        str += Now.getDate()+"."+ actualMounth + "." + Now.getFullYear() + " ";
        str += Now.getHours() + ":" + Now.getMinutes();
        document.write(str);
    }
</script>

<?php
$timestamp = date("YmdHis");
?>

<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

if(isset($_COOKIE['id']))
{
    $updateonlinesql = 'UPDATE `user` SET LastSeen = :curtime WHERE u_id = :userid';

    date_default_timezone_set('Europe/Moscow');
    $curdate_online = new DateTime();

    $updateonline = $forum->prepare($updateonlinesql);
    $updateonline->execute(['curtime' => date_format($curdate_online, 'Y-m-d H:i:s'), 'userid' => $_COOKIE['id']]);
}
?>
```

Рис. 3.1.4. Файл «globalscripts.php»

В файле «leftmenu.php» навигация по всем раздела форума. В тем содержится название темы, а также ссылка для перехода в неё. Leftmenu будет отображено на каждой странице под баннером слева как ещё одна панель навигации.

```

> leftmenu.php
<div class="leftmenu">
  <div class="content">
    <a class="header">Форумы</a><br/>
    <a class="header">Профессии</a>
    <a href="/forums/topics.php?id=1" class="itemlink">Веб</a>
    <a href="/forums/topics.php?id=2" class="itemlink">Базы данных</a>
    <a href="/forums/topics.php?id=3" class="itemlink">Микроконтроллеры</a>
    <a class="header">Языки программирования</a>
    <a href="/forums/topics.php?id=4" class="itemlink">Java</a>
    <a href="/forums/topics.php?id=5" class="itemlink">C#</a>
    <a href="/forums/topics.php?id=6" class="itemlink">C++</a>
    <a href="/forums/topics.php?id=7" class="itemlink">PHP</a>
    <a class="header">Общение</a>
    <a href="/forums/topics.php?id=8" class="itemlink">Новости сайта</a>
    <a href="/forums/topics.php?id=9" class="itemlink">Флудилка</a>
    <br> <br>
    <a class="header">Информация</a>
    <a href="/" class="itemlink" style="font-size: 14px;">Правила пользования</a>
    <a href="/" class="itemlink" style="font-size: 14px;">Политика конфиденциальности</a>
  </div>
</div>

```

Рис. 3.1.5. Файл «leftmenu.php»

В файле «head.php» содержится ссылка на иконку сайта для её отображения в браузере. Также там присутствует подключение к таблице стилей сайта, чтобы её правила могли использоваться в основном HTML-файле.

```

<head>
  <link rel="shortcut icon" href="/img/icon.png" type="image/x-icon">
  <link rel="stylesheet" href="/site.css?v=<?php echo $timestamp;?>">
</head>

```

Рис. 3.1.6. Файл «head.php»

В каждом файле сайта происходит последовательное подключение описанных выше файлов с помощью функции `require`. Данная функция вставляет в программу описанный в каждом из файлов код.

## 3.2 Таблица стилей

Набор правил, применяемый при работе с каждой страницей, описан в файле «site.css». Правило определяет то, как тот или иной элемент будет выглядеть на сайте. В данном файле содержатся все правила, которые

использует веб-приложение (кроме встраиваемых, которые описываются непосредственно в HTML-тэге). Итоговый файл получился достаточно объемным и содержит в себе 888 строк кода.

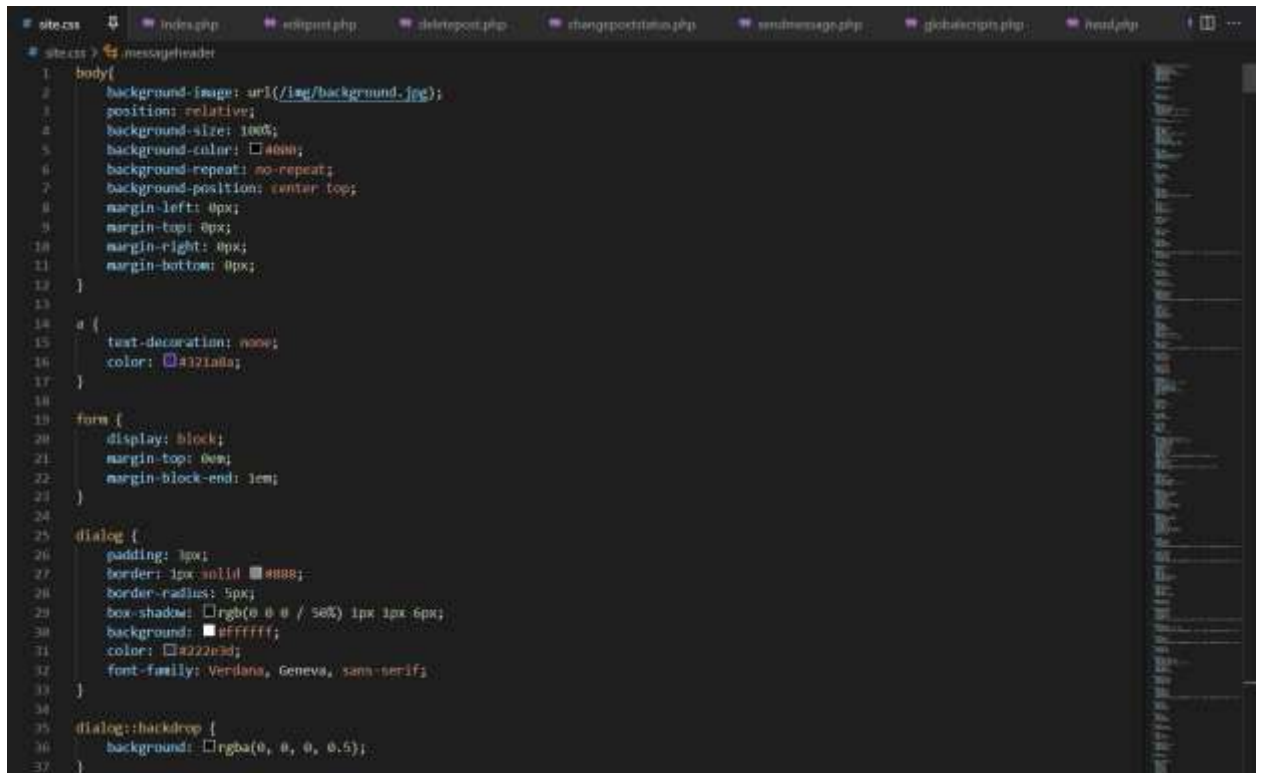


Рис. 3.1.9. Начало файла «site.css»

### 3.3 Регистрация и авторизация

Frontend-часть данного процесса была реализована в файле «login.php»:

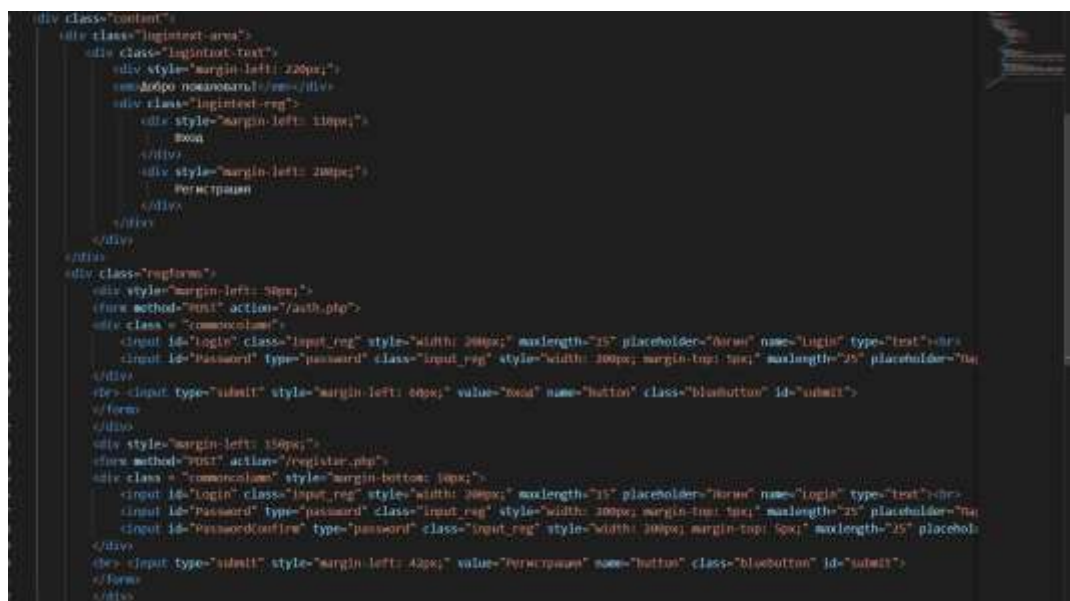


Рис. 3.3.1. Файл «login.php»

С помощью технологий HTML в браузер выводится две формы: форма для входа в уже существующий аккаунт и форма для регистрации нового пользователя. Эти формы привязаны к разным Backend-действиям.

Форма для входа в аккаунт спрашивает логин пользователя и его пароль. Форма для регистрации по своему содержанию почти аналогична форме для входа, только в ней требуется повторить введённый пароль для того, чтобы свести к минимуму вероятность ошибок при его написании.

Поля для ввода данных реализованы с помощью HTML-тэга «input», которым был присвоен созданный ранее стиль. Максимальная длина логина – 15 символов. HTML-тэг «form» предоставляет возможность сбора информации её передачи в другую часть программы. Данные передаются либо в глобальный массив `$_POST`, либо в глобальный массив `$_GET`, тип передачи указывается в самом тэге формы. Внизу добавлены кнопки с типом «submit» – после нажатия на них произойдет отправка формы и программа перейдет к указанным в заголовке формы действиям.

Backend-логика входа на сайт реализована в файле «auth.php»:

```

auth.php
<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

$comand = 'SELECT uPassword, u_id, Moder, Avatar FROM `user` WHERE uLogin = :uLog';
$query = $forum->prepare($comand);
$query->bindValue(":uLog", $_POST['Login'], PDO::PARAM_STR);
$query->execute();
$row = $query->fetch(PDO::FETCH_ASSOC);

if(!is_countable($row) || count($row) == 0 || current($row) != $_POST['Password']) {
    echo 'Ошибка, введены неверные данные';
    exit();
}

$isblocked = 'SELECT MAX(BlockEnd) FROM `bans` WHERE banned_id = :curuser';
$querycheckblock = $forum->prepare($isblocked);
$querycheckblock->execute(['curuser' => next($row)]);
$dateend = $querycheckblock->fetch(PDO::FETCH_ASSOC);

date_default_timezone_set('Europe/Moscow');
$datecheck = new DateTime();
$datecheck = date_format($datecheck, 'Y-m-d H:i:s');

$userid = current($row);

if(current($dateend) == null || $datecheck >= current($dateend)) {
    setcookie("Login", $_POST['Login'], time() + 3600*24);
    setcookie("id", current($row), time() + 3600*24);
    setcookie("moder", next($row), time() + 3600*24);
    setcookie("avatar", next($row), time() + 3600*24);
}

$path = 'location: /userpage.php?id='.$userid;
header($path);
?>

```

Рис. 3.3.2. Файл «auth.php»

В данном файле происходит подключение к базе данных, после чего к ней отправляется запрос на получение информации о пользователе с указанной при авторизации логином.

Затем происходит проверка данных на корректность: если результат запроса пуст, то это означает, что при входе был указан неправильный логин. Если логин совпал, но не совпал пароль, то вход также не может быть произведен. В обоих этих случаях пользователю будет выведено сообщение об ошибке.

После этого делается запрос к таблице с банами для получения максимальной даты окончания блокировки (на случай, если блокировок несколько) для пользователя, который пытается войти на сайт. После чего производится проверка: если текущая дата меньше найденного результата, то

это означает, что в данный момент пользователь заблокирован, и ему нельзя войти на сайт. Программа перенаправит пользователя на его страницу, где будет указано, что пользователь заблокирован. Также на странице можно будет посмотреть причину блокировки.

Если максимальная дата не была найдена (что значит, что у пользователя нет блокировок), либо дата окончания блокировки меньше текущей даты, то пользователю разрешен вход на сайт.

Устанавливаются куки-файлы, содержащие информацию об id пользователя, его логине, его статусе («модератор» или «пользователь»), и его аватаре. После чего пользователь перенаправляется на свою страницу.

В целом логика запроса к базе данных в php реализована следующим образом: сначала создается строка, содержащая в себе sql-запрос. Если это запрос с параметром, то перед параметром ставится символ «:». После чего происходит подготовка к выполнению команды и её выполнение. При выполнении команды в функцию передается ассоциативный массив, содержащие в себе значения для ключей-параметров, представленных в запросе. После чего каждая строка полученного результата переводится либо в объект, либо в ассоциативный массив, что делает возможным работу с полученными данными стандартными PHP-средствами. В дальнейшем процесс подключения к базе данных, формирования запросов и получения результатов будет абсолютно аналогичным.

Функция `setcookie` устанавливает в браузер куки-файл на заданное время. Доступ к значениям куки в PHP можно получить с помощью стандартного массива `$_COOKIE`.

Backend-логика регистрации нового пользователя реализована в файле «register.php»:

```

1  register.php
2
3  5  if($$_POST['Password'] == '' || $_POST['login'] == '') {
4      6      echo 'Ошибка. Пожалуйста, введите логин и пароль.';
5      7      exit();
6      8  }
7      9
8      10 if($_POST['Password'] != $_POST['PasswordConfirm']) {
9      11     echo 'Пароли не совпадают.';
10     12     exit();
11     13 }
12     14
13     15 $command = 'SELECT COUNT(*) FROM user WHERE uLogin = :uLogin';
14     16 $query = $forumposts->prepare($command);
15     17 $query->bindValue(":uLogin", $_POST['login'], PDO::PARAM_STR);
16     18 $query->execute();
17     19 $row = $query->fetch(PDO::FETCH_ASSOC);
18     20
19     21 if(current($row) > 0) {
20     22     echo 'Данное имя пользователя уже занято. Выберите другой ник.';
21     23     exit();
22     24 }
23     25
24     26 date_default_timezone_set('Europe/Moscow');
25     27 $curdate = new DateTime();
26     28
27     29 $sqlc = "INSERT INTO user(uLogin, uPassword, Moder, Avatar, LastSeen) VALUES(:uLogin, :pass, :mod, :ava, :ls)";
28     30
29     31 $query1 = $forumposts->prepare($sqlc);
30     32 $query1->execute(['uLogin' => $_POST['login'], 'pass' => $_POST['Password'], 'mod' => 0, 'ava' => 15, 'ls' => date_format($curdate, 'Y-m-d H:i:s')]);
31     33
32     34 $command1 = 'SELECT u_id, Moder FROM user WHERE uLogin = :uLogin';
33     35 $query1 = $forumposts->prepare($command1);
34     36 $query1->bindValue(":uLogin", $_POST['login'], PDO::PARAM_STR);
35     37 $query1->execute();
36     38 $row1 = $query1->fetch(PDO::FETCH_OBJ);
37     39
38     40 setcookie("login", $_POST['login'], time() + 3600*24);
39     41 setcookie("id", $row1->u_id, time() + 3600*24);

```

Рис. 3.3.3. Файл «register.php»

В данном файле производится регистрация нового пользователя. Программа проверяет, ввел ли пользователь логин и пароль, и совпадает ли пароль с его подтверждением. Если найдена ошибка, то программа выведет соответствующее сообщение.

После этого происходит проверка того, занят ли данный логин другим пользователем, или нет. Если занят, то программа попросит пользователя выбрать другой логин.

В противном случае происходит добавление нового пользователя в таблицу «User», после чего устанавливаются соответствующие cookie-файлы и пользователь перенаправляется на свою страницу.

### 3.4 Профиль пользователя, его редактирование, выход

За представление профиля пользователя и за работу с ним отвечает файл «userpage.php». Рассмотрим его.



В самом начале файла из базы данных достаются основные данные о пользователе, подсчитывается количество оставленных им постов и комментариев и считается их сумма:

```
$comand1 = 'SELECT uLogin, moder, avatar FROM `user` WHERE u_id = :posterid';
$query1 = $forum->prepare($comand1);
$query1->bindValue(":posterid", $_GET["id"], PDO::PARAM_STR);
$query1->execute();
$row1 = $query1->fetch(PDO::FETCH_OBJ);

$comand2 = 'SELECT COUNT(*) FROM `forumposts` WHERE u_id = :posterid';
$query2 = $forum->prepare($comand2);
$query2->bindValue(":posterid", $_GET["id"], PDO::PARAM_STR);
$query2->execute();
$row2 = $query2->fetch(PDO::FETCH_ASSOC);

$comand3 = 'SELECT COUNT(*) FROM `comments` WHERE u_id = :posterid';
$query3 = $forum->prepare($comand3);
$query3->bindValue(":posterid", $_GET["id"], PDO::PARAM_STR);
$query3->execute();
$row3 = $query3->fetch(PDO::FETCH_ASSOC);

$countuserposts = current($row2) + current($row3);
?>
```

Рис. 3.4.1. Запросы к базе данных в файле «userpage.php»

Далее выводится основная информация о пользователе. Если пользователь с указанным в URL страницы id не существует, то программа выдаст соответствующую ошибку. В противном случае будет выведена информация о пользователе. На скриншоте ниже представлен вывод логина пользователя и количества оставленных им постов:

```
<div class="main">
  <div class="content">
    <?php
    if(!isset($row1->uLogin)){
      echo 'Ошибка. Данный пользователь либо не создан, либо удалён.';
      exit();
    }
    ?>
    <a class="header">
      Профиль пользователя <?php echo $row1->uLogin;?>
    </a>
    <div class="userpage-maininfo">
      <span class="smaller-header">
        <?php
        echo 'Пользователь оставил '.$countuserposts.' сообщений, из них
        <div style="margin-left: 332px;">'.current($row2).' постов <br>и '.current($row3).' комментариев</div>';
```

Рис. 3.4.2. Вывод основной информации о пользователе



Далее происходит проверка того, заблокирован ли в данный момент пользователь. Если да, то будет выведено соответствующее сообщение и информация об окончании блокировке. Логика запроса аналогична логике в рассмотренном ранее файле с авторизацией.

```
$isblocked = 'SELECT MAX(BlockEnd) FROM `bans` WHERE banned_id = :curuser';
$querycheckblock = $forum->prepare($isblocked);
$querycheckblock->execute(['curuser' => $_GET['id']]);
$dateend = $querycheckblock->FETCH(PDO::FETCH_ASSOC);

date_default_timezone_set('Europe/Moscow');
$datecheck = new DateTime();
$datecheck = date_format($datecheck, 'Y-m-d H:i:s');

if(count($dateend) > 0 && $datecheck < current($dateend))
{
    echo '<div style="font-size: 20px; color: #a50404;">Пользователь заблокирован';
    if(current($dateend) > date('Y-m-d H:i:s', strtotime($datecheck.' + 1 years')))
        echo ' навсегда.';
    else
        echo '</br> Дата окончания блокировки: '.current($dateend).'';
    echo '</div>';
}
?>
```

Рис. 3.4.3. Проверка того, заблокирован ли пользователь

Далее осуществляется вывод основной информации о пользователе в специальном блоке: аватар, логин, количество оставленных сообщений и количество «звёзд», которые считаются как число постов, оставленных пользователем, делённое на два плюс один.

```
<div class="userpage-block">
    <div class="userpage-avatar">
        <?php
        echo '';
        ?>
    </div>
    <div class="userpage-info">
        <?php
        echo '<div class="post-author-username" style="font-size: 16px;">'.$row1->uLogin.'</div>';
        if($row1->moder) {
            echo '<div class="post-moder" style="font-size: 14px;">Модератор</div>';
        }
        else {
            echo '<div class="post-user" style="font-size: 14px;">Пользователь</div>';
        }
        $starsnumber = floor(min($countuserposts/2, 6)+ 1);
        echo '';
        ?>
    </div>
    <div class="userpage-controls">
```

Рис. 3.4.4. Вывод детальной информации о пользователе

На этой же панели снизу представлены кнопки для осуществления возможных действий с пользователем. Множество возможных действий напрямую зависит от роли пользователя. Если пользователь авторизован и находится на своей страничке, то он может сменить аватар или логин/пароль. Если пользователь авторизован, но находится чужой страничке, то он сможет отправить пользователю личное сообщение. Модератор также будет иметь кнопку «Забанить пользователя». Модератор не может банить модераторов. Если пользователь авторизован, но находится не на своей страничке, то для него будет доступна функция отправки пользователю нового сообщения. Также абсолютно любой посетитель сайта (даже не авторизованный) имеет возможность посмотреть историю блокировок данного пользователя.

Все данные действия на сайте реализованы с помощью всплывающих окон – диалогов, которые позволяют работать с сайтом без ненужных обновлений страниц.

Реализация диалога для просмотра истории блокировок представлена ниже:

```
<dialog id="showAllBans">
  <div class="userdata-change-form" id="table">
    <div class="userdata-info">
      История блокировок пользователя
    </div>
    <div class="userdata-area">
      <?php
        $getcntbans = 'SELECT COUNT(*) FROM `bans` WHERE banned_id = :cur_user';
        $querycntbans = $forum->prepare($getcntbans);
        $querycntbans->execute(['cur_user' => $_GET['id']]);
        $cntbans = $querycntbans->fetch(PDO::FETCH_ASSOC);

        if(current($cntbans) == 0)
          echo '<div style="font-size: 16px; margin-bottom: 35px; margin-top: 18px;">У пользователя нет блокировок.</div>';
        else {
          echo '<table style="margin-left: auto; margin-right: auto; margin-bottom: 35px;">
            <thead style = "color: #1a075f;">
              <tr style="font-size: 14px;">
                <td width="100">Модератор</td>
                <td width="165">Время окончания</td>
                <td width="180">Причина</td>
              </tr>
            </thead>';
          $get_all_u_bans = 'SELECT * FROM `bans` WHERE banned_id = :curuser';
          $querygetallbans = $forum->prepare($get_all_u_bans);
          $querygetallbans->execute(['curuser' => $_GET['id']]);
```

Рис. 3.4.5. Реализация диалога для просмотра блокировок пользователя

На рисунке выше делается запрос к базе данных для получения информации о количестве блокировок пользователя. Если у пользователя нет

блокировок, то будет выведено соответствующее сообщение, и кроме него не будет представлено никакой информации. В противном случае будет построена таблица, содержащая id модератора, который забанил пользователя (с возможностью перехода на его страницу), дата окончания блокировки, и причина блокировки.

Для получения детальной информации о блокировках делается соответствующий запрос к базе данных, после чего информация выводится следующим образом:

```
while($rowbans = $querygetallbans->fetch(PDO::FETCH_OBJ)){
    echo '<tr style="font-size: 12px; color: #000;"><td>';
    $getmoderlogin = 'SELECT uLogin FROM `user` WHERE u_id = :moder';
    $querygetmoderlogin = $forum->prepare($getmoderlogin);
    $querygetmoderlogin->execute(['moder' => $rowbans->moder_id]);
    $moderlogin = $querygetmoderlogin->fetch(PDO::FETCH_OBJ);

    echo '<a href = "/userpage.php?id='.$rowbans->moder_id.'">'.$moderlogin->uLogin.'</a></td>';
    echo '<td>';
    if($rowbans->BlockEnd > date('Y-m-d H:i:s', strtotime($datecheck.'+ 1 years'))){
        echo '-';
    } else echo $rowbans->BlockEnd;
    echo '</td>';
    echo '<td>'.$rowbans->Explanation.'</td></tr>';
}
echo '</table>';
?>
<div class="userdata-controls">
    <button class="beigebutton" type="button" id="closebans">Закрыть</button>
</div>
</div>
</dialog>
```

Рис. 3.4.6. Вывод информации о блокировках

Данный код «идет» по строкам результата сделанного запроса и выводит в таблицу соответствующую информацию. Чтобы можно было перейти на страницу модератора, который вынес данный бан, делается дополнительный запрос в базу данных для получения его id. В конце файла выводится кнопка для закрытия диалогового окна.

Следующий диалог, реализующий логику для смены пользовательского аватара, можно открыть, только если пользователь находится на своей собственной странице. На сайте есть ограниченное количество аватаров, которые содержатся в рабочей папке. Каждый аватар имеет свой id. Чтобы пользователь понимал, какой аватар он выбрал, реализована следующая логика: после того, как пользователь нажимает на какой-либо аватар, данный

аватар обводится в рамку. При нажатии на другой аватар в рамку будет обведен он, а с предыдущего выбранного аватара рамка спадет. При открытии окна в рамку обводится текущий аватар пользователя. При регистрации на сайте все пользователи получают аватар по умолчанию, который имеет  $id = 15$ . Всего существует 16 аватаров. Аватар с индексом 0 был установлен администратору сайта напрямую через базу данных, не существует способа получить этот аватар в рамках сайта.

```
<dialog id="ChangeAvatar">
  <div class="userdata-change-form">
    <div class="userdata-info">
      <span class="avatar-info-text">Изменить аватар</span>
    </div>
    <form id="AvatarForm" method="POST" action="/changeinfo.php">
      <?php
        echo '<input id="Avatar" name="Avatar" type="hidden" value="'. $row1->avatar. "'>';
      <?>
    </form>
    <div class="userdata-area">
      <div class="explanation">
        <em>Аватары используются для представления вас другим участникам форума. Щелкните на картинку ниже, чтобы назначить её своей.</em>
      </div>
      <?php
        for($i = 1; $i < 16; $i++) {
          echo '';
        }
      <?>
    </div>
    <div class="userdata-controls">
      <button class="bluebutton" type="submit">Сохранить</button>
      <button class="beigebutton" type="button" id="cancel" onclick="destroyCurAvatar();">Отмена</button>
    </div>
  </div>
</dialog>
```

Рис. 3.4.7. Диалоговое окно для смены аватара

Снизу представлены кнопки для сохранения информации о текущем пользователе, а также о выходе из диалогового окна на случай, если пользователь передумает.

Следующее диалоговое окно предназначено для смены логина и пароля. При вводе нового пароля также требуется ввести его повторно для дополнительной проверки. Для того, чтобы данные были изменены, необходимо также ввести свой текущий пароль, это служит дополнительной защитой от взлома.

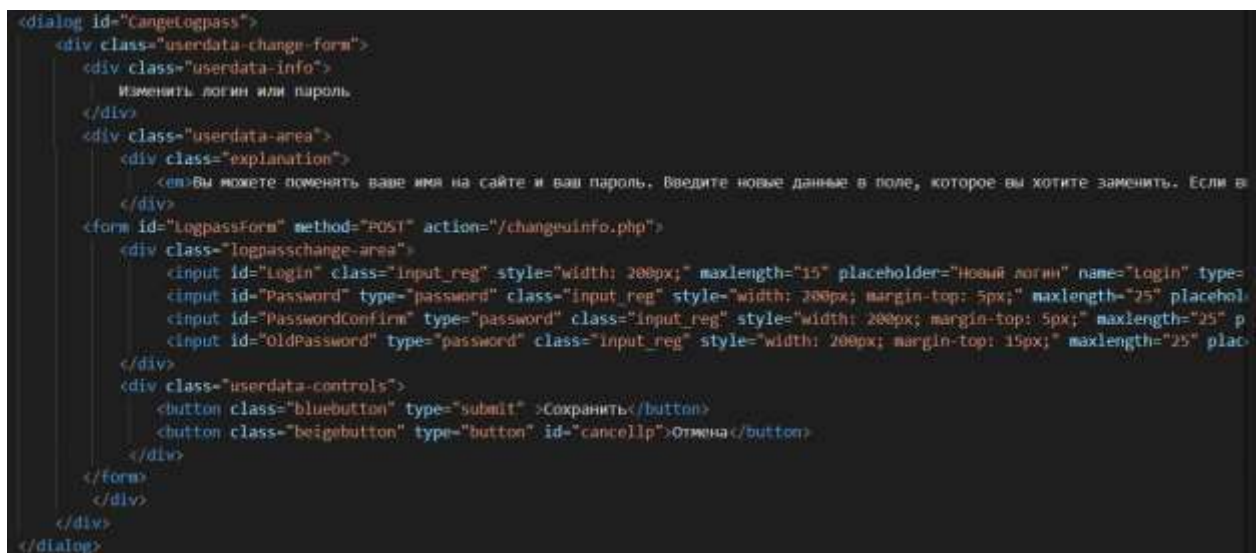


Рис. 3.4.8. Диалоговое окно для смены логина/пароля

Следующее диалоговое окно доступно только модераторам, оно дает возможность заблокировать пользователю доступ к сайту. Реализована достаточно обширная логика, согласно которой время блокировки пользователя определяется по количеству ранее полученных им блокировок. Если число блокировок равно нулю, то пользователь будет забанен на 10 минут, если пользователь уже получал одну блокировку, то он будет забанен на день, две блокировки – бан на неделю, три блокировки – бан на месяц. Если пользователь ранее был заблокирован четыре раза, то он будет заблокирован навсегда.

В диалоге также присутствует чекбокс, который позволяет сразу же забанить пользователя навсегда, минуя «предупредительные» блокировки. Также выводится текстовое окно для описания причины бана.



```

if(isset($ _COOKIE['moder']) && $ _COOKIE['moder'] == 1 && $row1->moder == 0)
{
    $newban = current($cntbans) + 1;
    echo '<div id="BanUserForm">
    <div class="userdata-change-form">
        <div class="userdata-info">
            Забанить пользователя
        </div>
        <div class="userdata-area">
            <div class="explanation">
                Вы можете заблокировать пользователю доступ к сайту. <br>Обратите внимание, это его '$newban.' блокировка. Длительность
                if(current($cntbans) == 0)
                    echo '10 минут';
                else if(current($cntbans) == 1)
                    echo '1 день';
                else if(current($cntbans) == 2)
                    echo '1 неделю';
                else if(current($cntbans) == 3)
                    echo '1 месяц';
                else
                    echo 'навсегда';
                echo ' .</b></br></br>Пожалуйста, убедитесь, что выдаёте бан заслуженно и доходчиво опишите причину.
            </div>
            <form id="BanForm" method="POST" action="/phpactions/banuser.php">
                <div class="logpasschange-area">
                    <input type="hidden" name="banned" value="'. $ _GET['id']. ' ">
                    <input type="hidden" name="moder" value="'. $ _COOKIE['id']. ' ">
                    <textarea class="input_reg" style="width: 350px; height: 90px; resize: vertical;" name="Reason" placeholder="Опишите причину">
                    </br>
                    <input type="checkbox" id="checkperma" name="isperma" value="perma">
                    <label for="checkperma">Заблокировать навсегда?</label>
                </div>
                <div class="userdata-controls">
                    <button class="bluebutton" type="submit">Заблокировать</button>
                    <button class="beigebutton" type="button" id="cancelban">Отмена</button>
                </div>
            </form>
        </div>
    </div>
}

```

Рис. 3.4.9. Диалоговое окно для вынесения блокировки пользователю  
Снизу также представлены кнопки для закрытия диалогового окна.

Далее в описанную ранее панель с детальной информацией выводятся кнопки для вызова диалоговых окон (следует обратить внимания, что без вызова диалоговые окна не отобразятся).

```

<?php
echo '<button class="userbutton" id="ShowBanHistory">История блокировок (<span>';
if(current($cntbans) > 0)
    echo '<span style="color: #a50404;">'.current($cntbans). '</span>';
else
    echo current($cntbans);
echo '</button>';
if(isset($ _COOKIE['id'])){
    if($ _COOKIE['id'] == $ _GET['id']){
        echo '<button class="userbutton" id="ShowAvatarMenu" onclick="setOriginalAvatar('.$row1->avatar.');">Изменить аватар</button>';
        echo '<button class="userbutton" id="ShowLogpassMenu">Изменить логин или пароль</button>';
    }
    else {
        echo
        '<form method="GET" action="/community/newmessage.php" style="margin-bottom: 0px;">
        <input type="hidden" name="to" value="'. $row1->ulogin. ' ">
        <button class="userbutton" type="submit">Сообщение</button>
        </form>';
        if($ _COOKIE['moder'] == 1 && $row1->moder == 0)
            echo '<button class="userbutton" id="BanUser" style="color: #a50404;">Забанить пользователя</button>';
    }
}
}

```

Рис. 3.4.10. Кнопки для вызова диалоговых окон

Для того, чтобы диалоговые окна открывались без перезагрузки страницы, необходимо задействовать язык программирования JavaScript, который позволяет создавать динамические веб-страницы.

Элементы языка JavaScript дают возможность работать с содержанием веб-страниц напрямую путем обращения к объекту document, содержащим в себе всю информацию о представлении веб-страницы.

Следует рассмотреть скрипт для открытия диалогового окна для смены аватара и работы с ним:

```
<script> //avatar
var showb = document.getElementById('ShowAvatarMenu');
var cancelb = document.getElementById('cancel');
var avatarDialog = document.getElementById('ChangeAvatar');
var curAvatar;

showb.addEventListener('click', function() {
    avatarDialog.showModal();});

cancelb.addEventListener('click', function() {
    avatarDialog.close();});

function setOriginalAvatar(number) {
    curAvatar = number;
    var avalid = 'avatar' + String(number);
    var avafame = document.getElementById(avalid);
    avafame.setAttribute("border", 6);
}

function destroyCurAvatar(number) {
    var oldava = 'avatar' + String(curAvatar)
    var oldframe = document.getElementById(oldava);
    oldframe.setAttribute("border", 1);
}

function changeAvatarId(number){
    var oldava = 'avatar' + String(curAvatar)
    var oldframe = document.getElementById(oldava);
    oldframe.setAttribute("border", 1);

    var ava = document.querySelector('#Avatar');
    ava.setAttribute("value", number);

    curAvatar = number;
    var newava = 'avatar' + String(number);
    var avafame = document.getElementById(newava);
    avafame.setAttribute("border", 6);
}
</script>
```

Рис. 3.4.11. Диалоговое окно для смены аватара

Объекты кнопок для открытия и закрытия диалоги инициализируются в JS по указанному в соответствующих тэгах HTML id, аналогичным образом инициализируется и сам диалог.

Далее к объектам кнопок привязываются события, которые срабатывают при нажатии. Данные события открывают и закрывают соответствующий диалог.

Переменная curAvatar нужна для отслеживания текущего аватара, выбранного пользователем. Это нужно для того, чтобы убирать с него рамку при смене выбора.

Функция setOriginalAvatar вызывает при первом обращении к диалогу. Она инициализирует переменную curAvatar и выполняет обводку изображения с соответствующим id.

Функция destroyCurAvatar вызывается при закрытии диалогового окна (на случай, если пользователь передумает и решит не менять аватар). Она уберет обводку с текущего выбранного пользователем аватара, тем самым вернет диалог к исходному состоянию, не нарушая его логику.

Функция changeAvatarId вызывается каждый раз, когда пользователь переключает аватары. Она убирает обводку с предыдущего выбранного пользователем аватара и ставит обводку на новый, при этом обновляя переменную CurAvatar.

Все остальные скрипты для других диалоговых окон имеют абсолютно идентичную реализацию. При этом они более примитивны, поскольку не содержат дополнительных функций для работы с формой:



```

<script> //logpass
var showb_lp = document.getElementById('ShowLogpassMenu');
var cancel_lp = document.getElementById('cancellp');
var Dialog_lp = document.getElementById('CangeLogpass');

showb_lp.addEventListener('click', function() {
    Dialog_lp.showModal();});

cancel_lp.addEventListener('click', function() {
    Dialog_lp.close();});
</script>

<script> //ban!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
var showb_ban = document.getElementById('BanUser');
var cancel_ban = document.getElementById('cancelban');
var Dialog_ban = document.getElementById('BanUserForm');

showb_ban.addEventListener('click', function() {
    Dialog_ban.showModal();});

cancel_ban.addEventListener('click', function() {
    Dialog_ban.close();});
</script>

<script> //bans
var show_hban = document.getElementById('ShowBanHistory');
var cancel_hban = document.getElementById('closebans');
var Dialog_hban = document.getElementById('ShowAllBans');

show_hban.addEventListener('click', function() {
    Dialog_hban.showModal();});

cancel_hban.addEventListener('click', function() {
    Dialog_hban.close();});
</script>

```

Рис. 3.4.12. Все JS-функции для работы с диалоговыми окнами

Если пользователь изменил информацию, то данные отправляются в файл-обработчик «changeuserinfo.php». Если пользователь устанавливал аватар, то программа просто его поменяет, форма отправки построена таким образом, что при смене аватара не может возникнуть ошибок. Как можно заметить по коду выше, пользователь не может изменить одновременно и аватар, и логин/пароль. Поэтому несмотря на то, что данные действия обрабатываются в одном файле, будут выполняться разные части кода, и никаких ошибок не возникнет, хотя логически процесс смены информации будет сильно упрощен.

В противном случае входные данные нужно проверять. Поэтому сначала данный обработчик проверяет входные данные на наличие всех возможных ошибок:

```
<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

if(isset($_POST['Avatar'])) {
    $sql = 'UPDATE `user` SET Avatar = :ava WHERE u_id = :userid';
    $query = $forum->prepare($sql);
    $query->execute(['userid' => $_COOKIE['id'], 'ava' => $_POST['Avatar']]);
    setcookie ("avatar", "", time() - 3600*24);
    setcookie ("avatar", $_POST['Avatar'], time() + 3600*24);
}

else if($_POST['OldPassword'] != '' || $_POST['Password'] != '' || $_POST['Login'] != '') {

    if($_POST['Login'] == '' && $_POST['Password'] == ''){
        echo 'Ошибка. Введите данные для смены логина или пароля.';
        exit();
    }
    if($_POST['OldPassword'] == '') {
        echo 'Ошибка. Введите текущий пароль для подтверждения смены.';
        exit();
    }

    $comandpass = 'SELECT uPassword FROM `user` WHERE u_id = :userid';
    $querycheckp = $forum->prepare($comandpass);
    $querycheckp->execute(['userid' => $_COOKIE['id']]);
    $rowpass = $querycheckp->fetch(PDO::FETCH_ASSOC);

    if(current($rowpass) != $_POST['OldPassword']) {
        echo 'Ошибка. Неправильно введён текущий пароль.';
        exit();
    }
}
```

Рис. 3.4.13. Проверка данных для изменения на наличие ошибок

Программа проверяет, заполнил ли пользователь все требуемые поля, совпадает ли новый пароль с подтверждением нового пароля, а также правильность введения текущего пароля. Если найдена ошибка, то выводится соответствующее сообщение, и процесс смены данных прерывается.

Следует отметить, что в диалоговом окне есть опция одновременно и для смены логина, и для смены пароля, но будут изменены лишь те данные, соответствующее поле с которым пользователь заполнит. Он может написать пароль, оставив пустым логин, и тогда будет изменен только логин, и наоборот. При этом пользователь может поменять одновременно и логин и пароль.

Если пользователь установил логин, то выполняется следующая часть кода:

```
if($_POST['Login'] != '') {
    $comandlogin = 'SELECT COUNT(*) FROM `user` WHERE uLogin = :uLog';
    $querycheck = $forum->prepare($comandlogin);
    $querycheck->execute(['uLog' => $_POST['Login']]);
    $rowlogin = $querycheck->fetch(PDO::FETCH_ASSOC);

    if(!is_countable($rowlogin) || current($rowlogin) > 0){
        echo 'Данный логин уже занят. Выберите другой логин.';
        exit();
    }

    $sql = 'UPDATE `user` SET uLogin = :userl WHERE u_id = :userid';
    $query = $forum->prepare($sql);
    $query->execute(['userid' => $_COOKIE['id'], 'userl' => $_POST['Login']]);

    setcookie ("Login", "", time() - 3600*24);
    setcookie ("Login", $_POST['Login'], time() + 3600*24);
}
```

Рис. 3.4.14. Изменение логина

Сначала выполняется запрос к базе данных для подсчета числа пользователей с таким же логином. Если данный логин уже занят, то пользователю будет отказано в смене логина. В противном случае происходит обновление базы данных, данному пользователю устанавливается новый логин, а также обновляется соответствующий куки-файл.

Если пользователь установил пароль, то будет выполнен следующий фрагмент кода:

```
if($_POST['Password'] != '') {
    if($_POST['Password'] != $_POST['PasswordConfirm']) {
        echo 'Ошибка. Пароли не совпадают.';
        exit();
    }

    $sql = 'UPDATE `user` SET uPassword = :userp WHERE u_id = :userid';
    $query = $forum->prepare($sql);
    $query->execute(['userid' => $_COOKIE['id'], 'userp' => $_POST['Password']]);
}
```

Рис. 3.4.15. Смена пароля пользователя

Если новый пароль совпадает с подтверждением пароля, то он будет изменен. В противном случае будет выдана ошибка и в изменении будет отказано.

Else в конце обрабатывает ситуацию, когда пользователь не ввел абсолютно никакой информации. Если смена произошла успешно, то пользователь будет переадресован на свою страницу:

Рис.  
3.4.16.  
Конец  
файла-

```
else {
    echo 'Ошибка. Вы не можете отправить пустую форму.';
    exit();
}

$path = 'Location: /userpage.php?id='.$_COOKIE['id'];
header($path);
?>
```

обработчика смены данных

При нажатии на иконку «выход» (описана в пункте 3.1) будет вызвана функция для подтверждения действий пользователя. Данная функция много где используется на сайте, поэтому реализована в файле «banner.php»:

```
<script>
function confirmClick(){
    var sure = confirm('Вы уверены?');
    if(sure) {
        window.location="/logout.php";
    }
}
</script>
```

Рис. 3.4.17. Скрипт для подтверждения действий пользователя

Если пользователь подтвердил выход, то запрос будет передан в файл «logout.php». В противном случае ничего не произойдет.

Файл «logout.php» просто удаляет установленные куки-файлы, тем самым осуществляется вход из аккаунта пользователя:

```

<?php
setcookie ("Login", "", time() - 3600*24);
setcookie ("id", "", time() - 3600*24);
setcookie ("moder", "", time() - 3600*24);
setcookie ("avatar", "", time() - 3600*24);

header('Location: /');
?>

```

Рис. 3.4.18. Файл «logout.php»

### 3.5 Реализация блокировок пользователя

То, каким образом выносятся блокировки, и как происходит отправка в backend запроса для её установления, было рассмотрено выше. Теперь следует рассмотреть сам файл-обработчик, который данный запрос обрабатывает.

Данная логика реализована в файле «banuser.php».

Он имеет следующую структуру:

```

<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

if($_POST['reason'] == ''){
    echo 'Пользователь, опишите причину блокировки.';
    exit();
}

$getcntbans = 'SELECT COUNT(*) FROM bans WHERE banned_id = :cur_user';
$querycntbans = $forum->prepare($getcntbans);
$querycntbans->execute(['cur_user' => $_POST['banned']]);
$cntbans = $querycntbans->fetch(PDO::FETCH_ASSOC);

date_default_timezone_set('Europe/Moscow');
$curdate = new DateTime();
$curdate_form = date_format($curdate, 'Y-m-d H:i:s');
$block_end;

if(isset($_POST['isperma']) && $_POST['isperma'] == 'perma'){
    $block_end = date('Y-m-d H:i:s', strtotime($curdate_form.'+ 1000 years'));
} else if(current($cntbans) == 0){
    $block_end = date('Y-m-d H:i:s', strtotime($curdate_form.'+ 10 minutes'));
} else if(current($cntbans) == 1){
    $block_end = date('Y-m-d H:i:s', strtotime($curdate_form.'+ 1 days'));
} else if(current($cntbans) == 2){
    $block_end = date('Y-m-d H:i:s', strtotime($curdate_form.'+ 1 weeks'));
} else if(current($cntbans) == 3){
    $block_end = date('Y-m-d H:i:s', strtotime($curdate_form.'+ 1 months'));
} else {
    $block_end = date('Y-m-d H:i:s', strtotime($curdate_form.'+ 1000 years'));
}

$sql = 'INSERT INTO bans(banned_id, moder_id, blockEnd, Explanation) VALUES(:banned, :moder, :BE, :reason)';
$query = $forum->prepare($sql);
$query->execute(['banned' => $_POST['banned'], 'moder' => $_POST['moder'], 'BE' => $block_end, 'reason' => $_POST['reason']]);

header('Location: /userpage.php?id='.$_POST['banned']);

```

Рис. 3.5.1. Файл-обработчик запроса на бан пользователя

В начале происходит проверка: если модератор забыл указать причину блокировки, то программа попросит его это сделать.

В противном случае программа считает количество блокировок пользователя, в зависимости от чего по описанной ранее логике определяется срок блокировки. Также если был установлен чекбокс «Заблокировать навсегда», то пользователь будет забанен навсегда несмотря на количество ранее полученных блокировок.

После этого в базу данных с банами добавляется новая запись, а сайт перебрасывает модератора на страницу заблокированного пользователя.

### 3.6 Список тем на форуме

Рассмотренная в пункте 3.1 боковая навигация позволяет переключаться между разделами форума. Для каждого раздела появляется табличка со списком размещенных тем, содержащая основную информацию о каждой из них.

Логика данного процесса реализована в файле «topics.php».

Сначала делается запрос к GlobalTopics на получение названия текущего раздела по его id:

```
<div class= content >
  <div style="position:relative; width:705px; height:69px; left:0; top: 11px;">
    <a class="header">
      <?php
        $sqltopics = 'SELECT * FROM `globaltopics` WHERE topic_id = :cur_t_id';
        $query_t = $forum->prepare($sqltopics);
        $query_t->execute(['cur_t_id' => $_GET['id']]);
        $row_t = $query_t->fetch(PDO::FETCH_OBJ);
        echo $row_t->Name;
      ?>
    </a>
  </div>
```

Рис. 3.6.1. Получение информации о названии раздела на форуме

Если пользователь авторизован, то у него будет возможность добавить новую тему на форум, и выводится соответствующая кнопка:

```

<?php
if(isset($_COOKIE['Login']) && ($_GET['id'] != 8 || $_COOKIE['moder'] == 1)) {
echo '<div id="forum-controls">
    <form method="GET" action="/forums/newtopic.php">
        <input type="hidden" name="id" value="'.$_GET['id'].'>
        <input type="submit" value="Новая тема" class="bluebutton">
    </form>
</div>';
}

```

Рис. 3.6.2. Вывод кнопки для добавления новой темы

Данный if также реализует описанное в постановки задачи условие. Предполагается, что существует отдельный раздел, содержащий в себе исключительно новости сайта, когда могут писать только модераторы. Если пользователь не является модератором, и в данный момент находится в данном разделе, то кнопка для добавления новой темы показана не будет.

После этого строится и выводится таблица с информацией о постах на форуме. Для получения списка тем, размещенных в данном разделе, производится соответствующий запрос к базе данных. Выводится информация о статусе темы: если она закрыта, то будет выведен значок с замком, если открыта, то ничего выведено не будет.

Затем выводится название темы. Если на него нажать, то будет выполнен переход к странице с самим постом. Снизу будет написан логин автора поста, при нажатии на который произойдет переход на его страницу. Для получения логина делается соответствующий запрос к таблице users.

Потом с помощью запроса к таблице comments подсчитывается и выводится количество оставленных к теме комментариев.

В конце выводится дата последнего комментария, оставленного к посту.



```

<div class="forumcontent">
  <div class="tableheader">
    <div style="margin-top: 3px;">
      <span style="margin-left:15px;">Тема</span>
      <span style="margin-left:354px">Ответы</span>
      <span style="margin-left:20px">Последний комментарий</span>
    </div>
  </div>
  <table id="postlist">
  </table>
  <?php
  $sqlposts = 'SELECT * FROM `forumposts` WHERE topic_id = :cur_t_id ORDER BY lastcomment DESC';
  $query = $forum->prepare($sqlposts);
  $query->execute(['cur_t_id' => $_GET['id']]);

  while($row = $query->fetch(PDO::FETCH_OBJ)) {
    $sqlcoms = $forum->query('SELECT COUNT(*) FROM `comments` WHERE post_id = '.$row->post_id);
    $countc = $sqlcoms->fetchColumn();
    echo '<tr><td class="status-post">';
    if($row->isclosed)
      echo '';
    echo '</td>';
    echo '<td class="topic"><a href="/forums/post.php?id='.$row->post_id.'">'.$row->title.'</a><br> <div style="fo

    $command1 = 'SELECT ulogin FROM `user` WHERE u_id = :posterid';
    $query1 = $forum->prepare($command1);
    $query1->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
    $query1->execute();
    $row1 = $query1->fetch(PDO::FETCH_OBJ);

    echo '<a href="/userpage.php?id='.$row->u_id.'">'.$row1->ulogin.'</a></div></td>
    <td class="activity"><img class
    </div></td>
    <td class="lastpost"><div class="common-text">'.$row->LastComment.'</div></td>
    </tr>';
  }
  </?php>

```

Рис. 3.6.3. Вывод всех тем в заданном разделе форума

### 3.7 Добавление новой темы

При нажатии на описанную в пункте выше кнопку «Новая тема» пользователь будет перенаправлен на страницу для создания новой темы.

Файл, в котором описана данная страница: «newtopic.php» Он имеет следующую структуру:



```

<?php require "D:/Stuff/XXXhtdocs/layout/head.php" />
<title>Новая тема</title>
<body>
  <div class="container">
    <?php require "D:/Stuff/XXXhtdocs/layout/banner.php" />
    <div class="flexarea">
      <?php require "D:/Stuff/XXXhtdocs/layout/leftmenu.php" />
    <div class="mainfullarea">
      <div class="main">
        <div class="content">
          <div style="position: relative; margin-top: 10px; margin-bottom: 10px; line-height: 20px; font-size: 20px;">
            <em>Создание новой темы</em>
          </div>
          <form method="POST" action="/phpactions/add.php">
            <div class="commoncolumn">
              <input id="Title" class="input_reg" maxlength="45" placeholder="Название темы..." name="Title" type="text">
            </div>
            <div class="editor">
              <?php
                echo '<div class="editor-author" style="width: 90px;">
                  <div class="editor-author-avatar">
                  <div class="post-author-username">'.$_COOKIE['Login'].'</div>
                </div>';
              ?>
            <div class="editor-text">
              <div class="editor-text-content">
                <textarea id="Content" class="input_reg" cols="50" rows="25" name="Content" style="margin-top: 0px; margin-bottom: 0px;">
              </div>
            </div>
            <?php
              echo '<input type="hidden" name="id" value="'.$_GET['id'].'" />';
            ?>
            <div class="editor-text-footer" id="post">
              <input type="submit" value="Отправить" name="button" class="beigebutton" id="submit">
              <input type="button" value="Отмена" name="button" class="bluebutton" id="cancel" onclick="window.location=
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>

```

Рис. 3.7.1. Файл «newtopic.php»

Также следует обратить внимание, что в начале файла осуществляется проверка на возможность доступа к данной странице. Если пользователь не авторизован, то он не сможет добавить новую тему, программа перенаправит его на главную страницу.

```

<?php
if(is_null($_GET['id']) or !(isset($_COOKIE["Login"]))) {
    header('Location: /');
}
?>

```

Рис. 3.7.2. Начало файла «newtopic.php»

Сам файл описывает два поля для введения текста: одно для заголовка, второе для содержание. Сбоку выводится аватар пользователя и его логин как средства для предварительного просмотра.

Внизу также представлены кнопки «отправить» и «отмена». При нажатии на кнопку «отправить» запрос на добавление новой темы будет передан в файл-обработчик, при нажатии на кнопку «отмена» пользователь будет перенаправлен в тот раздел, в котором он хотел разместить новый пост.

Обработка запроса на создание новой темы осуществляется в файле «add.php». Сначала программа проверяет, ввел ли пользователь текст нового поста и его содержание. Если нет, то программа выведет ошибку и новый пост добавлен не будет. В противном случае в таблицу forumposts добавляется вся информация о новой теме. Важно отметить, что за дату последнего комментария к теме при её создании считается само время создания данной темы. Это позволяет сохранить выбранную логику сортировки сообщений.

```
</php>

$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

if($_POST['title'] == '' || $_POST['content'] == '') {
    echo "Ошибка. Пожалуйста, введите название и содержание темы.";
    exit();
}

$sqlc = 'INSERT INTO forumposts(title, Content, PostDate, LastComment, topic_id, u_id) VALUES(:title, :content, :postd, :lcom, :topid, :p
date_default_timezone_set("Europe/Moscow");
$curdate = new DateTime();

$query = $forum->prepare($sqlc);
$query->execute(['title' => $_POST['title'], 'content' => nl2br($_POST['content']), 'postd' => date_format($curdate, 'Y-m-d H:i:s'), 'lcc
$path = 'location: /forums/topics.php?id='.$_POST['id'];
header($path);
?>
```

Рис. 3.7.3. Файл «add.php»

### 3.8 Тема на форуме

То, как будет отображаться конкретная тема на форуме и комментарии к ней, описано в файле «post.php». Стоит отметить, что логика отображения комментария и поста практически не отличается. Единственное отличие: для поста будет выведена дополнительная кнопка, при нажатии на которую пост можно будет закрыть для комментирования (но только в том случае, если просматривающий страницу пользователь – модератор). Также фон поста отличается от фона комментариев к нему. Поэтому целесообразно рассмотреть лишь отображения самого поста.

В самом начале осуществляется много запросов к базе данных. Получается информация о самом посте, потом о его авторе, потом о количестве сообщений на форуме, оставленных автором.

```

<div id="forumheader">
    <?php
    $comand_post = 'SELECT * FROM `forumposts` WHERE post_id = :postid';
    $query = $forum->prepare($comand_post);
    $query->execute(['postid' => $_GET['id']]);
    $row = $query->fetch(PDO::FETCH_OBJ);
    echo $row->Title;

    $comand1 = 'SELECT uLogin, Moder, Avatar FROM `user` WHERE u_id = :posterid';
    $query1 = $forum->prepare($comand1);
    $query1->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
    $query1->execute();
    $row1 = $query1->fetch(PDO::FETCH_OBJ);

    $comand2 = 'SELECT COUNT(*) FROM `forumposts` WHERE u_id = :posterid';
    $query2 = $forum->prepare($comand2);
    $query2->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
    $query2->execute();
    $row2 = $query2->fetch(PDO::FETCH_ASSOC);

    $comand3 = 'SELECT COUNT(*) FROM `comments` WHERE u_id = :posterid';
    $query3 = $forum->prepare($comand3);
    $query3->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
    $query3->execute();
    $row3 = $query3->fetch(PDO::FETCH_ASSOC);

    $countuserposts = current($row2) + current($row3);
    ?>
</div>

```

Рис. 3.8.1. Запросы к базе данных в файле «post.php»

Сначала выводится название темы, а затем само её содержание с применением специально созданного стиля. Слева темы будет выведен аватар пользователя, оставившего данный пост, и его логин. При нажатии на них будет осуществлен переход на страницу данного пользователя.

```

<div class="postframe" id="main">
    <div class="postauthor">
        <div class="post-author-avatar">
            <?php
            echo '';
            ?>
        </div>
        <?php
        echo '<div class="post-author-username"><a href="/userpage.php?id='.$row->u_id.'" style="color: white;">'.$row1->uLogin.'</div>';
        if($row1->Moder) {
            echo '<div class="post-moder">Модератор</div>';
        }
        echo '<div class="post-author-stat">Активность:<br/>'.$countuserposts.' сообщений</div>';
        ?>
    </div>

```

Рис. 3.8.2. Вывод информации об авторе поста

Сверху будут выведены элементы управления постом. Если просматривающий страницу пользователь – модератор, то он сможет отключить к нему комментарии, редактировать название или содержание поста, или удалить его. Если просматривающий страницу пользователь – автор

поста, то он сможет удалить пост (в любое время), а также изменить его название или содержание, если с момента публикации до текущего времени прошло не более суток. Если просматривающий страницу пользователь не подходит под описанные выше критерии, то у него не будет никаких прав на действия с данным постом, соответствующие кнопки не будут выведены.

```

if(isset($_COOKIE["login"]) && ($_COOKIE["moder"] == 1 || $_COOKIE["id"] == $row->u_id))
{
    echo '<div class="post-actions">';

    if($_COOKIE["moder"] == 1){
        echo '<form method="POST" action="/phpactions/changepoststatus.php" onsubmit="return confirmcomClick();">
        <input type="hidden" name="postid" value="'.$_GET['id'].'">
        <input type="hidden" name="gtopicid" value="'.$row->topic_id.'">
        <input type="hidden" name="post_status" value="'.$row->isClosed.'">
        <button type="submit" class="postico"></button>
        </form>';
    }

    $curdate = new DateTime();
    $curdate_form = date_format($curdate, 'Y-m-d H:i:s');
    $is_aviable_to_edit = date('Y-m-d H:i:s', strtotime($curdate_form.' - 1 day'));

    if((($_COOKIE["id"] == $row->u_id && $row->PostDate >= $is_aviable_to_edit) || $_COOKIE["moder"] == 1){
        echo '<form method="POST" action="/forums/editpost.php">
        <input type="hidden" name="is_post" value="1">
        <input type="hidden" name="id" value="'.$_GET['id'].'">
        <input type="hidden" name="postid" value="'.$_GET['id'].'">
        <input type="hidden" name="title" value="'.$row->Title.'">
        <input type="hidden" name="Content" value="'.str_replace("\n", "&#14;", $row->Content).'">
        <button type="submit" class="postico"></button>
        </form>';
    }

    echo '<form method="POST" action="/phpactions/deletepost.php" name="delpost" onsubmit="return confirmcomClick();">
    <input type="hidden" name="postid" value="'.$_GET['id'].'">
    <input type="hidden" name="gtopicid" value="'.$row->topic_id.'">
    <button type="submit" class="postico"></button>
    </form>
    </div>';
}

```

Рис. 3.8.3. Вывод кнопок для осуществления действий с темой

Далее выводится сам текст поста, больше в данном файле не присутствует никакой логики.

Для получения списка комментариев создается соответствующий запрос к базе данных:

```

<?php
$sql_comments = 'SELECT * FROM `comments` WHERE post_id = :postid';
$querycom = $forum->prepare($sql_comments);
$querycom->execute(['postid' => $row->post_id]);

while($comrow = $querycom->fetch(PDO::FETCH_OBJ)) {
    echo '

```

Рис. 3.8.4. Запрос на получение списка комментариев к теме

Далее программа просто идёт по полученному списку и выводит каждый комментарий в описанном выше формате с описанными выше отличиями.

### 3.9 Добавление комментария

Рассмотрим конец описанного в пункте 3.8 файла. Существует три возможных варианта отображения области для комментирования темы.

Если тема открыта для обсуждения, и пользователь авторизован на сайте, то ему будет доступна редактируемая текстовая область, а также кнопка «отправить», которая передает файлу-обработчику запрос на добавление нового комментария.

```
<?php
if(isset($_COOKIE["id"]) && !$row->isClosed) {
    echo '<div class="editor-text-comment">
        <em>Добавить новый комментарий</em>
    </div>
    <div class="editor-author">
        <div class="editor-author-avatar">'.$_COOKIE["login"].'</div>
    </div>
    <div class="editor-text" id="comment">
        <div class="editor-text-content">
            <textarea id="Content-com" class="input_reg-com" cols="50" rows="15" name="Content" style="margin-top: 0px; margin-bottom:
        </div>
        <input type="hidden" name="id" value="'.$_GET['id'].'>
        <div class="editor-text-footer">
            <input type="submit" value="Отправить" name="button" class="beigebutton" id="submit">
        </div>
    </div>';
}
```

Рис. 3.9.1. Область для добавления нового комментария

Если тема, закрыта для обсуждения, то будет выведено соответствующее сообщение.

```
}
else if($row->isClosed){
    echo '<div class="editor-text-comment">
        <em>Данный пост закрыт для обсуждения</em>
    </div>';
}
```

Рис. 3.9.2. Вывод информации о том, что данная тема закрыта

Если тема открыта для обсуждения, но пользователь не авторизован на сайте, то программа попросит его войти в свой аккаунт или же



зарегистрироваться. При нажатии на выделенный текст пользователь будет перенаправлен на страницу для регистрации и авторизации.

```
else{
    echo '<div class="editor-text-comment">
        <em><a href="/login.php">Войдите или зарегистрируйтесь</a>, чтобы оставлять комментарии</em>
    </div>';
}
```

Рис. 3.9.3. Вывод информации для неавторизованного пользователя  
Файлом-обработчиком, используемым при добавлении нового комментария, является файл «addcomment.php».

```
<?php

$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

if($_POST['Content'] == '') {
    echo "Ошибка. Пожалуйста, введите название и содержание темы.";
    exit();
}

date_default_timezone_set('Europe/Moscow');
$curdate = new DateTime();

$sqlc = 'INSERT INTO comments(comContent, comDate, post_id, u_id) VALUES(:content, :comd, :postid, :poster)';
$query = $forum->prepare($sqlc);
$query->execute(['content' => nl2br($_POST['Content']), 'comd' => date_format($curdate, 'Y-m-d H:i:s'), 'postid' => $_POST['id'], 'poster' => $_POST['u_id']]);

$sqlpc = 'UPDATE forumposts SET lastcomment = :comdate WHERE post_id =:postid';
$query1 = $forum->prepare($sqlpc);
$query1->execute(['comdate' => date_format($curdate, 'Y-m-d H:i:s'), 'postid' => $_POST['id']]);

$path = 'Location: /forums/post.php?id='.$_POST['id'];
header($path);
/>>
```

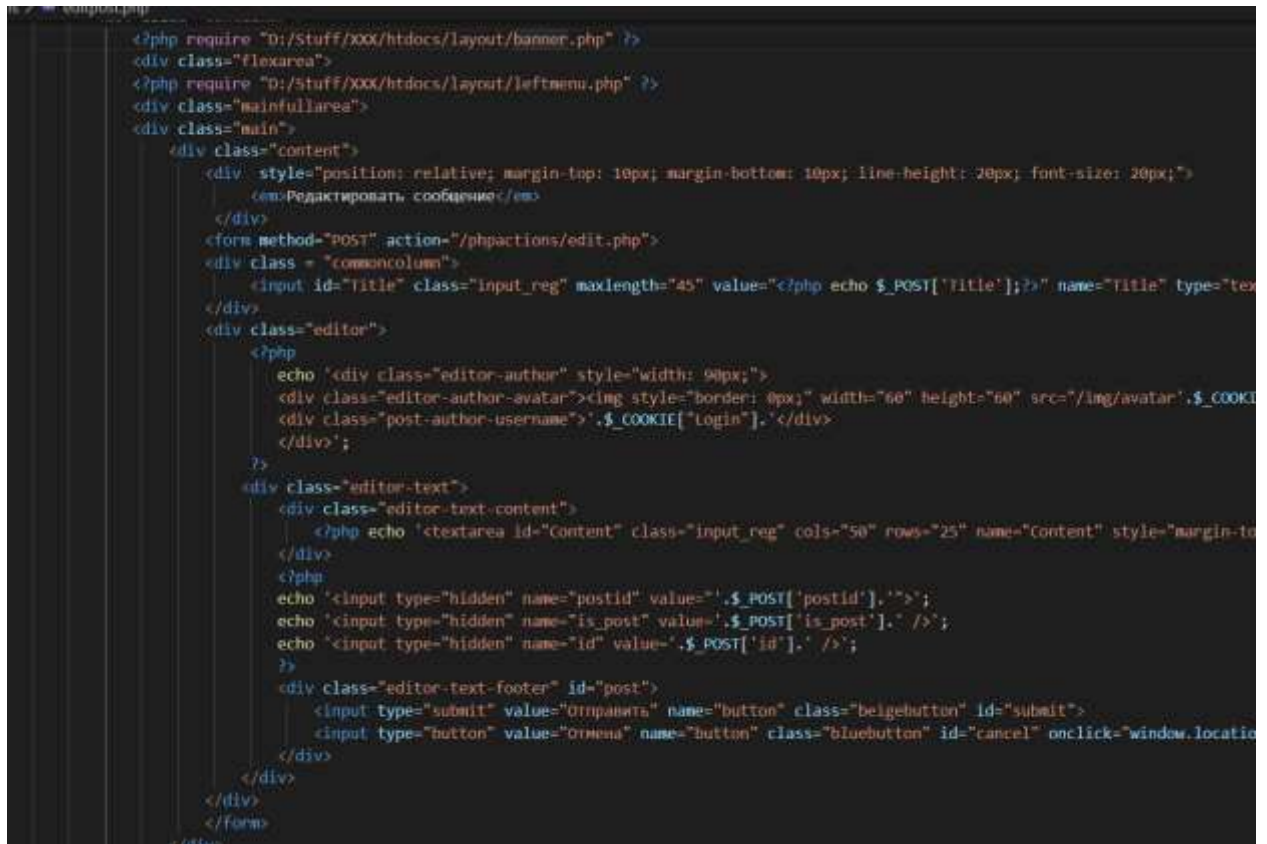
Рис. 3.9.4. Файл «addcomment.php»

Если пользователь ввел пустой комментарий, то он не будет добавлен, а пользователю будет выведено сообщение об ошибке. В противном случае в базу данных comments вносится запись с информацией о новом комментарии. Также происходит обновление информации о посте, к которому добавлен комментарий. Для него будет установлена новая дата последнего оставленного к нему комментария.

### 3.10 Редактирование комментариев и тем

За редактирование тем и комментариев отвечают одни и те же файлы. Frontend-часть процесса описана в файле «editpost.php», Backend-часть описана в файле «edit.php». По своей логике они абсолютно идентичны описанным в пункте 3.7 файлам, но имеют небольшие отличия.

В файле «editpost» пользователю будут выведены абсолютно аналогичные файлу «newtopic» поля для ввода текста, но теперь они будут предварительно заполнены.



```

<?php require "D:/Stuff/XXX/htdocs/layout/banner.php" ?>
<div class="flexarea">
<?php require "D:/Stuff/XXX/htdocs/layout/leftmenu.php" ?>
<div class="mainfullarea">
<div class="main">
  <div class="content">
    <div style="position: relative; margin-top: 10px; margin-bottom: 10px; line-height: 20px; font-size: 20px;">
      <em>Редактировать сообщение</em>
    </div>
    <form method="POST" action="/phpactions/edit.php">
      <div class="commoncolumn">
        <input id="title" class="input_reg" maxlength="45" value="<?php echo $_POST['title'];>" name="title" type="text">
      </div>
      <div class="editor">
        <?php
          echo '<div class="editor-author" style="width: 90px;">';
          <div class="editor-author-avatar">'. $_COOKIE['login']>';
          </div>';
        ?>
        <div class="editor-text">
          <div class="editor-text-content">
            <?php echo '<textarea id="Content" class="input_reg" cols="50" rows="25" name="Content" style="margin-top: 10px;">';
          </div>
          <?php
            echo '<input type="hidden" name="postid" value="',$_POST['postid'],'">';
            echo '<input type="hidden" name="is_post" value="',$_POST['is_post'],'" />';
            echo '<input type="hidden" name="id" value="',$_POST['id'],'" />';
          ?>
          <div class="editor-text-footer" id="post">
            <input type="submit" value="Отправить" name="button" class="beigebutton" id="submit">
            <input type="button" value="Отмена" name="button" class="bluebutton" id="cancel" onclick="window.location.href='<?php echo $_POST['id'];>'>'" />
          </div>
        </div>
      </div>
    </form>
  </div>
</div>

```

Рис. 3.10.1. Файл «editpost.php»

Если пользователь хочет отредактировать пост, то в маленькое текстовое поле будет выведено его название (при этом оно будет доступным для редактирования), в большое – содержание поста. При этом данные поля будут доступны для редактирования.

Если пользователь хочет отредактировать комментарий, то в малое поле будет выведено название поста, но оно не будет доступно для редактирования. В большое поле будет выведен текст комментария, и пользователь сможет его изменить.

Внизу также выведены две кнопки. При нажатии на кнопку «отправить» данные будут переданы в файл-обработчик, при нажатии на кнопку «отмена» пользователь будет перенаправлен в тему, которую (или комментарий к которой) он хотел отредактировать.

Файл «edit.php» проверяет введенные данные на корректность по описанным ранее алгоритмам, после чего либо выводит сообщение об ошибке, либо обновляет данные в базе данных.

```
<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

if((isset($_POST['Title']) && $_POST['Title'] == '') || $_POST['Content'] == '') {
    echo 'Ошибка. Пожалуйста, введите название и содержание темы.';
    exit();
}

$sql='';

if($_POST['is_post'])
    $sql = 'UPDATE `ForumPosts` SET Content = :new_content, Title = :new_title WHERE post_id = :id';
else
    $sql = 'UPDATE `Comments` SET ComContent = :new_content WHERE com_id = :id';

$query = $forum->prepare($sql);

if($_POST['is_post'])
    $query->execute(['id' => $_POST['id'], 'new_content' => nl2br($_POST['Content']), 'new_title' => $_POST['Title']]);
else
    $query->execute(['id' => $_POST['id'], 'new_content' => nl2br($_POST['Content'])]);

$path = 'location: /forums/post.php?id='.$_POST['postid'];
header($path);
}
```

Рис. 3.10.2. Файл «edit.php»

### 3.11 Удаление комментариев и тем

Следует отметить, что при нажатии на кнопки для удаления темы или комментария будет вызван скрипт для подтверждения действия пользователя. Только в том случае, если пользователь подтвердит свое действие, запрос будет передан в файл-обработчик.

Файл, который обрабатывает запрос на удаление постов – «deletepost.php». Сначала он получает список всех комментариев, оставленных к данному посту, и удаляет их из базы данных, после чего удаляет сам пост. Пользователь будет перенаправлен в раздел на форуме, в котором данный пост был размещен.



```

<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

$selectallcomments = 'SELECT com_id FROM `comments` WHERE post_id = :postid';
$querycomslist = $forum->prepare($selectallcomments);
$querycomslist->execute(['postid' => $_POST['postid']]);

while($row = $querycomslist->fetch(PDO::FETCH_OBJ)) {
    $deletecomments = 'DELETE FROM `comments` WHERE com_id = :commentid';
    $querydelcom = $forum->prepare($deletecomments);
    $querydelcom->execute(['commentid' => $row->com_id]);
}

$deletepost = 'DELETE FROM `forumposts` WHERE post_id = :postid';
$querydelpost = $forum->prepare($deletepost);
$querydelpost->execute(['postid' => $_POST['postid']]);

$path = 'Location: /forums/topics.php?id='.$_POST['gtopicid'];
header($path);
?>

```

Рис. 3.11.1. Файл «deletepost.php»

Обработчиком удаления комментария является файл «deletecomment.php». Он просто удаляет из таблицы комментариев соответствующую запись, после чего пользователь перенаправляется в тему, комментарий к которой он удалил.

```

<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

$comdel = 'DELETE FROM `comments` WHERE com_id = :comid';
$query = $forum->prepare($comdel);
$query->execute(['comid' => $_POST['comid']]);

$path = 'Location: '.$_POST['urladdress'];
header($path);
?>

```

Рис. 3.11.2. Файл «deletecomment.php»

### 3.12 Отключение/включение возможности комментирования темы

Следует напомнить, что данную возможность могут использовать только модераторы. При нажатии на кнопку для закрытия темы также будет вызван описанный ранее скрипт для подтверждения действия пользователя.

Также следует отметить, что если модератор нажмет на данную кнопку в посте, к которому комментарии уже закрыты, то комментарии будут открыты вновь.

Если модератор подтвердит свои действия, то запрос на их выполнение будет передан в файл «changepoststatus.php». Он просто обновляет запись о соответствующем посте в базе данных, устанавливая статус, противоположный текущему. То есть если комментарии были закрыты, то теперь они станут открыты, и наоборот.

```
<?php
$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

$sql = 'UPDATE `forumposts` SET isClosed = :newstatus WHERE post_id = :p_id';
$query = $forum->prepare($sql);
$query->execute(['newstatus' => !($_POST['post_status']), 'p_id' => $_POST['postid'] ]);

$path = 'Location: /forums/topics.php?id='.$_POST['gtopicid'];
header($path);
?>
```

Рис. 3.12.1. Файл «changepoststatus.php»

### 3.13 Система личных сообщений

Как было описано в пункте 3.1, в центр личных сообщений можно попасть, нажав на иконку сообщения (или на выведенное число новых сообщений) в верхней панели навигации.

Сам центр сообщений описан в файле «messages.php».

```

<div>ваши личные сообщения</div>
</div>
<div class="editor-text-footer" style="width: 275px;">
  <input type="button" value="Новое сообщение" name="button" class="bluebutton" id="cancel" onclick="window.location.href='/'">
</div>
</div>
<div style="margin-left: 50px;">
  <a href="/community/messages.php?order=received" <?php if(!isset($_GET['order']) || $_GET['order'] != 'sent') echo 'Полученные сообщения';>
</div>
<script language="JavaScript">
  var path = '/community/newmessage.php';
</script>
<div class="tableheader" id="message" style="margin-top: 10px;">
  <div style="margin-top: 3px;">
    <span style="margin-left: 44px;">Тема сообщения</span>
    <span style="margin-left: 276px;"><?php if(isset($_GET['order']) && $_GET['order'] == 'sent') {echo 'Получатель';}>
  </div>
</div>
<table id="msglist">
  <?php
  $getmessages = '';
  if(isset($_GET['order']) || $_GET['order'] != 'sent')
    $getmessages = 'SELECT * FROM `message` WHERE u_id_r = :curuser ORDER BY m_id DESC';
  else
    $getmessages = 'SELECT * FROM `message` WHERE u_id_s = :curuser ORDER BY m_id DESC';
  $query_getmessages = $forum->prepare($getmessages);
  $query_getmessages->execute(['curuser' => $_COOKIE['id']]);
  while($row = $query_getmessages->FETCH(PDO::FETCH_OBJ)){
    echo '<tr><td class="status">';
    if($row->isRead == 0)
      echo '';
    else
      echo '';
    echo '</td><td class="title"><a href="/community/message.php?id='.$row->m_id.'">'.$row->Title.'</a></td>';

    $getsenderlog = 'SELECT uLogin FROM `user` WHERE u_id = :sender';
    $querygetlog = $forum->prepare($getsenderlog);
    $neededid;
  }
  </table>

```

Рис. 3.13.1. Файл «messages.php» (фрагмент)

В самом начале файла выводится кнопка, ведущая на страницу для создания и отправки нового сообщения.

Данный файл предусматривает два «режима»: просмотр полученных сообщений и просмотр отправленных. В зависимости выбранного режима в базу данных будут сделаны два разных запроса:

Далее по полученным результатам строится таблица сообщений, по своей логике очень похожая на таблицу для просмотра списка тем на форуме. Сначала выводится иконка, обозначающая статус сообщения: восклицательный знак, если сообщение не было прочитано адресатом, галочка, если сообщение было прочитано. Стоит отметить, что для режима «полученные сообщения» адресатом является текущий пользователь.

Затем выводится название сообщения. При нажатии на название пользователь будет перенаправлен на страницу для просмотра данного сообщения.

В конце идет столбец с отправителем (для режима «полученные сообщения») или с получателем (для режима «отправленные сообщения») с возможностью перехода на соответствующую страницу пользователя при нажатии.

### **3.13 Создание и отправка нового сообщения**

На вкладку для создания нового сообщения можно попасть двумя способами: либо нажав на кнопку «Новое сообщение» в описанном выше центре сообщений, либо при нажатии кнопки «Сообщение» в профиле пользователя (описано в пункте 3.4). Разница между данными способами в том, что во втором случае поле получателя сообщения будет автоматически заполнено.

Страница для создания и отправки нового сообщения представлена в файле «newmessage.php». Он абсолютно аналогичен рассмотренному ранее файлу «newtopic.php» с разницей в том, что newmessage привязан к другому обработчику действий, а также, помимо названия сообщения и его содержания, пользователю придется указать и получателя сообщения. При нажатии на кнопку «Отправить» сообщение будет отправлено, при нажатии на кнопку «Отмена» пользователь будет переадресован на главную страницу сайта.

```

<div class="main">
<div class="content">
<form method="POST" action="/phpactions/sendmessage.php" style="margin-top: 20px;">
<div class="messageheader">
    Новое сообщение
</div>
<div id="msg-editor">
    <span class="editorlabel">Получатель: </span>
    <?php
    echo '<input class="input_reg" name="Receiver" style="width: 250px; font-size: 13px;"';
    if(isset($_GET['to'])){
        echo 'value="'.$_GET['to'].'"';
    }
    echo ' type="text">';
    <br/>
    <span class="editorlabel">Тема: </span>
    <input class="input_reg" maxlength="30" name="Topic" style="width: 250px; font-size: 13px;" type="text">
    <br/>
    <span class="editorlabel" style="vertical-align: top;">Сообщение: </span>
    <textarea class="input_reg" name="Msg_text" id="message" rows="10"></textarea>
    <?php
    echo '<input type="hidden" name="Sender" value="'.$_COOKIE['id'].'">';
    </div>
<div class="editor-text-footer">
    <input type="submit" value="Отправить" name="button" class="beigebutton" id="submit">
    <input type="button" value="Отмена" name="button" class="bluebutton" id="cancel" onclick="window.location=pa
</div>
<script language="JavaScript">
    var path = '/';
</script>
</form>
</div>
</div>

```

Рис. 3.13.1. Файл «newmessage.php»

Обработчик отправки сообщений реализован в файле «sendmessage.php». Он проверяет, существует ли пользователь, которому адресовано сообщение, и указал ли отправитель название темы и содержание. Если проверка не пройдена, то будет выведено соответствующее уведомление об ошибке, иначе в базу данных в таблицу message будет добавлена информация о новом сообщении. По умолчанию сообщению присваивается статус «Не прочитано».

```

<?php
if(!isset($_POST['Receiver']) || !isset($_POST['Topic']) || !isset($_POST['Msg_text'])){
    echo 'Ошибка. Введите имя получателя, тему сообщения и его содержание.';
    exit();
}

$dbinfo = 'mysql:host=localhost;dbname=forum';
$forum = new PDO($dbinfo, 'root');

$checkuser = 'SELECT u_id FROM `user` WHERE ulogin = :uolog';
$query_check = $forum->prepare($checkuser);
$query_check->execute(['uolog' => $_POST['Receiver']]);
$uinfo = $query_check->fetch(PDO::FETCH_ASSOC);

if(!is_countable($uinfo) || count($uinfo) == 0){
    echo 'Ошибка. Такого пользователя не существует.';
    exit();
}

date_default_timezone_set('Europe/Moscow');
$curdate = new DateTime();

$sql_msg = 'INSERT INTO `message` (Title, Content, IsRead, u_id_s, u_id_r, MessageSent) VALUES(:title, :content, :readfalse, :sender, :rec
$query_sendmsg = $forum->prepare($sql_msg);
$query_sendmsg->execute(['title' => $_POST['Topic'], 'content' => nl2br($_POST['Msg_text']), 'readfalse' => 0, 'sender' => $_POST['sender
header('Location: /community/messages.php');

?>

```

Рис. 3.13.2. Файл «sendmessage.php»

### 3.14 Просмотр сообщения, быстрый ответ

При нажатии на название сообщения в центре сообщений можно перейти к его содержанию. Пользователь будет переадресован на страницу «message.php», куда будет выведено соответствующее сообщение.

В самом начале файла происходит проверка: если пользователь не является ни адресатом, ни получателем сообщения, то он будет перенаправлен на главную страницу сайта.

```

<?php
if(!isset($_GET['id'])){
    header('Location: /');
}

$msginfo = 'SELECT * FROM `message` WHERE m_id = :mid';
$querymsg = $forum->prepare($msginfo);
$querymsg->execute(['mid' => $_GET['id']]);
$message = $querymsg->fetch(PDO::FETCH_OBJ);

if($message->u_id_s != $_COOKIE['id'] && $message->u_id_r != $_COOKIE['id']){
    header('Location: /');
}

```

Рис. 3.14.1. Получение информации о сообщении и проверка  
возможности пользователя прочитать его



Далее выполняется ещё одна проверка: если пользователь является адресатом сообщения, и сообщение ранее не было прочитано, то сообщение будет помечено, как прочитанное. Для этого будет обновлен статус сообщения в соответствующей записи в базе данных.

```
if($message->isRead == 0 && $_COOKIE['id'] == $message->u_id_r)
{
    $changestatus = 'UPDATE `message` SET isRead = 1 WHERE m_id = :mid';
    $updatestatus = $forum->prepare($changestatus);
    $updatestatus->execute(['mid' => $_GET['id']]);
}
```

Рис. 3.14.2. Обновление статуса сообщения

Само по себе сообщение строится и отображается абсолютно аналогично темам и комментариям на форуме с отличием лишь в том, что теперь немного иначе строится заголовок сверху. Кроме того, над сообщениями никто не может произвести никаких действий: ни удалить их, ни ИЗМЕНИТЬ.

```
<div class="messageheader" style="width: 674px;">
  <?php echo $message->Title;
  echo <div style="margin-left: 465px;">'. $message->MessageSent.' </div>; />
</div>
<div id="msg-show">
  <?php
  $getsinfo = 'SELECT uLogin, Avatar FROM `user` WHERE u_id = :sender';
  $querysinfo = $forum->prepare($getsinfo);
  $querysinfo->execute(['sender' => $message->u_id_s]);
  $sender = $querysinfo->fetch(PDO::FETCH_OBJ);

  echo <div id="msg-avatar">
    <div id="msg-info">
      <div class="post-author-avatar">
        
      </div>
      <div class="post-author-username"><a href="/userpage.php?id='.$message->u_id_s.'" style="color: white;">'. $sen
    </div>
  </div>
  <div id="msg-text">
    '. $message->Content.'
  </div>;
  ?>
</div>
```

Рис. 3.14.3. Отображение содержимого сообщения

Если пользователь является адресатом сообщения, то внизу будет представлена форма для быстрого ответа. Форма быстрого ответа по своей логике идентична форме создания нового сообщения, но разница будет заключаться в том, что в форму автоматически будет подставлен адресат сообщения (это будет человек, отправивший данное сообщение), а также название нового сообщения (представляет собой название полученного



сообщения с приставкой «Re: »). Отправленные таким образом сообщения также будут обработаны обработчиком «sendmessage.php».



```

<?php
if($_COOKIE['id'] == $message->u_id_r)
{echo '<form method="POST" action="/phpactions/sendmessage.php" style="margin-top: 20px;">
  <div class="messageheader">
    Быстрый ответ
  </div>
  <div id="msg-editor">
    <span class="editorlabel">Получатель: </span>
    <input class="input_reg" name="Receiver" style="width: 250px; font-size: 13px;" value="'. $sender->u_login. '" ty
    <br/>
    <span class="editorlabel">Тема: </span>
    <input class="input_reg" maxlength="30" name="Topic" style="width: 250px; font-size: 13px;" value="Re: ' . $mess
    <br/>
    <span class="editorlabel" style="vertical-align: top;">Сообщение: </span>
    <textarea class="input_reg" name="Msg_text" id="message" rows="10"></textarea>
    <input type="hidden" name="Sender" value="'. $_COOKIE['id']. '">
  </div>
  <div class="editor-text-footer">
    <input type="submit" value="Отправить" name="button" class="beigebutton" id="submit">
    <input type="button" value="Отмена" name="button" class="bluebutton" id="cancel" onclick="window.location=pat
  </div>
</form>';}
}>
<script language="JavaScript">
  var path = '/';
</script>

```

Рис. 3.14.4. Форма быстрого ответа

Если пользователь не является адресатом данного сообщения, то это значит, что он является адресантом, и данная форма выведена не будет.

### 3.15 Список пользователей сайта, поиск

Просмотр списка пользователей сайта, его сортировка, и их поиск, может быть осуществлен на странице «allusers.php».

В самом начале будет выведена панель для поиска: поле, в которое можно ввести фрагмент логина пользователя, а также отдельные поля, где можно выбрать опции их сортировки. Если данные значения были установлены ранее, то они сохраняются, и будут подставлены автоматически. В базу данных для получения списка пользователей будет сделан соответствующий установленным параметрам запрос. Пользователей можно сортировать по их id, логину, и дате последнего посещения в порядке возрастания или убывания.

```

<div class="header" style="margin-bottom: 12px; font-size: 18px;">Поиск пользователей</div>
<form method="GET" action="/community/allusers.php">

  <input class="input_reg" style="width: 200px;" maxlength="15" <?php if(isset($_GET['user'])) echo 'value="'.$_GET['user'].'"';>

  <div class="input-select-area">
    <div class="smaller-header" style="font-size: 16px; margin-left: 0px;">Сортировать по: </div>
    <div class="smaller-header" style="font-size: 16px; margin-left: 50px;">В порядке: </div>
  </div>
  <select name="sort" size="1">
    <option value="u_id" <?php if(isset($_GET['sort']) || $_GET['sort'] == 'u_id') echo 'selected';>id пользователя</option>
    <option value="ulogin" <?php if(isset($_GET['sort']) && $_GET['sort'] == 'ulogin') echo 'selected';>Имя пользователя</option>
    <option value="lastSeen" <?php if(isset($_GET['sort']) && $_GET['sort'] == 'lastSeen') echo 'selected';>Дата посещения</option>
  </select>
  <select name="by" size="1" style="margin-left: 22px; margin-top: 5px;">
    <option value="ASC" <?php if(isset($_GET['by']) || $_GET['by'] == 'ASC') echo 'selected';>Возрастания</option>
    <option value="DESC" <?php if(isset($_GET['by']) && $_GET['by'] == 'DESC') echo 'selected';>Убывания</option>
  </select>
</br>
<input type="submit" style="margin-top: 15px;" value="Поиск" class="beigebutton">

```

Рис. 3.15.1. Вывод панели для поиска пользователей

Сам список пользователей представляет собой таблицу. В самом начале выводится значок-статус пользователя. Если пользователь проявлял активность 5 и менее минут назад, то считается, что он онлайн, ему будет присвоен зелёный значок. Если пользователь проявлял активность 8 и менее минут назад, то считается, что он отошел. Ему будет присвоен желтый значок. Иначе пользователю присваивается красный значок.

```

<tr style="font-size: 16px;">
  <td width="20"></td>
  <td width="160">Имя пользователя</td>
  <td width="105">Роль</td>
  <td width="70">Статус</td>
  <td width="180">Последняя активность</td>
  <td width="80" style="text-align: center;">Значок</td>
</tr>
</thead>
<tbody>
<?php
$set_user_list = '';
if(isset($_GET['sort']))
  $set_user_list = 'SELECT u_id, ulogin, Moder, lastSeen FROM `user` WHERE ulogin LIKE \''.$_GET['user'].'%' ORDER BY '.$_GET['sort'];
else
  $set_user_list = 'SELECT u_id, ulogin, Moder, lastSeen FROM `user`';
$userlist = $forum->prepare($set_user_list);
$userlist->execute();
$color = 0;

while($row = $userlist->fetch(PDO::FETCH_OBJ))
{
  echo '<tr border="0">'; if($color%2 == 0) echo ' style="background-color: #ddd4bf;"'; echo '>';

  date_default_timezone_set('Europe/Moscow');
  $last_seen_u = $row->lastSeen;
  $is_online = date('Y-m-d H:i:s', strtotime($last_seen_u.' + 5 minutes'));
  $is_idle = date('Y-m-d H:i:s', strtotime($last_seen_u.' + 10 minutes'));

  $curdate = new DateTime();
  $curdate_form = date_format($curdate, 'Y-m-d H:i:s');

  if($curdate_form <= $is_online)
    echo '<td style="text-align: center;"></td>';
  else if($curdate_form <= $is_idle)
    echo '<td style="text-align: center;"></td>';
  else echo '<td style="text-align: center;"></td>';
}

```

Рис. 3.15.2. Построение таблицы и вывод значка-статуса пользователя

Далее будет выведен логин пользователя, при нажатии на который будет открыта его личная страница.

```
echo '<td><a href="/userpage.php?id='.$row->u_id.'">'.$row->uLogin.'</a></td>';
```

Рис. 3.15.3. Вывод логина пользователя

Затем выводится роль пользователя: «пользователь» или «модератор».

```
$color++;
if($row->Moder)
    echo '<td><span style="color: green;">Модератор</span></td>';
else
    echo '<td>Пользователь</td>';
```

Рис. 3.15.4. Вывод роли пользователя

Затем будет выведено текстовое описание его статуса: «онлайн», «офлайн», или «отошел».

```
if($curdate_form <= $is_online)
    echo '<td><span style="color: green;">Онлайн</span></td>';
else if($curdate_form <= $is_idle)
    echo '<td><span style="color: #ee9a4d;">Отошел</span></td>';
else echo '<td><span style="color: #731d08;">Офлайн</span></td>';
```

Рис. 3.15.5. Вывод описания статуса пользователя

Затем выводится дата и время, когда пользователь в последний раз проявлял активность на сайте.

```
//echo '<td>'.$row->LastSeen.'</td>';
$lastseen_date= date('H:i d/m/Y', strtotime($row->LastSeen));
echo '<td><span style="margin-left: 30px;">'.$lastseen_date.'</span></td>';
```

Рис. 3.15.6. Вывод информации о последней активности пользователя

В конце выводится количество «звезд» пользователя (как считается описано в пункте 3.4).

```
$comandcntp = 'SELECT COUNT(*) FROM "forumposts" WHERE u_id = :posterid';
$querycntp = $forum->prepare($comandcntp);
$querycntp->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
$querycntp->execute();
$cntp_res = $querycntp->fetch(PDO::FETCH_ASSOC);

$comandcntc = 'SELECT COUNT(*) FROM "comments" WHERE u_id = :posterid';
$querycntc = $forum->prepare($comandcntc);
$querycntc->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
$querycntc->execute();
$cntc_res = $querycntc->fetch(PDO::FETCH_ASSOC);

$countuserposts = current($cntp_res) + current($cntc_res);
$starsnumber = floor(min($countuserposts/2, 6)+ 1);

echo '<td><p style="text-align: center;"></p></td>';
```

Рис. 3.15.7. Подсчет количества «звезд» пользователя и их вывод

### 3.16 Главная страница сайта

На главную страницу сайта должны быть выведены 3 последних сообщения, оставленных в специальной теме «Новости сайта» (описано ранее). Для этого делается соответствующий запрос в базу данных с указанием кодового слова LIMIT:

```
<?php
$getlastnews = 'SELECT * FROM `forumposts` WHERE topic_id = 8 ORDER BY post_id DESC LIMIT 3';
$querylastnews = $forum->prepare($getlastnews);
$querylastnews->execute();

while($row = $querylastnews->fetch(PDO::FETCH_OBJ))
{
```

Рис. 3.16.1. Запрос на получение трех последних постов в заданном разделе и итерирование по ним

Далее посты выводятся на главную страницу. Логика вывода абсолютно идентична логике вывода темы в файле «post.php», только не выводится панель для управления темой. Ниже будет выведена ссылка, нажав на которую можно будет попасть на страницу «post.php» в соответствующую тему.

```

$row1 = $query1->fetch(PDO::FETCH_OBJ);

$comand2 = 'SELECT COUNT(*) FROM `forumposts` WHERE u_id = :posterid';
$query2 = $forum->prepare($comand2);
$query2->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
$query2->execute();
$row2 = $query2->fetch(PDO::FETCH_ASSOC);

$comand3 = 'SELECT COUNT(*) FROM `comments` WHERE u_id = :posterid';
$query3 = $forum->prepare($comand3);
$query3->bindValue(":posterid", $row->u_id, PDO::PARAM_STR);
$query3->execute();
$row3 = $query3->fetch(PDO::FETCH_ASSOC);

$countuserposts = current($row2) + current($row3);

echo '<div class="postframe" id="main">
    <div class="postauthor">
        <div class="post-author-avatar">
            
        </div>
        <div class="post-author-username"><a href="/userpage.php?id='.$row->u_id.'" style="color: white;">'.$row1->u_id.'
        if($row1->Moder) {
            echo '<div class="post-moder">Модератор</div>';
        }
        echo '<div class="post-author-stat">Активность:<br/>'.$countuserposts.' сообщений</div>';
    </div>
    <div class="posttext">
        <div class="postheader">
            <span style="margin-left: 15px;">Опубликовано: '.$row->PostDate.'
            </span>
        </div>
        <div class="posttextcontent">'.$row->Content.'
        </div>
    </div>
</div>
<div class="index-link"><a href="/forums/post.php?id='.$row->post_id.'">Перейти к новости на форуме</a></div>';

```

Рис. 3.16.2. Вывод содержимого постов на главную страницу



## 4 Обзор программы

Полученное приложение реализует весь описанный в п. 1 функционал. Протестируем его. При подключении к сайту он выглядит следующим образом:

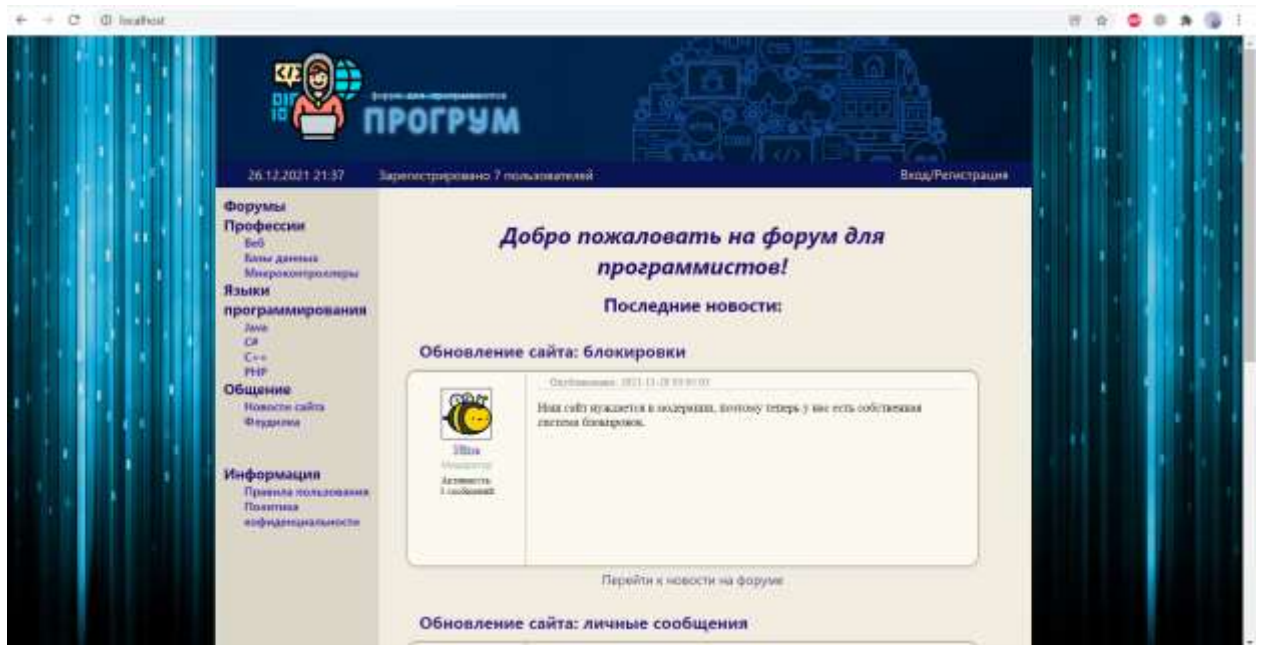


Рис. 4.1. Главная страница сайта для неавторизованного пользователя

Перейдём на страницу для авторизации или регистрации и выполним вход в аккаунт модератора:

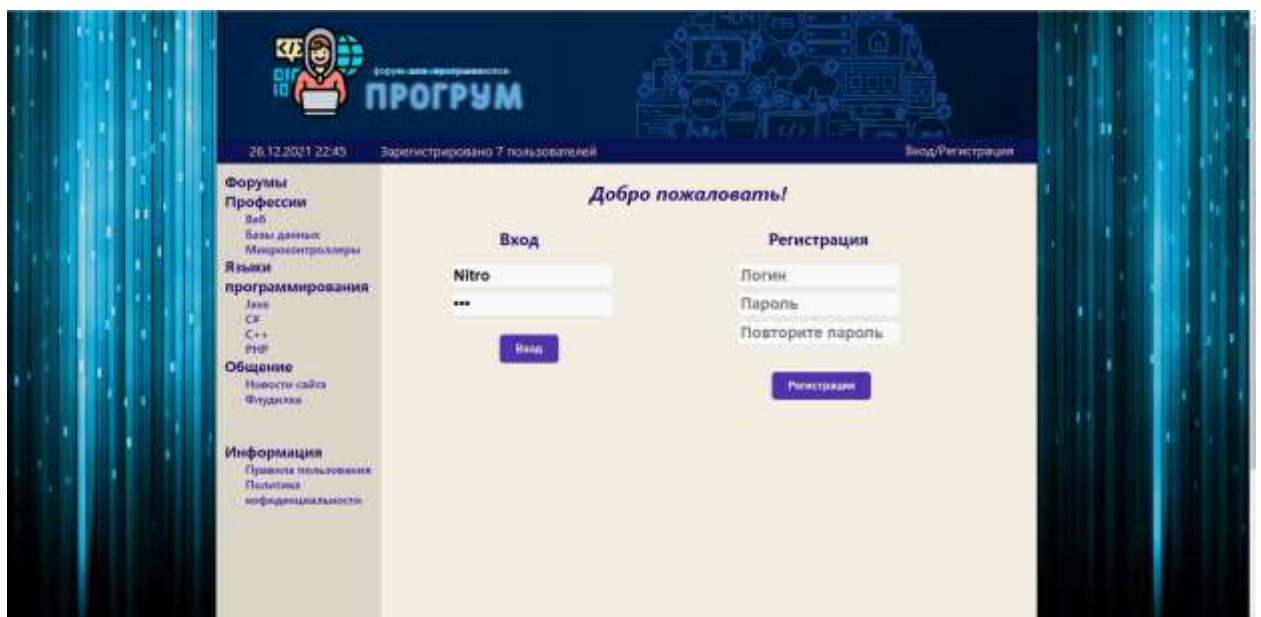


Рис. 4.2. Выполнение входа на сайт

Мы будем переадресованы на нашу страницу. Также навигационная панель сверху поменяет свой вид:



Рис. 4.3. Профиль, в который был выполнен вход

При нажатии на кнопку «История блокировок» будет вызвано следующее окно:

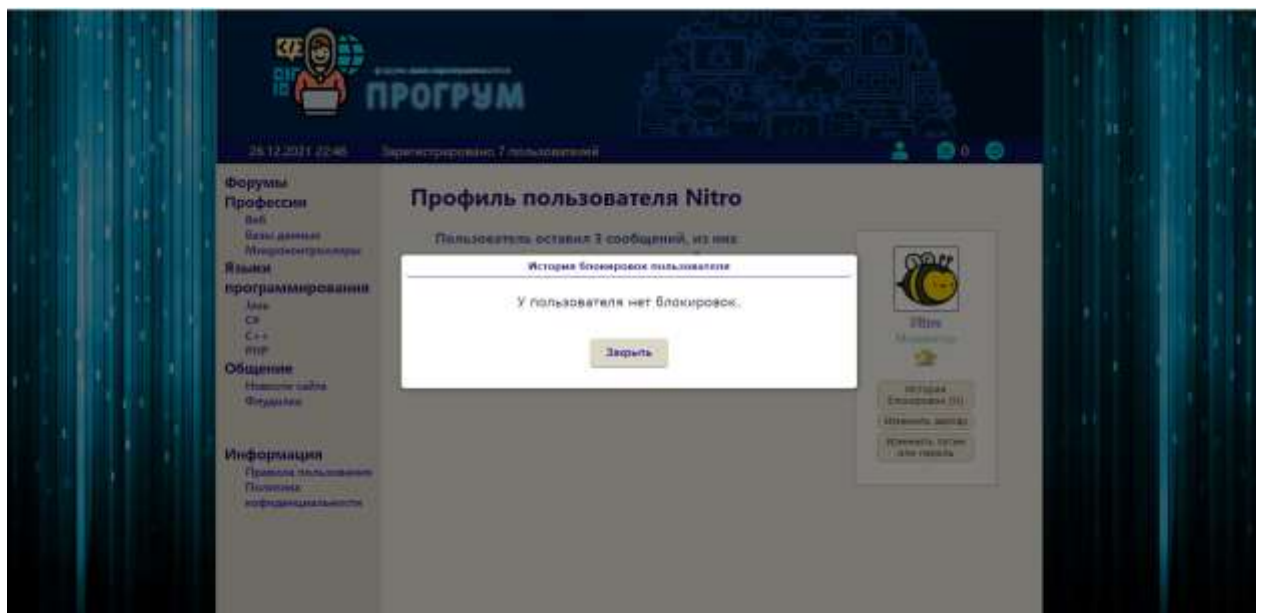


Рис. 4.4. Окно с историей блокировок пользователя

При нажатии на кнопку «Изменить аватар» будет вызван описанный ранее диалог для смены аватара. Текущий аватар обводится в рамку.



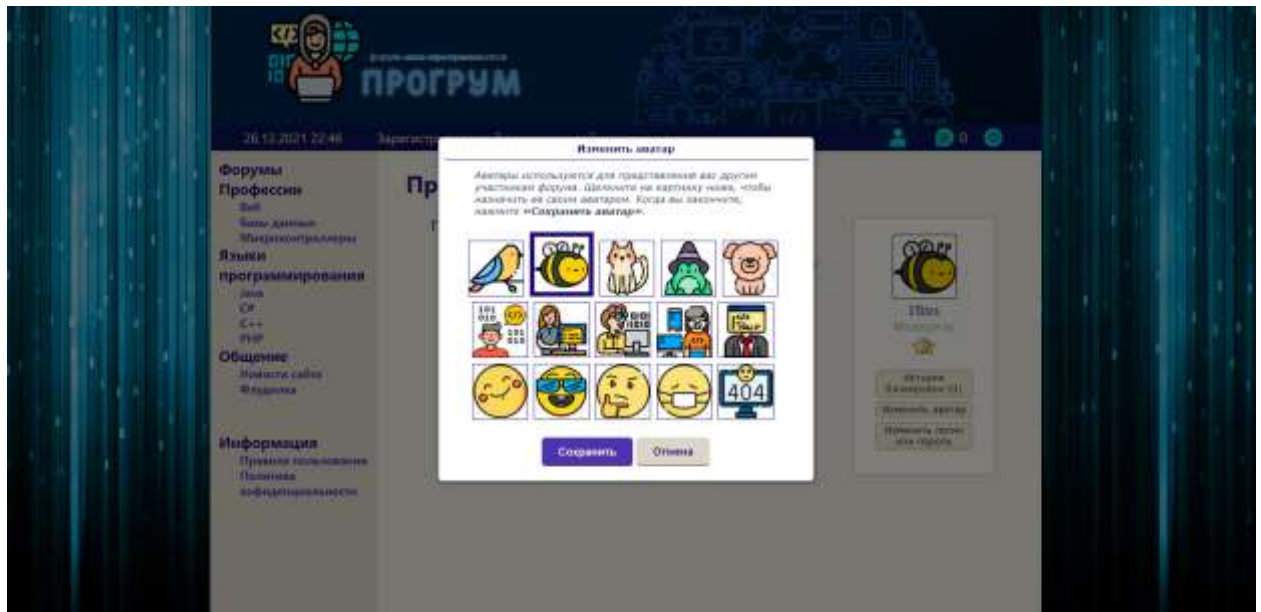


Рис. 4.5. Окно для смены аватара

При нажатии на другой аватар он будет обведен в рамку, что сигнализирует о выборе данного аватара для смены. Рамка с предыдущего аватара удаляется.

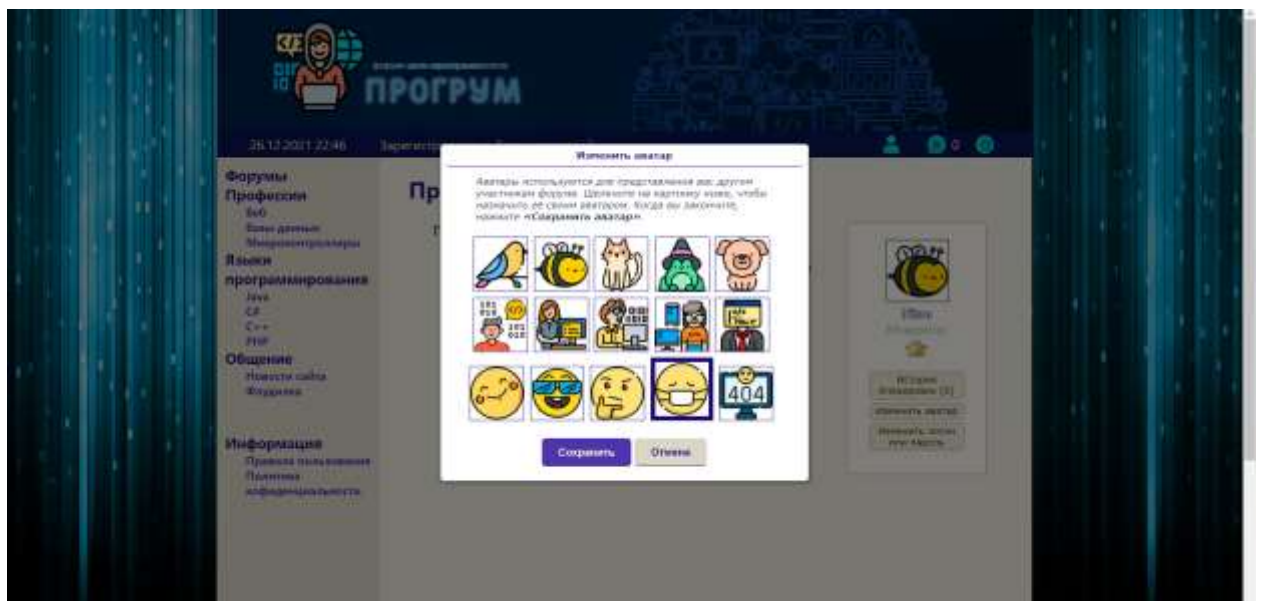


Рис. 4.6. Переключение аватара

При нажатии на кнопку «Сохранить» аватар будет изменен:

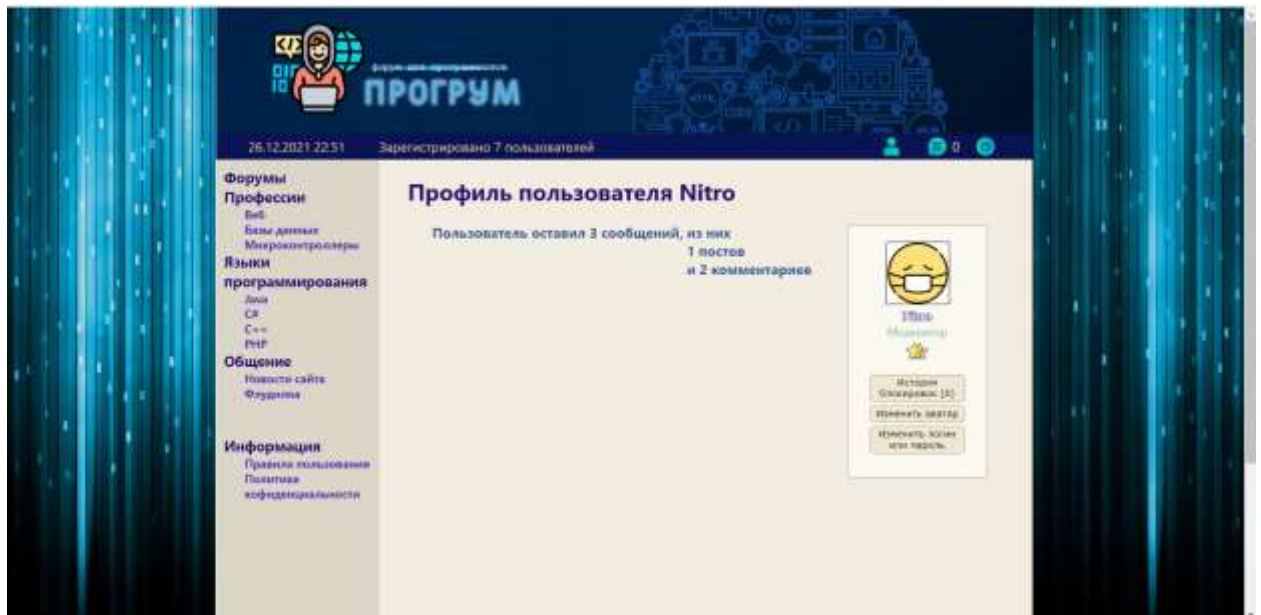


Рис. 4.7. Аватар успешно изменен

При нажатии на кнопку «Изменить логин или пароль» будет открыто диалоговое окно для смены логина или пароля. Поменяем логин для тестирования данной функции.

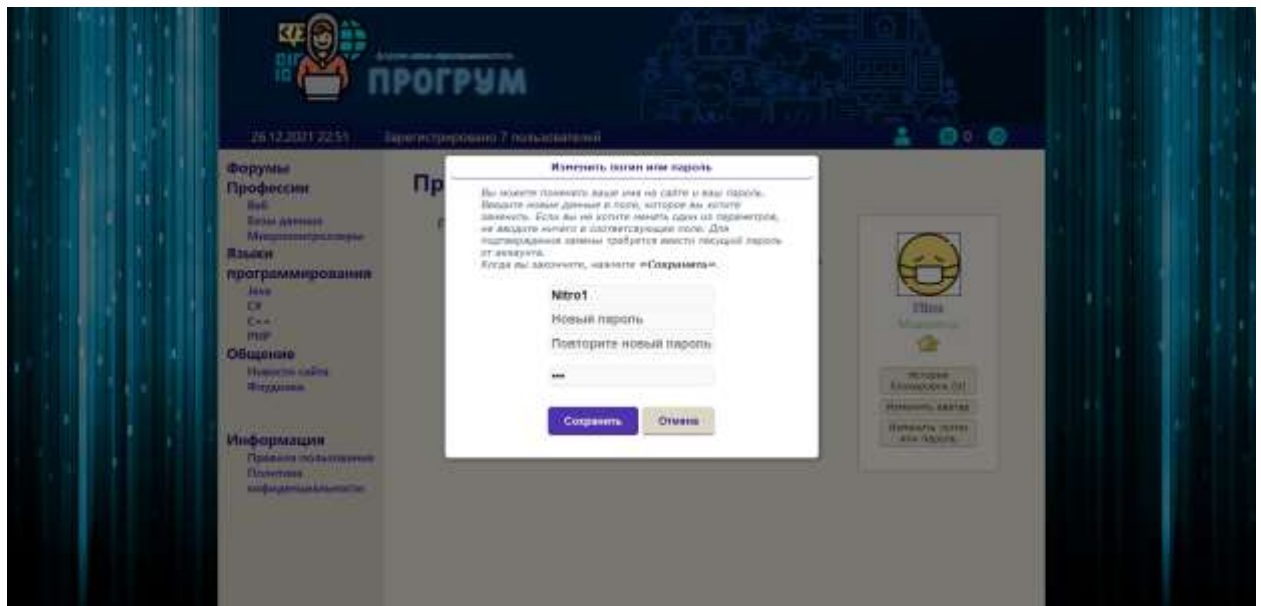


Рис. 4.8. Смена логина и/или пароля

Логин был успешно изменен:



Рис. 4.9. Логин был успешно изменен

Перейдём в какой-нибудь раздел на форуме с помощью боковой панели, например «Новости сайта».



Рис. 4.10. Раздел «Новости сайта» на форуме

Поскольку мы – модератор, нам доступна кнопка создания новой темы в данном разделе. Создадим новую тему.

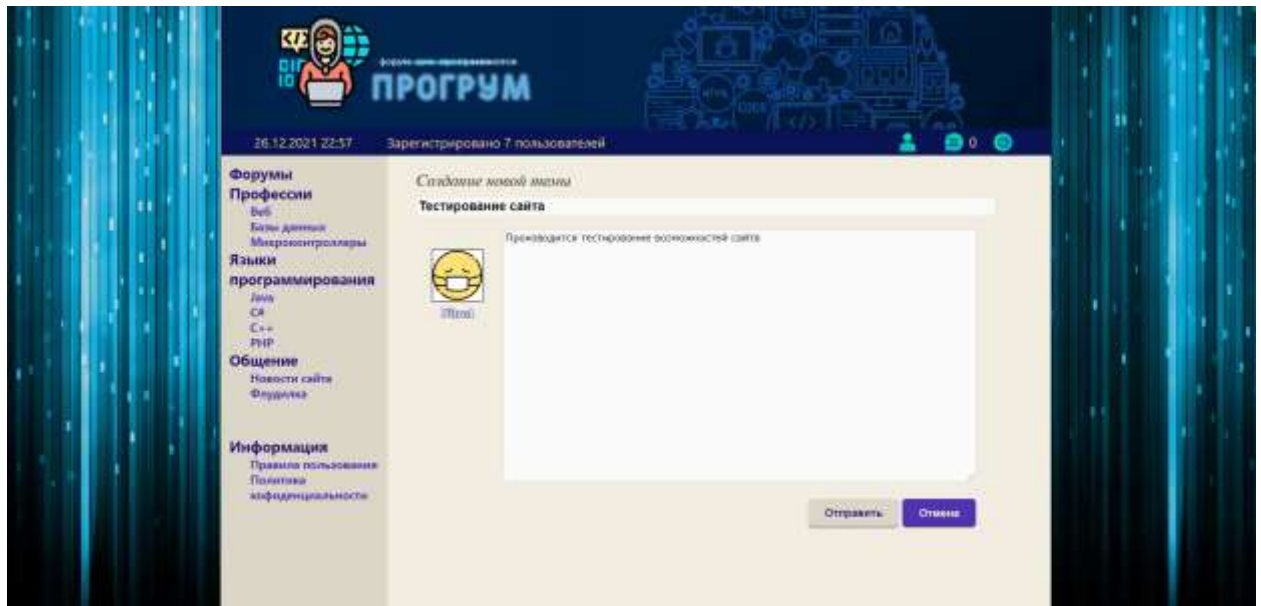


Рис. 4.11. Создание новой темы на форуме

Тема была успешно создана:



Рис. 4.12. Тема была успешно создана

Также созданная тема была выведена на главную страницу сайта, а счетчик оставленных нами сообщений на форуме увеличился.





Рис. 4.13. Созданная тема на главной странице сайта

При нажатии на кнопку «перейти к новости на форуме» будет осуществлен переход в саму тему. Оставим к ней какой-либо комментарий.

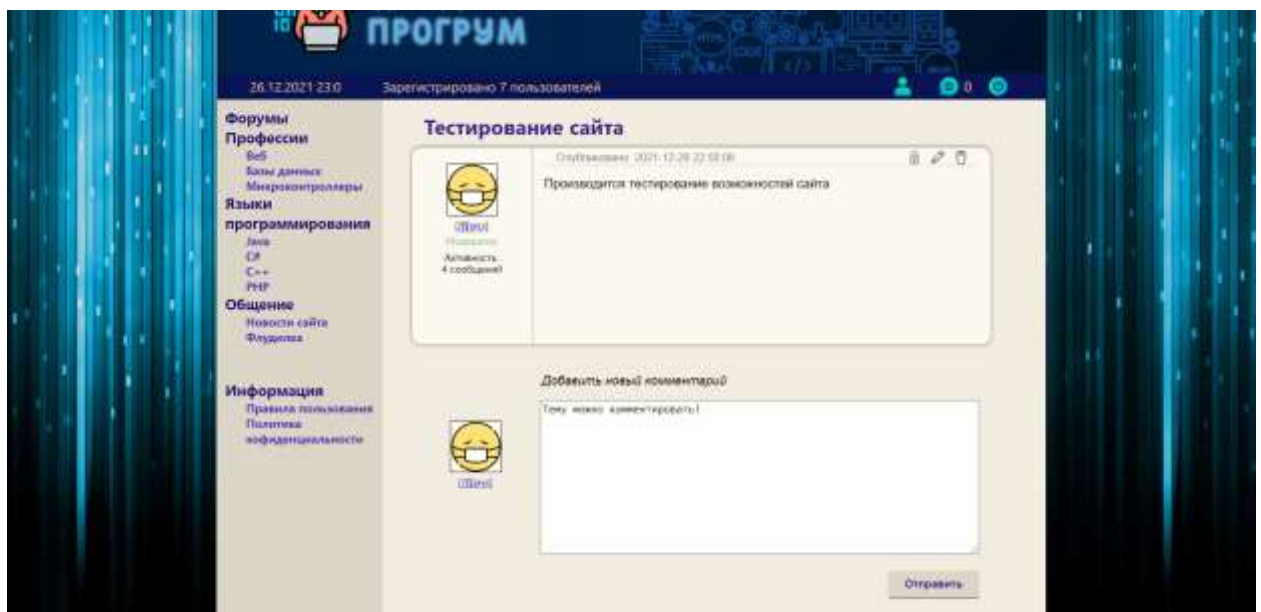


Рис. 4.14. Вкладка с темой и добавление нового комментария



Рис. 4.15. Комментарий успешно добавлен

Закроем данную тему для обсуждения, нажав на значок замка у соответствующей темы. Будет выведено окно для подтверждения действий:

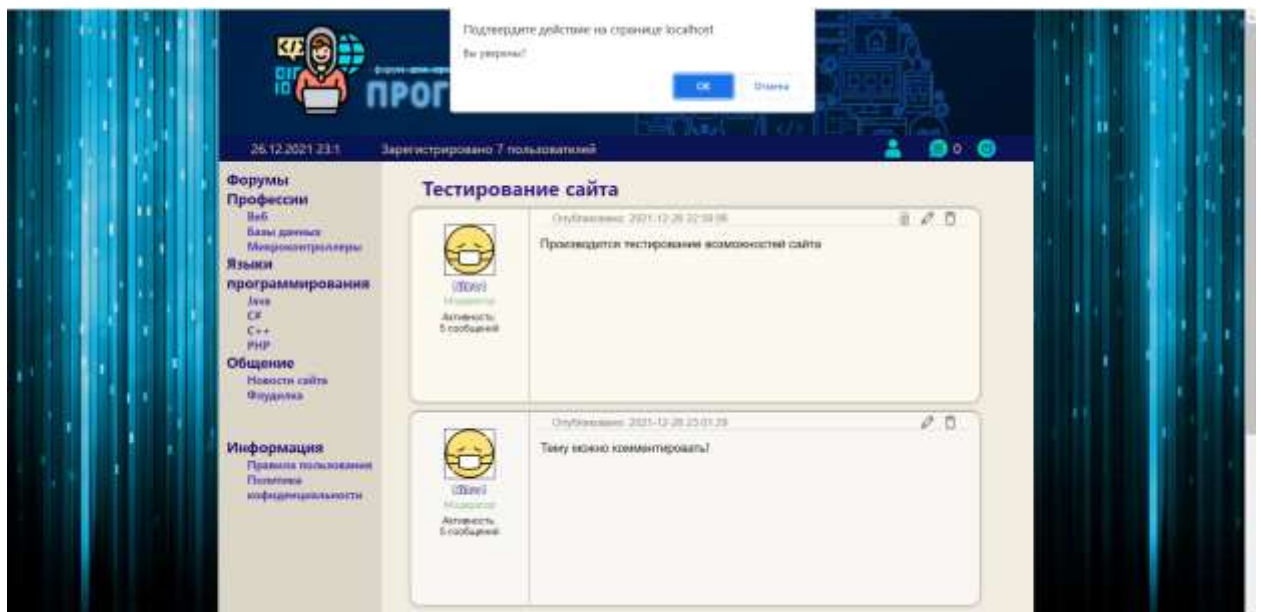


Рис. 4.16. Окно для подтверждения действия

Нажимаем «ок», теперь тема закрыта для обсуждения:

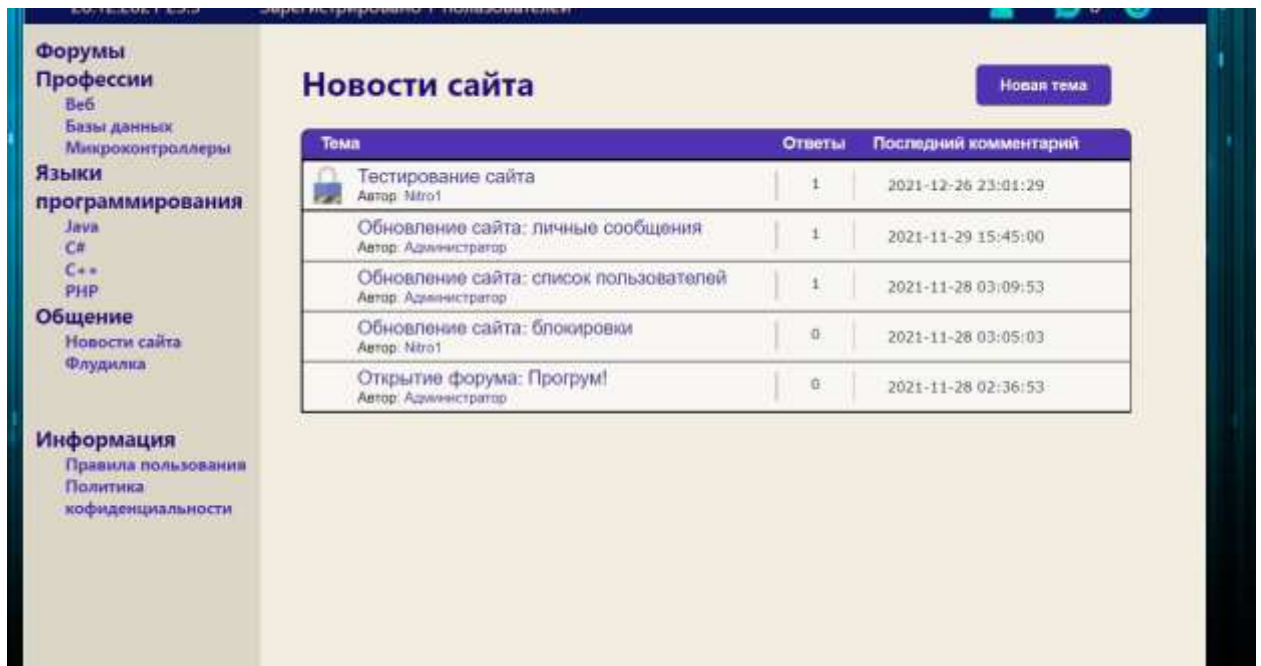


Рис. 4.17. Тема закрыта для обсуждения

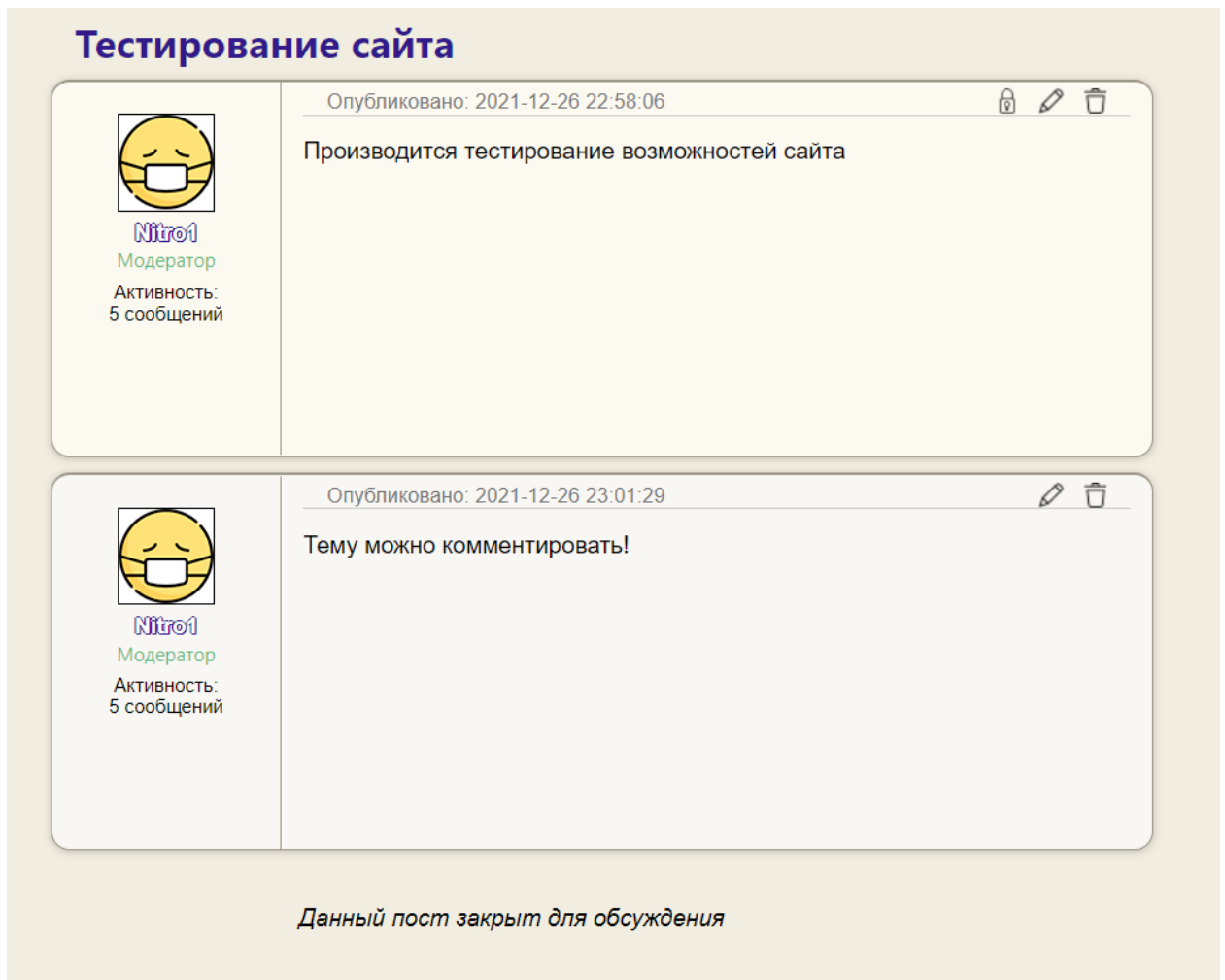


Рис. 4.18. Тема закрыта для обсуждения

Отредактируем пост, нажав на значок с карандашом.



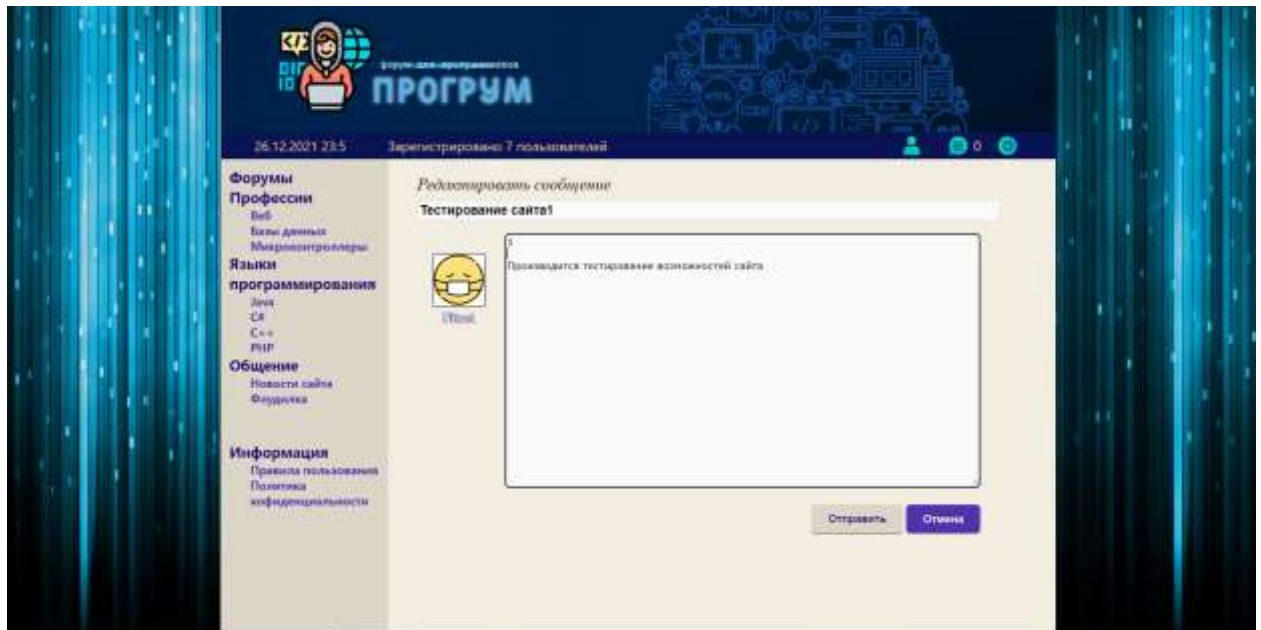


Рис. 4.19. Редактирование оставленного сообщения

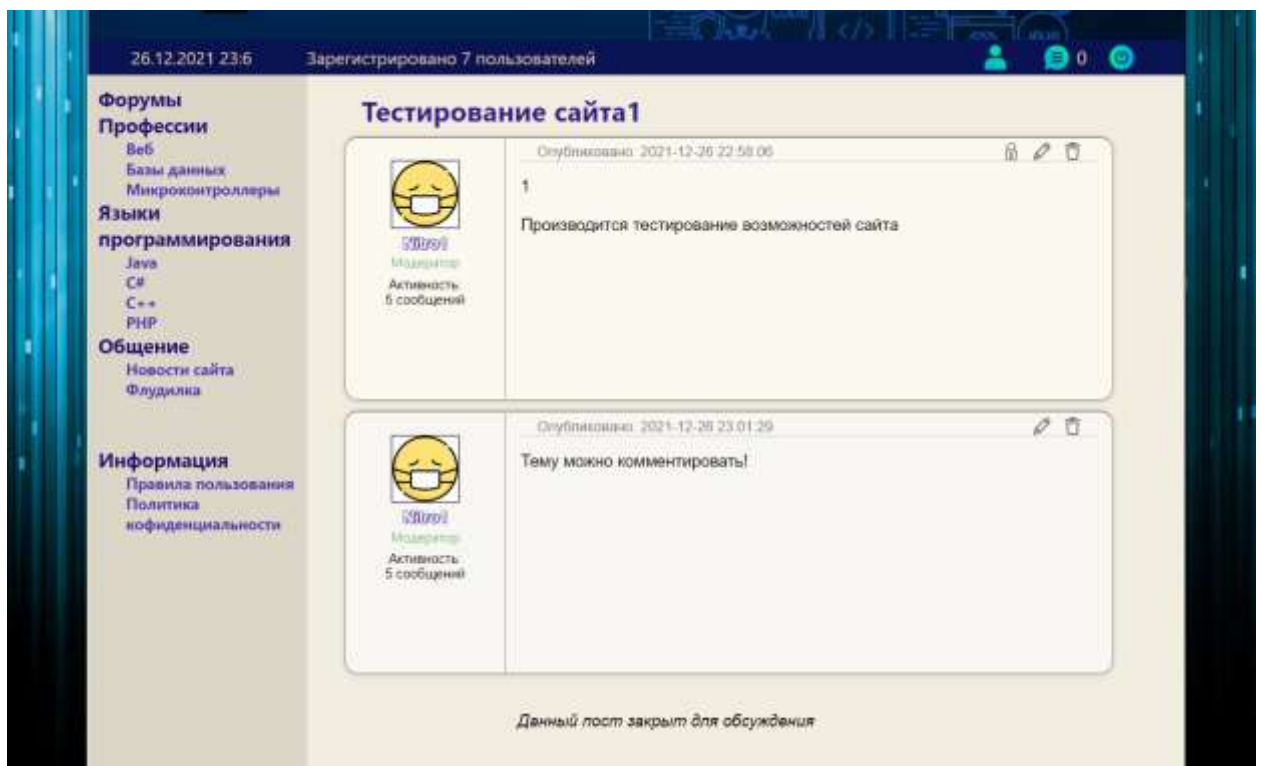


Рис. 4.20. Данные успешно обновлены

Удалим данный пост и комментарий к нему, нажав на значок корзины. Пост и комментарий к нему будут удалены.

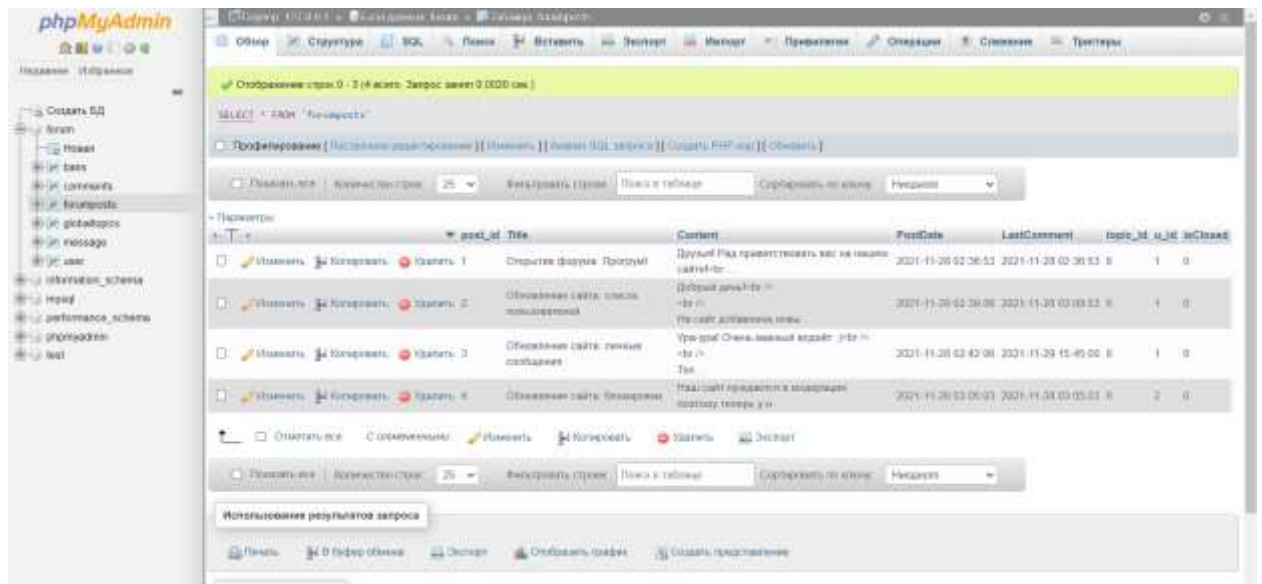


Рис. 4.21. Пост пропал из базы данных

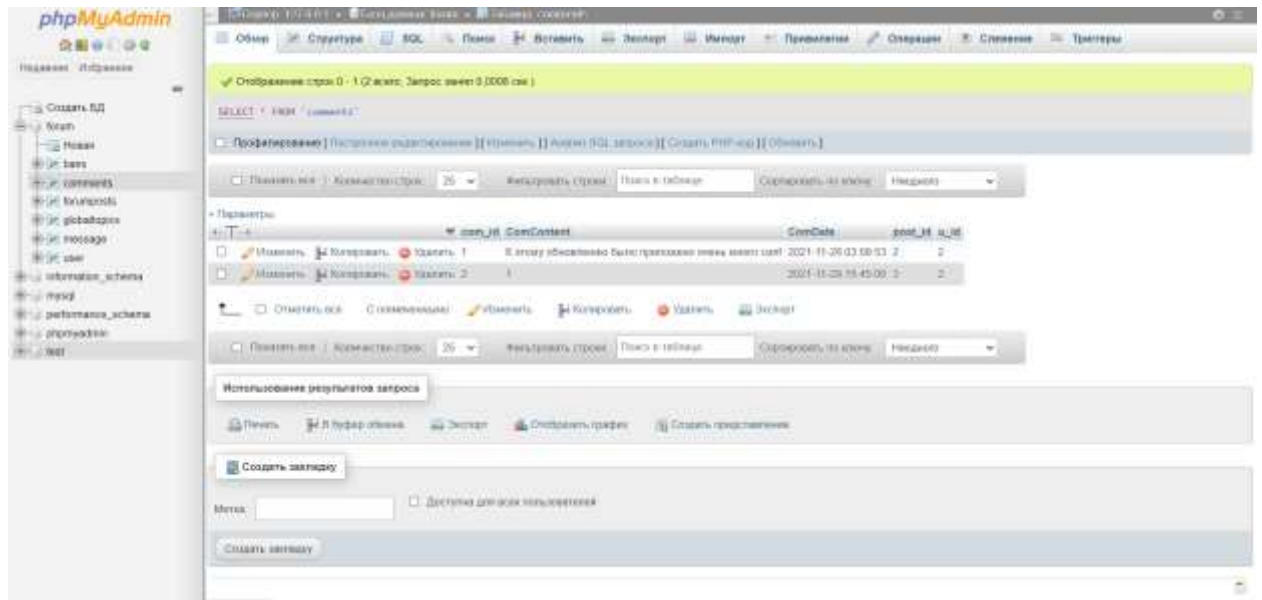


Рис. 4.22. Комментарий к посту также пропал из базы данных

Теперь поста снова нет на главной странице:



Рис. 4.23. Пост пропал с главной страницы

Также видно, что счетчик оставленных нами сообщений вернулся к первоначальному состоянию

Перейдем на вкладку со списком пользователей, нажав на надпись «Зарегистрировано x пользователей» и выполним некоторый поиск. Также отметим, что функция онлайн работает исправно:

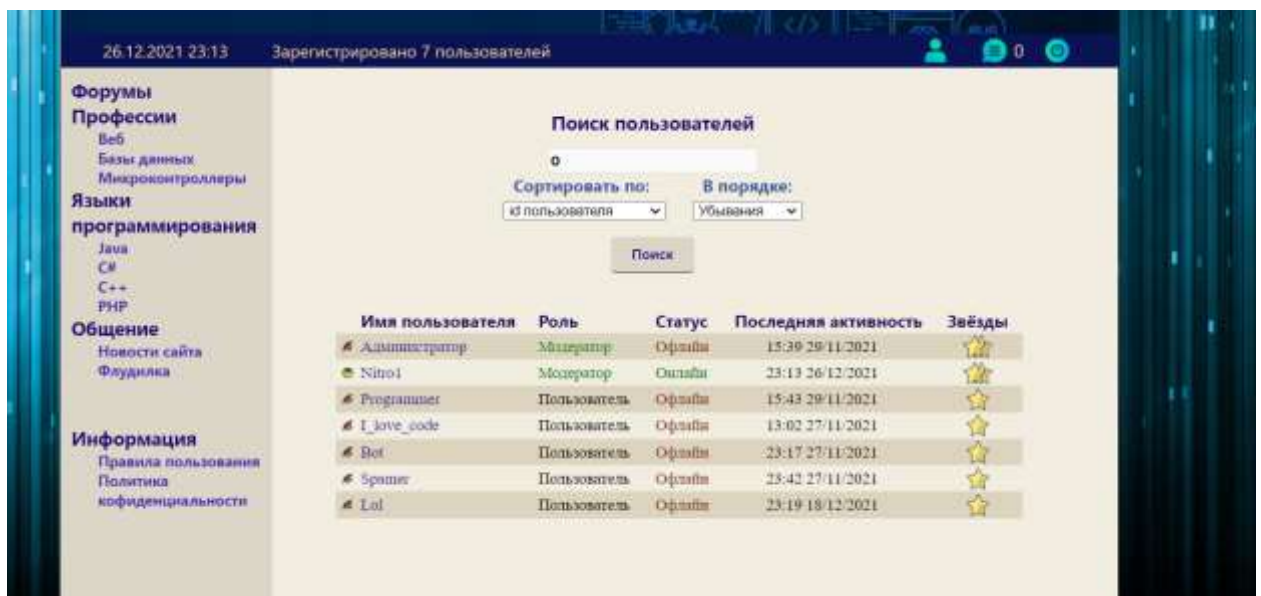


Рис. 4.24. Список пользователей сайта

Был выполнен поиск пользователей сайта, в логине которых есть английская буква «o», результаты отсортированы по id пользователя в порядке убывания (от самых новых страниц к самым старым). Получены следующие результаты:



Рис. 4.25. Результаты поиска

Перейдем в профиль пользователя «Bot». Как видно, пользователь заблокирован навсегда:



Рис. 4.26. Страница пользователя Bot

При нажатии на кнопку «История блокировок» будут показаны все блокировки, которые получил пользователь, и их описание:



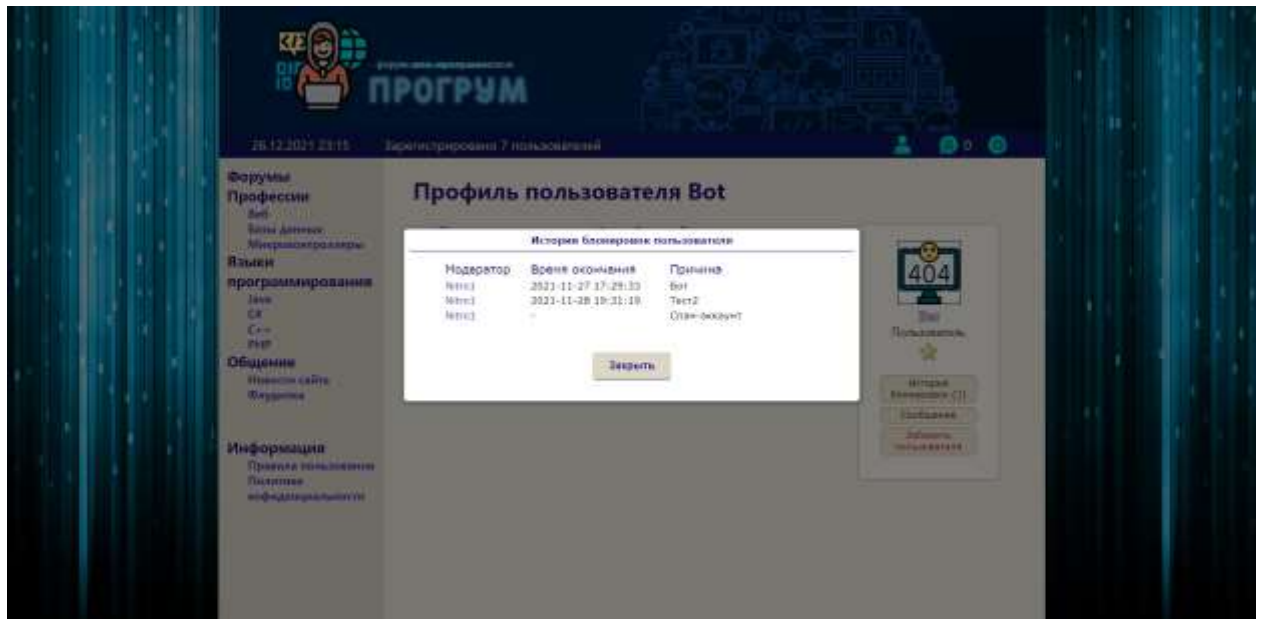


Рис. 4.27. Баны пользователя Bot

Как видно, данный пользователь в третий раз был забанен с помощью чекбокса «забанить навсегда».

Перейдём на другую страницу пользователя с логином Lol. Заблокируем его аккаунт на 10 минут:

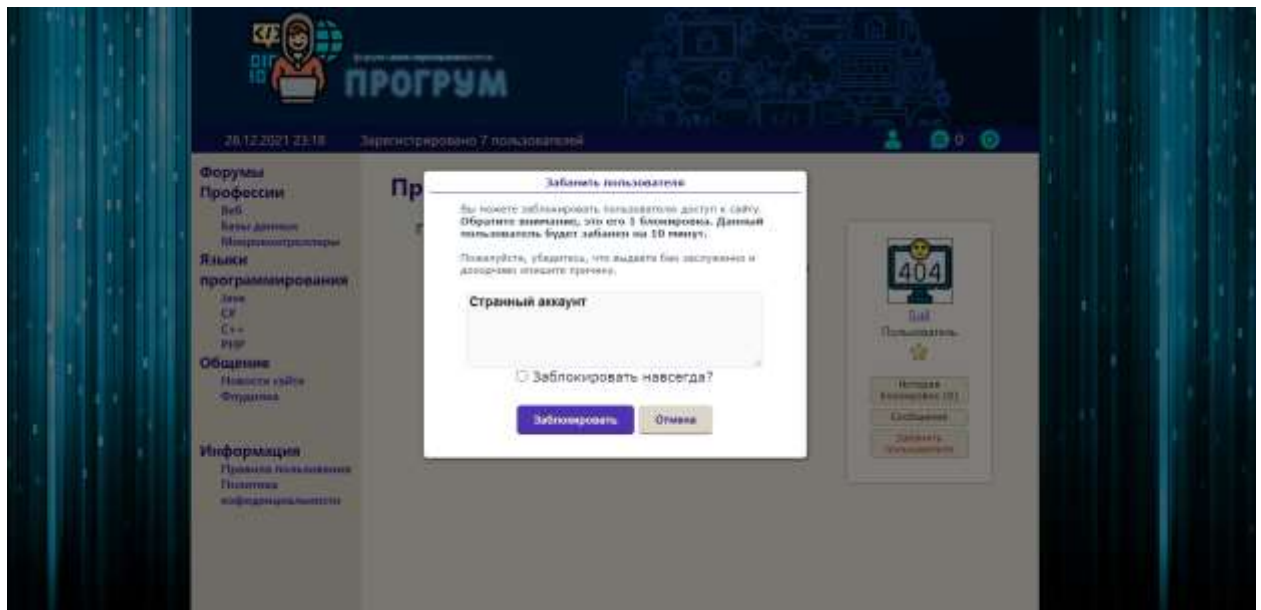


Рис. 4.28. Окно блокировки пользователя

Теперь пользователь заблокирован. На его страницу выведена соответствующая информация, а его история блокировок обновлена:

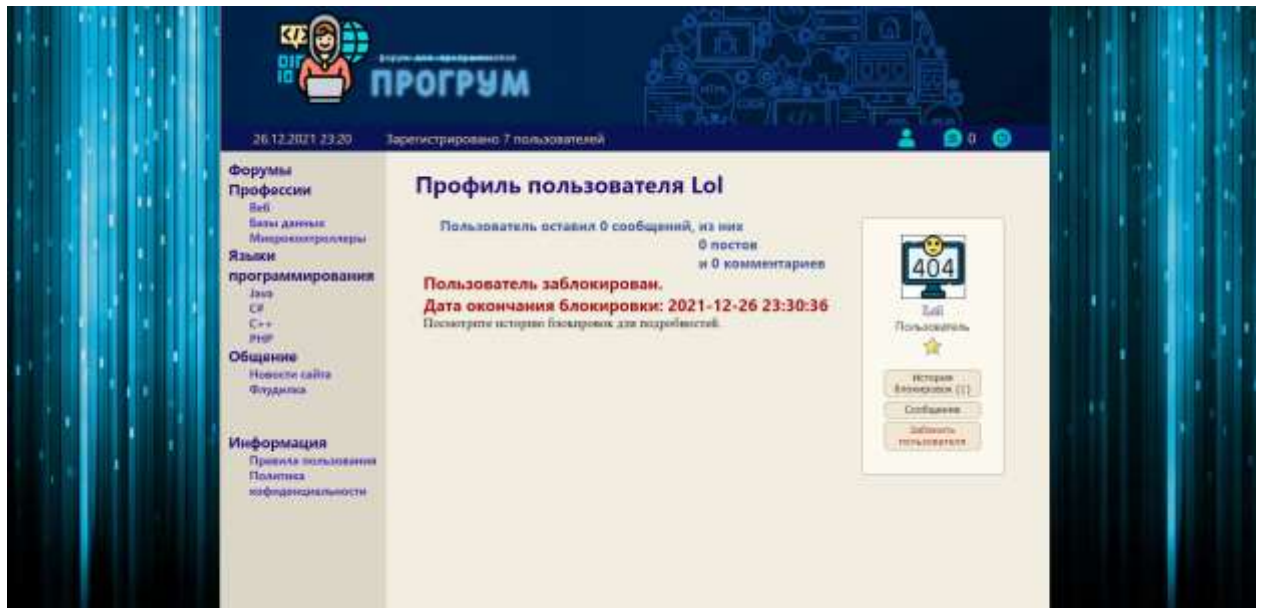


Рис. 4.29. Аккаунт пользователя после блокировки

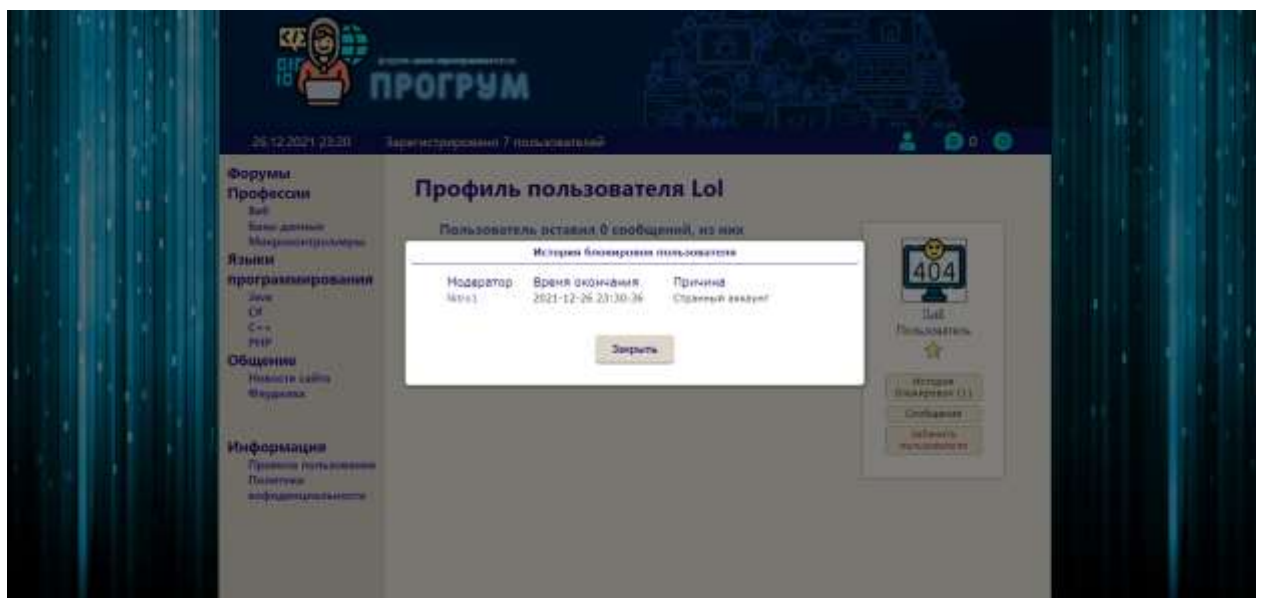


Рис. 4.30. Обновленная история блокировок пользователя

Если попытаться выполнить вход под данным пользователем, то он не будет произведен до окончания этого срока блокировки.

Следует обратить внимание, что возможности заблокировать другого модератора нет:



Рис. 4.31. Профиль другого модератора

Отправим личное сообщение какому-либо пользователю, например «Programmer». Для этого на его личной странице нажимаем на кнопку «Сообщение».



Рис. 4.32. Профиль пользователя Programmer



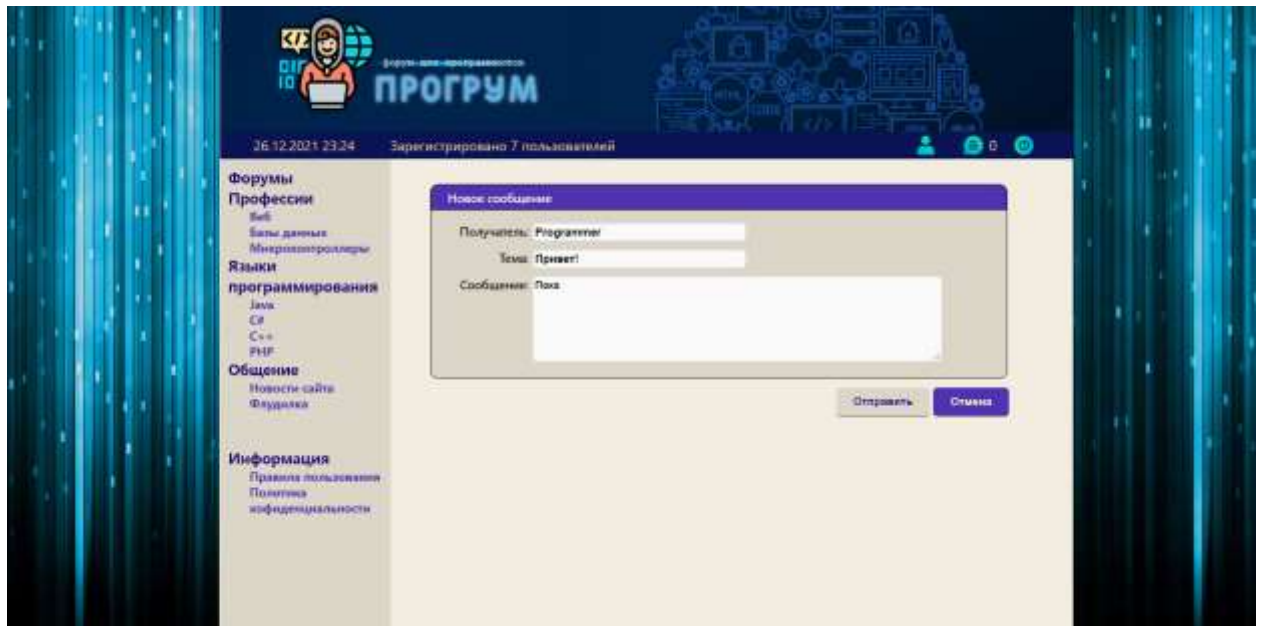


Рис. 4.33. Окно отправки нового сообщения

После отправки сообщения мы будем переадресованы в центр сообщений. Можно посмотреть как полученные письма:

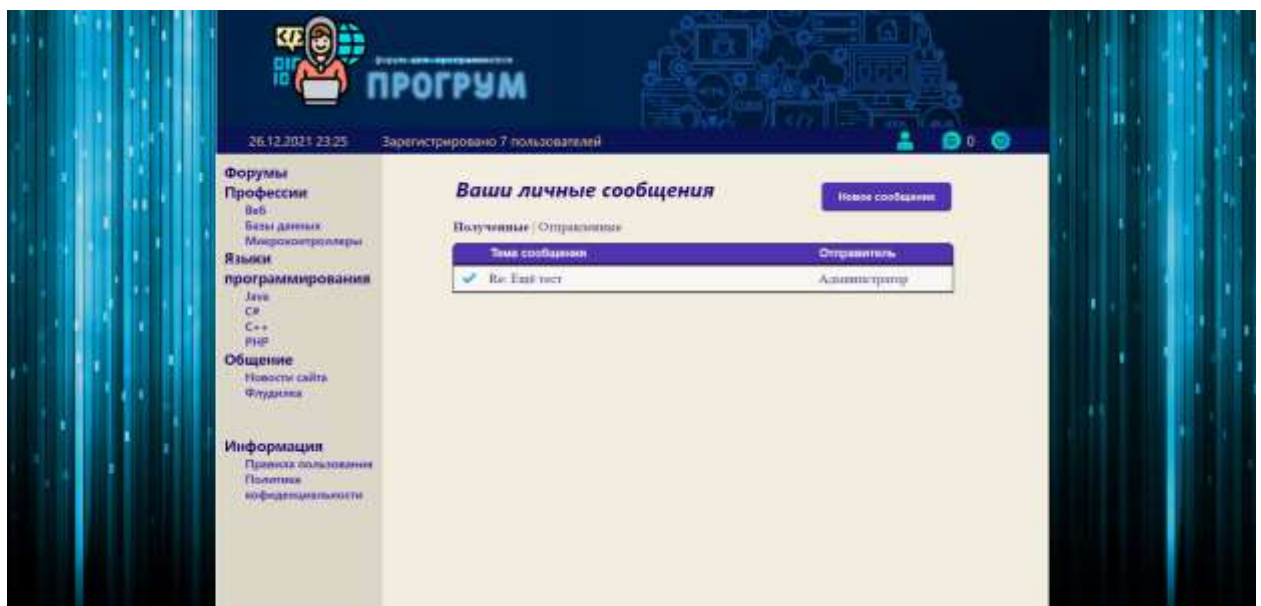


Рис. 4.34. Полученные сообщения

так и отправленные:

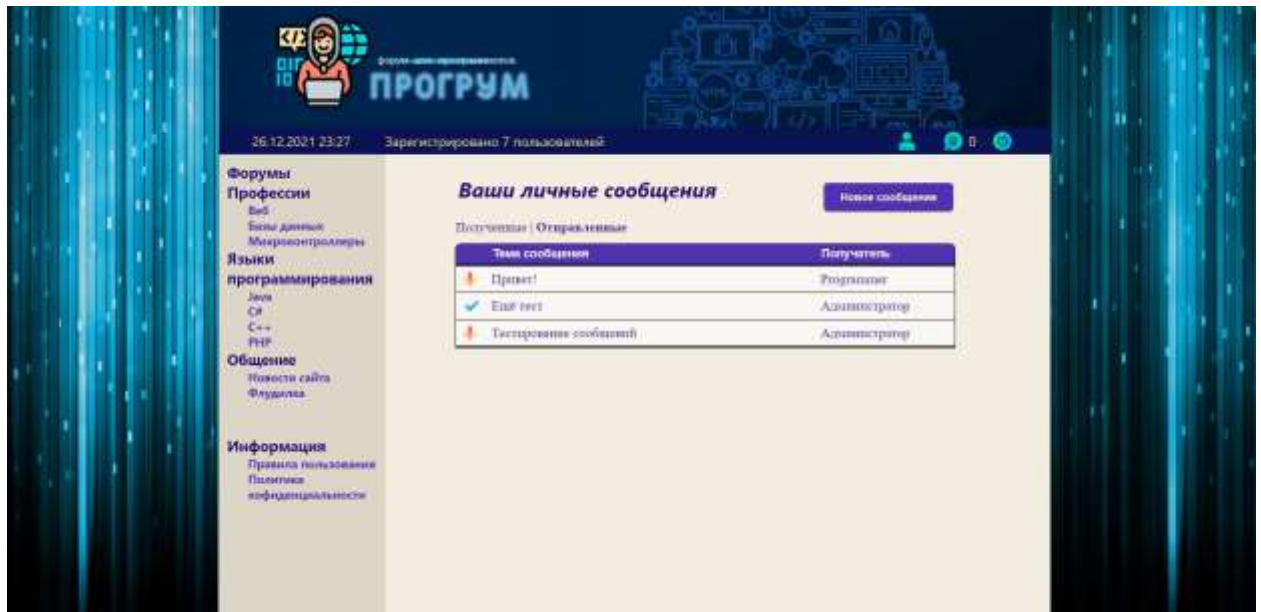


Рис. 4.35. Отправленные сообщения

Следует обратить внимание на разный статус сообщений. Синей галочкой отмечают прочитанные адресатом сообщения. Красным восклицательным знаком – непрочитанные.

Выйдем из аккаунта и войдем в аккаунт пользователя Programmer. Теперь центр сообщений уведомляет о наличии нового сообщения:



Рис. 4.36. Вход под пользователем Programmer

Перейдем в центр сообщений, и откроем полученное сообщение. Стоит обратить внимание, что теперь сообщение считается прочитанным, и центр

сообщений перестает уведомлять о новом сообщении. Также к сообщению доступна форма отправки быстрого ответа.

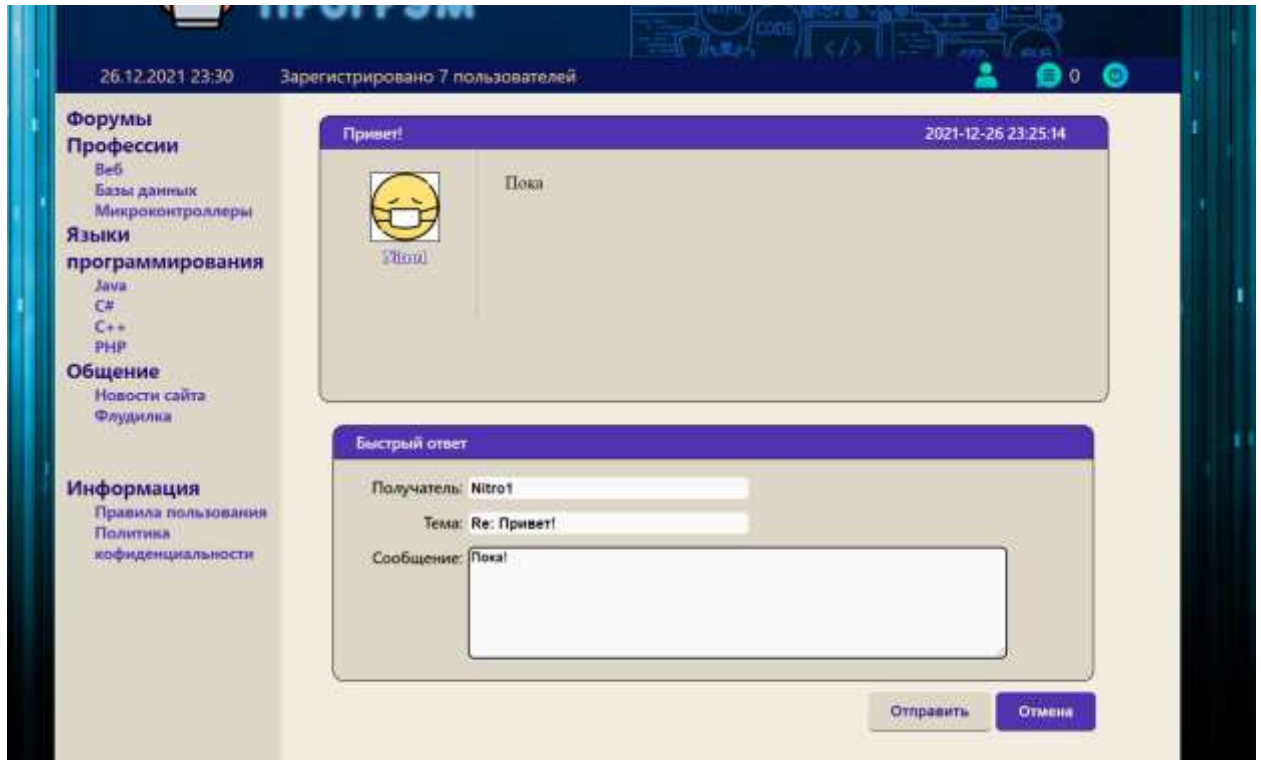


Рис. 4.37. Полученное сообщение

Отправим ответ и через раздел «Отправленные» в центре сообщений перейдем к нему.

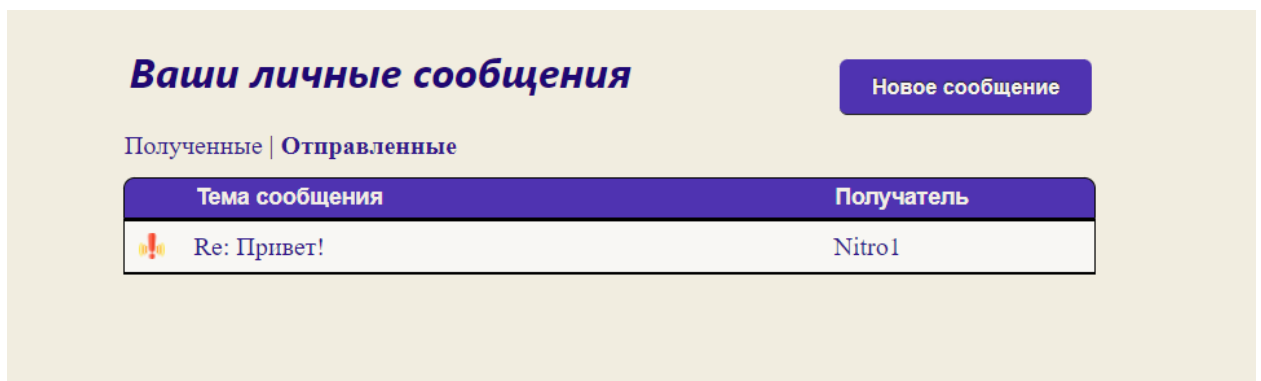


Рис. 4.38. Раздел «Отправленные»



Рис. 4.39. Отправленное ранее сообщение

Как видно, форма для быстрого ответа недоступна.

Во вкладке «Полученные» можно убедиться в том, что полученному ранее сообщению присвоен статус «Прочитано»:

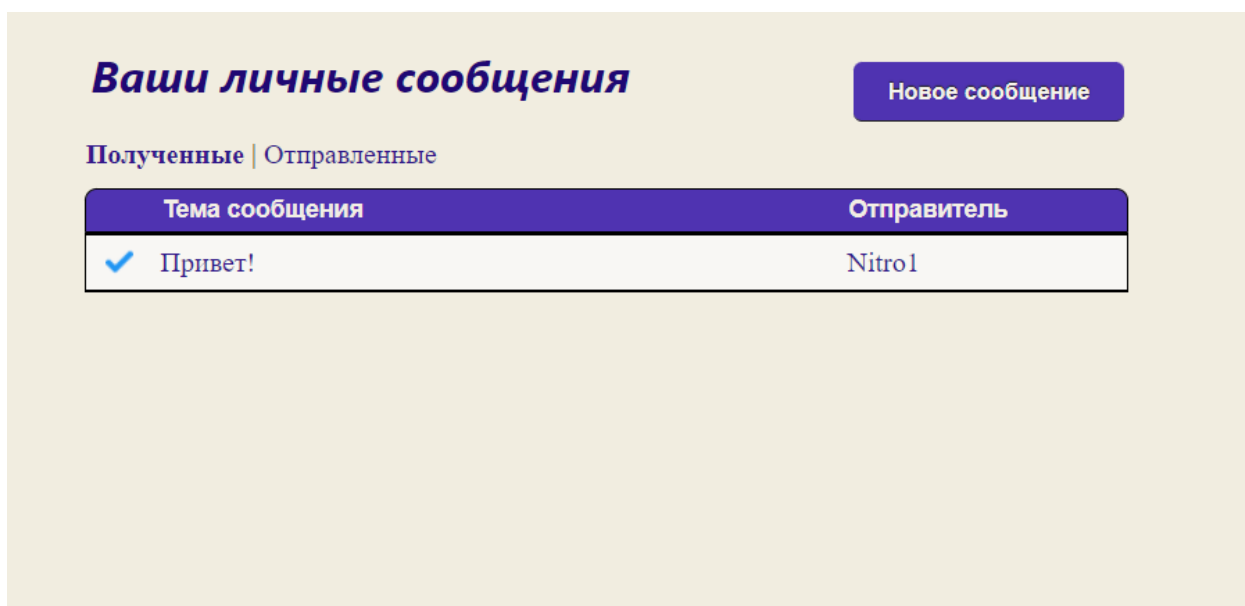


Рис. 4.40. Новый статус сообщения

Перейдем в раздел с новостями сайта. Мы не можем добавить к нему новую тему, хотя мы можем создавать новые темы во всех остальных разделах:



Рис. 4.41. Раздел «Новости сайта» для пользователя

Перейдем в любую тему. Как видно, мы можем комментировать тему, но мы не можем её удалить/изменить. Чужие комментарии нам также недоступны для редактирования:



Рис. 4.42. Тема на сайте от лица пользователя

Но если мы напишем новый комментарий, то мы сможем его отредактировать или удалить. Аналогичная ситуация будет и с добавленными постами.





Рис. 4.43. Добавленный новый комментарий

В других разделах форума мы можем создавать новые темы:



Рис. 4.44. Другой раздел форума

Выйдем из аккаунта и посмотрим, как интерфейс выглядит для гостя. Теперь мы не можем нигде создать новую тему:





Рис. 4.45. Форумы для неавторизованного пользователя

Также отсутствует функция комментирования:

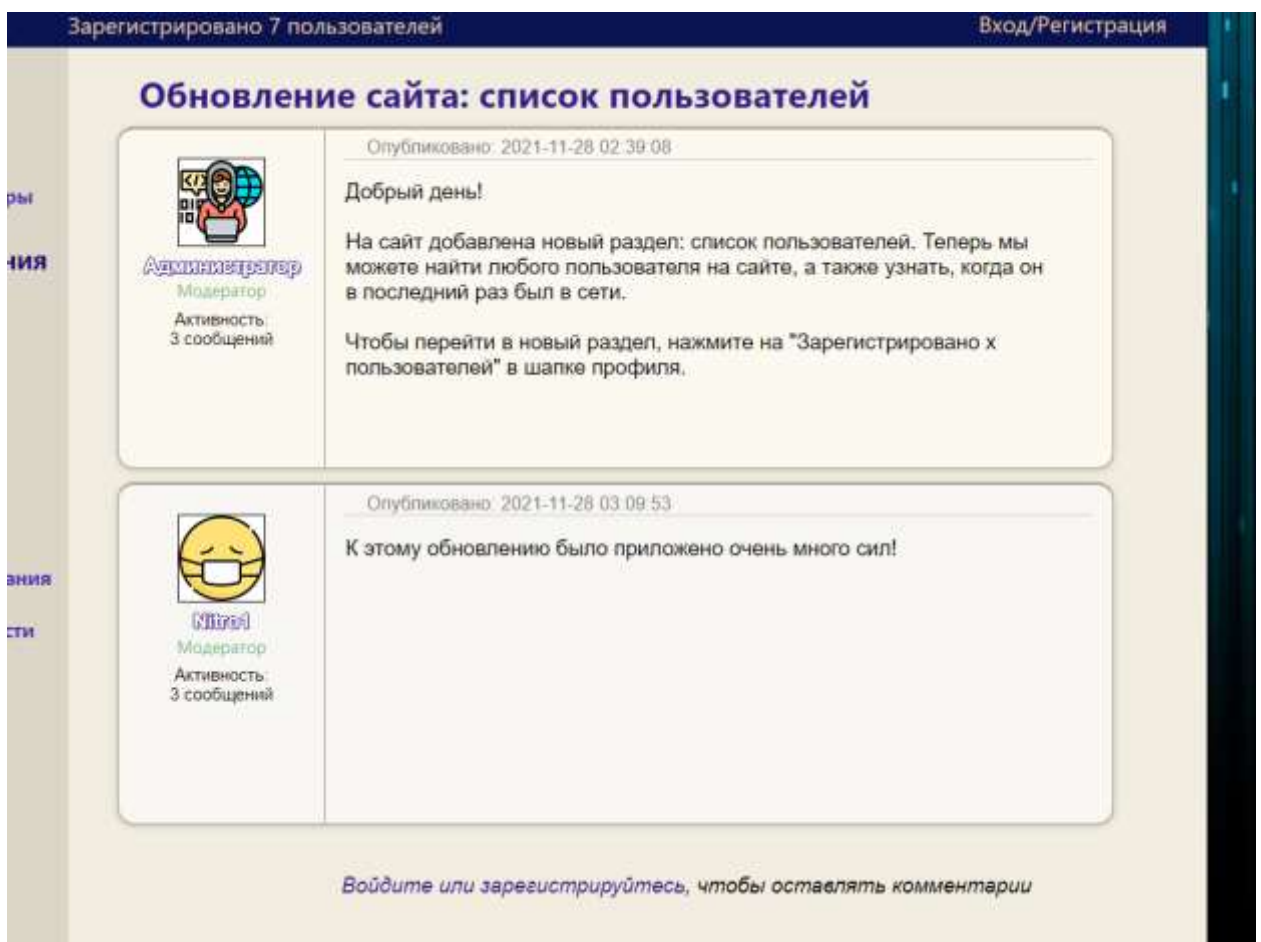


Рис. 4.46. Форма для отправки комментария отсутствует

На страницах пользователей мы можем только посмотреть их историю блокировок:



Рис. 4.47. Страницы пользователей для гостя

Зарегистрируем нового пользователя:

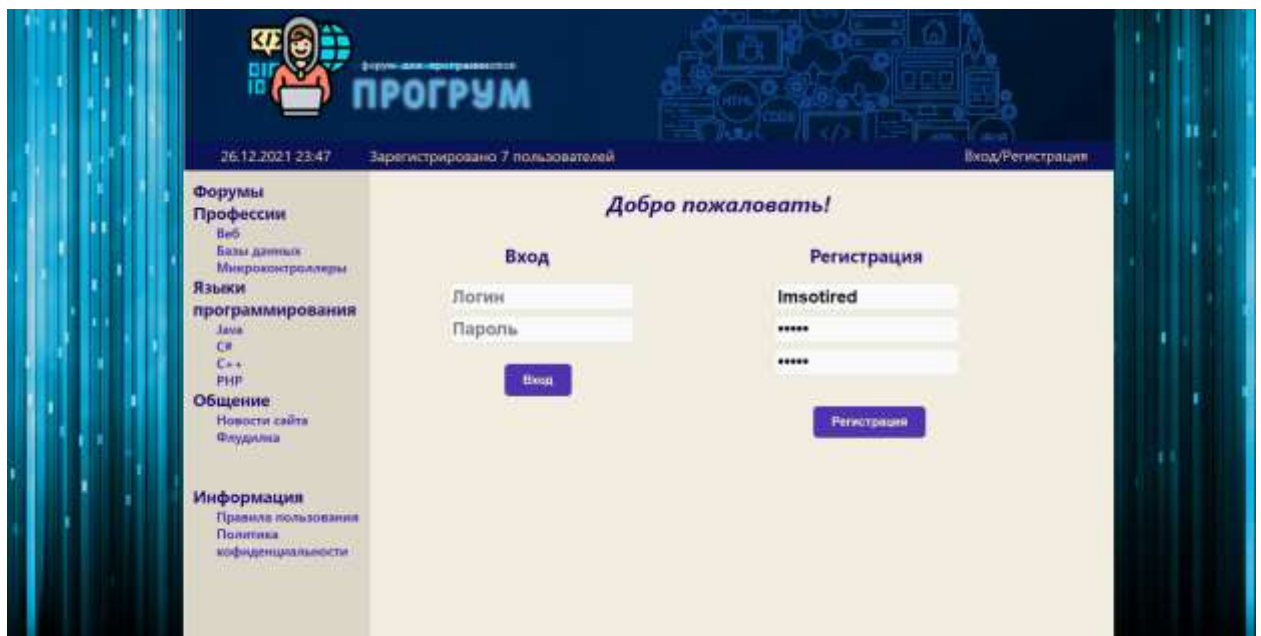


Рис. 4.48. Регистрация нового пользователя

Теперь счетчик пользователей в баннере увеличится и новый пользователь появится в списке пользователей:

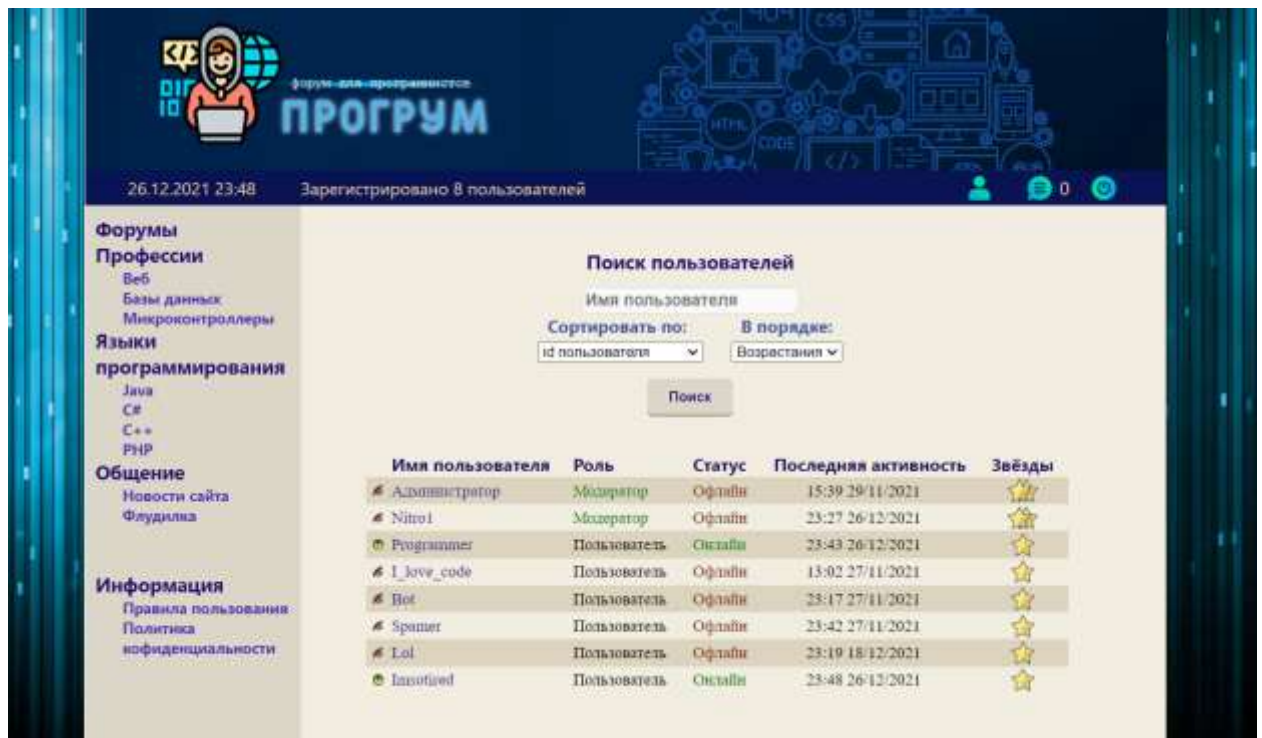


Рис. 4.49. Новый пользователь успешно добавлен

## **5 Заключение**

В результате было разработано веб-приложение, работающее с базой данных. Были реализованы все поставленные перед разработкой задачи, такие как разработка дизайна сайта, написание Frontend и Backend частей приложения, внедрение всех функций, необходимых для комфортного взаимодействия с сайтом, реализация всех возможностей для общения на форуме и в личных сообщениях.

Проделанная работа поспособствовала глубокому пониманию веб-разработки и работы с базами данных.

## 6 Список использованных источников

1. Руководство по PHP [электронный ресурс]  
<https://www.php.net/manual/ru/index.php> (дата обращения 11.11.2021)
2. Уроки PHP для начинающих с нуля! [электронный ресурс]  
[https://www.youtube.com/watch?v=GfHSbgyHN\\_I&list=PLDyJYA6aTY1m5zGQVcEYIoSFz2GD8u7cC](https://www.youtube.com/watch?v=GfHSbgyHN_I&list=PLDyJYA6aTY1m5zGQVcEYIoSFz2GD8u7cC) (дата обращения 20.11.2021)
3. CSS - MDN Web Docs [электронный ресурс]  
<https://developer.mozilla.org/ru/docs/Web/CSS> (дата обращения 28.11.2021)
4. Справочник по HTML [электронный ресурс]  
<http://htmlbook.ru/html> (дата обращения 28.11.2021)
5. PHP, MySQL и другие веб-технологии [электронный ресурс]  
<http://www.php.su/> (дата обращения 28.11.2021)