Дмитрий Приходько

PHP. Разработка модуля комментариев для сайта

6+

Дмитрий Приходько

PHP. Разработка модуля комментариев для сайта

http://www.litres.ru/pages/biblio_book/?art=67276092
SelfPub; 2022

Аннотация

В книге рассмотрен вариант разработки модуля комментариев для сайта на чистом языке php в процедурном стиле.

Содержание

Дмитрий Приходько PHP. Разработка модуля комментариев для сайта

О себе

Сначала немного представлюсь. Я кандидат технических наук. Специализация тепло, механика, энергетика. Разработка сайтов и программирование не являются для меня основным занятием, это просто хобби. В своё время изучал язык программирования Delphi, писал на нем программы, связанные с обработкой баз данных. На одно из разработанных мною приложений, написанных на Delphi, получил официальное свидетельство на программу для ЭВМ.

Благодарности

В первую очередь хотел бы поблагодарить всех web-программистов, занимающихся, кроме основной работы, еще и просветительской деятельностью. Большое Вам всем спасибо. Естественно хотел поблагодарить своих родителей, ведь без них этой книги бы не было. И конечно старших товарищей. Моего бывшего научного руководителя, доктора технических наук, профессора, Буслаева Виктора Федоровича и бывшего начальника и руководителя, доктора технических наук, профессора, Гальянова Ивана Васильевича.

Для кого написана эта книга

В первую очередь книга рассчитана на читателя, который как минимум, должен знать общий синтаксис PHP, а также иметь некоторый опыт использования языка на практике. Думающему, активному новичку в PHP, который хочет проверить себя и разобраться с возможностью реализации обратной связи между посетителями сайта при помощи языка сереверного программирования.

Для тех, кто вечно спешит и кому нужен просто модуль комментариев для их сайта, а не беллетристика. Они могут не читать, а просто воспользоваться кодом, приведенным в книге и начать им пользоваться.

Введение

Книга никоим образом не претендует на роль учебника по PHP и MySQL, а является в некотором роде решебником. Показывает путь решения задачи, вставшей перед разработчиком сайтов уровня любитель. Частный случай по оживлению силициума. В ней показано как с ноля можно решить конкретную прикладную задачу. Причем решить на языке PHP, языке серверном, сложность которого, несмотря на внешнюю простоту, в действительности вполне можно сравнить со сложностью дифференциальных уравнений. В книге основной упор сделан не на рассмотрение функций языка, с этим гораздо лучше справится мануал от разработчиков, а на логику решения задач, возникающих при написании кода.

Ну и собственно говоря, о самой задаче. Она встала передо мной на ровном месте и заключалась в следующем. В ходе проводимой исследовательской работы у меня возникла необходимость в своем личном блоге, в котором, в числе прочего можно было бы быстро обсудить с коллегами последние новости. Что можно было сделать?

1. Готовые решения

Первый и совершенно очевидный вариант: использование готовых CMS (Content Management System) т.е. программ управляющих содержимым сайтов. Ранее мне приходилось делать для организаций, в которых я работал сайты. Обычно использовал Joomla. Начинал ещё с её первой версии. Остались самые положительные впечатления. А первый свой сайт вообще написал в блокноте с использованием табличной верстки. И было это в те годы, когда про CMS ещё не слышали. Поэтому решил опять взяться за Joomla и одновременно посмотреть, может есть, что поинтереснее. Так вот. Изучение бесплатных систем управления сайтами по состоянию на 2019 год, меня честно говоря, обескуражило. Делюсь впечатлениями.

Joomla. Проста и дружелюбна для новичков, но последние версии стали использовать шаблоны, распухшие до неприличных размеров, что привело к сильному падению скорости работы сайтов. Бесплатность относительная. За действительно нужные модули просят денег. С бесплатными модулями есть постоянная проблема с обновлением CMS до последней актуальной версии.

Общее впечатление: при всех плюсах последней Joomla ложкой дегтя являются «современные» шаблоны. Неуклюжие и примитивные, с жутко медленной настройкой, при

этом едят кучу ресурсов, медленно стартуют и так же нехотя работают, чем напрочь отбивают желание пользоваться этой CMS.

Drupal. Крайне не дружелюбен к новичкам. Без хороших навыков программирования чуть более, чем бесполезен. Хотя для программистов очень даже неплох. Был. Насколько хорош он был в предыдущей седьмой версии, настолько же ухудшился в последней восьмой. На мой взгляд, восьмая версия Drupal это полный провал. Заметно тормозит даже в пустой сборке по умолчанию.

Wordpress. Вроде простой, вроде удобный. Но в нем только по рельсам. Добавить что-либо своё – это огромные затраты времени, совершенно не пропорциональные результату. Проще написать новую CMS, что кстати многие разработчики и делают, чем пытаться «прикрутить» к Wordpress что-то своё.

Maxisite. Бесплатная CMS, шаблоны которой мне понравились своей аккуратностью. Но использовать авторские CMS типа Maxisite – это значит потратить энное количество времени на то, чтобы разобраться с их кодом. Тоже не вариант. Разбираться с чужим кодом, это слишком долго.

Движок форума типа phpBB. Разворачивать на личном блоге форум, как-то слишком серьезно. Опять же потребуется довольно много времени на настройку под свои нужды.

2. Конкретизация задачи

Небольшое «лирическое отступление». Самое интересное по моему мнению в современной web разработке состоит в том, что скорость работы языка на котором в основном разрабатываются системы управления контентом, а именно PHP, от англ. PHP: Hypertext Preprocessor препроцессор гипертекста[6] постоянно увеличивается, а работа сайтов замедляется.

При этом контент не изменился. Как было на заре интернета, несколько десятков лет тому назад: текст, картинка, видео и звук. Так и осталось: текст, картинка, видео и звук. Только вот программы для их обработки распухли до монструозных размеров.

Таким образом, пришло решение использовать для блога простейший HTML и CSS шаблон, к которому подключать модуль комментариев.

И в очередной раз столкнулся с проблемой. Поиск таких модулей в интернете показал, что бесплатные или не устанавливаются или не работают, а рабочие стоят существенных денег. Поэтому решил совместить приятное с полезным, т.е. изучить язык программирования и заодно написать на нем свой модуль комментариев. Сразу встал вопрос: на каком языке писать? В данном случае вариантов было два. Javascript или PHP. Выбрал PHP, потому что его

код выполняется на стороне сервера и будет работать всегда и везде невзирая на настройки пользователя, в отличии от Javascript, выполнение команд которого пользователь может отключить. Итак, задача: разработать модуль комментариев для сайта на языке PHP.

Изучение литературы по данному вопросу показало, что отечественных книг по данной теме нет. Может я и ошибаюсь, можете меня поправить. Но в основном это справочники, в которых есть все обо всем и ни о чем конкретно. Список использованной литературы в конце книги. Из понравившихся, особо хотел бы отметить книгу Кевина Янка «PHP и MySQL. От новичка к профессионалу» [1] – отличное изложение и совершенно практичное применение, освоивший её, без проблем сможет написать свою первую CMS. Собственно говоря, именно освоение данной книги и сподвигло меня на самостоятельную разработку модуля. Из книг отечественных авторов не нашел ничего подходящего, зато могу отметить сайт Михаила Русакова [3] и Евгения Попова [4] пишут коротко и по делу, некоторые их материалы оказались очень полезны.

1. Подготовка
1.1 Выбор редактора кода

Первое с чем необходимо определиться, это где писать код. Редакторов кода много. Выбор есть. При этом сам по себе выбор редактора очень субъективен, потому что функционально они все практически одинаковы. Все зависит от личных предпочтений пользователя. Ниже очень краткий обзор редакторов, которые нравятся лично мне.

Notepad++. Да, я использую текстовый редактор Notepad ++. Самый быстрый редактор. Строг и лаконичен, в отличие от приведенных далее и которые больше похожи на детские книжки «разукрашки», а не на рабочие инструменты. На мой взгляд для разработки на PHP его возможностей более чем достаточно. Отлично подобранная цветовая гамма по умолчанию. Небольшая начальная настройка плюс прокачка плагинами и он становится не только самым быстрым, но и самым удобным редактором кода. Что касается меня, то когда я пишу код в нем я отдыхаю, когда пользуюсь другими редакторами я работаю. Он удобен как домашние тапочки, все просто и практично без лишней мельтешащей суеты. Официальный сайт https://notepad-plus-plus.org/. Установка проста. Скачиваете установщик, запускаете. Всё. По его настройкам в интернете есть очень много материалов.

Поэтому вкратце. В первую очередь настраиваем автозавершение. Идем в Опции->Настройки->Автозавершение. Ставим галочки. Теперь у нас будут автоматически завершаться выбранные парные теги, слова и функции.

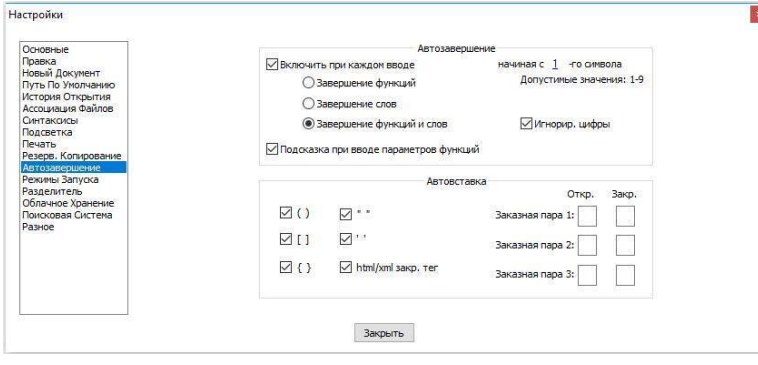


Рис. 1 Настройка автозавершения в Notepad++.

Важно! Сразу устанавливаем кодировку файлов в редакторе Utf-8 без BOM. Для этого открываем вкладку «Кодировки» и устанавливаем нужную: Utf-8.

Далее устанавливаем плагины. Для этого переходим на вкладку плагины и жмем кнопку управление плагинами. Выбираем нужный и устанавливаем. Всё интуитивно понятно.

Из полезных на мой взгляд это «Snippets», который позволяет вставлять готовые блоки кода HTML, CSS и PHP и плагин «TextFx» для обработки текста. Подробные описания

плагинов и их установка есть в интернете, поэтому останавливаться на этом не буду. Скриншот с плагинами, установленными на своем редакторе прилагаю.

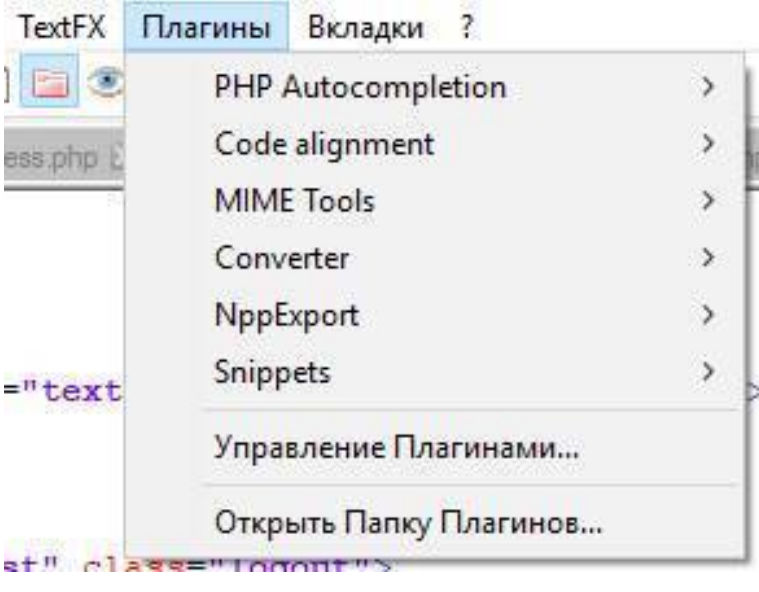


Рис. 2 Список установленных плагинов в Notepad++.

2. Sublimetext 3. Прост в установке и настройке, не критичен к ОС, запускается даже на Windows XP, но на мой взгляд уступает в скорости Notepad. По возможностям аналогичен Notepad. Официальный сайт https://www.sublimetext.com/3 .

В целом очень неплох, но только после прокачки. Однако рекомендовать его не могу. Слишком уж он ненадежен, «глючит» на ровном месте. После того как он перестал запускаться на четвертом подряд компьютере, я решил что жизнь слишком коротка, чтобы тратить время на выяснение, а почему же не работает Sublimetext 3. Очень сырой продукт. Привел его здесь в качестве антипримера.

3. VS Code. Хорошая машинка, хотя заметно уступает в скорости вышеприведенным редакторам. Если Notepad++ можно использовать сразу из коробки и все будет работать замечательно и быстро, то VS Code как и Sublimetext необходимо настраивать. Причем VS Code достаточно капризен, не ставится на старые версии Windows и на его настройку под себя, гарантированно потратите гораздо больше времени, чем планировали. По моему мнению, под простые проекты использовать его не имеет смысла. Зато очень радует красивая оболочка, на которую вполне можно медитировать, забыв о работе и настраивая всё и вся. Официальный сайт https://code.visualstudio.com/ .

Первый и последний редакторы без проблем одолеют любой написанный код. Выбор за вами.

1.2 Справочник языка

PHP

Далее нам понадобится справочник языка PHP от разработчиков. Он доступен в разных вариантах. Я предпочитаю локальный файл справки в формате chm. Эта справ-

ка будет доступна даже при отключении от сети. Для получения справочника заходим на сайт разработчиков https://www.php.net/ переходим на вкладку «Documentation». Ищем на этой странице раздел «Downloads». Переходим по приведенной там ссылке на страницу скачивания документации https://www.php.net/download-docs.php . Выбираем «HTML Help file» – «Russian» в формате chm. Скачиваем. Переходим в место расположения загруженного файла. Жмем правую кнопку мыши. В контекстном меню выбираем: «Свойства», затем «Общие». Жмем кнопку «Разблокировать». Всё. Теперь можно пользоваться справочником по языку программирования PHP на русском языке.

Внимание: ответы На большинство вопросов, которые связаны с пониманием работы кода вы найдете именно в этом справочнике.

1.3 Сервер

Разработка ведется локально, поэтому нужен сервер. Здесь все просто: Open Server. В нем уже встроено все, что нужно в данной ситуации. А именно: Apache, PHP, MySQL, phpMyAdmin и отправка писем с сервера. Поэтому идем на сайт разработчика https://ospanel.io/. Качаем нужную версию. Устанавливаем. Запускаем. Щелкаем на значке «Open Server» в виде флажка и переходим на вкладку «Настройки» затем «Модули». Выбираем нужную версии языка PHP желательно самую последнюю, она будет зависеть от версии Windows установленной на вашем компьютере, выбираем модуль Apache, программы, которая позволяет пользователю просматривать веб-документы, совместимый с версией PHP и нужный модуль базы данных MySQL.

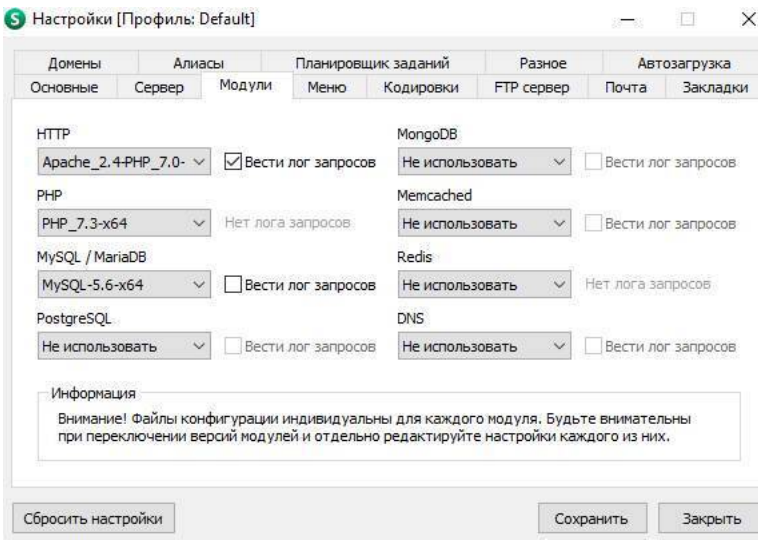


Рис. 3 Настройки модулей Open Server

Итак, выбран редактор кода, есть справочник по PHP и развернут локальный сервер, на котором включены необходимые модули. Осталось определиться с отладчиком.

1.4 Отладчик

Для отладки использовалась доработанная функция dumper(), предложенная в книге [2.С.225]. Моя доработка заключалась в замене функции each(), которая не поддерживается в версиях языка PHP выше 7.2, на цикл foreach. Скрипт с функциями размещаем в отдельном файле dumper.php и будем подключать в шапке «проблемной» страницы для вызова функции dumper().

Листинг 1. dumper.php

```
<?php
// Функция для вывода содержимого переменной
// Распечатывает дамп переменной на экран
function dumper($obj)
{
echo
"<pre>",
htmlspecialchars(dumperGet($obj)),
"</pre>";
}

// Возвращает строку – дамп значения переменной в древовидной форме
// (если это массив или объект). В переменной $leftSp хра-
```

нится

```
// строка с пробелами, которая будет выводиться слева от текста.
function dumperGet(&$obj, $leftSp = "")
{
if (is_array($obj)) {
$type = "Array[" . count($obj) . "]";
} elseif (is_object($obj)) {
$type = "Object";
} elseif (gettype($obj) == "boolean") {
return $obj ? "true" : "false";
} else {
return "\"$obj\"";
}
$buf = $type;
$leftSp .= " ";
    foreach ($obj as $k => $v) {
    Reset($obj);
if ($k === "GLOBALS") {
continue;
}
$buf .= "\n$leftSp$k => " . dumperGet($v, $leftSp);
}
return $buf;
}
```

Данная функция выводит содержимое любой, сколь угод-

но сложной переменной, будь то массив, объект или простая переменная, в гораздо более удобном виде чем стандартные print_r() или var_dump().

После того как я некоторое время поработал с ней, то согласился с мнением ее разработчиков, в том, что при отладке она действительно хороша.

Ложим файл dumper.php в корень папки chat.

Скорее всего при работе с книгой рано или поздно у вас возникнут сложности с пониманием того как действует тот или иной код. Поэтому советую создать пустой файл php и назвать его допустим test.php. В шапке этого файла подключить данную функцию при помощи инструкции include. Разместить test.php в корне сайта и использовать как страницу отладки непонятного кода. Отладку делать эмпирическим путем. Вставляя непонятный код и запуская страницу для проверки того, что этот код делает, ну или не делает.

Проект относительно небольшой, поэтому с моей точки зрения, это гораздо удобнее и нагляднее, да и полезнее, чем использовать XDebug или отладчик VS Code.

Постановка задачи

2.1 Требования к

модулю

Определимся с требованиями к модулю. Модуль должен:

Устанавливаться на любой сайт.

Использовать базу данных.

Пользоваться модулем могут только зарегистрированные

пользователи.

Корректно выглядеть не нарушая основной дизайн сайта.

2.2 Предварительная логика работы

Оцениваем варианты выполнения раздела 2.1 по пунктам.

Установка:

а) Для того чтобы модуль мог работать на любых сайтах в первую очередь необходимы настройки файла .htaccess. Дело в том, что файлы модуля будут включаться в тело HTML файлов. А некоторые серверы не обрабатывают PHP код, внедренный в HTML документы. Решаем созданием файла .htaccess с нужной командой.

б) В связи с тем, что многие файлы будут подключаемыми необходимо указать корректные пути к этим файлам.

в) для простоты установки модуль должен быть выполнен в виде папки с файлами, и устанавливаться простым копированием этой папки в корень сайта.

База данных:

Модуль должен автоматически подключаться к базе данных и далее работать с ней. При отсутствии базы данных он должен её создать, а также автоматически создать в этой базе таблицы необходимые для работы. Решаем с помощью запросов к MySQL.

Доступ:

а) Необходима авторизация пользователей. Регистрация пользователей должна исключать регистрацию ботов. Поэтому модуль должен содержать минимальную защиту от подоб-

ных регистраций. Решаем с помощью капчи и подтверждения регистрации по адресу электронной почты.

б) У администратора модуля должна быть возможность управления пользователями и комментариями. Решаем созданием административного раздела модуля.

4. Внешний вид:

а) Элементы модуля должны иметь свой стиль. Решаем в файле style.css модуля.

б) чтобы не сильно влиять на дизайн сайта элементы регистрации, авторизации и элементы непосредственного вывода комментариев необходимо разделить. Решаем созданием раздельных контроллеров для элементов авторизации и печати.

2.3 Реализация логики

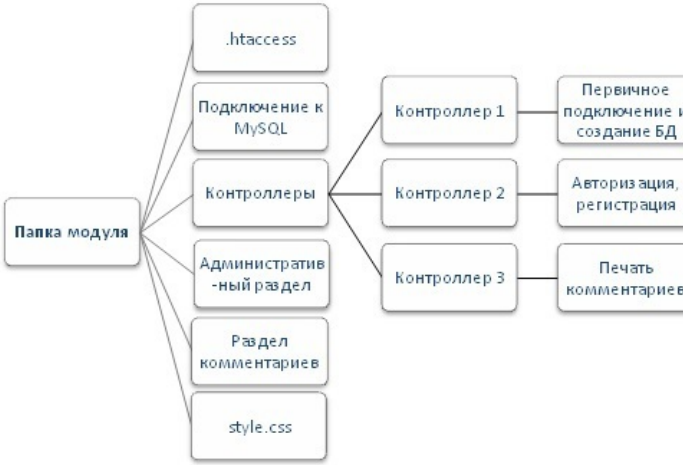


Рис. 4 Блок схема модуля,

Учитывая вышеизложенное, изобразим полученные результаты графически. Результат представлен на рис.3. Модуль комментариев будет представлять из себя папку, содержащую блоки кода в виде файлов и вложенных папок, указанных на рисунке 3.

2.4 Подключения, корень сайта

Как видно файлов достаточно много даже при самом схематичном рассмотрении. возникает вопрос, а как же они будут друг с другом взаимодействовать? Совершенно очевидно, что придется делать ссылки и вставки частей кода друг в друга. Возникает следующий вопрос: если модуль планируется подключать неизвестно к чему, т.е. не известен сайт, неизвестен хостинг, неизвестны директории, то как с этим неизвестно, чем взаимодействовать? Ведь необходимо прописывать пути. Каким образом? Немного теории.

Любой сайт существует в двух измерениях[5]: реальном и виртуальном. Для всех посетителей – это виртуальный веб-сервер. На котором нет файлов, а есть виртуальные адреса URI. Если вы видите строку в браузере http://site.ru/ufo/phantom.html – это не файл. Это URI, виртуальный адрес. Никакого файла с именем phantom.html на сервере может вообще не быть. Вполне вероятно, что и папки ufo там тоже нет, а все URL адреса обрабатываются одним единственным PHP файлом. И браузер работает именно с виртуальными адресами. Браузер не может видеть реальную файловую структуру сервера. Он не видит никаких дисков C, D, E и т.д. он видит только URL адреса, а не файлы.

Для разработчиков, сайт – это программа, выполняющаяся на совершенно конкретном реальном компьютере. С

совершенно конкретным жестким диском, директориями и файлами. И скрипт, работая со своими данными, подгружая другие скрипты, работает именно с реальными файлами, на реальном физическом диске.

Еще раз: ПУть к файлу скрипта и виртуальный адрес этого скрипта для просмотра в браузере – это не одно и то же

Как достучаться до файла? Вариантов два: абсолютный и относительный путь.

Если путь указывается от корня системы, то это путь абсолютный. Абсолютный путь в PHP – это полный путь к папке или файлу. Вот пара примеров для разных операционных систем:

- C:\OpenServer\domains\test.ru\index.php – для Open Server на Windows
- /var/www/html/test.ru/index.php – для Ubuntu

Как видим, это полный путь от корня диска до конкретного файла или папки.

Виртуальный адрес этого скрипта при просмотре через браузер, будет:

http://www. test.ru\index.php

Для браузера это самый полный путь, который только может быть. Он начинается от корня сайта.

В юникс-системах и на веб сайтах корень обозначается косой чертой "/", в Windows начинается с буквы диска.

Относительный путь – это путь без указания корня.

У относительных путей в PHP есть один недостаток – они строятся по своей логике, которую не интересуют наши желания. В результате этого нужный файл зачастую просто не обнаруживается. Происходит это потому что когда мы подключаем скрипт по относительному пути, например include 'ufo.php', то сначала PHP попытается найти этот файл в папках, указанных в директиве include_path, затем будет искать файл в папке, в которой находится подключающий скрипт, и под конец попытается найти файл в папке текущего рабочего каталога.

Что это значит на практике. Допустим у нас есть в корне сайта две папки chat и ufo, и в папке ufo у нас есть скрипт phantom.html. Так вот, при помощи относительного пути вставить наш phantom.html в файлы папки chat не получится. От слова никак не получится. Почему? Попробую привести аналогию с движением автомобиля.

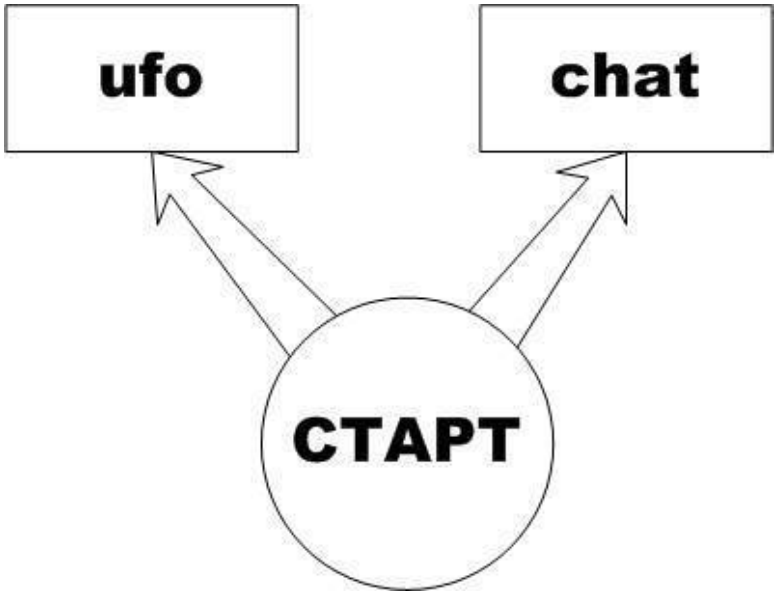


Рис. 5 относительный путь – дорога в одну сторону, где СТАРТ – корень сайта.

Движение из точки старт возможно только в одном направлении. Или налево или направо. Мы не можем взять предметы в месте chat и затем отвезти их в место ufo. Если автомобиль выезжает из позиции старт в направлении ufo и по прибытии в ufo обнаружится, что нужные предметы для этого места отсутствуют, то все. Мы оказались у разбитого

корыта. Вернуться назад за нужными предметами мы не можем. Движение одностороннее.

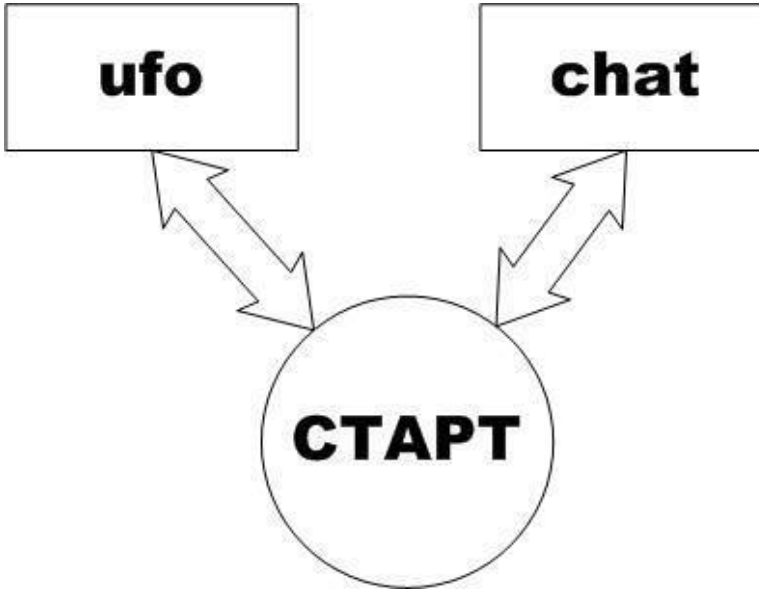


Рис. 6 Абсолютный путь – двустороннее движение, где СТАРТ – корень сайта.

При абсолютном же пути движение двустороннее и мы всегда можем вернуться обратно на старт. Отправится в ufo.

Узнать, что там необходимо. Сделать крюк допустим до места chat и вернуться обратно уже с полным пакетом всего что надо. Если и сейчас что-то забыли, не вопрос. Можно опять съездить куда надо и вернуться. Т.е. абсолютный путь дает спокойное и предсказуемое поведение в точках назначения.

Поэтому использовать полностью относительные пути в PHP я бы не советовал вообще. Целесообразнее определить с помощью PHP корневую директорию веб-сервера, а местоположения файлов указывать относительно ее.

Значит остается абсолютный путь. Применительно к решению данной задачи есть два варианта по созданию абсолютных путей:

Вариант А: использовать константу __DIR__.

__DIR__ константа для получения абсолютного пути к папке, это директория файла. Если используется внутри подключаемого файла, то возвращается директория этого файла. Это эквивалентно вызову dirname(__FILE__). Возвращаемое имя директории не оканчивается на слеш, за исключением корневой директории[6].

Применительно к нашей задаче данная константа не совсем удобна, зато будет работать везде. Неудобство будет заключаться в том, что в первую очередь нас интересует путь к корню сайта, а поскольку мы не знаем заранее его название, то придется использовать обходной способ. И этот способ будет работать только когда мы будем знать названия папок из которых будут вызываться include. Заключается способ в

том, что на тех страницах, где будут вызываться include прописываются папки для поиска загружаемых файлов. Это делается при помощи функции set_include_path которая задает значение настройки конфигурации include_path на время выполнения скрипта.

Конфигурационная директива include_path указывает список директорий, в которых функции require, include, fopen(), file(), readfile() и file_get_contents() ищут файлы. Формат соответствует формату системной переменной окружения PATH: список директорий, разделенных двоеточием в Unix или точкой с запятой в Windows[7].

При поиске подключаемых файлов PHP отдельно рассматривает каждое значение в include_path. Он проверяет первый путь, если файл в нем не найден, то он переходит к следующему, и так до тех пор, пока не найдет подключаемый файл, либо вернет E_WARNING или E_ERROR.

Проще говоря, необходимо указать папку, в которой будут искаться подключаемые файлы.

Таким образом, чтобы подключить файлы, в начале страницы, где требуется включение, будет нужно прописать код:

Листинг А. Получение пути к корню сайта. Размещается вначале страницы.

```
<?php
$p = explode('ufo', __DIR__);
echo $p[0];
```

```
set_include_path(get_include_path() .
PATH_SEPARATOR . $p[0]);
```

где ufo имя папки из которой вызывается файл. В примере указано ufo, но по факту любое.

Для сайта news и папки ufo в константе __DIR__ будет такой путь:

C:\OSPanel\domains\news\ufo

При помощи функции explode этот путь делится на две части. Разделителем будет служить название каталога, в данном примере ufo. В итоге в $p[0] будет содержаться левая часть:

C:\OSPanel\domains\news\

т.е. путь к корню сайта.

И теперь можно подключать файлы из любых каталогов следующим образом:

include "chat/нужный скрипт"; или include "ufo/нужный скрипт";. Мне кажется это довольно неудобно, хотя и будет работать везде. Поэтому рассмотрим вариант Б.

Вариант Б: использовать $_SERVER['DOCUMENT_ROOT'].

'DOCUMENT_ROOT' это директория корня документов, в которой выполняется текущий скрипт, в точности та, которая указана в конфигурационном файле сервера[mnl]. Значение этой переменной дает путь к корню сайта. И подключение файла будет выглядеть так:

include $_SERVER['DOCUMENT_ROOT']."/нужный файл";

Как мне кажется, вариант Б более удобен, поэтому в дальнейшем все подключения будут делаться через $_SERVER['DOCUMENT_ROOT'].

3. Создание тестового сайта

Ну, что ж приступим. Для упрощения разработки модуля на этапе разработки, воспользуемся не готовым шаблоном, а создадим свой простейший сайт одностраничник. Для этого переходим в папку с установленным «Open Server». По умолчанию она находится по пути: C:\OSPanel. Далее переходим в папку «domains» и создаем там тестовый отладочный сайт. Для этого создадим в папке «domains» подпапку. Название выбирайте какое хотите. Название этой папки будет именем вашего сайта. Я назвал её news. Затем открываем редактор и создаем в нем текстовый документ со следующим кодом:

Листинг 2. index.html Путь: news/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Test site</title>
</head>
<body>
<p>Значимость этих проблем настолько очевидна, что начало повседневной работы по формированию позиции способствует подготовке и реализации новых предложений!
```

Значимость этих проблем настолько очевидна, что повышение уровня гражданского сознания влечет за собой процесс внедрения и модернизации модели развития. Практический опыт показывает, что рамки и место обучения кадров способствует повышению актуальности соответствующих условий активизации.
</body>
</html>

Сохраняем файл в папке news как index.html. Теперь у нас есть стартовая страница тестового сайта. В тегах содержится обычный текст заглушка. Стили к данной странице применять не будем.

Запускаем «Open Server». В разделе мои сайты должен появиться новый сайт news. Запускаем его. Если все сделано правильно, в браузере должен появиться сайт «news» и на нем текст заглушка.

Так как дальнейшая работа предполагает работу с PHP, создаем в редакторе в этой же папке «news» файл .htaccess со следующим содержимым

Листинг 3. .htaccess Путь: news/.htaccess

```
#Включаем обработку php в html файлах
AddType application/x-httpd-php .php .htm .html
#Устанавливаем кодировку по умолчанию
AddDefaultCharset utf-8
```

Сохраняем в корне сайта, т.е. в папке «news».

При сохранении в меню «Файл» выбрать «Сохранить как».

В выпадающем списке «Тип файла» выбрать «Все файлы».

Ввести в качестве имени .htaccess.

Нажать на кнопку «Сохранить».

Файл .htaccess (англ. hypertext access) используется для настройки веб-сервера на котором хранится сайт пользователя. А меняя настройку веб-сервера, мы влияем на работу сайта. В данном случае серверу дается команда обрабатывать PHP код в HTML документах и устанавливать по умолчанию кодировку «UTF-8»

Создаем папку модуля комментариев. Ее можно создать в любом месте, но разместить в корне сайта. Назовем папку «chat». В этой папке создадим две подпапки:

admin – для файлов администрирования;

say – для файлов управления комментариями.

В этой же папке chat создадим файл style.css для того чтобы настраивать внешний вид элементов модуля. Пока этот файл будет пустой. Должно получиться как на рис. 7, 8.

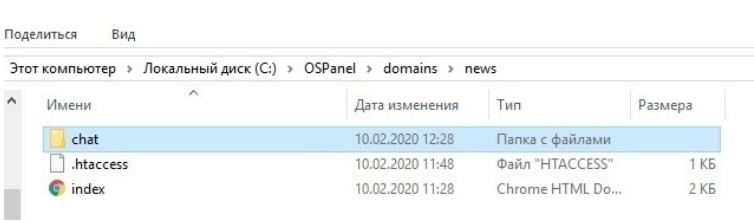


Рис. 7 Содержание тестового сайта news.

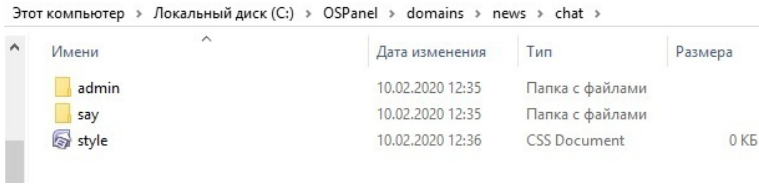


Рис. 8. Содержание папки chat сайта news.

Комментарии без смайлов, как-то не интересно. Поэтому добавим праздника. Смайлов. Для этого в папке «say» создаем подпапку «smiles» в которую загружаем набор смайлов. В этом модуле загружен набор «Колобки» в формате GIF. Вы можете загрузить любые другие смайлы. Папка «smiles» будет проверяться на наличие изображений при первом запуске модуля и в случае если в ней будут обнаружены изображения, то они автоматически будут обработаны в качестве

смайлов.

OSPanel › domains › news › chat › say › smiles


acute


bad


be


book


bye


cla


cray


dance


d

Рис. 9. Содержимое папки smiles.

На рисунке 7 представлена структура тестового сайта на данном этапе.

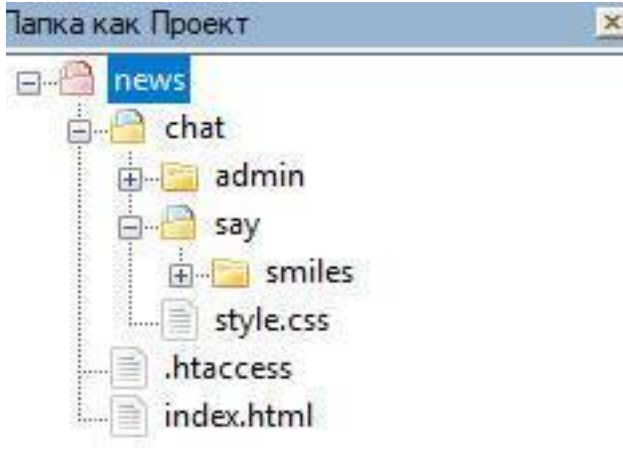


Рис. 10. Содержание тестового сайта news.

Таким образом, у нас получился рабочий тестовый сайт с размещенным на нем модулем комментариев с набором смайлов.

База данных

Теперь необходимо определиться для чего нам нужна база данных (далее БД), с ее системой управления и с тем,

что в ней будет храниться. Я выбрал систему управления БД MySQl. Во-первых потому что ранее уже приходилось с ней работать, во-вторых, это самая распространенная на сегодня БД и на нее есть огромное количество справочной информации. Итак: MySQl – система управления реляционными, т.е. представленными в виде таблиц, базами данных (далее СУБД). Что в ней хранить?

В контексте задачи храниться будут:

Комментарии.

Ответы на комментарии.

Данные пользователей.

Данные о странице на которой размещаются комментарии.

Смайлы (без смайлов я комментарии не представляю).

При создании таблиц используем правило: каждый объект должен располагаться в отдельной таблице.

Объекты будут следующие: комментарии, ответы на комментарии, пользователи, права пользователей, смайлы. Еще одна таблица будет связывать права пользователя с данными в других таблицах.

Для хранения комментариев создаем таблицу «say». Таблица будет содержать следующую информацию о комментарии:

id – уникальный номер комментария;

saytext – текст комментария;

userid – идентификатор пользователя;

page_id – идентификатор страницы;

saydate – дата добавления комментария;

Для ответов на комментарии создаем таблицу «reply». Таблица будет содержать следующую информацию.

id – уникальный номер ответа на комментарий;

replytext – текст комментария;

userid – идентификатор пользователя;

replyid – идентификатор комментария;

saydate – дата добавления комментария;

Мне кажется в этих двух таблицах все понятно, вопросов быть не должно.

Для администрирования создаем три таблицы: «users», «authorrole», «role».

Таблица «users», будет содержать следующие данные:

id – уникальный номер пользователя;

login – логин пользователя;

password – пароль пользователя;

email – адрес электронной почты пользователя;

img – адрес расположения аватара пользователя;

activation – проверка активации пользователя;

date – дата регистрации пользователя.

Для идентификации автора комментария, поле «id» таблицы «users» будет в запросах к бд связываться с полем «userid» таблиц «say» и «reply»

Таблица «role» будет содержать следующие данные:

id – роль (права) пользователя;

description – описание прав пользователя;

Таблица «authorrole» – промежуточная и будет содержать следующие данные:

authorid – id пользователя;

roleid – роль (права) пользователя;

Таблица «authorrole» связывает пользователей «users» и их права «role»

Для смайлов создаем таблицу «smiles». Таблица будет содержать следующие данные:

id – id смайла;

smile – условное обозначение смайла;

path – путь к смайлу.

Пишем файл, создающий базу данных и необходимые таблицы. Назовем этот файл «createbase.php» и разместим в папке «admin». Сохраняем в формате php.

Листинг 4. createbase.php Путь: news/admin/createbase.php

```
<?php
error_reporting(E_ALL);//включаем вывод ошибок

/* 1. Создаем переменные подключения */

$host = "localhost"; /* Имя хоста */

$root = "root"; /* Имя пользователя БД */
```

```
$root_password = ""; /* Пароль к БД */

$db = "beseder"; /* название БД */

/* 2. Создаем базу данных */
try {
$dsn = new PDO("mysql:host=$host", $root, $root_password);
$dsn->exec("CREATE DATABASE IF NOT EXISTS `$db` CHARACTER SET utf8 COLLATE utf8_general_ci;");
echo 'База создана (OK!)<br>' . ' Имя БД: ' . '<b>' . $db . '</b>' . '<br> Пользователь: ' . '<b>' . $root . '</b>';
} catch (PDOException $e) {
echo $e->getMessage();
echo $e->getLine();
exit('Ошибка при создании базы');
}

$dsn = null;//обнуляем подключение

/* 3. Формируем запросы для создания таблиц */

try {
$dsn=newPDO("mysql:host=$host;dbname=$db", $root,root_password);//подключаемся к созданной БД
```

```
   $dsn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

   /* Нумерация страниц */

   $sql_page = "CREATE TABLE IF NOT EXISTS page (
   id INT(11) AUTO_INCREMENT PRIMARY KEY NOT
NULL,
   sayid int(11),
   userid int(11),
   pageid text
   )";

   //пользователи

   $sql_users = "CREATE TABLE IF NOT EXISTS users (
   id int(11) AUTO_INCREMENT PRIMARY KEY NOT
NULL,
   login VARCHAR(15) NOT NULL,
   password VARCHAR(256) NOT NULL,
   youtext text NOT NULL,
   email text NOT NULL,
   img text NOT NULL,
   activation int(11),
   date int(11)
   )";
```

```
//роль автора определение

$sql_role = "CREATE TABLE IF NOT EXISTS role (
id VARCHAR(255) NOT NULL PRIMARY KEY,
description VARCHAR(255)
)
DEFAULT CHARACTER SET utf8 ENGINE=InnoDB";

//роль автора id
$sql_authorrole = "CREATE TABLE IF NOT EXISTS authorrole (
authorid INT NOT NULL,
roleid VARCHAR(255) NOT NULL,
PRIMARY KEY (authorid, roleid)
)
DEFAULT CHARACTER SET utf8 ENGINE=InnoDB";

$sql_roledesc = "REPLACE INTO role (id, description) VALUES
('user', 'Контроль своих комментариев'),
('admin', 'Full control'),
('Site Administrator', 'Контроль комментариев')";

$sql_userole = "REPLACE INTO authorrole (authorid, roleid) VALUES
```

```
(1, 'admin')";

/* Для комментариев */

$sql_say = "CREATE TABLE IF NOT EXISTS say (
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
saytext TEXT,
userid int(11),
saydate int(11)
) DEFAULT CHARACTER SET utf8 ENGINE=InnoDB";

/* Для ответов на комментарии */

$sql_reply = "CREATE TABLE IF NOT EXISTS reply (
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
replytext TEXT,
userid int(11),
replyid int(11),
replydate int(11)
) DEFAULT CHARACTER SET utf8 ENGINE=InnoDB";

/* 4. Создаем таблицы */

$dsn->exec($sql_users);
```

```
$dsn->exec($sql_authorrole);
$dsn->exec($sql_role);
$dsn->exec($sql_roledesc);
$dsn->exec($sql_userole);
$dsn->exec($sql_say);
$dsn->exec($sql_page);
$dsn->exec($sql_reply);
} catch (PDOException $e) {
echo $e->getMessage();
}
/* 5. Смайлы */
try {
$sql_smile = "CREATE TABLE IF NOT EXISTS smiles (
id INT(11) AUTO_INCREMENT PRIMARY KEY NOT NULL,
smile text,
path text)";
$dsn->exec($sql_smile);
} catch (PDOException $e) {
echo $e->getMessage();
echo $e->getLine();
exit();
}
$dir = $_SERVER['DOCUMENT_ROOT'].'/chat/say/smiles/';//строим путь к папке smiles
```

```
$files1 = preg_grep('~\.(jpeg|jpg|png|gif)$~', scandir($dir));//делаем массив из картинок в папке smiles

try {
$sql = 'INSERT INTO smiles SET
smile = :smile,
path = :path';

$s = $dsn->prepare($sql);

foreach ($files1 as $val) {
$smile = pathinfo($val, PATHINFO_FILENAME); //получаем путь к смайлу
$smile = str_replace($smile, ":$smile:", $smile); //делаем условное обозначение смайла
$path = '/chat/say/smiles/'.$val;//строим наш путь к смайлу

$s->bindValue(':smile', $smile);
$s->bindValue(':path', $path);

$s->execute();
}
} catch (PDOException $e) {
echo $e->getMessage();
echo $e->getLine();
exit();
```

```
}
echo '<br>'.'Все таблицы успешно созданы';
```

В этом файле мы сначала включаем вывод всех сообщений об ошибках кодом: error_reporting(E_ALL). Это необходимо на этапе разработки, чтобы иметь возможность сразу выявлять и обрабатывать ошибки. После отладки всего приложения данную строку необходимо будет или удалить или закомментировать.

Затем подключаемся к СУБД MySQL. Для подключения необходимы следующие параметры:

Имя хоста: $host = "localhost";

Имя пользователя СУБД: $root = "root";

Пароль к СУБД: $root_password = "";

Для подключения к БД:

Название БД: $db = "beseder",

Первые три значения по умолчанию. На реальном сайте предоставляются «хостером». Четвертый параметр это название для создаваемой новой базы данных. В ней будут хранится комментарии, ее название придумывается самостоятельно. У меня БД названа «beseder».

Далее подключаемся к СУБД MySQL при помощи PDO (PHP Data Objects), PDO определяет интерфейс для доступа к базам данных в PHP[manual]. Подключение выполняется согласно мануала PDO с явным перехватом ошибок в блоке catch[7].

Внимание! далее все подключения к бд будут выполнятся только с использованием блоков **try** и **Catch**

При успешном создании БД выводится сообщение: 'База создана (OK!)Имя базы $db'. После создания базы закрываем подключение.

Снова открываем подключение, но теперь коннектимся не к СУБД, а к созданной БД. Создаем и затем выполняем SQL запросы на создание таблиц.

Типы полей для хранения данных принимаем следующие:

– для чисел «int»;

– для текста «text»;

– для коротких строк «varchar».

Смайлы, выделены в отдельный блок. Принцип работы следующий, проверяем папку «smiles» на наличие картинок:

```
$files1 = preg_grep('~\.(jpeg|jpg|png|gif)$~', scandir($dir));
```

где preg_grep возвращает массив, состоящий из элементов входящего массива, которые соответствуют заданному шаблону, в данном случае из папки «smiles» возвращаются только «картинки».

Создаем в цикле условные обозначения для смайлов:

```
$smile = pathinfo($val, PATHINFO_FILENAME);
$smile = str_replace($smile, ":$smile:", $smile);
```

где pathinfo – возвращает информацию о пути к файлу,

str_replace (что меняем, на что меняем, где меняем) – заменяет все вхождения строки поиска на строку замены;

и строим пути к картинкам:

$path = '/say/smile/'.$val;

При успешном завершении кода выводим сообщение:

echo '<br>'.'Все таблицы успешно созданы';

Запускаем сайт. Добавляем после «news» в адресную строку /chat/admin/createbase.php. Обновляем страницу. Должно получиться следующее.

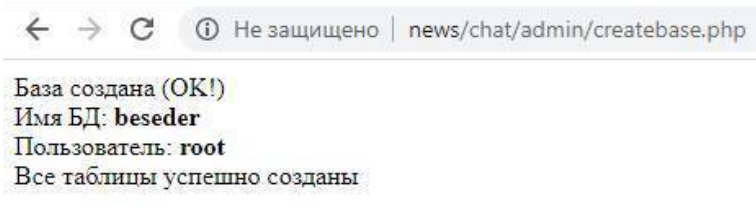


Рис. 11 Результат работы файла createbase.php

Заходим в phpMyAdmin и проверяем. В списке БД должна появится новая база «beseder».

phpMyAdmin находится на вкладке «дополнительно» «Open Server».

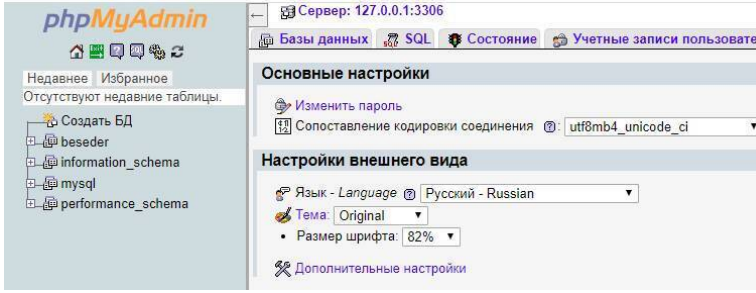


Рис. 12. Проверяем наличие созданной БД

Файл работает. Для интереса, можете посмотреть, какие таблицы созданы.

На данном этапе БД не нужна, поэтому удаляем. В phpMyAdmin:

– выбираем базу «beseder»;

– жмём кнопку «Операции»;

– жмем кнопку «Удалить базу данных (DROP)».

Для того чтобы выполнять удаление не входя в phpMyAdmin средствами PHP, создаем файл «drop» и размещаем его также в папке «admin». Теперь можно будет удалять базу переходя по адресу: news/chat/admin/drop.php

Листинг 5. drop.php Путь: news/chat/admin/drop.php

```
<?php
```

```
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/
dsn.php';
try {
$sql = 'DROP DATABASE beseder';
$s = $dsn->exec($sql);
} catch (PDOException $e) {
echo $e->getMessage();
exit('Ошибка подключения к базе данных');
}
header('Location: /');//переходим на главную страницу
```

В этом скрипте мы сначала подключаемся к нашей базе данных beseder, а затем выполняем SQL запрос DROP DATABASE удаляющий все таблицы в указанной базе данных и саму базу. В результате выполнения данного оператора база удаляется полностью.

*файл подключения к БД dsn.php будет рассмотрен в следующем разделе.

5. Файл подключения к базе данных

Теперь нужен файл, при помощи которого можно будет создавать подключения к БД. В корне папки «chat» создаем файл «dsn.php» со следующим содержимым:

Листинг 6. dsn.php Путь: news/chat/dsn.php

```
<?php
try {
$dsn = new PDO('mysql:host=localhost;dbname=beseder', 'root', '');
$dsn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$dsn->exec('SET NAMES "utf8"');
} catch (PDOException $e) {
echo $e->getMessage();
echo $e->getLine();
exit();
}
```

Он будет вставляться в те PHP файлы, где потребуется обращение к базе данных.

Внимание! В строке подключения к БД:

```
$dsn = new PDO('mysql:host=localhost;dbname=beseder', 'root', '');
```

необходимо вставить соответствующие значения: имя хоста, имя БД, имя пользователя, пароль пользователя, если они отличны от используемых по умолчанию

Выводу ошибок PDO::ATTR_ERRMODE задаем режим выброса исключений PDO::ERRMODE_EXCEPTION свойства которого будут отражать код ошибки и ее описание. Этот режим полезен при отладке, так как сразу известно, где в программе произошла ошибка. Это позволяет быстро локализовать и решить проблему. Режим исключений также полезен, так как дает возможность структурировать обработку ошибок более тщательно, нежели с обычными предупреждениями PHP, а также с меньшей вложенностью кода, чем в случае работы в тихом режиме с явной проверкой возвращаемых значений при каждом обращении к базе данных [7]. Устанавливаем кодировку обращения к БД по умолчанию как UTF-8.

6.
Установка
администратора
базы данных

Теперь необходимо установить «Администратора» базы данных, т.е. пользователя сайта которому разрешены операции с базой данных. Для этого в папке «admin» создаем подпапку «users», в которой будут находиться файлы по работе с пользователями.

Задавать данные администратора удобнее через форму. В

папке «users» создаем файл «form_create_admin.php». Это обычная HTML форма.

Расширение для этого и всех последующих HTML файлов, содержащих php код, принимаем как .php

Делается это затем, что лучше лишний раз подстраховаться, на предмет сохранения работоспособности кода в той конфигурации, в которой программа будет работать, а не в той, в которой она разрабатывается. Это замечание касается обработки PHP кода в HTML файле на хостинге.

Листинг 7. form_create_admin.php Путь: /news/chat/admin/users /form_create_admin.php

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/style.css" />
<title>Установить администратора</title>
</head>

<body>
<h2>Установить администратора</h2>
<form class="adminform" action="" method="post">

<p>Введите учетные данные Администратора</p>
<div class="label">
```

```
<label for="name">Логин:
<input type="text" name="login" id="login">
</label>
</div>
<hr>
<div class="label">
<label for="password">Пароль:
<input type="password" name="password" id="password">
</label>
</div>
<hr>
<p><i>Для запуска базы данных введите данные,
в дальнейшем вы сможете поменять их в разделе администрирования</i></p>
<div class="runcreateadmin">
<input type="hidden" name="action" value="start">
<input type="submit" value="Отправить">
</div>
</form>
</body>
</html>
```

В этой форме устанавливается логин и пароль администратора и передаются на обработку.

В дальнейшем в целях безопасности передаваемых формами данных их необходимо будет предварительно подготовить. Создаем вспомогательный файл «clean.php» и разме-

щаем в папке «admin». Подготовка осуществляется функциями:

htmlspecialchars – преобразует специальные символы в HTML-сущности;

stripslashes – удаляет экранирующие символы.

Листинг 8. clean.php Путь: news\chat\admin\clean.php

```
<?php
/* Обрабатываем текст в форме
Преобразуем специальные символы в HTML сущности
Удаляем экранирующие бэкслэши */

function html($text)
{
return htmlspecialchars($text, ENT_QUOTES, 'UTF-8');
return stripslashes($text);
}

//печатаем предыдущую функцию
function htmlout($text)
{
echo html($text);
}

// Очистить данные сессии для текущего сценария
function cleansession(){
$_SESSION = [];
```

```
// Удалить cookie, соответствующую SID
unset($_COOKIE[session_name()]);
// Уничтожить хранилище сессии
session_destroy();
}
```

*В файле clean.php также размещены функции, которые понадобятся в дальнейшем.

Для обработки переданных формой данных в папке «users» создаем файл «createadmin.php».

Листинг 9. createadmin.php Путь: news/chat/admin/users/createadmin.php

```
<?php
error_reporting(E_ALL);
if (session_id() == '') {
session_start();
}
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/dsn.php';

if (isset($_POST['login'])) {
$login = $_POST['login'];
if ($login == '') {
unset($login);
}
}
```

```
if (isset($_POST['password'])) {
$password = $_POST['password'];
if ($password =='') {
unset($password);
}
}

if (empty($login) or empty($password)) {
exit("<h4>Вы ввели не всю информацию, заполните все поля!</h4>");
}
/* если логин и пароль введены, то обрабатываем их */
$login = stripslashes($login); //
$login = htmlspecialchars($login);
$password = stripslashes($password);
$password = htmlspecialchars($password);
$login = trim($login);//удаляем лишние пробелы
$password = trim($password);

/* Заносим данные в базу */

try {
$sql = 'INSERT INTO users SET
login = :login,
activation = :activation';
$activation=1;
```

```
$s = $dsn->prepare($sql);
$s->bindValue(':login', $_POST['login']);
$s->bindValue(':activation', $activation);
$s->execute();
} catch (pdoException $e) {
exit('Ошибка при добавлении пользователя');
}

$authorid = $dsn->lastInsertId();
$password = md5($_POST['password'] . 'swl');
try {
$sql = 'UPDATE users SET
password = :password
WHERE id = :id';

$s = $dsn->prepare($sql);
$s->bindValue(':password', $password);
$s->bindValue(':id', $authorid);
$s->execute();
} catch (pdoException $e) {
exit('Ошибка установки пароля');
}

try {
$role = 'admin';
$sql = 'INSERT INTO authorrole SET
```

```
authorid = :authorid,
roleid = :roleid';

$s = $dsn->prepare($sql);
$authorid = $dsn->lastInsertId();
$s->bindValue(':authorid', $authorid);
$s->bindValue(':roleid', $role);
$s->execute();
} catch (pdoException $e) {
exit('Ошибка при назначении прав автору');
}
$_POST['action'] = '';
```

В этом файле:

– запускаем сессию, если она еще не начиналась;

– подключаемся к БД;

– проверяем данные, которые введены в форму, если все нормально, шифруем пароль в md5;

– устанавливаем активацию пользователя.

– заносим информацию об администраторе в БД.

Забегая вперед хотел сказать следующее: при авторизации мною принято решение о том, что пользователь считается активированным, если $activation=1, поэтому в запросе для создания администратора в этой переменной пишем единичку.

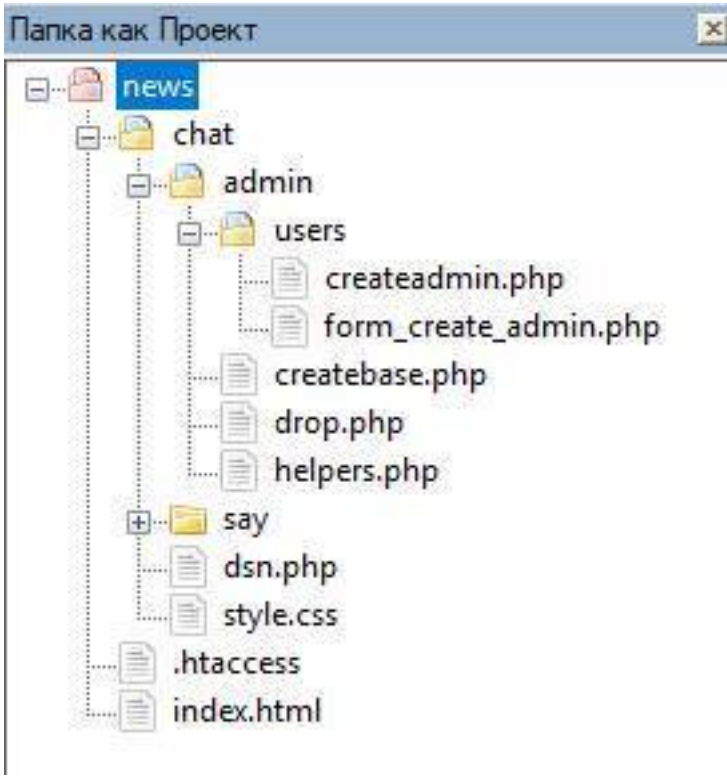


Рис. 13. Структура тестового сайта на момент создания БД

Все файлы необходимые для создания базы данных и ее администратора, а также вспомогательный файл для обработки передаваемых данных из форм и файл подключения

к БД созданы.

7. Контроллеры

Контроллеры в данном случае – это PHP файлы которые будут вставляться в страницы сайта и управлять выводом комментариев и авторизацией. Для данной задачи как мне кажется оптимально использовать три контроллера, размещаемых в нужных местах страницы сайта.

Контроллеры будут вставляться в необходимые места страниц сайта.

Подключение файла контроллера происходит согласно указанному к файлу пути. Есть два варианта пути: относительный и абсолютный. Относительный – это указание пути к подлючаемому файлу относительно файла с инструкцией подключения. Абсолютный – указание полного пути к подключаемому файла.

Подключение может осуществляться при помощи инструкций require или include. Они полностью идентичны за исключением лишь одной особенности – при использовании require в случае возникновения проблем выполнение скрипта будет остановлено (сценарий дальше не будет считываться), в то время как include просто выведет предупреждение и продолжит дальнейшее выполнение скрипта.

Так как по условиям задачи мы не знаем, как будет называться сайт, на котором будут размещаться комментарии и не хотим, чтобы в случае проблем с модулем сайт перестал

работать, для всех дальнейших подключений будем использовать инструкцию include и указывать абсолютный путь к файлу. Абсолютный путь будем строить при помощи переменной $_SERVER['DOCUMENT_ROOT'] (см. раздел 2.4).

8. Контроллер 1

Как было определено в разделе 2.3 контроллер 1 отвечает за создание базы данных и её администратора. Все необходимые для этого файлы готовы, поэтому пишем код файла, который будет ими управлять.

Контроллер будет размещаться в самом начале страницы сайта до блока <!DOCTYPE>.

Создаем в папке «chat» файл «createbase_controller.php».

Листинг 10. createbase_controller.php Путь: news/chat/ createbase_ controller.php

```
<?php
error_reporting(E_ALL);
if (session_id() == '') {
session_start();
}
/* Проверяем наличие базы данных и наличие в ней админа */
try {
$dsn = new PDO('mysql:host=localhost;dbname=beseder', 'root', '');
$dsn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
$dsn->exec('SET NAMES "utf8"');
} catch (PDOException $e) {
/* Мы здесь т.к. базы нет поэтому Создаем базу */
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/createbase.php';
}
/* Создаем админа */
/* Проверяем передавались ли данные формы на установку админа */
/* Если ДА создаем админа */
if (isset($_POST['action']) and $_POST['action'] == 'start') {
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/createadmin.php';
header('Location:.'); //перегружаемся чтобы убрать $_POST['action']
exit('контроллер 1 очистите кэш');
}
/* проверяем счетчик записей в таблице users */
/* если записи есть выходим, если нет вставляем форму создания админа */
try {
$count = $dsn->query("SELECT count(1) FROM users")->fetchColumn();
if ($count <= 0) {
include_once $_SERVER['DOCUMENT_ROOT']. '/chat/admin/users/form_create_admin.php';
```

```
exit('controller 1: Нет админа ');
}
} catch (PDOException $e) {
exit('Ошибка на первом входе в админку controller 1');
}
```

Здесь основная идея состоит в том, чтобы создавать БД на выбросе исключений в блок «catch» и последующей проверке наличия администратора при помощи счетчика записей.

Логика работы контроллера 1 следующая.

1.Запускаем сессию.

2. Пробуем подключиться к БД «beseder».

Внимание!

Не забываем ввести корректные данные строки подключения

$dsn = new PDO('mysql:host=**localhost**;dbname=**beseder**', '**root**', '');

если они отличаются от установленных по умолчанию.

3. Если БД нет, получаем ошибку и создаем БД в блоке обработки исключений catch при помощи файла createbase.php

4. На этом шаге БД или уже была или только что была создана.

5. Проверяем, передавались ли данные из формы form_create_admin.php

на создание администратора, если передавались, обрабатываем их файлом createadmin.php и создаем запись данных

администратора в БД.

6. Проверяем счетчик записей в таблице «users», если счетчик записей больше 1, т.е. в есть пользователи, то выходим. Контроллер выполнил работу. Есть БД и есть админ.

7. Если записи в БД отсутствуют вставляем форму для ввода данных администратора.

form_create_admin.php

заполняем значения и отправляем. Свойство «action» этой формы пустое, поэтому произойдет перезагрузка страницы и контроллер повторит свою работу, но теперь уже точно сработает п.5. и соответственно п.6. Контроллер выполнил работу. Есть БД и есть админ.

Важно! При наличии $_POST['action'] == 'start' контроллер 1 будет каждый раз создавать в таблице «users» дубликаты администратора при перезагрузке страницы, а соответственно появятся ошибки и сайт станет недоступен. Поэтому в последней строке листинга 5. createadmin.php мы обнуляем данные $_POST['action']= ''; и перезагружаем страницу в контроллере header('Location:.');exit();.

Проверяем работу контроллера. Вставляем в файл index.html т.е. главную страницу сайта «news» код:

<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/createbase.php'?>

Вставляем в самый верх. Получится:

Листинг 11. Тестируем контроллер 1

```
<?
php include_once $_SERVER['DOCUMENT_ROOT'].'/
chat/createbase_controller.php'?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>News</title>
</head>
<body>
<p>Значимость этих проблем настолько очевидна, что начало повседневной работы по формированию позиции способствует
подготовке и реализации новых предложений!
<br>
Значимость этих проблем настолько очевидна, что повышение уровня гражданского сознания влечет за собой процесс
внедрения и модернизации модели развития.
<br> Практический опыт показывает, что рамки и место обучения кадров способствует повышению актуальности
соответствующих условий активизации.</p>
</body>
</html>
```

Запускаем сайт, в случае с Open Server просто кликаем по его названию в выпадающем меню. Должно получиться следующее

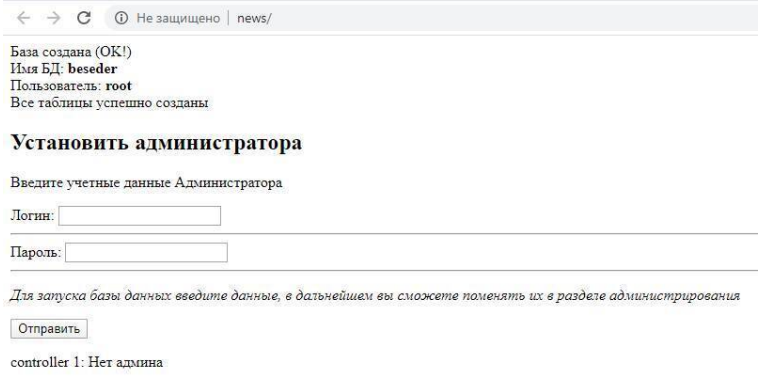


Рис. 14. Вывод формы создания администратора

Появляются сообщения о том, что БД и таблицы созданы и затем выводится форма для задания логина и пароля администратора модуля.

Вводим свой логин и пароль, отправляем. Получается:

Рис. 15. Работа сайта

Мы видим содержание сайта, значит все прошло успешно и контроллер 1 свою функцию выполнил. Можете войти в phpMyAdmin и проверить таблицы и записи.

9. Контроллер 2. описание

Контроллер 2 будет заниматься администрированием пользователей: регистрацией, авторизацией и распределением прав. Исходя из этого он должен предоставить пользователю необходимый интерфейс. В состав интерфейса будут входить управляющие кнопки:

• Авторизация – отвечает за вход в модуль для возможности оставления комментариев ;

• Регистрация – установка параметров авторизации нового пользователя;

• Кабинет – отвечает за вход в личный кабинет пользователя;

• Выход.

Контроллер должен обеспечивать вход в личный кабинет пользователя и предоставлять ему возможности для изменения и управления данными.

В данном случае достаточно двух основных ролей пользователя: «admin» и «user». «admin» может управлять добавлением, редактированием и удалением любых пользователей, имеет доступ для редактирования и удаления всех комментариев и ответов на них, «user» имеет те же возможности, но только для себя.

Есть также дополнительный резервный профиль «Site Administrator» он не активирован.

Перед написанием кода контроллера необходимо создать файлы его окружения.

9.1 Файлы окружения контроллера 2
9.1.1 Личный кабинет

Создаем в папке «admin» файл «index.php». Это будет страница администрирования модуля, на которой будут располагаться ссылки на соответственные разделы.

Листинг 12. index.php Путь: /news/chat/admin/index.php

```
<?php
/* Проверяем уровень доступа к странице */

include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/access.php';

if (!userIsLoggedIn()) {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/form_login.php';
exit();
}

if (!userHasRole('admin') and !userHasRole('user')) {
$error = 'Вход только для администратора';
include  $_SERVER['DOCUMENT_ROOT'].'/chat/admin/
```

```
accessdenied.html.php';
    exit();
    }
    ?>
    <html lang="en">

    <head>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="/style.css" />
    <title>Панель управления</title>
    </head>

    <body class="chatbody">
    <?php include_once $_SERVER['DOCUMENT_ROOT'] . '/
chat/admin/button_logout.html'; ?>
    <div class="wrap_apanel">
    <div class="blockapanel">
    <h2>Панель управления</h2>
    <ul>
    <li><a                    href="/chat/admin/comment.php"
class="apanel">Комментарии</a></li>
    <li><a                  href="/chat/admin/users/index.php"
class="apanel">Пользователи</a></li>
    </ul>
    </div>
```

```
<p><a href="/" class="apreturn">Вернуться на главную страницу</a></p>
</div>
</body>

</html>
```

В самом начале файла проверяем данные пользователя, если пользователь авторизован и имеет статус «admin» или «user» открываем страницу, если нет, выводим страницу запрета доступа accessdenied.html.php. Если пользователь не авторизован, выводим форму авторизации form_login.php.

На самой странице панели управления выводим:

• форму выхода пользователя из сессии button_logout.html с кнопкой «Выйти»;

• ссылки для перехода на страницы «Комментарии», «Пользователи»;

• ссылку «Вернуться на главную страницу» для перехода на начальную страницу сайта.

9.1.2 Управление доступом

Для управления доступом не будем изобретать велосипед, а воспользуемся уже проверенным решением из шуточной CMS [1] и создаем файл «access.php» листинг которого за некоторыми доработками аналогичен, приведенному в данной книге.

В исходный листинг [1.С.233] добавлены проверка активации и установление «id» пользователя.

Листинг 13. access.php Путь: /news/chat/admin/access.php

```
<?php
error_reporting(E_ALL);

/* Проверяем пароль и логин стартуем или удаляем сессию */
/* Функция возвращает true или false в зависимости от результатов проверки */
function userIsLoggedIn()
{
if (isset($_POST['action']) and $_POST['action'] == 'out')
{
if (!isset($_POST['login']) or $_POST['login'] == '' or
!isset($_POST['password']) or $_POST['password'] == '')
```

```
    {

    $GLOBALS['loginError'] = 'Пожалуйста, заполните оба
поля';
    return FALSE;
    }

    $password = md5($_POST['password'] . 'swl');

    if (databaseContainsAuthor($_POST['login'], $password))
    {
        /* Если пользователь существует в БД, то проверяем
его активацию */

        try {
            include  $_SERVER['DOCUMENT_ROOT'].'/chat/
dsn.php';

        $activation ='';
        $login = $_POST['login'];
        $password = md5($_POST['password'] . 'swl');

         $sql = 'SELECT activation FROM users WHERE login
= :login AND password = :password';
        $s = $dsn->prepare($sql);
        $s->bindValue(':login', $login);
```

```
        $s->bindValue(':password', $password);
        $s->execute();
        }
        catch (PDOException $e) {
            echo $e->getMessage();
            exit();
        }

        $activation = $s->fetch(PDO::FETCH_COLUMN);

        if ($activation !=1) {

             include $_SERVER['DOCUMENT_ROOT'].'/chat/
admin/erroractivation.html';
            exit();
        }

        /* Если активирован задаем значения сессии */

        if(session_id() == '') {session_start();}

    $_SESSION['loggedIn'] = TRUE;
    $_SESSION['login'] = $_POST['login'];
    $_SESSION['password'] = $password;
    return TRUE;
    }
```

```
else
{
    if(session_id() == '') {session_start();}

unset($_SESSION['loggedIn']);
unset($_SESSION['login']);
unset($_SESSION['password']);
$GLOBALS['loginError'] =
'Указанный логин или password не совпадают.';
return FALSE;
}
}

if (isset($_POST['action']) and $_POST['action'] == 'logout')
{
if(session_id() == '') {session_start();}

unset($_SESSION['loggedIn']);
unset($_SESSION['login']);
unset($_SESSION['password']);
unset($_SESSION['userid']);
header('Location: ' . $_POST['goto']);

exit();
}
if(session_id() == '') {session_start();}
```

```
    if (isset($_SESSION['loggedIn']))
    {
    return databaseContainsAuthor($_SESSION['login'],
$_SESSION['password']);
    }
    }

    /* Функция проверяет наличие в БД пользователя с пере-
данной парой логин – пароль */

    function databaseContainsAuthor($login, $password)
    {

    include $_SERVER['DOCUMENT_ROOT'].'/chat/
dsn.php';
    try
    {
    $sql = 'SELECT COUNT(*) FROM users WHERE login
= :login AND password = :password';
    $s = $dsn->prepare($sql);
    $s->bindValue(':login', $login);
    $s->bindValue(':password', $password);
    $s->execute();
    }
    catch (PDOException $e)
```

```
{
echo $e->getMessage();
exit();
}

$row = $s->fetch();

if ($row[0] > 0)
{
    return TRUE;
}
else
{
return FALSE;
}
}

/* Права для пользователя */

function userHasRole($role)
{
include          $_SERVER['DOCUMENT_ROOT'].'/chat/
dsn.php';
if (isset($_SESSION['login'])) {

try
```

```
{
$sql = "SELECT COUNT(*) FROM users
INNER JOIN authorrole ON users.id = authorid
INNER JOIN role ON roleid = role.id
WHERE login = :login AND role.id = :roleId";
$s = $dsn->prepare($sql);
$s->bindValue(':login', $_SESSION['login']);
$s->bindValue(':roleId', $role);
$s->execute();
}
catch (PDOException $e)
{
echo $e->getMessage();
echo $e->getLine();
exit('Ошибка поиска прав пользователя');
}

$row = $s->fetch();

if ($row[0] > 0)
{
return TRUE;
}
else
{
return FALSE;
```

```
    }
    }
    }
    /* Определяем id пользователя */

    function userId() {
    if (isset($_SESSION['login'])) {

        try {
                include $_SERVER['DOCUMENT_ROOT'].'/
dsn.php';
            $sql = 'SELECT id FROM users WHERE login
= :login';
            $s = $dsn->prepare($sql);

            $s->bindValue(':login',$_SESSION['login']);
            $s->execute();

            $rowid=$s->fetch();
            $_SESSION['userid'] = $rowid['id'];
        }
        catch (PDOException $e) {

            echo $e->getMessage();
            echo $e->getLine();
            exit('Ошибка добавления пользователя');
```

```
    }
}
}
```

Файл содержит функции для проверки данных пользователя.

Функция userIsLoggedIn()проверяет наличие и правильность данных переданных формой авторизации, активацию пользователя. В зависимости от результатов проверки этих данных устанавливает или уничтожает значения переменных сессии $_SESSION. Возвращает «true» или «false».

Для активации пользователя используется $activation. Пользователь считается активированным, если $activation = 1. Проверяем запросом к БД для указанного логина.

Функция databaseContainsAuthor() проверяет наличие в БД пользователя с переданной парой логин – пароль. Если пользователь существует возвращает «true» если нет «false».

Функция userHasRole($role)определяет уровень пользователя: «admin» или «user». В качестве параметра передается значение $role. Если $role равна значению «id» таблицы «role» установленному для данного логина при авторизации, то возвращает значение «true» иначе «false».

Функция userId() определяет «id» пользователя. Устанавливает значение переменной сессии $_SESSION['userid'].

9.1.3 Форма авторизации

Форма авторизации «form_login.php» служит для ввода логина и пароля пользователя.

Листинг 14. form_login.php Путь: /news/chat/admin/form_login.php

```
<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/admin/clean.php'; ?>
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
<title>Авторизация</title>
</head>

<body>

<?php if (isset($loginError)): ?>
<p><?php htmlout($loginError); ?></p>
<?php endif; ?>
```

```
<form action="" method="post" class="chatform">

<h4 class="formname">Авторизация</h4>
<hr>

<div class="login_input">
<label for="email">Логин:
<input type="text" name="login" id="login" class="inputs">
</label>
</div>
<hr>
<div>
<label for="password">Пароль:
<input type="password" name="password" id="password">
</label>
</div>
<hr>
<div>
<input type="hidden" name="action" value="out">
<input type="submit" value="Отправить">
</div>
</form>

</body>
```

</html>

В самом начале файла вставляем скрипт «clean.php», который будет чистить получаемые данные ($loginError).

Выводим форму. Заполняем, отправляем данные.

9.1.4 Страница ошибки активации

Страница ошибка активации erroractivation.html. Форма будет вставляться в случае ошибки активации, а именно, если переменная активации не равна 1:

$activation != 1;

проверка ведется в скрипте access.php.

Листинг 15. erroractivation.html Путь: /news/chat/admin/erroractivation.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
>
<title>Ошибка активациии</title>
</head>
<body>
<div style="float:none; display:inline-block; ">
<h5 class="ingress" style="text-align:center;">
Вы не активировали аккаунт, <br>
проверьте сообщения в почтовом ящике,<br>
указанном вами при регистрации
```

```
</h5>
</div>
<div>
<h5 class="ingress"><a href="/">Вернуться на главную страницу</a></h5>
</div>
</body>
</html>
```

Здесь вопросов быть не должно. Обычная html страница без кода. Часть стилей CSS заданы непосредственно на странице.

9.1.5 Кнопка выхода из раздела администрирования

Кнопка выхода из раздела администрирования «button_logout.php»

Листинг 16. button_logout.php Путь: /news/chat/admin/ button_logout.php

```
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="stylesheet" type="text/css" href="/chat/style.css" /
>
</head>

<div class="logouts">
<form action="/chat/admin/logout.php" method="post"
class="logout">
<input type="hidden" name="action" value="logout" />
<input type="hidden" name="goto" value="/admin/" />
<input type="submit" name="action" value="Выйти" />
</form>
</div>
```

</html>

Кнопка передает данные скрипту logout.php (Листинг 25) который обнуляет соответствующие переменные сессии.

После создания этих фалов пробуем зайти на страницу администрирования

news/chat/admin и посмотреть что получилось.

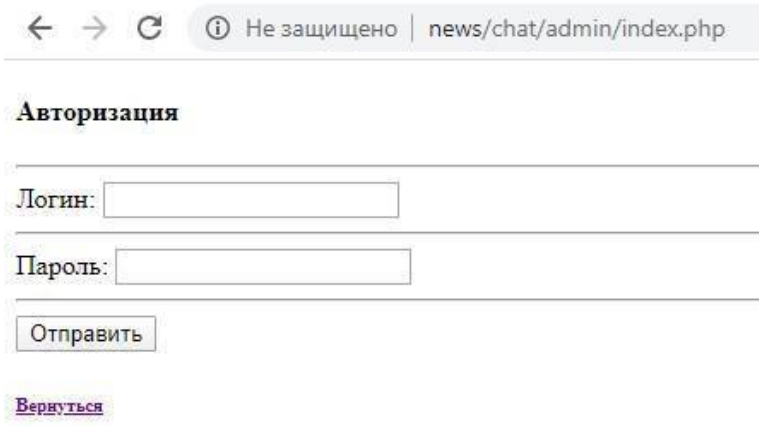


Рис. 16. Форма авторизации закрывает вход в личный кабинет

Заполняем поля формы и отправляем данные.

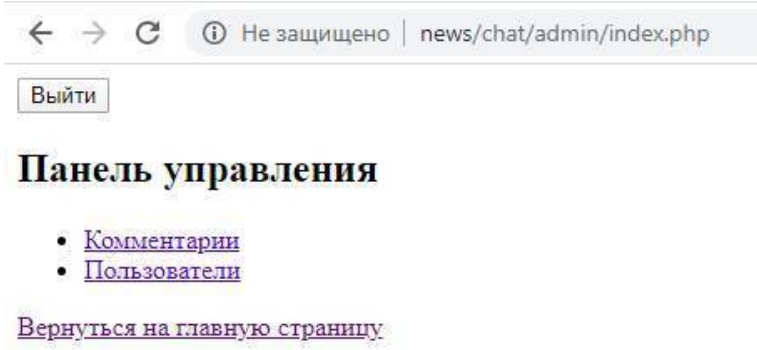


Рис. 17. Страница личного кабинета открыта

Если всё прошло нормально открывается страница доступа к личному кабинету в административном разделе. На этой странице имеется кнопка «Выход», ссылки на страницы «Комментарии» и «Пользователи», а также ссылка «Вернуться на главную страницу».

9.1.6 Страница доступ запрещен

Сами понимаете, без такой страницы никуда.

Листинг 17. accessdenied.html.php Путь: news/chat/admin/accessdenied.html.php

```
<?php
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/admin/clean.php';
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/admin/button_logout.html';
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Access Denied</title>
</head>
<body>
<h1>Доступ запрещен</h1>
<p><b><i><?php htmlout($error); ?></i></b></p>
<p><a href="/">Вернуться на главную страницу</a></p>
</body>
</html>
```

Страница будет выводиться, в случае если у пользователя нет прав для доступа к панели управления. Допустим, user захочет войти на страницу, доступную только пользователю с правами admin. «Доступ запрещен» будет выводиться по результатам проверки в файле access.php, на странице index раздела admin вместо основной страницы.

9.2. Статистика комментариев
9.2.1 Страница комментариев

Создаем страницу просмотра статистики комментариев «comment.html.php»

Листинг 18. comment.html.php Путь: news/chat/admin/ comment.html.php

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css" /
>
<title>Листинг комментариев</title>
</head>

<body class="chatbody">
<h1 class="user">Статистика комментариев</h1>

<div class="statwrap" id="">
<!– Вставляем блок формы поиска –>
<div class="statone" id="">
```

```
<?php include_once $_SERVER['DOCUMENT_ROOT'].'/
chat/admin/search.html.php'; ?>
</div><!– END statone –>

<div class="stattwo" id="">
<h5 class="user">Комментарии</h5>
<?php
if (isset($says)) {
foreach ($says as $say) : ?>
<p class="stat">
<?php
echo '<span style="color:cadetblue;">' . ' ' . $say['id'] . ' ' . '</
span>';
//убираем смайлы
$patterns = '/(:([^>]+):)/U';
$replace = ' smile* ';
$clean = $say['text'];
$text = preg_replace($patterns, $replace, $clean);
echo $text;

$t = time($say['saydate']);
echo '<span style="color:cadetblue;">' . ' ' . date("d.m.Y",
"$t") . '</span>'; ?>
</p>

<?php endforeach;
```

```
    } ?>

    </div><!– END stattwo –>

    <div class="statthree" id="">
    <h5 class="user">Ответы на комментарии</h5>
    <?php
    if (isset($replys)) {
    foreach ($replys as $reply) : ?>
    <p class="stat"><?php
    echo '<span style="color:cadetblue;">' . ' ' . $reply['replyid'] .
' ' . '<-' . ' ' . '</span>';

    //убираем смайлы
    $patterns = '/(:([^>]+):)/U';
    $replace = ' smile* ';
    $clean = $reply['replytext'];
    $text = preg_replace($patterns, $replace, $clean);
    echo $text;
    //время
    $t = time($reply['replydate']);
    echo '<span style="color:cadetblue;">' . ' ' . date("d.m.Y",
"$t") . '</span>'; ?>
    </p>
    <?php endforeach;
    } ?>
```

```
</div><!– END statthree –>

</div><!– END statwrap –>
<br />
<div class="return"><a href="/chat/admin/">Вернуться</a></div>
</body>
</html>
```

Страница функционально состоит из трех блоков:

- Блок с данными пользователей и формой поиска.
- Блок, отображающий комментарии.
- Блок, отображающий ответы на комментарии.

Предусмотрена ссылка для возврата на предыдущую страницу «Вернуться».

Содержание блоков формируется в файле comment.php в соответствии с данными отправленными формой поиска search.html.php. По умолчанию отображаются все имеющиеся данные. Комментарии и ответы на них выводятся без картинок.

9.2.2 Форма поиска

Так как не исключена ситуация, что в комментариях нам надо будет, что-нибудь искать, поэтому создадим для страницы комментариев форму поиска «search.html.php»

Листинг 19. search.html.php Путь: news/chat/admin/search.html.php

```
<form action="" method="post" class="stat">
<center>
<p>Панель информации:</p>
</center>

<fieldset>
<div>
<label for="author">Пользователь:</label>
<select name="author" id="author">
<option value="">Все пользователи</option>
<?php foreach ($users as $user) : ?>
<option value="<?php htmlout($user['id']); ?>"><?php
htmlout($user['login']); ?></option>
<?php endforeach; ?>
</select>
</div>
```

```
<div>
<p>
<label for="category">Раздел:</label>
<select name="category" id="category">
<option value="all">Все разделы</option>
<option value="say">Комментарии</option>
<option value="reply">Ответы</option>
</select>
</p>
</div>

<div class="textsearch">
<label for="text">Содержит текст:</label>
<input type="text" name="text" id="text">
</div>

<div class="textsearch_button">
<input type="hidden" name="action" value="search">
<input type="submit" value="Искать">
</div>
</fieldset><!– END fieldset –>

<fieldset>
<legend>Выборка для: </legend>
<h5 class="user"><?= $legend ?></h4>
```

```
</fieldset>

<fieldset>
<legend>Список пользователей:</legend>

<ul style="list-style-type:none;padding:0px;margin:5px;">
<?php foreach ($users as $user): ?>

<li style="border-bottom:1px solid lightgray; margin-bottom:5px;">
<?php
        htmlout($user['id']); echo '.   ';
htmlout($user['login']);
            ?>
</li>
<?php endforeach; ?>
</ul>

</fieldset>
</form>
```

Обычная HTML форма в полях <fieldset> которой, сгруппированы параметры для поиска и фильтрации данных. Поиск ведется по тексту. Результаты можно отфильтровать по логину пользователя и разделу.

Форма сгруппирована из блоков:

• Панель информации – общий контейнер для блоков.

• Пользователи – выпадающий список <select> пользователей.

• Раздел – выпадающий список <select> разделов, их всего три «Все разделы», «Комментарии» и «Ответы».

• Поле ввода поискового запроса и кнопка для оправки этого запроса «Искать».

Данные из формы отправляются на обработку в файл comment.php.

9.2.3 Скрипт обработки страницы комментариев

Обработкой данных на странице комментариев будет заниматься файл «comment.php».

Листинг 20. comment.php Путь: news/chat/admin/comment.php

```
<?php
error_reporting(E_ALL);
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/clean.php';

//Пользователи для панели информации
try {
$result = $dsn->query('SELECT id, login FROM users');
} catch (PDOException $e) {
echo $e->getMessage();
echo $e->getLine();
exit('Ошибка поиска пользователя в базе комментариев');
}
```

```
    foreach ($result as $row) {
    $users[] = array('id' => $row['id'], 'login' => $row['login']);
    }
    /* Получаем логин пользователя при запросе */

    if (isset($_POST['author']) and $_POST['author'] != '') {
    try {
    $sql = 'SELECT login FROM users WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['author']);
    $s->execute();
    $legend = $s->fetchColumn(0);
    } catch (PDOException $e) {
    echo $e->getMessage();
    echo $e->getLine();
    exit('Ошибка поиска логина в базе комментариев');
    }
    } else {
    $legend = '';
    }
    /* Start. Работаем с комментариями */
    //если выбрано поле "ответы" выходим и ничего не делаем
в разделе "комментарии"
    if (isset($_POST['category']) and $_POST['category'] ==
'reply') {
    }
```

```
/* иначе формируем запрос к базе */
elseif (isset($_POST['action']) and $_POST['action'] == 'search' or !isset($_POST['action'])) {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';
// Базовое выражение SELECT.
$select = 'SELECT say.id, say.userid, say.saytext, say.saydate';
$from = ' FROM say INNER JOIN users ON say.userid = users.id';
$where = ' WHERE TRUE';
$sequence = array();
// Автор выбран
if (isset($_POST['author']) and $_POST['author'] != '') {
$where .= " AND userid = :userid";
$sequence[':userid'] = $_POST['author'];
}
// Была указана какая-то искомая строка
if (isset($_POST['text']) and $_POST['text'] != '') {
$where .= " AND saytext LIKE :saytext";
$sequence[':saytext'] = '%' . $_POST['text'] . '%';
}
try {
$sql = $select . $from . $where;
$s = $dsn->prepare($sql);
$s->execute($sequence);
```

```
    } catch (PDOException $e) {
    echo $e->getMessage();
    echo $e->getLine();
    exit('Ошибка при извлечении комментариев');
    }
    foreach ($s as $row) {
    $says[] = array('id' => $row['id'], 'userid' => $row['userid'],
'text' => $row['saytext'], 'saydate' => $row['saydate']);
    }
    }
    /* End. Завершаем обработку комментариев */
    /* Start. Работаем с ответами на комментарии */
    //если выбрано поле "комментарии" выходим и ничего не
делаем в разделе "ответы"
    if (isset($_POST['category']) and $_POST['category'] ==
'say') {
    }
    /* иначе формируем запрос к базе */ elseif
(isset($_POST['action']) and $_POST['action'] == 'search' or !
isset($_POST['action'])) {
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';
    // Базовое выражение SELECT.
    $select = 'SELECT reply.userid, reply.replytext,reply.replyid,
reply.replydate ';
    $from = ' FROM reply INNER JOIN users ON reply.userid
```

```
= users.id ';
    $where = ' WHERE TRUE';
    $order = ' ORDER BY reply.replyid';
    $sequencer = array();
    if (isset($_POST['author']) and $_POST['author'] != '') { // Автор выбран
    $where .= " AND userid = :userid";
    $sequencer[':userid'] = $_POST['author'];
    }
    if (isset($_POST['text']) and $_POST['text'] != '') { // Была указана какая-то искомая строка
    $where .= " AND replytext LIKE :replytext";
    $sequencer[':replytext'] = '%' . $_POST['text'] . '%';
    }
    try {
    $sql = $select . $from . $where . $order;
    $s = $dsn->prepare($sql);
    $s->execute($sequencer);
    } catch (PDOException $e) {
    echo $e->getMessage();
    echo $e->getLine();
    exit('Ошибка при извлечении ответов на комментарии');
    }
    foreach ($s as $row) {
    $replys[] = array('replyid' => $row['replyid'], 'userid' => $row['userid'], 'replytext' => $row['replytext'], 'replydate' =>
```

```
$row['replydate']);
    }
    }
    include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/
admin/comment.html.php';
```

Вывод информации производится в соответствии с условиями заданными в форме поиска: по имени пользователя, по разделу, по искомому тексту.

В начале скрипта делаем выборку из БД всех пользователей. Затем получаем из БД логин пользователя, для которого выполняется поиск.

Создаем запрос SELECT, зависящий от указанных в форме условий. Сначала определим строки, объединение которых формирует запрос SELECT в том случае, если не выбран ни один критерий, строки находятся в переменных:

$select, $from и $where.

Если кнопка «Искать» не нажималась, то $_POST['action'] будет отсутствовать и будут, отображены все комментарии и ответы них.

Т.к. базовое выражение SELECT собирается из тех критериев, что выбраны в форме, то в запросе будут операторы FROM и WHERE. Если критерии не заданы (то есть нужно получить все данные из БД), WHERE будет ненужной. Добавлять выражения к несуществующему оператору проблематично, поэтому подставляем такую команду, которая ни-

как не повлияет на результат, если ее опустить. Здесь подойдет выражение WHERE TRUE, которое всегда истинно.

Для добавления новой строки к существующей используется оператор конкатенации (.=). В этом скрипте к оператору WHERE добавляется условие, по которому содержимое поля userid из таблицы say должно совпадать со значением псевдопеременной :userid. Вначале скрипта с помощью метода bindValue установить требуемое общее значение $_POST['author'] нельзя т.к. еще не подготовлен объект параметризированного запроса, из которого этот метод вызывается. Вследствие этого запрос разбросан по вышеперечисленным строкам ($select, $from и $where).

Инициализируем массив $sequence для сохранения параметризированных переменных, используя их имена в качестве индексов. В массиве будет храниться id пользователя и текст поискового запроса. Содержимое массива:

```
$sequence (
:userid => "$_POST['author']",
:saytext => "'%'.$_POST['text'].'%'"
);
```

где "$_POST['author']" = $user['id'].

Чтобы получить значение параметризированной переменной для оператора LIKE, содержимое $_POST['text'] размещено между двумя знаками процента (%). LIKE воспринимает этот знак как групповой символ, поэтому при поиске строки $_POST['text'] в поле text формы search.html.php в

расчет будут приниматься также строки, где до и после этого значения находится другой текст.

Запускаем. Если все сделано без ошибок, то при запущенном на «Open Server» тестовом сайте при переходе по адресу: news/chat/admin/comment.php видим страницу статистики комментариев (см. рис. 15)

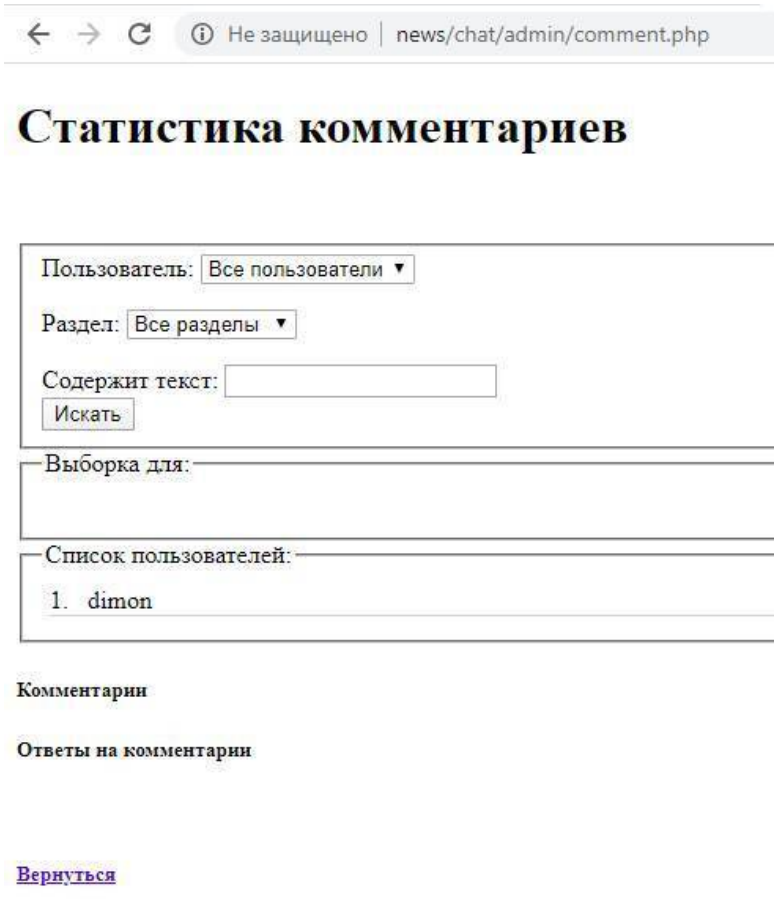


Рис. 18. Вид страницы статистики комментариев

На этом этапе необходимо заняться контроллером 2 и кнопками авторизации, так как опробовать работу страницы в деле пока нельзя в связи с отсутствием комментариев и ответов на них.

10. Контроллер 2

Контроллер 2 будет заниматься контролем авторизации пользователей и их возможностям для работы с комментариями.

Напишем код для входа на страницу личного кабинета и связанных с ней страниц. Контроллер 2 назовем «login_controller.php» и разместим в корне папки «chat», т.е. там же, где и контроллер 1 (createbase_controller.php)

Листинг 21. login_controller.php Путь: news/chat/login_controller.php

```
<?php
error_reporting(E_ALL);
/* Проверяем данные пользователя, устанавливаем данные сессии */

include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/access.php';
userIsLoggedIn();
userId();

if (isset($_SESSION['login'])) {
echo '<div class="welcom">Welcome '. $_SESSION['login'].'</div>'; //Выводим "Привет юзер"
```

```
/* Вставляем кнопки "Кабинет" и "Выход" */

include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/button_cabinet.html';
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/button_logout.html';
} else {
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/input_button_block.html';

if (!empty($_GET['name'])) {
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/form_login.php';
}
}
```

В подключаемом файле access.php проверяем данные пользователя и устанавливаем данные сессии.

После проверки данных функциями userIsLoggedIn() и userId()если пользователь авторизован приветствуем его и вставляем кнопки «Кабинет» и «Выход». Если нет, выводим блок кнопок «Вход», «Регистрация», «На главную» и форму авторизации.

Напишем коды кнопок и опробуем его в деле, это можно будет сделать после добавления файлов рассматриваемых в следующем разделе.

11. Кнопки раздела администрирования

11.1 Кнопки вход и регистрация

Кнопки «Вход» «Регистрация» и «На главную» сгруппированы в одном файле «input_button_block.html»

Листинг 22. input_button_block.html Путь: news/chat/admin/ users/ input_button_block.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
</head>
<div class="inputbutton">
<div class="ingress">
<a href="?name=door">Вход</a>
</div>

<div class="ingress">
<a href="admin/users/reg.php">Регистрация</a>
</div>
```

```
<div class="ingress">
<a href="/">На главную</a>
</div>
</div>
</html>
```

Назначаем обработчик нажатия кнопки «Вход».

11.2 Обработчик кнопки вход

Для обработки нажатия кнопки вход используем метод GET передавая в нем переменную «door». При нажатии кнопки происходит перезагрузка страницы и если контроллер видит переменную door то вставляет файл form_login.php

Листинг 23. form_login.php Путь: news/chat/admin/form_login.php

```
<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/admin/clean.php'; ?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/style.css" />
<title>Авторизация</title>
</head>

<body>
<?php if (isset($loginError)): ?>
<p><?php htmlout($loginError); ?></p>
<?php endif; ?>
```

```
<form action="" method="post">
<h4 class="formname">Авторизация</h4>
<hr>
<div>
<label for="email">Логин:
<input type="text" name="login" id="login" class="inputs">
</label>
</div>
<hr>
<div>
<label for="password">Пароль:
<input type="password" name="password" id="password">
</label>
</div>
<hr>
<div>
<input type="hidden" name="action" value="out">
<input type="submit" value="Отправить">
</div>
</form>
</body>
</html>
```

В эту форму пользователь вставляет свои логин и пароль и отправляет дальше на проверку.

11.3 Кнопки кабинет и выход

Страница опять перегрузилась. Если логин и пароль введены корректно, то выводим кнопки «Кабинет» и «Выход», а заодно напечатаем в контроллере 2 приветствие:«Welcome $user».

Кнопка «Кабинет» файл «button_cabinet.html».

Листинг 24. button_cabinet.html Путь: news/chat/admin/button_cabinet.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="stylesheet" href="/chat/style.css" />
</head>
<div class="cabinets">
<form action="/chat/admin/index.php" method="post" class="cabinet">
<input type="submit" name="action" value="Кабинет" />
</form>
</div>
</html>
```

Кнопка «Выход» файл «button_logout.html» рассмотрена

ранее в Листинге 16.

Обработчик события кнопки «Выйти» файл «logout.php»

Листинг 25. logout.php Путь: news/chat/admin/ logout.php

```
<?php
if (session_id() == '') {
session_start();
}
/* Проверяем была ли нажата кнопка "Выйти" если 'Да' уничтожаем сессию */
if (isset($_POST['action']) and $_POST['action'] == 'Выйти') {
unset($_SESSION['loggedIn']);
unset($_SESSION['login']);
unset($_SESSION['password']);
}
header("Location: " . $_SERVER["HTTP_REFERER"]);
```

При нажатии кнопки «Выйти» идет переход на страницу logout.php, где происходит сброс значений сессии, а затем делается редирект обратно на исходную страницу.

Для того чтобы вывести кнопки и формы необходимо в исходный текст стартовой страницы вставить (инклюдить) следующий код

<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/login_controller.php'?>

Листинг 26. index.html

Вставка контроллера 2: login_controller.php в файл: news/index.html

<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/createbase_controller.php'?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>NEWS</title>
</head>
<body>

<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/login_controller.php'?>

<p>Значимость этих проблем настолько очевидна, что начало повседневной работы по формированию позиции способствует

подготовке и реализации новых предложений!

Значимость этих проблем настолько очевидна, что повышение уровня гражданского сознания влечет за собой процесс

внедрения и модернизации модели развития.

 Практический опыт показывает, что рамки и место обучения кадров способствует повышению актуальности

соответствующих условий активизации.</p>

</body>

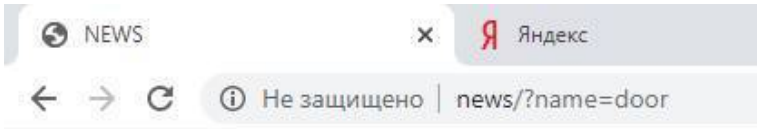
</html>

Код строк вставок контроллера1 и контроллера 2 выделен цветом.

кнопки будут выведены в том месте документа, где вставлен код строки подключения login_controller.php.

Можете поэкспериментировать с местом размещения «инклюда» в документ. Далее приведены скриншоты, иллюстрирующие работу контроллеров 2 и 1.

Начинают появляться выводимые элементы, поэтому понемногу надо задавать им стили css. Дальнейшие скриншоты работы программы будут с использованием стилей, но сама итоговая страница стилей будет приведена в конце книги, чтобы не было лишней путаницы.



ВХОД РЕГИСТРАЦИЯ НА ГЛАВНУЮ

Авторизация

Логин:

Пароль:

Отправить

Значимость этих проблем настолько очевидна, что начало повседневной работы по формированию позиции способствует подготовке и реализации новых предложений!
Значимость этих проблем настолько очевидна, что повышение уровня гражданского сознания влечет за собой процесс внедрения и модернизации модели развития.
Практический опыт показывает, что рамки и место обучения кадров способствует повышению актуальности соответствующих условий активизации.

Рис. 19. Пользователь не авторизован

Рис. 20. Пользователь авторизовался

12. Страница панель управления

На страницу «Панель управления» попадаем при нажатии кнопки «Кабинет». За вывод страницы отвечает файл index.php в папке «users».

Листинг 27. index.php Путь: news/chat/admin/users/index.php

```
<?php
error_reporting(E_ALL);

include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/access.php';

/*.....Проверяем полномочия пользователя.....*/

if (!userIsLoggedIn()) {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/form_login.php';
exit();
}

if (!userHasRole('admin') and !userHasRole('user')) {
exit('Доступ только для Администратора');
```

```
    }
    /*.....Если admin разрешаем все. Start admin.....*/
    //Начинаем основной цикл для администратора

    if (userHasRole('admin')) {

    /*.....Подключение нового пользователя.....*/

    if (isset($_GET['add'])) {
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';

    $pageTitle = 'Подключение нового пользователя';
    $action = 'addform';
    $name = '';
    $email = '';
    $id = '';
    $button = 'Добавить пользователя';

    // Выводим уровни доступа
    try {
    $result = $dsn->query('SELECT id, description FROM role');
    } catch (pdoException $e) {
    $error = 'Ошибка при получении списка ролей.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
```

```
    exit();
    }

    foreach ($result as $row) {
    $roles[] = array(
    'id' => $row['id'],
    'description' => $row['description'],
    'selected' => false
    );
    }
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/form_add_user.php';
    exit();
    }
    /*..... 1. Добавление нового пользователя.....*/

    if (isset($_GET['addform'])) {
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/avatar.php';

    /*.... 1.1 Проверяем все ли поля формы заполнены.....*/

    if (empty($_POST['login']) or empty($_POST['password'])
or empty($_POST['email']) or empty($_POST['roles'])) {
```

```
        exit("Вы ввели не всю информацию, вернитесь назад и заполните все поля!");
    }

    if (mb_strlen($_POST['login']) < 3 or mb_strlen($_POST['login']) > 15) { // проверяем длину логина

    exit("Логин должен состоять не менее чем из 3 символов и не более чем из 15."); //останавливаем выполнение сценариев
    }

    /*..... 1.2 Проверка корректности email.....*/

    if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $_POST['email'])) {
    exit("Неверно введен e-mail!");
    }

    /*.... 1.3 Проверка существования логина.....*/

    try {
    $logresult = $dsn->query('SELECT login FROM users');
    } catch (pdoException $e) {
    $error = 'Логин не существует.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
```

```
users/error.html.php';
    exit();
    }

    //заносим введенный пользователем логин в переменную
$login, если он пустой, то уничтожаем переменную

    if (isset($_POST['login'])) {
    $login = $_POST['login'];
    if ($login == '') {
    unset($login);
    $error = 'Логин пустой.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    foreach ($logresult as $row) {
    $logins[] = array(
    'login' => $row['login']
    );

    if ($row['login'] == $_POST['login']) {
    $error = 'Логин занят.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
```

```
exit();
}
}
}
/*....Сохранение данных пользователя.....*/
//Блок вложен в основной цикл админа

try {
$sql = 'INSERT INTO users SET
login = :login,
email = :email,
    activation = :activation,
    date = :date';
$s = $dsn->prepare($sql);
$s->bindValue(':login', $_POST['login']);
$s->bindValue(':email', $_POST['email']);
$s->bindValue(':activation', "1");
$s->bindValue(':date', time());
$s->execute();
} catch (pdoException $e) {
$error = 'Ошибка добавления отправленного пользовате-
ля.';
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
exit();
}
```

```
    if ($_POST['password'] != '') {
    $password = md5($_POST['password'] . 'swl');
    $authorid = $dsn->lastInsertId();

    try {
    $sql = 'UPDATE users SET
    password = :password
    WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':password', $password);
    $s->bindValue(':id', $authorid);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка 1 установки пароля.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
    }

    try {
    $sql = 'UPDATE users SET
    img = :img
    WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':img', $avatar);
```

```
    $s->bindValue(':id', $authorid);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка 2 установки пароля.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    if (empty($_POST['roles'])) {
    $error = 'Вы не отметили полномочия нового пользовате-
ля';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    if (isset($_POST['roles'])) {
    foreach ($_POST['roles'] as $role) {
    try {
    $sql = 'INSERT INTO authorrole SET
    authorid = :authorid,
    roleid = :roleid';
    $s = $dsn->prepare($sql);
    $s->bindValue(':authorid', $authorid);
    $s->bindValue(':roleid', $role);
```

```
$s->execute();
} catch (pdoException $e) {
$error = 'Ошибка при назначении выбранной роли пользователю.';
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/error.html.php';
exit();
}
}
}
header('Location: .');
exit();
}

/*.....Редактирование пользователя.....*/
//Блок вложен в основной цикл админа

if (isset($_POST['action']) and $_POST['action'] == 'Редактировать') {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';

try {
$sql = 'SELECT id, login, email, img FROM users WHERE id = :id';
$s = $dsn->prepare($sql);
```

```
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка при получении сведений об авторе.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    $row = $s->fetch();

    $pageTitle = 'Редактирование профиля';
    $action = 'editform';
    $login = $row['login'];
    $email = $row['email'];
    $id = $row['id'];
    $avatar = $row['img'];
    $button = 'Отправить';

    //Уровень доступа автора

    try {
    $sql = 'SELECT roleid FROM authorrole WHERE authorid
= :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $id);
```

```
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка при получении назначенной роли пользователя.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/error.html.php';
    exit();
    }

    $selectedRoles = array();
    foreach ($s as $row) {
    $selectedRoles[] = $row['roleid'];
    }

    // Готовим список уровней доступа
    try {
    $result = $dsn->query('SELECT id, description FROM role');
    } catch (pdoException $e) {
    $error = 'Ошибка при получении списка ролей.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/error.html.php';
    exit();
    }

    foreach ($result as $row) {
    $roles[] = array(
```

```
    'id' => $row['id'],
    'description' => $row['description'],
    'selected' => in_array($row['id'], $selectedRoles)
    );
    }

    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/form_add_user.php'; //выводим форму добавления и ре-
дактирования пользователя
    exit();
    }

    if (isset($_GET['editform'])) {
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/avatar.php'; //инициализируем аватар пользователя

    /*..... 2.1 Проверяем все ли поля формы заполнены.....*/

    if (empty($_POST['login']) or empty($_POST['password'])
or empty($_POST['email']) or empty($_POST['roles'])) {
    exit("Вы ввели не всю информацию, вернитесь назад и за-
полните все поля!");
    }
```

```
if (mb_strlen($_POST['login']) < 3 or mb_strlen($_POST['login']) > 15) { // проверяем длину логина
exit("Логин должен состоять не менее чем из 3 символов и не более чем из 15.");
}

/*..... 2.2 Проверка корректности email.....*/

if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $_POST['email'])) {
exit("Неверно введен e-mail!");
}

try {
if (!isset($date)) {
$date = time();
}
$sql = 'UPDATE users SET
login = :login,
email = :email,
    img = :img,
    date = :date
WHERE id = :id';
$s = $dsn->prepare($sql);
$s->bindValue(':id', $_POST['id']);
$s->bindValue(':login', $_POST['login']);
```

```
    $s->bindValue(':email', $_POST['email']);
    $s->bindValue(':date', $date);
    $s->bindValue(':img', $avatar);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка при обновлении пользователя.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    if ($_POST['password'] != '') {
    $password = md5($_POST['password'] . 'swl');

    try {
    $sql = 'UPDATE users SET
    password = :password
    WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':password', $password);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка установки пароля автора.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
```

```
    exit();
    }
    }

    if (empty($_POST['roles'])) {
    $error = 'Вы не отметили полномочия нового пользовате-
ля';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    try {
    $sql = 'DELETE FROM authorrole WHERE authorid = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка удаления устаревших записей роли
пользователя.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    if (isset($_POST['roles'])) {
```

```
foreach ($_POST['roles'] as $role) {
try {
$sql = 'INSERT INTO authorrole SET
authorid = :authorid,
roleid = :roleid';
$s = $dsn->prepare($sql);
$s->bindValue(':authorid', $_POST['id']);
$s->bindValue(':roleid', $role);
$s->execute();
} catch (pdoException $e) {
$error = 'Ошибка при назначении выбранной роли авто-
ру.';
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
exit();
}
}
}

header('Location: .');
exit();
}
/*....Удаление 2.Результат работы формы, вызываемой ко-
дом ниже(Удаление 1)…*/

if (isset($_POST['action']) and $_POST['action'] == 'ДА') {
```

```
    // Удаляем ассоциации пользователя

    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';

    try {
    $sql = 'DELETE FROM authorrole WHERE authorid = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка удаления роли пользователя.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    // Удаляем пользователя

    try {
    $sql = 'DELETE FROM users WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
```

```
    $error = 'Ошибка удаления пользователя индекс.';
    $e->getMessage();
    $e->getLine();
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
    header('Location: .');
    exit();
    }
    /*…Первичный запрос на удаление. Выводит форму вы-
бора ДА или НЕТ. Удаление 1…*/

    if (isset($_POST['action']) and $_POST['action'] == 'Уда-
лить') {
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/delete.php';
    exit();
    }

    //Готовим лист с данными пользователей

    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';

    try {
```

```
    $result = $dsn->query('SELECT users.id,users.login, users.img, authorrole.roleid FROM `users` INNER JOIN authorrole ON users.id =authorrole.authorid');
    } catch (pdoException $e) {
    $error = 'Ошибка при получении пользователей из базы данных!';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/error.html.php';
    exit();
    }

    foreach ($result as $row) {
    $authors[] = array('id' => $row['id'], 'login' => $row['login'], 'roleid' => $row['roleid'], 'img' => $row['img']);
    }

    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/authors.html.php'; //вставляем листинг авторов
    }
    //Закончили основной цикл для администратора

    /*..... END admin.....*/

    /*..... Start user..... */

    //Работа с простым пользователем
```

```
if (userHasRole('user')) {
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/edituser.php';
}
//Закончили работу простым пользователем
/*..... END user..... */

/* Удаляем неактивных пользователей.
Удаление происходит из двух таблиц при обновлении страницы */

try {
$sql = "SELECT * FROM users INNER JOIN authorrole ON users.id = authorrole.authorid WHERE users.activation IS NULL OR users.activation !='1' ";
$s = $dsn->query($sql);
$no_active = $s->Fetchall();
} catch (pdoException $e) {
exit('Ошибка при выборке неактивных пользователей');
}

foreach ($no_active as $nouser) {
$r = time() – $nouser["date"];
$id = $nouser['id'];
```

```
    if ($r > 3600) {
    try {
    $sql = "DELETE FROM users WHERE id = :id AND
activation IS NULL OR users.activation !='1'";
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $id);
    $s->execute();
    } catch (pdoException $e) {
    exit('Ошибка при выборке неактивных пользователей0');
    }

    try {
    $sql = "DELETE FROM authorrole WHERE authorid = :id";
    $s = $dsn->prepare($sql);

    foreach ($no_active as $nouser) {
    $authorid = $nouser['id'];
    $s->bindValue(':id', $authorid);
    $s->execute();
    }
    } catch (pdoException $e) {
    exit('Ошибка при выборке неактивных пользователей');
    }
    }
    }
```

Файл состоит из трех частей. В первой обрабатываются данные администратора. Во второй части подключается файл edituser.php и в нем обрабатываются данные обычного пользователя. Вторая часть начинается с комментария / * *Start user* */*. В третьей части идет проверка неактивных пользователей и их удаление.

Скрипты, которые обрабатывает файл index.php:

– /chat/admin/access.php';
– /chat/admin/users/authors.html.php;
– /chat/admin/users/edituser.php;
– /chat/admin/users/error.html.php';
– /chat/admin/users/form_add_user.php';
– /chat/admin/users/avatar.php';
– /chat/admin/users/form_add_user.php';
– /chat/admin/users/delete.php';

Обработка данных администратора заключается в следующем:

1. Сквозной вариант без нажатия ссылок и кнопок

1.1 Проверяются полномочия пользователя для захода на страницу, если пользователь не «admin» и не «user» доступ к странице закрывается.

1.2 Если права «admin», то проверяются нажатия ссылок

«Добавить нового пользователя», кнопок «Редактировать» и «Удалить».

1.3 Если кнопки не нажимались идем дальше и проверяем пользователя на права «user».

1.4 Если права не «user», переходим к последней, третьей части в которой идет удаление неактивных пользователей.

1.5 Удаление не активных пользователей сделано для тех пользователей, которые активируются самостоятельно, а не добавлены администратором. Принцип удаления таков: если пользователь начал регистрацию на сайте, получил письмо на свою электронную почту и в течении часа не ответил на него, перейдя по указанной в письме активации ссылке, то его данные удаляются из БД. Сам алгоритм определения неактивных пользователей следующий:

Делается SQL запрос к таблице «users» и связанной с ней таблицей «authorrole», в котором выбираются все пользователи, у которых значение activation не равно 1. Полученный массив обрабатывается построчно в цикле foreach по условию:

```
$r = time() – $nouser["date"];
```

где:

time() – текущее время;

$nouser["date"]– время регистрации пользователя (соответствует значению «date» таблицы «users»).

За время активации отвечает параметр $r. Если у пользователя $r > 3600 идет удаление значений данного пользователя из таблиц «users» и «authorrole».

Внимание! В файле много sql запросов, если некоторые не очень понятны, можно перейти в phpMyAdmin зайти в базу beseder выбрать вкладку sql и самостоятельно посмотреть, что выдают запросы в коде. при вставке не забудьте заменить переменные их значениями.

2. Нажата ссылка «Добавить нового пользователя»

2.1. Страница перегружается и появляется переменная $_GET['add'].

2.2 Инициализируем переменные добавления пользователя, вставляем форму добавления пользователя form_add_user.php. После заполнения формы проверяются переданные ею данные и устанавливается «аватар» в подключаемом скрипте avatar.php.

2.3 Данные заносятся в БД.

2.4 Проверяются и удаляются не активированные пользователи.

3. Нажата кнопка «Редактировать».

3.1 Появляется $_POST['action'] == 'Редактировать'

3.2. Также вставляется form_add_user.php, но в ней уже заранее установлены данные конкретного пользователя.

3.3 Данные изменяются администратором, проверяются и заносятся в БД.

3.4 Проверяются и удаляются не активированные пользо-

ватели.

4. Нажата кнопка «Удалить»

4.1. Появляется $_POST['action'] == 'Удалить', выполняется условие вставки скрипта подготовки удаления delete.php и формы удаления form_delete.php. Если в форме удаления соглашаемся и нажимаем «Да» то $_POST['action'] == 'Да' и запускается код удаления. Удаление происходит из таблицы «users» и связанной с ней таблицей «authorrole».

13. Управление пользователями

Попадаем при переходе по ссылке «Пользователи» на странице «Панель управления». За вывод страницы отвечает файл authors.html.php .

Листинг 28. authors.html.php Путь: news/chat/admin/users/authors.html.php

```
<?php
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/clean.php';
?>
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
<title>Управление пользователями</title>
</head>

<body>
<h2 class="user">Управление пользователями</h2>
<div class="addusers"><a href="?add">Добавить нового
```

```
пользователя</a></div>

    <div class="ourwrapper" id="">
    <!– <ul>–>
    <?php foreach ($authors as $author) : ?>

    <!– <li style="list-style-type: none">–>
    <div class="formuser">
    <form action="" method="post" class="formusers">

    <div class="wrapuser">
    <div class="topuser" id="">
    <p class="login_art">
    <span class="number"># <?php htmlout($author['id']); ?></
span>
    <?php htmlout($author['login']); ?>
    </p>

    </div><!– end topuser –>
    <?php
    $ava = $author['img'];
    echo '<img class="imgava" src="' . $ava . '">' . ' ';
    echo '<p>';
    htmlout($author['roleid']); //выводим уровень доступа
пользователя
    echo '</p>' . '<br>';
```

```
?>
<input type="hidden" name="id" value="<?php echo $author['id']; ?>">
<input type="submit" name="action" value="Редактировать">
<input type="submit" name="action" value="Удалить" onclick="">

</div><!– end wrapuser –>

</form>
</div><!– end formuser –>
<!–</li>–>

<?php endforeach; ?>
<!–</ul>–>
</div><!– end ourwrapper –>

<p><a href="..">Вернуться</a></p>
</body>

</html>
```

На этой странице выводится список пользователей и предоставляется возможность их редактирования.

Страница получает массив authors от страницы index.php

обрабатывает и выводит на печать.

14. Форма добавления пользователя

Для добавления пользователя служит файл формы «form_add_user.php»

Листинг 29. form_add_user.php Путь: news/chat/admin/users/ form_add_user.php

```
<?php
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/admin/clean.php'; ?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css"/>
<title><?php htmlout($pageTitle); ?></title>
</head>
<body>
<h1><?php htmlout($pageTitle); ?></h1>
<form action="?<?php htmlout($action); ?>" method="post" enctype="multipart/form-data">
```

```
<fieldset class="usernames">
<div class="username">
<label for="name">Имя:      
      <input type="text" name="login"
id="login" value="<?php if (isset($login)){
          htmlout($login);} ?>" >
      </label>
</div> <!– end.username –>

<div class="username">
<label for="email">Email:    
     <input type="email" name="email"
id="email" value="<?php htmlout($email); ?>"></label>
</div><!– end.username –>

<div class="username">
<label for="password">Пароль: 
     <input type="password"
name="password" id="password"></label>
</div><!– end.username –>
</fieldset><!– end.fieldset usernames–>
<br/>

<fieldset class="ava">
<legend>Загружаем аватар:</legend>
```

```
<p>
<label><p>Выберите аватар. Изображение должно быть формата jpg, gif или png:</p>
<p>Загружаем готовые аватары или картинки только квадратной формы</p><br>
    </label>
<input type="FILE" name="fupload">
</p>
</fieldset><!– end.fieldset ava –>

<fieldset class="level">
<legend>Уровень доступа пользователя:</legend>
<?php for ($i = 0; $i < count($roles); $i++): ?>
<div>
<label for="role<?php echo $i; ?>">
            <input type="checkbox"
name="roles[]" id="role<?php echo $i; ?>"
value="<?php htmlout($roles[$i]['id']);?>"<?php
if ($roles[$i]['selected']) {
echo ' checked';
} ?>
        >

        <?php htmlout($roles[$i]['id']); ?></label>:
<?php htmlout($roles[$i]['description']); ?>
<?php endfor; ?>
```

```
</fieldset><!– end.fieldset level –>

<div>
<input type="hidden" name="id" value="<?php
htmlout($id); ?>">
<input type="submit" value="<?php htmlout($button); ?>">
</div>

</form>
<a href="/chat/admin/">Вернуться</a>
</body>
</html>
```

Форма принимает данные для создания нового пользователя «Имя», «Электронная почта», «Пароль», «Аватар», «Роль пользователя» и передает их обработчику на той же странице index.php форма вставляется при нажатии ссылки «Добавить нового пользователя», которое обновляет страницу и создает переменную $_GET['add'], являющуюся условием вставки формы.

Поскольку в форме будет передаваться картинка аватар, то необходимо указать атрибут формы enctype="multipart/form-data".

15. Форма удаления пользователя

Для удаления выбранного пользователя служит форма «form_delete.php»

Листинг 29. form_delete.php Путь: news/chat/admin/users/form_delete.php

```
<?php
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/clean.php';
?>
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
<title>Подтверждение удаления</title>
</head>

<body class="chatbody">
<h1>Удаление</h1>
```

```
<ul>
<?php foreach ($authors as $author) : ?>
<li style="list-style-type: none">
<form action="" method="post" class="chatform">
<div>
<p>Вы действительно хотите удалить этого пользователя?</p>
<hr>
<p>
<input type="hidden" name="id" value="<?php echo $author['id']; ?>">
<input type="submit" name="action" value="ДА">
<input type="submit" name="action" value="НЕТ">
</p>
<?php
$ava = $author['img'];
echo '<img src="' . $ava . ' ">'.'<p>';
htmlout($author['login']);
        echo ' ';
htmlout($author['roleid']);
        echo '<p>';
?>
</div>
</form>
</li>
<?php endforeach; ?>
```

```
</ul>

<p><a href="..">Вернуться</a></p>
</body>

</html>
```

Форма выводит кнопки «Да» и «Нет», по нажатии которых страница обновляется и если была нажата «Да» происходит удаление пользователя.

Логика работы следующая: если нажата кнопка «Удалить» вставляется файл обработчик «delete.php», который делает выборку данных пользователя и вставляет форму подтверждения удаления, если удаление подтверждено, нажата кнопка «Да», то страница перегружается и само удаление происходит уже в файле «index.php».

16. Скрипт подготовки удаления пользователя

Подготовку удаления пользователя выполняет файл «delete.php»

Листинг 30. delete.php Путь: news/chat/admin/users/delete.php

```
<?php
error_reporting(E_ALL);
//require_once $_SERVER['DOCUMENT_ROOT'] . '/admin/access.php';

include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';

/* Делаем выборку данных пользователя из связанных таблиц */

try{

$sql = 'SELECT users.id,users.login, users.img, authorrole.roleid FROM `users` INNER JOIN authorrole ON
```

```
users.id =authorrole.authorid
    WHERE users.id = :id';
    $s =$dsn->prepare($sql);
    $s->execute(array('id' => $_POST['id']));
    $result = $s->Fetchall();
    }
    catch (pdoException $e)
    {
    $error = 'Ошибка получения пользователя из БД';
    include $_SERVER['DOCUMENT_ROOT'] .'/chat/admin/
users/error.html.php';
    exit();
    }

    foreach ($result as $row)
    {
    $authors[] = array('id' => $row['id'], 'login' => $row['login'],
'roleid'=> $row['roleid'], 'img' => $row['img'] );
    }

    /* Вставляем форму согласия на удаление */

    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/form_delete.php';
```

Готовит данные для идентификации удаляемого пользо-

вателя и передает их на печать в форме form_delete.php.

Запрос как обычно, в таблицы «users» и «authorrole». Выводит форму «Вы действительно хотите удалить этого пользователя?». Удаление происходит на странице chat/admin/users/index.php в зависимости от ответа, выбранного в форме form_delete.php' (Листинг 29).

17. Вывод аватара

Для вывода аватаров пользователей я воспользовался готовым решением https://ruseller.com/lessons.php?rub_id=37&id=350. За вывод аватаров пользователей отвечает файл «avatar.php».

Листинг 31. avatar.php Путь: news/chat/admin/users/avatar.php

```
<?php
error_reporting(E_ALL);
/*.....avatar.....*/

if (empty($_FILES['fupload']['name']))
{
//если переменной не существует (пользователь не отправил изображение),то присваиваем ему заранее приготовленную картинку с надписью "нет аватара"
$avatar = 'avatars/noavatar.png'; //картинка в исходниках
}
else {
//иначе – загружаем изображение пользователя
$path = 'avatars/'; //папка, куда будет загружаться начальная картинка и ее сжатая копия
```

```
if(preg_match('/[.](JPG)|(jpg)|(gif)|(GIF)|(png)|(PNG)$/',
$_FILES['fupload']['name']))//проверка формата исходного
изображения
{
$filename = $_FILES['fupload']['name'];
$source = $_FILES['fupload']['tmp_name'];
$target = $path.$filename;
move_uploaded_file($source, $target);//загрузка оригинала
в папку $path

if(preg_match('/[.](GIF)|(gif)$/', $filename)) {

$im = imagecreatefromgif($path.$filename) ; //если ориги-
нал был в формате gif, то создаем изображение в этом же
формате. Необходимо для последующего сжатия
}
if(preg_match('/[.](PNG)|(png)$/', $filename)) {
$im = imagecreatefrompng($path.$filename) ;//если ориги-
нал был в формате png, то создаем изображение в этом же
формате. Необходимо для последующего сжатия
}
if(preg_match('/[.](JPG)|(jpg)|(jpeg)|(JPEG)$/', $filename))
{
$im = imagecreatefromjpeg($path.$filename); //если ориги-
нал был в формате jpg, то создаем изображение в этом же
```

формате. Необходимо для последующего сжатия
}

// Создание квадрата 90x90
// dest – результирующее изображение
// w – ширина изображения
// ratio – коэффициент пропорциональности

$w = 90; // квадрат 90x90
$h = 90;

// создаём исходное изображение на основе исходного файла и определяем его размеры

$w_src = imagesx($im); //вычисляем ширину
$h_src = imagesy($im); //вычисляем высоту изображения

if ($w_src !== $h_src) {
exit ('<h4>Стороны изображения для загрузки должны быть равны. Квадрат. Например 256*256.
Рекомендую использовать готовые аватары со специализированных сайтов.
Или подготовьте картинку в графическом редакторе</h4>
p.s. грузим аватары, а не картины');
}

```
// создаём пустую квадратную картинку именно truecolor!, иначе будет 8-битный результат

$dest = imagecreatetruecolor($w,$w);
$white = imagecolorallocate($dest, 255, 255, 255);
imagefill($dest, 0, 0, $white);

imagecopyresampled($dest, $im, 0, 0, 0, 0, $w, $w, $w_src, $h_src);

$date=time(); //вычисляем время в настоящий момент.
imagejpeg($dest, $path.$date.".jpg");//сохраняем изображение формата jpg в нужную папку, именем будет текущее время. Сделано, чтобы у аватаров не было одинаковых имен.

$avatar = $path.$date.'.jpg';//заносим в переменную путь до аватара.

$delfull = $path.$filename;
unlink ($delfull);//удаляем оригинал загруженного изображения, он нам больше не нужен. Задачей было – получить миниатюру.
}
else
{
    //в случае несоответствия формата, выдаем соответ-
```

ствующее сообщение

```
exit ("Аватар должен быть в формате <strong>JPG,GIF или PNG</strong>"); //останавливаем выполнение сценариев
}
//конец процесса загрузки и присвоения переменной $avatar адреса загруженной авы
}
```

Скрипт создает и присваивает пользователю аватар из выбранной им картинки и загружает его под уникальным именем в папку avatars. Листинг подробнейший, мне кажется, здесь пояснения не нужны.

18. Работа с обычным пользователем
18.1 Скрипт работы с пользователем

Скрипт аналогичен коду используемому для работы с администратором в индексном файле папки «users» за исключением того, что пользователь видит и может редактировать только свои данные. Кроме того он не может поменять свой логин. За это отвечает свойство Readonly в поле ввода логина.

За обработку данных отвечает файл «edituser.php».

Листинг 32. edituser.php Путь: news/chat/admin/users/edituser.php

```
<?php
error_reporting(E_ALL);
require_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/access.php';
/*.....Редактирование пользователя.....*/
if (isset($_POST['action']) and $_POST['action'] == 'Редактировать') {
```

```
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';
    try {
    $sql = 'SELECT id, login, email, img FROM users WHERE
id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка при получении сведений об авторе.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
    $row = $s->fetch();

    $pageTitle = 'Редактирование профиля';
    $action = 'editform';
    $login = $row['login'];
    $email = $row['email'];
    $id = $row['id'];
    $avatar = $row['img'];
    $button = 'Обновить профиль';

    // список ролей, прав автора
    try {
```

```
    $sql = 'SELECT roleid FROM authorrole WHERE authorid
= :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $id);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка выборки прав юзера.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }

    $selectedRoles = array();
    foreach ($s as $row) {
    $selectedRoles[] = $row['roleid'];
    }

    // построение списка ролей автора
    try {
    $result = $dsn->query('SELECT id, description FROM role');
    } catch (pdoException $e) {
    $error = 'Ошибка построения списка ролей юзера.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
```

```
foreach ($result as $row) {
$roles[] = array(
'id' => $row['id'],
'description' => $row['description'],
'selected' => in_array($row['id'], $selectedRoles)
);
}
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/form_edit_user.php';
exit();
}
if (isset($_GET['editform'])) {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/users/avatar.php';

/* 2.1 Проверяем все ли поля формы заполнены */

if (empty($_POST['login']) or empty($_POST['password']) or empty($_POST['email'])) {
exit("Вы ввели не всю информацию, вернитесь назад и заполните все поля!");
}
/* 2.2 Проверка корректности email */
if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i",
```

```
$_POST['email'])) {
    exit("Неверно введен e-mail!");
    }
    try {
    $sql = 'UPDATE users SET
    login = :login,
    email = :email,
        img = :img
    WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['id']);
    $s->bindValue(':login', $_POST['login']);
    $s->bindValue(':email', $_POST['email']);
    $s->bindValue(':img', $avatar);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка добавления юзера.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
    if ($_POST['password'] != '') {
    $password = md5($_POST['password'] . 'swl');
    try {
    $sql = 'UPDATE users SET
    password = :password
```

```
    WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':password', $password);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка обновления пароля.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
    }
    header('Location: .');
    exit();
    }
    /* Удаление 2…Результат работы формы, вызываемой ко-
дом ниже(Удаление 1) */
    if (isset($_POST['action']) and $_POST['action'] == 'ДА') {
    // удаление ролей автора
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';
    try {
    $sql = 'DELETE FROM authorrole WHERE authorid = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
```

```
    } catch (pdoException $e) {
    $error = 'Ошибка удаления роли юзера.';
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
    try {
    $sql = 'DELETE FROM users WHERE id = :id';
    $s = $dsn->prepare($sql);
    $s->bindValue(':id', $_POST['id']);
    $s->execute();
    } catch (pdoException $e) {
    $error = 'Ошибка удаления юзера.';
    $e->getMessage();
    $e->getLine();
    include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
    exit();
    }
    header('Location: .');
    exit();
    }
    /* Первичный запрос на удаление. Выводит форму…ДА
или НЕТ (Удаление 1) */
    if (isset($_POST['action']) and $_POST['action'] == 'Уда-
лить') {
```

```
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/delete.php';
exit();
}
// страница юзера
include $_SERVER['DOCUMENT_ROOT'] . '/chat/
dsn.php';
try {
$sql = 'SELECT users.id,users.login, users.img,
authorrole.roleid FROM `users` INNER JOIN authorrole ON
users.id =authorrole.authorid
WHERE users.login = :login';
$s = $dsn->prepare($sql);
$s->execute(array('login' => $_SESSION['login']));
$result = $s->Fetchall();
} catch (pdoException $e) {
$error = 'Ошибка извлечения данных юзера из БД!';
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
users/error.html.php';
exit();
}
foreach ($result as $row) {
$authors[] = array('id' => $row['id'], 'login' => $row['login'],
'roleid' => $row['roleid'], 'img' => $row['img']);
}
include $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/
```

users/list_edit_user.php';

Подключаемся к БД проверяем какие кнопки были нажаты, проверяем данные пользователя, обрабатываем результаты. Формируем запросы и создаем первичные массивы для дальнейшей их обработки в соответствующих формах. Обновляем, удаляем или оставляем без изменений данные пользователя в БД.

18.2 Страница пользователя

Файл страницы пользователя «list_edit_user.php»

Листинг 33. list_edit_user.php Путь: news/chat/admin/users/ list_edit_user.php

```
<?php
include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/clean.php';
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
<title>Редактировать профиль</title>
</head>
<body class="chatbody">
<h2>Редактировать профиль</h2>
<ul>
<?php foreach ($authors as $author) : ?>
<li style="list-style-type: none">
<form action="" method="post" class="chatform">
```

```
<div class="list_edituser">
<?php
$ava = $author['img'];
echo '<img src="' . $ava . '">';
htmlout($author['login']);
echo " ";
htmlout($author['roleid']);
?>
<input type="hidden" name="id" value="<?php echo $author['id']; ?>">
<input type="submit" name="action" value="Редактировать">
<input type="submit" name="action" value="Удалить">
</div>
</form>
</li>
<?php endforeach; ?>
</ul>
<p><a href=".." class="apreturn">Вернуться</a></p>
</body>
</html>
```

Выводит страницу содержащую данные пользователя и форму для выбора действий с этими данными: «Редактировать», «Удалить».

18.3 Форма редактирования пользователя

Файл формы «form_edit_user.php».

Листинг 34. form_edit_user.php Путь: news/chat/admin/users/ form_edit_user.php

```
<?php
include_once $_SERVER['DOCUMENT_ROOT'] .'/chat/admin/clean.php'; ?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/chat/style.css"/>
<title><?php htmlout($pageTitle); ?></title>
</head>

<body class="chatbody">
<h1><?php htmlout($pageTitle); ?></h1>
<p style='font-style:italic'>Здесь вы можете поменять аватар, пароль и email</p>
```

```
<form action="?<?php htmlout($action); ?>" method="post"
enctype="multipart/form-data" class="chatform">
<div>
<label for="name">Имя:      
    <input type="text " READONLY name="login"
id="login" value="<?php if (isset($login)){
        htmlout($login);} ?>">
    </label>
</div>
<br/>
<div>
<label for="email">Email:    
        <input type="email" name="email"
id="email" value="<?php htmlout($email); ?>"></label>
</div>
<br/>
<div>
<label for="password">Пароль: <input type="password"
name="password" id="password"></label>
</div>
<p>
<label>Выберите аватар. Изображение должно быть фор-
мата jpg, gif или png:<br></label>
<input type="FILE" name="fupload">
</p>
<div>
```

```
<input type="hidden" name="id" value="<?php
htmlout($id); ?>">
<input type="submit" value="<?php htmlout($button); ?>">
</div>
</form>

<a href=".." class="apreturn">Вернуться</a>
</body>
</html>
```

Выводится форма, в которой пользователь может поменять свои аватар, пароль и email.

19. Регистрация пользователей

https://ruseller.com/lessons.php?rub_id=37&id=350

Для регистрации пользователей используются файлы:

- Страница с формой регистрации «form_registration.html».
- Страница сохранения пользователя «save_user.php».
- Страница активации пользователя «activation.php».
- Капча

Листинг 35. form_registration.html Путь: news/chat/admin/users/ form_registration.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
<title>Регистрация на сайте</title>
</head>

<body class="chatbody">
<h2>Регистрация на сайте</h2>
<form
```

```
   class="chatform"
   action="save_user.php"
   method="post"
   enctype="multipart/form-data"
   id="reg"
   >
   <fieldset class="chatfieldset">
   <legend>Данные для авторизации</legend>
   <label
   >Ваш логин *: <input name="login" type="text" size="15"
maxlength="15"
   /></label>
   <hr />
   <label
   >Ваш пароль *:
   <input name="password" type="password" size="15"
maxlength="15"
   /></label>
   <hr />
   <label
   >Ваш E-mail *:
   <input name="email" type="email" size="15"
maxlength="100"
   /></label>
   <br />
   </fieldset>
```

```
<br />
<fieldset class="chatfieldset">
<legend>Выберите аватар</legend>
<label>
Изображение должно быть квадратной формы<br />
и иметь расширение jpg, gif или png:<br />
<br />
<input type="FILE" name="fupload"
/></label>
</fieldset>
<br />
<!– В переменную fupload отправится изображение, которое выбрал пользователь. –>
<fieldset class="chatfieldset">
<legend>Введите код с картинки *:</legend>
<input type="text" name="capcha" />
<div><img src="/chat/capcha/captcha.php" /></div>
</fieldset>
<p>
<input type="submit" name="submit" value="Зарегистрироваться" />
<!– Кнопочка (type="submit") отправляет данные на страничку save_user.php –>
</p>
</form>
```

Звездочками (*) обозначены поля, обязательные для за-

полнения.

```
<p><a href="/../">Вернуться</a></p>
</body>
</html>
```

Выводит страницу «Регистрация на сайте» и форму регистрации. В форме предусмотрена защита при помощи «капчи» и предлагается отправка для регистрации следующих данных: «Логин», «email», «Пароль», «Аватар».

20. Капча

Файлы капчи: обработчик и папка с шрифтами, выводят на форме регистрации проверочный код, который пользователь должен будет ввести.

Листинг 36. captcha.php Путь:chat\capcha\captcha.php

```
<?php
if (session_id() == '') {
session_start();
}
$string = "";
for ($i = 0; $i < 5; $i++) {
$string .= chr(rand(97, 122));
}

$_SESSION['rand_code'] = $string;

$dir = $_SERVER['DOCUMENT_ROOT'] . '/chat/capcha/fonts/verdana.ttf';

$image = @imagecreatetruecolor(170, 60) or die('Невозможно инициализировать GD поток'); // Создаём изображение
$black = imagecolorallocate($image, 0, 0, 0);
$color = imagecolorallocate($image, 200, 100, 90);
```

```
$white = imagecolorallocate($image, 255, 255, 255);

imagefilledrectangle($image, 0, 0, 399, 99, $white);
imagettftext($image, 30, 0, 10, 40, $color, $dir, $_SESSION['rand_code']);

header("Content-type: image/png");
imagepng($image);
```

Здесь в цикле генерируется строка из случайных чисел, при этом числа сразу преобразуются в символьные эквиваленты. Диапазон чисел 97-122 соответствует буквам латинского алфавита в диапазоне a-z по ASCII http://www.asciitable.com/ . Полученный набор букв в виде строки сохраняется в переменную сессии, а затем оформляется в виде изображения.

21. Страница сохранения пользователя

За сохранение и обработку полученных при регистрации данных пользователя отвечает файл «save_user.php»

Листинг 37. save_user.php Путь: news/chat/admin/users/save_user.php

```
<?php
error_reporting(E_ALL);
if (session_id() == '') {
session_start();
}
/* заносим введенный пользователем логин в переменную $login, если он пустой, то уничтожаем переменную */
if (isset($_POST['login'])) {
$login = $_POST['login'];
if (mb_strlen($login) < 2 or mb_strlen($login) > 15) { //проверяем длину логина
exit("Логин должен состоять не менее чем из 3 символов и не более чем из 15."); //останавливаем выполнение сценариев
}
```

```
if ($login == '') {
unset($login);
}
}
/* заносим введенный пользователем пароль в переменную $password, если он пустой, то уничтожаем переменную */
if (isset($_POST['password'])) {
$password = $_POST['password'];
if (mb_strlen($password) < 3 or mb_strlen($password) > 15) { //проверяем длину пароля
exit("Пароль должен состоять не менее чем из 3 символов и не более чем из 15."); //останавливаем выполнение сценариев
}
if ($password == '') {
unset($password);
}
}
/* заносим введенный пользователем код в переменную $capcha, если он пустой, то уничтожаем переменную */
if (isset($_POST['capcha'])) {
$capcha = $_POST['capcha'];
if ($capcha == '') {
unset($capcha);
}
```

```
    }
    /* заносим введенный пользователем e-mail, если он пустой, то уничтожаем переменную */
    if (isset($_POST['email'])) {
    $email = $_POST['email'];
    if ($email == '') {
    unset($email);
    }
    }
    $capcha = stripslashes($capcha); //sequrity чистим капчу
    $capcha = htmlspecialchars($capcha);
    $capcha = trim($capcha); //удаляем лишние пробелы
    if (($capcha == $_SESSION["rand_code"]) && ($capcha != "")) { //проверяем капчу
    } else {
    exit("Капча введена неправильно");
    }

    if (empty($login) or empty($password) or empty($email) or empty($capcha)) { //если пользователь не ввел данные, то выдаем ошибку и останавливаем скрипт
    exit("Вы ввели не всю информацию, вернитесь назад и заполните все поля!");
    }
    if  (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $email)) { //проверка e-mail адреса регулярными выражени-
```

ями на корректность
exit("Неверно введен e-mail!");
}
$login = stripslashes($login); //если логин и пароль введены, то обрабатываем их, чтобы теги и скрипты не работали, мало ли что люди могут ввести
$login = htmlspecialchars($login);
$password = stripslashes($password);
$password = htmlspecialchars($password);
$login = trim($login); //удаляем лишние пробелы
$password = trim($password);

/*.....Шифруем пароль.....*/

//можно добавить несколько своих символов по вкусу, например, вписав "she".
//Если этот пароль будут взламывать методом подбора у себя на сервере этой же md5,то явно ничего хорошего не выйдет.
//Но советую ставить другие символы, можно в начале строки или в середине.
//При этом необходимо увеличить длину поля password в базе. Зашифрованный пароль может получится гораздо большего размера.
$password = md5($_POST['password'] . 'swl'); //шифруем пароль

```
include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php'; // подключаемся к базе
$sth = $dsn->prepare("SELECT id FROM users WHERE login=:login"); // проверка на существование пользователя с таким же логином
$sth->execute(array(':login' => $login));
if (!empty($sth->fetch(PDO::FETCH_ASSOC))) {
exit("Извините, введённый вами логин уже зарегистрирован. Введите другой логин.");
}
/*.....avatar.....*/
if (empty($_FILES['fupload']['name'])) {
//если переменной не существует (пользователь не отправил изображение),то присваиваем ему заранее приготовленную картинку с надписью "нет аватара"
$avatar = "avatars/noavatar.png"; //можете нарисовать net-avatara.jpg или взять в исходниках
} else {
//иначе – загружаем изображение пользователя
$path_to_90_directory = 'avatars/'; //папка, куда будет загружаться начальная картинка и ее сжатая копия
/* проверка формата исходного изображения */
if (preg_match('/[.](JPG)|(jpg)|(gif)|(GIF)|(png)|(PNG)$/', $_FILES['fupload']['name'])) {
$filename = $_FILES['fupload']['name'];
$source = $_FILES['fupload']['tmp_name'];
```

```
$target = $path_to_90_directory . $filename;
move_uploaded_file($source, $target); //загрузка оригинала в папку $path_to_90_directory
if (preg_match('/[.](GIF)|(gif)$/', $filename)) {
$im = imagecreatefromgif($path_to_90_directory . $filename); //если оригинал был в формате gif, то создаем изображение в этом же формате. Необходимо для последующего сжатия
}
if (preg_match('/[.](PNG)|(png)$/', $filename)) {
$im = imagecreatefrompng($path_to_90_directory . $filename); //если оригинал был в формате png, то создаем изображение в этом же формате. Необходимо для последующего сжатия
}
if (preg_match('/[.](JPG)|(jpg)|(jpeg)|(JPEG)$/', $filename))
{
$im = imagecreatefromjpeg($path_to_90_directory . $filename); //если оригинал был в формате jpg, то создаем изображение в этом же формате. Необходимо для последующего сжатия
}
// Создание квадрата 90x90
// dest – результирующее изображение
// w – ширина изображения
// ratio – коэффициент пропорциональности
```

```
$w = 90; // квадратная 90x90. Можно поставить и другой размер.
$h = 90;
// создаём исходное изображение на основе
// исходного файла и определяем его размеры
$w_src = imagesx($im); //вычисляем ширину
$h_src = imagesy($im); //вычисляем высоту изображения
if ($w_src !== $h_src) {
exit('<h4>Стороны изображения для загрузки должны быть равны. Например 256*256.<br>Рекомендую использовать готовые аватары со специализированных сайтов.<br>
Или подготовьте картинку в графическом редакторе</h4><br><i>p.s. грузим аватары, а не картины</i>');
}
// создаём пустую квадратную картинку
// важно именно truecolor!, иначе будем иметь 8-битный результат
$dest = imagecreatetruecolor($w, $w);
$white = imagecolorallocate($dest, 255, 255, 255);
imagefill($dest, 0, 0, $white);
imagecopyresampled($dest, $im, 0, 0, 0, 0, $w, $w, $w_src, $h_src);
$date = time(); //вычисляем время в настоящий момент.
imagejpeg($dest, $path_to_90_directory . $date . ".jpg"); // сохраняем изображение формата jpg в нужную папку, име-
```

нем будет текущее время. Сделано, чтобы у аватаров не было одинаковых имен.

```
$avatar = $path_to_90_directory . $date . ".jpg"; //заносим в переменную путь до аватара.
$delfull = $path_to_90_directory . $filename;
unlink($delfull); //удаляем оригинал загруженного изображения, он нам больше не нужен. Задачей было – получить миниатюру.
} else {
//в случае несоответствия формата, выдаем соответствующее сообщение
exit("Аватар должен быть в формате <strong>JPG,GIF или PNG</strong>"); //останавливаем выполнение сценариев
}
//конец процесса загрузки и присвоения переменной $avatar адреса загруженной авы
}
/*.....Сохраняем пользователя в базу.....*/
if (!isset($date)) {
$date = time();
}
try {
$sql = "INSERT INTO users (login,password,email,img,date) VALUES(:login,:password,:email,:img, :date)";
$result2 = $dsn->prepare($sql);
```

```
    $result2->execute([
    'login' => $login,
    'password' => $password,
    'email' => $email,
    'img' => $avatar,
    'date' => $date
    ]);
    $role = 'user';
    $authorid = $dsn->lastInsertId();
    $sql = "INSERT INTO authorrole (authorid,roleid) VALUE
(:authorid,:roleid)";
    $resultrole = $dsn->prepare($sql);
    $resultrole->execute([
    'authorid' => $authorid,
    'roleid' => $role
    ]);
    echo '<img src="' . $avatar . '">';
    echo ' ' . '<h3>' . $login . '</h3>' . ' ' . "Вы успешно зареги-
стрированы! <a href='/index.html'>Главная страница</a>";
    } catch (PDOException $e) {
    echo "You have an error: " . $e->getMessage() . "<br>";
    echo "On line: " . $e->getLine();
    }
    $activation = md5($authorid) . md5($login);
    $subject = "Подтверждение регистрации"; //тема сообще-
ния
```

```
$message = "Здравствуйте! Спасибо за регистрацию в модуле комментариев chat\nВаш логин: " . $login . "\n
Перейдите по ссылке, чтобы активировать ваш аккаунт:\nhttp://" . $_SERVER['HTTP_HOST'] . "/chat/admin/users/activation.php?login=" . $login . "&code=" . $activation . "\nС уважением,\n
Администратор модуля"; //содержание сообщение
mail($email, $subject, $message, "Content-type:text/plane; Charset=utf-8\r\n"); //отправляем сообщение

echo "<hr><h3>Вам на E-mail выслано письмо с ссылкой, для подтверждения регистрации.</h3> <br><b>Внимание! Ссылка действительна 1 час.</b>"; //говорим о отправленном письме пользователю
```

В этом скрипте проверяются данные пользователя и сохраняются в БД. Обработка изображений для аватара сделана только для квадрата, т.к. PHP это язык серверный и обработка изображений явно не его конек. Результаты при обработке прямоугольников мягко говоря разочаровывают, поэтому загрузка и обработка прямоугольных исходников не предусмотрена. Лучше взять готовый или подготовить аватар в графическом редакторе. Код в данном скрипте прокомментирован практически построчно, думается вопросов быть не должно.

С административным разделом закончили, приступаем к

комментариям.

22. Комментарии

Для работы с комментариями используем папку «say», созданную ранее в разделе 3. На данном этапе она пуста и кроме папки «smiles» в которой хранятся заранее подготовленные смайлы в ней ничего нет.

23. Контроллер 3

Для вывода комментариев создаем контроллер 3 «say_controller.php».

Листинг 38. say_controller.php Путь: news/chat/say_controller.php

```
<?php
error_reporting(E_ALL);

include_once $_SERVER['DOCUMENT_ROOT'] .'/chat/admin/clean.php';
include $_SERVER['DOCUMENT_ROOT'].'/chat/function/print_comment.php';
include $_SERVER['DOCUMENT_ROOT'].'/chat/function/print_smile_set.php';

$page_id = $_SERVER['PHP_SELF']; //индексируем страницу

/* Если пользователь авторизован вставляем ссылку Добавить комментарий */

if (isset($_SESSION['login'])) {
```

```
$userid = $_SESSION['userid'];
echo '<div class="addsay" id=""><a href="?addsay" class="aaddsays">Добавить комментарий</a></div>';
}
else {
$userid = '';
}

/* Устанавливаем условие видимости блока комментариев и кнопок показать и скрыть комментарии */

if (isset($_GET['opensay'])) {
$display_say = 'display:none;';
$display_but = 'display:flow-root;';
}
else {
$display_say = 'display:flow-root;';
$display_but = 'display:none;';
}

/* Применяем условие видимости к кнопкам показать и скрыть комментарии */

echo '<div class="opensay" style="'.$display_but.'"><a href="?" class="aopensays">Показать комментарии</a></
```

```
div>';
   echo '<div class="opensay" style="'.$display_say.'"><a
href="?opensay" class="aopensays">Скрыть коммента-
рии</a></div>';

   /* Вставляем форму добавить комментарий */

   if (isset($_GET['addsay'])){

   $pageid = strtok($_SERVER['PHP_SELF'],'?');

   include_once $_SERVER['DOCUMENT_ROOT'].'/chat/
say/form_addsay.html.php';
   }

   /* Выводим лист комментариев */
   //сортируем в обратном порядке наверху последний ком-
ментарий
   try {
   include $_SERVER['DOCUMENT_ROOT'].'/chat/
dsn.php';
      $sql = 'SELECT say.id, say.userid, say.saydate, say.saytext,
users.login, users.img FROM say INNER JOIN users ON
users.`id` = say.userid WHERE say.page_id = :page_id ORDER
BY say.id DESC';
```

```
        $s = $dsn->prepare($sql);
    $s -> bindValue(':page_id', $page_id);
    $s -> execute();
    }
    catch (PDOException $e) {
    echo $e->getMessage();
    echo $e->getLine();
    exit();
    }

    foreach ($s as $row) {

    $say[] = array(
    'id' => $row['id'],
    'saytext' => $row['saytext'],
    'saydate' => $row['saydate'],
    'img' => $row['img'],
    'login' => $row['login'],
            'userid' => $row['userid']);
    }

    include_once    $_SERVER['DOCUMENT_ROOT'].'/chat/
say/form_say.html.php';
```

Контроллер определяет условия видимости управляющих комментариями кнопок и форм ввода комментариев, индек-

сирует страницу и формирует массив say[] с комментариями, который будет выводится в форме form_say.html.php после обработки.

Как видим у него также есть файлы которые нужно обрабатывать. Если сейчас подключить контроллер, то на странице будет ошибка. Файлы еще не готовы. В папке «chat» создаем папку «function» в которой будут находиться функции по обработке комментариев.

24. Папка функций
24.1 Печать смайлов

Для вывода набора доступных смайлов написана функция «print_smile_set.php»

Листинг 39. print_smile_set.php Путь: news/function/print_smile_set.php

```
<?php
error_reporting(E_ALL);
/* печать блока доступных смайлов в виде кнопок */
function print_smile_set()
{
try {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';
$sql = "SELECT smile, path FROM smiles";
$s = $dsn->query($sql);
$ress = $s->fetchall();
foreach ($ress as $row) :
$smiles_key = $row['smile'];
$smile_path = $row['path']; ?>
<input type="submit" name="smile" title="<?=
```

```
$smiles_key; ?>" value="<?= $smiles_key; ?>"
   style="background:url(<?= $smile_path; ?> );
   background-repeat:                                    no-
repeat;float:left;height:45px;border:none;
   width:50px;font-size:0;" />
   <?php
   endforeach;
   } catch (PDOException $e) {
   echo $e->getMessage();
   echo $e->getLine();
   exit();
   }
   }
```

Смайлы сделаны в виде кнопок. В свойстве «background» кнопки идет картинка смайла. Нажатие кнопки вставляет смайл.

Внимание! Стили CSS для смайлов задаются непосредственно в скрипте в строке <input … style=””>

24.2 Печать комментариев

Для печати комментариев служит функция «print_comment.php»

Листинг 40. print_comment.php Путь: news/function/ print_comment.php

```
<?php
/* печать комментария со смайлами */
function comment_to_smile($comment)
{
try {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/ dsn.php';
$sql = "SELECT smile, path FROM smiles";
$s = $dsn->query($sql);
$ress = $s->fetchall();
foreach ($ress as $row) :

$smiles_key = $row['smile'];
$smile_path = $row['path'];

endforeach;
} catch (PDOException $e) {
```

```
echo $e->getMessage();
echo $e->getLine();
exit();
}
$smile_path = array_column($ress, 'path'); // Массив с кодами смайлов
$smiles_keys = array_column($ress, 'smile'); // Массив с соответствующими путями к изображениям смайлов

for ($i = 0; $i < count($smile_path); $i++) {
$smile_path[$i] = "<img src="" . $smile_path[$i] . "' alt='' />";//получаем изображение смайла
}
$comment = str_replace($smiles_keys, $smile_path, $comment); //Меняем в комментарии ключи смайлов на пути к смайлам
echo $comment; //печатаем комментарий со смайлами
}
```

Получаем из таблицы smiles массивы условных обозначений смайлов и путей к их изображениям. В цикле обрабатываем и выводим на печать комментарии со смайлами.

25. Страница сепарации данных

Страница separate_action.php получает данные из формы formaddsay.html.php и обрабатывает их. Введение этой страницы было необходимо для обнуления $_POST на странице обработки комментариев после их отправки. Если этого не сделать, то при размещении этого кода на странице с контроллером 3 при обычном обновлении страницы возникала бы проблема с его обработкой т.к. браузер обычно запоминает данные, которые отправляются с текущей страницы для того, чтобы отправить их снова при обновлении страницы и:

без обнуления переменной POST один и тот же комментарий будет добавляться до бесконечности при каждой перезагрузке страницы.

Если же мы уйдем со страницы на другую и отправим данные формы на следующую (в данном случае вернемся обратно) страницу, то браузер запомнит уже новые заголовки и при обновлении будет обрабатывать их, а при возврате они будут для страницы возврата пусты и соответственно при ее обновлении ничего не произойдет.

Листинг 41. separate_action.php Путь: news/chat/function/print_comment.php

```
<?php
```

```
if(session_id() == '') {session_start();}
include_once $_SERVER['DOCUMENT_ROOT'].'./chat/admin/clean.php';
/* включаем/выключаем видимость блока смайлов */
if (isset($_POST['smileblock'])) {
$_SESSION['smileblock'] = 'display:block';
}
if (isset($_POST['smileblock_close'])) {
$_SESSION['smileblock'] = 'display:none';
}
/* Комментарии */
//если была нажата кнопка "Добавить" вставляем форму добавления комментариев,
//если была нажата кнопка "Ответить" вставляем форму ответа на комментарии,
if (isset($_POST['action']) and $_POST['action'] == 'Добавить') {
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/say/makeformaddsay.php';
}
elseif (isset($_POST['action']) and $_POST['action'] == 'Ответить') {

include_once $_SERVER['DOCUMENT_ROOT'].'/chat/say/make_reply.php';
}
```

```
else {//Проверяем куда пойдет смайл в комментарии или в ответы
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/say/smile_make.php';
}
```

Скрипт управляет отображением/скрытием блока смайлов, вставкой формы добавления комментариев и вставкой формы добавления ответов на комментарии.

26. Форма для вывода комментариев

Форма form_say.html.php служит для вывода комментариев

Листинг 42. form_say.html.php Путь: news/chat/say/form_say.html.php

```
<!DOCTYPE html>
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
</head>
<div class="wrapsayform" style="<?= $display_say ?>">
<?php
include $_SERVER['DOCUMENT_ROOT'] . '/chat/say/list_reply.php';
```

```
    if (!empty($say)) {
    foreach ($say as $saylist) : ?>

    <div class="sayform" id="">

    <!– Выводим логин и дату –>

    <div class="topprintcomment" id="">
    <b><?php echo $saylist['login']; ?></b>
    <?php $t = $saylist['saydate'];
    echo '<span style="float:right;">' . date("d.m.Y", "$t") . '</
span>'; ?>
    </div>

    <!– Выводим аватар и комментарии –>

    <div class="printcomment" id="">
    <p></p><img src="<?= '/chat/admin/users/' .
$saylist['img'] ?>" class="sayavatar">
    <?php
    $comment = $saylist['saytext'];
    comment_to_smile($comment); //печатаем комментарии
    $post_id = $saylist['id'];

    /* Активация кнопок */
```

```
if (userHasRole('admin')) {
$buttonactive = 'submit'; //если админ делаем активными все кнопки
} elseif ($saylist['userid'] == $userid) {
$buttonactive = 'submit'; //делаем активными кнопки для юзера
} else {
$buttonactive = 'hidden'; //отключаем кнопки
} ?></p>
</div><!– END printcomment –>

<!– Выводим блок кнопок –>

<div class="block_button_say" id="">

<div class="wrappersaybutton">
<div class="reply_button" id="">
<div class="postnumber" id=""><?php echo '#' . $saylist['id']; ?> </div>
<a href="?reply=<?php htmlout($post_id); ?>" class="areply">Ответить</a>

</div>
<div class="sayright">
<div class="button_say_edit">
```

```
<form name="" method="post" action="/chat/say/sayedit.php" class="logout">

<input type="hidden" name="pageid" id="" value=" <?php echo $pageid; ?>" />
<input type="hidden" name="textedit" value="<?php echo $saylist['id']; ?>">
<input type="hidden" name="saytext" id="" value="<?php echo $saylist['saytext']; ?>" />

<input type="<?= $buttonactive ?>" name="actionedit" value="Редактировать" />

</form>
</div>
<div class="button_say_delete">

<form name="sayform" method="post" action="/chat/say/reset.php" class="logout">

<input type="hidden" name="pageid" id="" value=" <?php echo $pageid; ?>" />
<input type="hidden" name="deleteid" id="" value=" <?php echo $saylist['id']; ?>" />

<input type="<?= $buttonactive ?>" name="delete" id=""
```

```
value="Удалить" />
   </form>
   </div>
   </div>
   </div>

            <!– Форма ответить на комментарий –>

   <div class="add_reply" id="">
   <?php
   if (isset($_SESSION['login'])) {
   if (isset($_GET['reply']) and $_GET['reply'] == $post_id) {
   include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/
say/form_add_reply.html.php';
   }
   } ?>
   </div>

   </div><!– END block_button_say –>

   </div><!– END sayform –>

   <!– Выводим ответы на комментарии и кнопки –>
   <?php
   include $_SERVER['DOCUMENT_ROOT'] . '/chat/say/
print_reply.html.php';
```

```
endforeach;
} ?>
</div><!– END wrapsayform –>

</html>
```

Здесь вставляется, подготовленный в list_reply.php массив ответов на комментарии $reply, затем в цикле foreach – endforeach обрабатываются комментарии, содержащиеся в массиве $say, созданном в контроллере 3, которые выводятся в соответствующих блоках div и происходит активация кнопок для каждого комментария, затем в этом же цикле подключается файл print_reply.html.php, который печатает ответы и выводит кнопки ответов.

27. Форма добавления комментариев

Для добавления комментариев служит форма formaddsay.html.php

Листинг 43. form_addsay.html.php Путь: news/chat/say/form_addsay.html.php

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
<title>Добавление</title>
</head>
<!– При нажатии кнопок формы идем на страницу сепарации переменных –>
<form action="/chat/say/separate_action.php" method="post" class="say">
```

```
<div>
<label for="saytext" class="say">Введите ваш коммента-
рий:</label>
<textarea id="saytext" name="saytext" rows="5" cols="60"
autofocus><?php if (isset($_SESSION['txt'])) {
echo htmlout($_SESSION['txt']);} ?></textarea>
</div>
<br />
<div class="">
<div class="">
<div class="says">
<input type="hidden" name="userid" value="Добавить">
<input type="hidden" name="pageid" value="<?php echo
$pageid ?>">
<input type="hidden" name="sayid" value="<?php if
(isset($sayid)) {
echo $sayid;
} ?>">
<input type="submit" name="action" value="Добавить">
<input type="submit" name="smileblock" value="Показать
смайлы" style="">
<input type="submit" name="smileblock_close"
value="Скрыть смайлы" style="">
</div>

<div class="notaddsay" id=""><a href="?">Отмена</a></
```

```
div>
   </div>
   <br>
   <hr>
   <!– Вставляем лист доступных смайлов –>
   <div class="smilesetdiv" id="">

   <fieldset class="smileset" style="<?php if
(isset($_SESSION['smileblock'])) {
   echo $_SESSION['smileblock'];
   } ?>">
   <?php print_smile_set(); ?>
   </fieldset>
   </div>
   </div>
   <br />
   </form>
   </body>
   </html>
```

Это HTML форма состоящая из следующих частей:

• Поле ввода комментариев тип textarea,autofocus.

• Кнопки «Добавить», «Показать смайлы», «Скрыть смайлы» тип input submit.

• Ссылка «Отмена», при нажатии обнуляет все значения в $_GET.

• Блок смайлов, который выводит доступный набор смай-

ЛОВ.

28. Обработка комментариев

Для обработки комментариев служит скрипт makeformaddsay.php

Листинг 44. makeformaddsay.php Путь: news/chat/say/ makeformaddsay.php

```
<?php
error_reporting(E_ALL);

if(session_id() == '') {session_start();}

unset($_SESSION['txt']);

if (isset($_POST['nosay'])) {
$_GET['addsay'] ='';
header("Location:     ".$_SERVER["HTTP_REFERER"]);//
Делаем редирект
exit();
}

/* Проверяем наличие текста в форме */

if (empty($_POST['saytext'])) {
```

```
header("Location: ".$_SERVER["HTTP_REFERER"]);// Делаем редирект
exit();
}

if (isset($_POST['action']) and $_POST['action'] == "Добавить") {
include_once $_SERVER['DOCUMENT_ROOT'].'/chat/dsn.php';

include_once $_SERVER['DOCUMENT_ROOT'] .'/chat/admin/clean.php';

/* Получаем id текущего пользователя */
if(isset($_SESSION['userid'])) {
$userid = $_SESSION['userid'];
}

/* заносим текст в базу */

try {
$sql = 'INSERT INTO say SET
saytext = :saytext,
userid = :userid,
page_id = :page_id,
saydate = :saydate';
```

```
    $s = $dsn->prepare($sql);

    $saytext = html($_POST['saytext']);
    $saydate = time();
    $page_id = $_POST['page_id'];

    $s->bindValue(':saytext',$saytext);
    $s->bindValue(':saydate',$saydate);
    $s->bindValue(':userid',$userid);
    $s->bindValue(':page_id',$page_id);
    $s->execute();
    }
    catch (PDOException $e) {
        echo 'makeformaddsay ошибка вставки комментария';
        echo 'sdsd'.$page_id;
    echo $e->getMessage();
    echo $e->getLine();
    exit();
    }

    header("Location:     ".$_SERVER["HTTP_REFERER"]);//
Делаем редирект
    exit();
    }
```

```
header("Location:     ".$_SERVER["HTTP_REFERER"]);//
Делаем редирект
exit();
```

Скрипт также выполнен в отдельном файле во избежание проблем с сохранением $_POST в браузере. В нем проверяются данные отправленные из формы formaddsay.html.php «Добавить комментарии» и заносятся в БД.

29. Форма редактирования комментариев

Для редактирования комментариев служит форма form_editsay.html.php

Листинг 45. form_editsay.html.php Путь: news/chat/say/ form_editsay.html.php

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css" href="<?= $_SERVER['HTTP_ORIGIN'].'/chat/style.css'; ?>"/>
</head>
<body>
<form action="" method="post" class="sayedit">
<div>
<label for="saytext" class="say"><h4>Редактировать комментарий:</h4></label>
    <hr>
<textarea id="saytext" name="saytext" rows="6" cols="80"
```

```
maxlength="100" autofocus><?= $_POST['saytext'];?></
textarea>
   </div>
   <br/>
        <div class="">
   <div class="says">
   <input type="hidden" name="sayid" value="<?=
$_POST['textedit'];?>">
   <input type="submit" name="sayedit" value="Редактиро-
вать">
        <input type="submit" name="actionnot" value="Отме-
на">
   </div>
        <br><hr>
        <div class="smilesetdiv" id="">
            <fieldset class="smileset" style="">
                <?php print_smile_set();?>
        </fieldset>
        </div>
   </div>
   <br/>
   </form>
   </body>
   </html>
```

Это HTML форма в которой можно отредактировать комментарий. Состоит из:

• Поле редактирования textarea со значением по умолчанию равным $_POST['saytext'].
• Кнопок «Редактировать» и «Отмена».
• Набор для выбора доступных смайлов.

30. Скрипт редактирования комментариев

Для редактирования комментариев служит файл sayedit.php

Листинг 46. sayedit.php Путь: news/chat/say/ sayedit.php

```
<?php
error_reporting(E_ALL);
if(session_id() == '') {session_start();}
include_once $_SERVER['DOCUMENT_ROOT'] .'/chat/admin/clean.php';
include $_SERVER['DOCUMENT_ROOT'].'/chat/function/print_smile_set.php';
/* Сохраняем переменную идентефикатор поста */
if (isset($_POST['textedit']) and $_POST['textedit'] != ""){
$_SESSION['idsave'] = $_POST['textedit'];
} else {
$_POST['textedit'] = $_SESSION['idsave'];
}
/* Заносим в базу и делаем редирект */
if (isset($_POST['sayedit']) and $_POST['sayedit'] == "Редактировать") {
```

```
    try {
    include_once     $_SERVER['DOCUMENT_ROOT'].'/chat/
dsn.php';
    $sql = 'UPDATE say SET saytext = :saytext WHERE id = :id';
    $s = $dsn ->prepare($sql);
    $s->bindValue(':saytext', $_POST['saytext']);
    $s->bindValue(':id', $_POST['sayid']);
    $res = $s->execute();
    unset ($_SESSION['idsave']);
    header("Location: /");
    exit();
    }
    catch (PDOexeption $e) {
        echo $e->getMessage();
    echo $e->getLine();
    exit('error.html.php');
    }
    exit();
    }
    /* Контролируем нажатия кнопок */
    if (isset($_POST['actionedit']) and $_POST['actionedit'] ==
"Редактировать") {
    include        $_SERVER['DOCUMENT_ROOT'].'/chat/say/
form_editsay.html.php';
    exit();
    }
```

```
if (isset($_POST['smile'])) {
$_POST['saytext'] = $_POST['saytext'].$_POST['smile'];
include $_SERVER['DOCUMENT_ROOT'].'/chat/say/form_editsay.html.php';
}
if (isset($_POST['actionnot']) and $_POST['actionnot'] == "Отмена") {
header("Location: /");// Делаем редирект
exit();
}
```

В этом файле происходит обработка данных переданных из формы form_editsay.html.php. Контролируется нажатие кнопок «Редактировать», «Отмена» и вставки исмайлов. Производится обновление текста комментария и добавление обновленного текста в БД.

31. Создаем массив ответов на комментарии

Для этого используем скрипт list_reply.php

Листинг 47. list_reply.php Путь: news/chat/say/list_reply.php

```
<?php
/* Формируем массив ответов на комментарии */
/* SELECT say.id, say.userid, say.saydate, say.saytext, users.login, users.img FROM say INNER JOIN users ON users.`id` = say.userid */
try {
$sql = 'SELECT reply.id, reply.userid, reply.replydate, reply.replyid, reply.replytext, users.login, users.img FROM reply INNER JOIN users ON users.id = reply.userid';
$stm = $dsn->query($sql);
} catch (PDOException $e) {
echo $e->getMessage();
echo $e->getLine();
exit();
}
foreach ($stm as $rows) {
```

```
$reply[] = array(
'id' => $rows['id'],
'replytext' => $rows['replytext'],
'userid' => $rows['userid'],
'replyid' => $rows['replyid'],
'img' => $rows['img'],
'login' => $rows['login'],
'replydate' => $rows['replydate'],
'userid' => $rows['userid']
);
}
```

Выбираем данные из таблицы reply, где id пользователя в таблице users совпадает со значением reply.userid. Т.е. выбираем ответы на комментарии для конкретного пользователя, соответствующие номеру комментария.

32. Обертка вывода ответов на комментарии

Для печати комментариев служит страница print_reply.html.php

Листинг 48. print_reply.html.php Путь: news/chat/say/print_reply.html.php

```
<!DOCTYPE html>
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
</head>

<div class="wrap_reply_form" id="">

<?php
if (isset($reply)) {
foreach ($reply as $replylist) :
```

```
/* Активация кнопок */

if (userHasRole('admin')) {
$buttonactive = 'submit'; //если админ делаем активными все кнопки
} elseif ($replylist['userid'] == $userid) {
$buttonactive = 'submit'; //делаем активными кнопки для юзера
} else {
$buttonactive = 'hidden'; //отключаем кнопки
}

if ($post_id == $replylist['replyid']) {
$comment = $replylist['replytext'];

echo '<div class="block_reply">';    //делаем общий div для кнопки и ответа
include $_SERVER['DOCUMENT_ROOT'] . '/chat/say/form_reply.html';
include $_SERVER['DOCUMENT_ROOT'] . '/chat/say/reply_delete_button.html';
echo '</div>' . '</br>';
}
endforeach;
}
```

```
?>
</div> <!– .wrap_reply_form –>
</html>
```

В скрипте проверяем какие кнопки доступны для данного пользователя. Доступные выводим. Для ответов возможность редактирования не предусмотрена, поэтому только «Удалить». Эта кнопка доступна только автору ответа и администратору. Затем обрабатываем и печатаем ответ на комментарий, полученный из предыдущего скрипта list_reply.php.

33. Форма ответа на комментарий

Для вывода ответа на комментарий используем файл form_reply.html

Листинг 49. form_reply.html Путь: news/chat/say/ form_reply.html

```
<!DOCTYPE html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
</head>
<div class="reply_form">
<div class="topreply" id="">
<b><?= $replylist['login'] ?></b><span style="float:right;">
<?php $t = $replylist['replydate'];
echo date("d.m.Y", "$t");?>
</span>
</div>
<img src="<?='/chat/admin/users/'.$replylist['img']?>" class="avareply" align="left" />
<?php comment_to_smile($comment); ?>
```

Выводит логин пользователя, дату оставления ответа, аватар пользователя, текст ответа. Для span выводящего дату стили указаны непосредственно в файле, но можно присвоить класс при необходимости.

34. Страница ответов на комметарии

Выводим страницу с ответами на комментарии list_reply.html.php

Листинг 50. list_reply.html.php Путь: news/chat/say/ list_reply.html.php

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
</head>
<!– Обертка –>
<div class="wrapsayform" >
<?php
if (!empty($reply)) {
foreach ($reply as $replylist) :
?>
<div class="sayform" id="">
```

```
<div class="printcomment" id="">
<?php
if ($post_id == $replylist['replyid']) {
$comment = $replylist['replytext'];
comment_to_smile($comment);

echo '#' . $replylist['id'];
echo '#' . $replylist['replyid'] . $replylist['login'] . $replylist['img'];

/* Активация кнопок */

if (userHasRole('admin')) {
$buttonactive = 'submit'; //если админ делаем активными все кнопки
} elseif ($saylist['userid'] == $userid) {
$buttonactive = 'submit'; //делаем активными кнопки для юзера
} else {
$buttonactive = 'hidden'; //отключаем кнопки
} ?>
</div>
</div>
<div class="blocksayform" id="">
<div class="sayforms">
<form name="sayform" method="post" action="/chat/say/
```

```
reset.php" class="logout">
   <input type="hidden" name="pageid" id="" value=" <?php
echo $pageid; ?>" />
   <input type="hidden" name="deleteid" id="" value=" <?php
echo $replylist['id']; ?>" />

   <input type="<?= $buttonactive ?>" name="delete" id=""
value="Удалить" />
   </form>
   </div>
   <?php
   }
   endforeach;
   }
   ?>
   </div>
   <div class="add_reply" id="">
   <?php
   if (isset($_SESSION['login'])) {
   if (isset($_GET['reply']) and $_GET['reply'] == $post_id) {
   include_once  $_SERVER['DOCUMENT_ROOT']  .  '/say/
form_add_reply.html.php';
   }
   }
   ?>
   </div>
```

```
</div>
</html>
```

На этой странице в цикле foreach – endforeach выводим все ответы на данный комментарий. Показываем кнопку «Удалить». Проверяем необходимость вставки формы для добавления ответа form_add_reply.html.php' на данный комментарий. Необходимость определяется наличием переменной $_GET['reply'].

35. Форма добавления ответов на комментарии

Для добавления ответов на комментарии служит форма form_add_reply.html.php

Листинг 51. form_add_reply.html.php Путь: news/chat/say/form_add_reply.html.php

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
</style>
</head>
<form action="/chat//say/separate_action.php" method="post" class="say">
<fieldset class="chatfieldset">
<legend>Ответить на комментарий:</legend>
```

```
<textarea class="saytext" name="replytext" rows="5" cols="50" autofocus><?php if(isset($_SESSION['txt'])){
        echo htmlout($_SESSION['txt']);}?></textarea>
<br />
<div class="">
<div class="says">
<input type="hidden" name="userid" value="Добавить">
<input type="hidden" name="pageid" value="<?php echo $pageid ?>">
<input type="hidden" name="sayid" value="<?php if(isset($sayid)) {echo $sayid;} ?>">
<input type="hidden" name="deleteid" id="" value=" <?php echo $saylist['id'];?>" />
<input type="hidden" name="postid" id="" value=" <?php htmlout($post_id);?>" />
<input type="submit" name="action" value="Ответить">
<input type="submit" name="smileblock" value="Показать смайлы">
<input type="submit" name="smileblock_close" value="Скрыть смайлы">
</div>
<div class="notaddsay" id=""><a href="?" class="notaddsays">Отмена</a></div>
</div>
</fieldset>
<br>
```

```
<hr>
<div class="smilesetdiv">
<fieldset class="smileset" style="<?php if (isset($_SESSION['smileblock'])) {echo $_SESSION['smileblock'];} ?>">
<?php print_smile_set();?>
</fieldset>
</div>
<br />
</form>
</html>
```

Это HTML форма состоящая из следующих частей:

• Поле ввода комментариев тип textarea,autofocus.

• Кнопки «Добавить», «Показать смайлы», «Скрыть смайлы» тип input submit.

• Ссылка «Отмена», при нажатии обнуляет все значения в $_GET.

• Блок смайлов, который выводит доступный набор смайлов.

36. Обработчик добавления ответов

Для обработки ответов на комментарии служит файл make_reply.php

Листинг 52. make_reply.php Путь: news/chat/say/ make_reply.php

```
<?php
error_reporting(E_ALL);
if (session_id() == '') {
session_start();
}
unset($_SESSION['txt']);
if (isset($_POST['nosay'])) {
$_GET['reply'] = '';
exit();
}
/* Проверяем наличие текста в форме */
if (empty($_POST['replytext'])) {
header("Location: " . $_SERVER["HTTP_REFERER"]); // Делаем редирект
exit();
```

```
    }
    if (isset($_POST['action']) and $_POST['action'] == "Ответить") {
    include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';
    include_once $_SERVER['DOCUMENT_ROOT'] . '/chat/admin/clean.php';
    /* Получаем id текущего пользователя */
    if (isset($_SESSION['userid'])) {
    $userid = $_SESSION['userid'];
    }
    /* заносим текст в базу */
    try {
    $sql = 'INSERT INTO reply SET
    replytext = :replytext,
    userid = :userid,
    replydate = :replydate,
    replyid =:replyid';

    $s = $dsn->prepare($sql);

    $replytext = html($_POST['replytext']);
    $replyid = $_POST['postid'];
    $replydate = time();

    $s->bindValue(':replytext', $replytext);
```

```
$s->bindValue(':replyid', $replyid);
$s->bindValue(':replydate', $replydate);
$s->bindValue(':userid', $userid);

$s->execute();
} catch (PDOException $e) {
echo 'Error adding данного пользователя';
echo $e->getMessage();
echo $e->getLine();
exit();
}

header("Location: " . $_SERVER["HTTP_REFERER"]); // Делаем редирект
exit();
}
header("Location: " . $_SERVER["HTTP_REFERER"]); // Делаем редирект
exit();
```

Здесь мы проверяем данные переданные формой form_add_reply.html.php, вставляем данные в БД и делаем редирект обратно.

37. Кнопка удаления ответов на комментарии

Для вывода кнопки удаления ответов на комментарии служит файл reply_delete_button.html

Листинг 53. reply_delete_button.html Путь: news/chat/say/reply_delete_button.html

```
<!DOCTYPE html>
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" type="text/css" href="/chat/style.css" />
</head>
<div class="wrapreplyform">
<div class="idreplynum">
    <!– печатаем id комментария –>
    <span style=""><?php
echo '#'.$saylist['id'].' '.'^'.$replylist['id'];?></span>
</div>
<!– печатаем кнопку удалить –>
```

```
<div class="reply_delete_button">
<form name="sayform" method="post" action="/chat/say/reply_delete.php" class="reply_delete_button">

<input type="hidden" name="pageid" id="" value="<?php echo $pageid;?>" />
<input type="hidden" name="deleteid" id="" value="<?php echo $replylist['id'];?>" />

<input type="<?= $buttonactive ?>" name="reply_delete" id="" value="Удалить" />
</form>
</div>
<?php if($buttonactive =='hidden'){echo '.';} ?>
<!– печатаем точку и не даем схлопнуться div –>
</div>
```

Здесь выводим на печать идентификаторы ответа: номер комментария и номер ответа, и кнопку «Удалить».

38. Удаление комментариев

Для удаления служит скрипт reset.php
Листинг 54. reset.php Путь: news/chat/say/ reset.php

```
<?php
if (isset($_POST['delete']) and $_POST['delete'] == 'Удалить') {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';
try {
$sql = 'DELETE FROM say WHERE id = :id';
$s = $dsn->prepare($sql);
$saylist = $_POST['deleteid'];
$s->bindValue(':id', $saylist);
$s->execute();

} catch (pdoException $e) {
echo $e->getMessage();
echo $e->getLine();
exit();
}
}
header("Location:" . $_SERVER['HTTP_REFERER']);//
```

Делаем редирект обратно

Удаляется запись из таблицы say с номером, совпадающим с номером комментария.

39. Удаление ответов на комментарии

Для удаления ответов на комментарии служит файл reply_delete.php

Листинг 55. reply_delete.php Путь: news/chat/say/reply_delete.php

```
<?php
if (isset($_POST['reply_delete']) and $_POST['reply_delete'] == 'Удалить') {
include $_SERVER['DOCUMENT_ROOT'] . '/chat/dsn.php';

try {
$sql = 'DELETE FROM reply WHERE id = :id';
$s = $dsn->prepare($sql);
$del = $_POST['deleteid'];
$s->bindValue(':id', $del);
$s->execute();

} catch (pdoException $e) {
echo $e->getMessage();
```

```
    echo $e->getLine();
    exit();
    }
    }
    header("Location:"    .    $_SERVER['HTTP_REFERER']);//
```
Делаем редирект обратно

Удаляется запись из таблицы reply с номером, совпадающим с номером ответа.

40. Перенаправление смайлов

Для разделения смайлов служит файл smile_make.php

Листинг 56. smile_make.php Путь: news/chat/say/smile_make.php

```
<?php
if(session_id() == '') {session_start();}
/* Проверяем куда пойдет смайл в комментарии или в ответы */
if (isset($_POST['saytext'])) {
    $_SESSION['txt'] = $_POST['saytext'].$_POST['smile'];
header("Location: ".$_SERVER["HTTP_REFERER"]);// Делаем редирект в комментарии
exit();
}
elseif (isset($_POST['replytext'])){
                $_SESSION['txt'] = $_POST['replytext'].$_POST['smile'];
header("Location: ".$_SERVER["HTTP_REFERER"]);// Делаем редирект в ответы
exit();
}
```

Смайлы предусмотрены и в комментариях и в ответах на

них, поэтому в данном скрипте отрабатывается место вставки смала: в комментарий или в ответ. Проверка идет при вставке в файл separate_action.php.

41. Стили

CSS

Код отвечающий за отображение выводимых элементов находится в файле style.css

Листинг 57. style.css Путь: news/chat/style.css

```
/* body */

.chatbody {
background-color: #fef5e4;
max-width: 1366px;
width: auto;
background-image: url(/images/home/bg.jpg);
}

/* form */

.chatform {
margin: 0 auto;
max-width: fit-content;
box-sizing: border-box;
padding: 1.5%;
border-radius: 5px;
```

```
background: RGBA(255, 255, 255, 1);
–webkit-box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
}

/* Блок авторизации */

.inputbutton {
display: flow-root;
float: none;
}

/* div-ы ссылок */

.ingress {
text-align: right;
float: left;
padding: 0px 10px;
text-transform: uppercase;
}

.ingressout {
padding: 0px 10px;
text-transform: uppercase;
float: none;
text-align: left;
```

```
}

.ingressloginout {
padding: 0px 10px;
text-transform: uppercase;
float: none;
text-align: left;
font-family: verdana;
font-size: smaller;
}

.logout {
padding: 0px 10px;
}

/* Ссылки Вход, Регистрация, На главную */

.aingress {
color:-web-link;
}

.aingress:hover {
color: red;
}

/* Вывод комментариев */
```

```
.sayavatar {
float: left;
width: 60px;
height: auto;
border: 1px solid lightgray;
border-radius: 10px;
margin-right: 15px;
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.15);
}

.postnumber {
float: left;
margin-right: 50px;
color: brown;
}

.printcomment {
display: flow-root;
padding: 1px 15px 15px 15px;
}

.topprintcomment {
padding: 0 10px;
border-bottom: 1px solid #cacaca;
margin-left: 15px;
```

```
}

.reply_button {
float: left;
margin: 10px 20px 0px 10px;
}

.wrappersaybutton {
display: flow-root;
clear: both;
background-color: whitesmoke;
}

.says {
float: left;
padding-right: 10px;
}

.sayright {
display: inline-flex;
float: right;
}

.cabinets {
float: left;
}
```

```
.return {
text-align: right;
float: left;
padding: 0px 10px;
margin: 1%;
}

.areturns {
font-family: "Carrois Gothic", sans-serif;
color: #6b5344;
}

.areturns:hover {
color: #f25c05;
}

.adduser {
font-family: "Carrois Gothic", sans-serif;
color: #6b5344;
}

/* панель управления */

/* ссылки */
```

```
.apanel {
color: cadetblue;
}

.apreturn {
color: #6b5344;
}

.wrap_apanel {
margin: 0 auto;
width: fit-content;
}

/* Лист пользователей в админ разделе */

/* делаем сетку */
.ourwrapper {
display: flex;
flex-wrap: wrap;
display: grid;
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
grid-auto-rows: minmax(150px, auto);
grid-gap: 1em;
}
```

```
/* аватарка для админки */

.imgava {
width: 50px;
height: 50px;
padding-top: 5px;
float: left;
padding-right: 20px;
}

/* Страница статистики */

/* Форма статистики и поиска */

.stat {
background: none;
}

.statwrap {
float: none;
display: inline-block;
}

.statone {
float: left;
margin-left: 2%;
```

```
width: 30%;
}

.stattwo {
float: left;
width: 30%;
–webkit-box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
padding: 10px 15px;
border-radius: 5px;
}

.statthree {
margin-left: 2%;
float: left;
–webkit-box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
padding: 10px 15px;
border-radius: 5px;
}

#author {
float: right;
width: 140px;
}
```

```
#category {
float: right;
width: 140px;
}

.textsearch {
float: left;
}

.textsearch_button {
float: right;
}
/* Форма пользователя */

.formuser {
background: white;
display: flex;
align-items: center;
justify-content: center;
height: fit-content;
/* Flex Fallback */
margin-left: 5px;
margin-right: 5px;
flex: 1 1 200px;
padding: 4%;
border-radius: 8px;
```

```
}

.topuser {
border-bottom: 1px dashed #7e6d4c;
float: none;
}

.login_art {
margin: 0px;
font-weight: bold;
}

h3.vintage {
color: #f25c05;
border-bottom: 1px dashed #7e6d4c;
font-style: italic;
}

.number {
color: brown;
}

/* 2. Административный раздел */

/* 2.1 Подключение нового пользователя form.html.php */
```

```
.username {
padding: 2px;
float: none;
}

/* fieldset */

.usernames {
border: 1px solid lightgray;
}

/* поле ввода логина */

.inputs {
float: right;
}

/* заголовок формы */

.formname {
margin: 0;
}

.chatfieldset {
border: 1px solid lightgray;
}
```

```
/* заголовок создания админа */

.make_admin {
text-align: center;
color: brown;
}

/* addsay */

.addsayform {
margin: 0 auto;
max-width: fit-content;
box-sizing: border-box;
padding: 1.5%;
border-radius: 5px;
background: RGBA(255, 255, 255, 1);
–webkit-box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.45);
}

/* div кнопки добавить комментарии */

.addsay {
float: left;
margin: 0 10px;
```

```
}

/* ссылки показать скрыть комментарии */

.aopensays {
color: #6f6f6f;
}

.aopensays:hover {
color: #497270;
}

.aaddsays {
color: #6f6f6f;
}

.aaddsays:hover {
color: #53918e;
}

.areply {
color: #6b5344;
}

.areply:hover {
color: #53918e;
```

```
}

.notaddsays {
color: #6b5344;
}

.notaddsays:hover {
color: #f25c05;
}

/* из test */

/*.....Ответы..... */

.block_reply {
margin-left: 3%;
position: relative;
/* background:green; */
}

.wrap_reply_form {
position: relative;
/* background: orange; */
}

.reply_form {
```

```
/* border: 1px solid green; */
padding: 1%;
background: white;
float: none;
margin-right: 1%;
display: flow-root;
}

.reply_delete_button {
background: whitesmoke;
margin-right: 1%;
padding: 3px;
border-radius: 4px;
float: right;
}

.avareply {
margin: 3px 10px;
border-right: 1px solid #cacaca;
padding: 0 5px;
width: 40px;
height: auto;
}

.topreply {
padding: 2px 12px;
```

```
    border-bottom: 1px solid #cacaca;
    border-radius: 5px;
    }

    .wrapsayform {
    position: relative;
    border: 1px solid lightgray;
    border-radius: 4px;
    margin: 2% 0;
    z-index: 0;
    clear: both;
    /* background:Градиент */
    /* background: linear-gradient(45deg, #3e84bf 25%,
transparent 25%, transparent 75%, #292929 75%),
    linear-gradient(45deg, #292929 25%, transparent 25%,
transparent 75%, #292929 75%) 0.1875em 0.1875em,
    radial-gradient(at 50% 0, #484847, #090909);
    background-size: 0.375em 0.375em, 0.375em 0.375em,
100% 100%; */
    /* В линейку */
    /* background-color: #FFFFEF;
    background-image:
    linear-gradient(90deg, transparent 98px, #ED82AD 98px,
#ED82AD 100px, transparent 100px),
    linear-gradient(#eee 1px, transparent 0px),
    linear-gradient(90deg, #eee 1px, transparent 0px);
```

```
background-size:100% 100%, 20px 20px, 20px 20px;
background-position: 0 0, -1px -1px, -1px 1px; */
}

.sayform {
background: white;
margin: 1%;
padding: 0;
border-radius: 5px;
border: 1px solid lightgray;
}

.topprintcomment {
padding: 0 10px;
border-bottom: 1px solid #cacaca;
margin-left: 15px;
}

.wrapreplyform {
display: flow-root;
background: whitesmoke;
margin-right: 1%;
}

.idreplynum {
float: left;
```

```
color: brown;
padding: 5px 15px;
}

h3.user {
margin: 0 1em 0 1em;
padding: 0 0 5px 0;
color: cadetblue;
font-weight: normal;
position: relative;
text-shadow: 0 2px 0 rgba(255, 255, 255, 0.5);
font-size: 24px;
line-height: 40px;
font-family: "Carrois Gothic", sans-serif;
border-bottom: 1px dashed cadetblue;
}

h4.user {
margin: 0 1em 0 1em;
padding: 0 0 5px 0;
color: cadetblue;
font-weight: normal;
position: relative;
text-shadow: 0 2px 0 rgba(255, 255, 255, 0.5);
font-size: 24px;
line-height: 40px;
```

```
font-family: "Carrois Gothic", sans-serif;
border-bottom: 1px dashed cadetblue;
}

h4.formname {
margin: 0 1em 0 1em;
padding: 0 0 5px 0;
color: cadetblue;
font-weight: normal;
position: relative;
text-shadow: 0 2px 0 rgba(255, 255, 255, 0.5);
font-size: 24px;
line-height: 40px;
font-family: "Carrois Gothic", sans-serif;
border-bottom: 1px dashed cadetblue;
}

h5.user {
margin: 0 1em 0 1em;
padding: 0 0 5px 0;
color: cadetblue;
font-weight: normal;
position: relative;
text-shadow: 0 2px 0 rgba(255, 255, 255, 0.5);
font-size: 24px;
line-height: 40px;
```

```
font-family: "Carrois Gothic", sans-serif;
border-bottom: 1px dashed cadetblue;
}

h1.user {
margin: 1em 1em 0.75em 1em;
padding: 0 0 5px 0;
color: #6b5344;
font-weight: normal;
position: relative;
text-shadow: 0 2px 0 rgba(255, 255, 255, 0.5);
font-size: 36px;
line-height: 40px;
font-family: "Carrois Gothic", sans-serif;
border-bottom: 1px dashed #7e6d4c;
}

h2.user {
margin: 1em 1em 0.75em 1em;
padding: 0 0 5px 0;
color: #6b5344;
font-weight: normal;
position: relative;
text-shadow: 0 2px 0 rgba(255, 255, 255, 0.5);
font-size: 36px;
line-height: 40px;
```

```
font-family: "Carrois Gothic", sans-serif;
border-bottom: 1px dashed #7e6d4c;
}

.addusers {
margin: 1%;
}
```

Стили в дальнейшем, можно настроить как угодно.

42. Установка модуля на сайт

Модуль готов. Приступаем к практической его установке. Ищем подходящий шаблон. Мне понравился выбор шаблонов на сайте Эрика Байгузина [5]. Один из выложенных на нем шаблонов я и решил использовать. Это шаблон EUCLID. Ссылка:

https://bayguzin.ru/main/shablonyi/shablonyi-dlya-bloga/kachestvennyj-sovremennyj-dizajn-bloga.-minimalistichnyj-prostoj-chitabelnyj.html

Естественно, что можно использовать любой другой доступный шаблон. На его примере посмотрим установку модуля. Последовательность действий следующая:

1. Скачиваем шаблон.

2. Размещаем шаблон на сервере (хостинге). В нашем случае размещаем в папке C:\OSPanel\domains\. Задаем (или используем зарегистрированное) имя сайта.

В данном случае я назвал сайт euclid.com.

3. Вкладываем в корень сайта папку chat.

4. Проверяем наличие в корне сайта файла .htaccess:

• Если он отсутствует, переносим наш файл .htaccess из папки chat в корень сайта.

• Если присутствует, копируем в него содержимое нашего файла .htaccess.

5. Выбираем страницу, на которой будут размещать-

ся комментарии. Смотрим содержимое сайта. Как видно из рис. 21 сайт небольшой, состоит из трех HTML страниц about, blog, index, папки со стилями CSS и папки с Javascript. Файл .htaccess отсутствовал, поэтому на скриншоте файл .htaccess из папки chat.

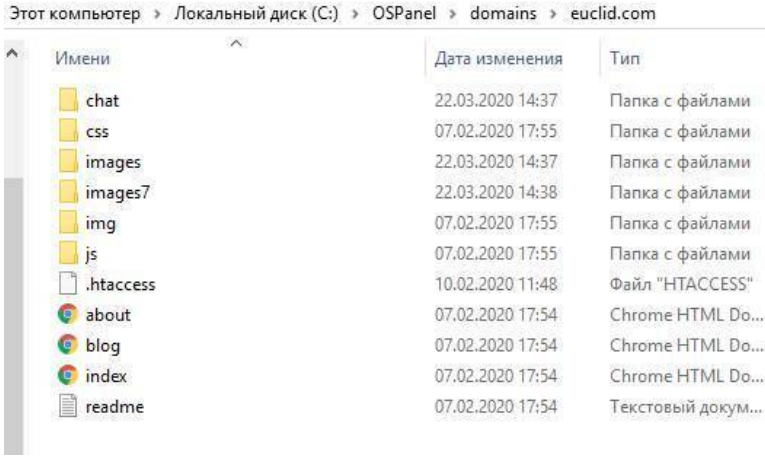


Рис. 21. Содержимое сайта euclid.com.

Внимание! Если запускаете модуль на хостинге пропишите действующие параметры подключения к СУБД и БД в файлах: **dsn.php и createbase.php**

Давайте начнем с главной страницы. Открываем файл index.html сайта в редакторе. Контроллер 1 будет распола-

гаться в самом начале документа. Запускаем сайт определяемся с местами, где у нас будут располагаться кнопки авторизации. Я выбрал сразу после блока <body>. Здесь разместим контроллер 2. Далее смотрим, где будем выводить комментрии. Давайте выведем после первого блока с материалом. Он называется PORT HARBOR. Материал заканчивается строкой со словом Etiam. Ищем в редакторе это слово. Как мы видим на этом слове заканчивается и div с данным материалом. После него вставляем контроллер 3. Получаем следущее:

Листинг ТЕСТ 1. файл index.html Путь: euclid.com/index.html

```
<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/createbase_controller.php'?>
<!DOCTYPE html>
<html lang="en">
<head>
…//Содержимое блока head
</head>

<body class="home blog">
<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/login_controller.php'?>
<!– Start Header –>
```

…//Содержимое блока body

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris sit amet auctor ligula. Donec eu placerat lacus, pellentesque tincidunt felis. Aliquam dictum cursus elit, et sagittis nibh tincidunt quis. Vestibulum leo dui, ullamcorper quis erat nec, accumsan imperdiet ligula. Maecenas ut dui sed arcu sodales consequat. Nulla et est ac lacus congue volutpat. Aliquam vehicula tincidunt sem eget cursus. Nam sed mollis diam. Pellentesque id felis ut diam dignissim egestas id non ipsum.Ut id magna eu eros vehicula sollicitudin at et odio. Mauris consectetur tortor in mauris aliquet feugiat. Etiam

<?php include_once $_SERVER['DOCUMENT_ROOT'].'/chat/say_controller.php'?>

…//Продолжение блока body

<!– End Footer –>

</body>

</html>

Внимание! Если по каким либо причинам на вашем хостинге **$_SERVER['DOCUMENT_ROOT']** работает некорректно заменяем все её вхождения на приведенный ниже код

Пример замены $_SERVER['DOCUMENT_ROOT'] (по-

дробности см. в гл. 2.4.):

```
<?php
$p = explode('chat', __DIR__);
set_include_path(get_include_path().PATH_SEPARATOR.
$p[0]);
include "chat/login_controller.php";
?>
```

Здесь после include вставляете относительный путь к вставляемому файлу, для примера указан login_controller.php, но вы вставляете нужный.

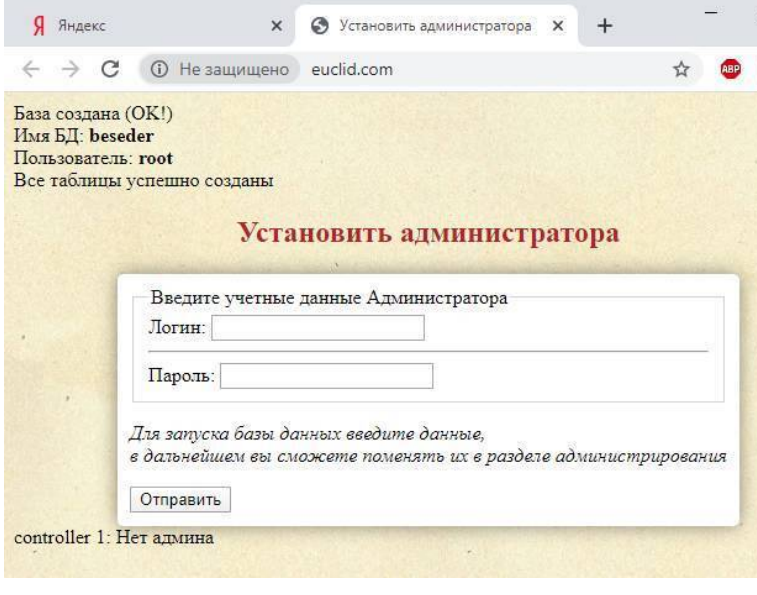


Рис. 22 Первый запуск.

Во время первого запуска выводиться сообщение о создании БД и форма для задания учетных данных Администратора модуля.

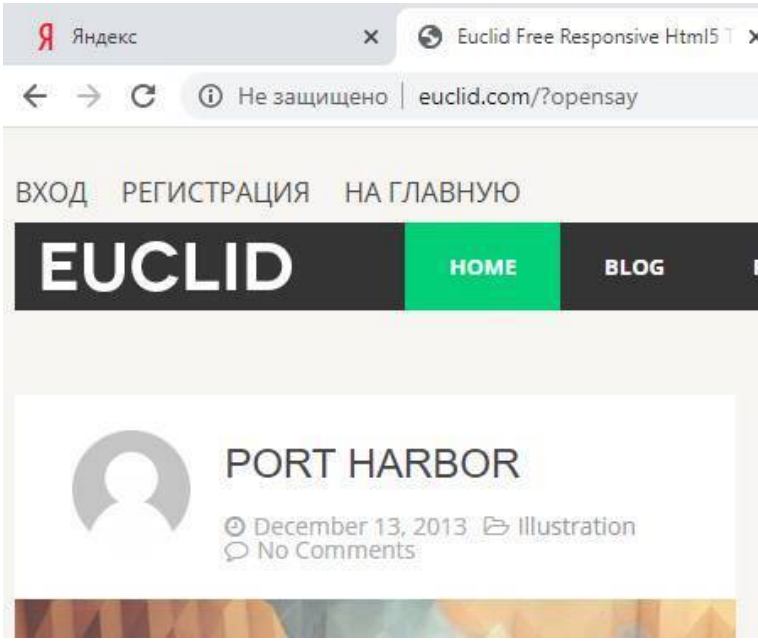


Рис. 23 После отправки данных администратора. Верх.

Как видим появились кнопки «Вход», «Регистрация» и

«На главную» в верхней части сайта.

Рис. 24 После отправки данных администратора. Материал блога.

И появилась кнопка «Показать комментарии»/ «Скрыть комментарии» в завершении одного из материалов блога.

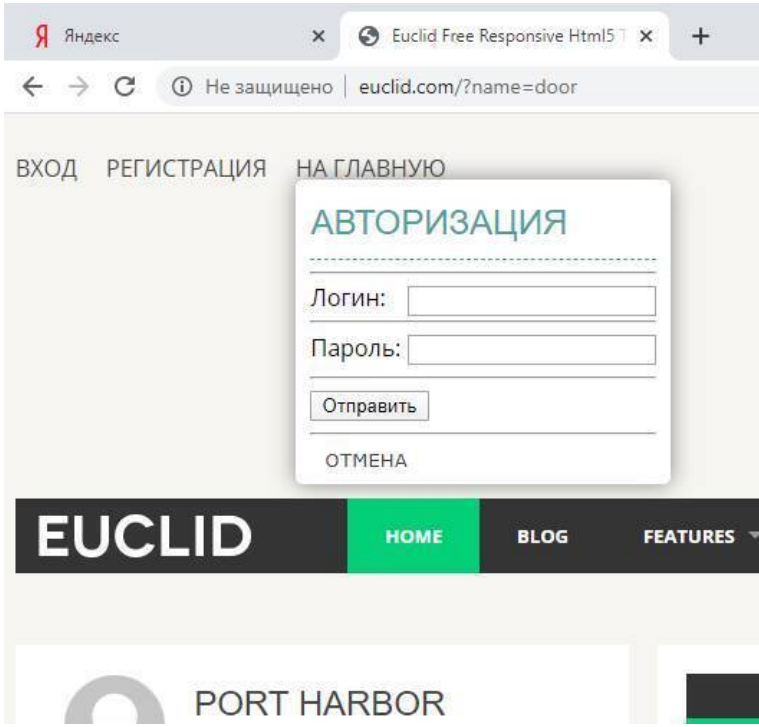
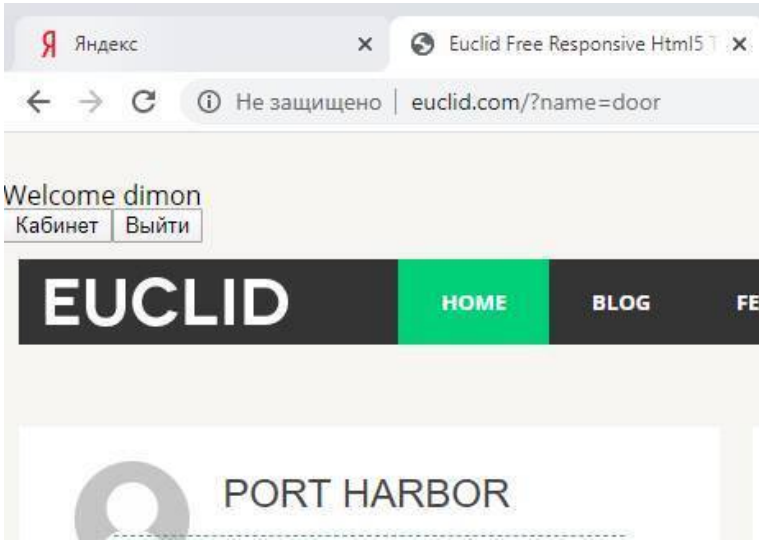


Рис. 25 Форма авторизации

Форма авторизации появляется при нажатии кнопки «Вход».



Приветствие и кнопки

Рис. 26 Шапка сайта после авторизации пользователя

После авторизации пользователя становятся доступны кнопки «Кабинет» и «Выйти». Сейчас желательно сразу зайти в кабинет и установить аватар для администратора.

Рис. 27 Появилась возможность добавления комментариев

После авторизации появляется ссылка для добавления комментариев. Жмем.

id non ipsum. Ut id magna eu eros vehicula sollicitudin at et odio.
Mauris consectetur tortor in mauris aliquet feugiat. Etiam

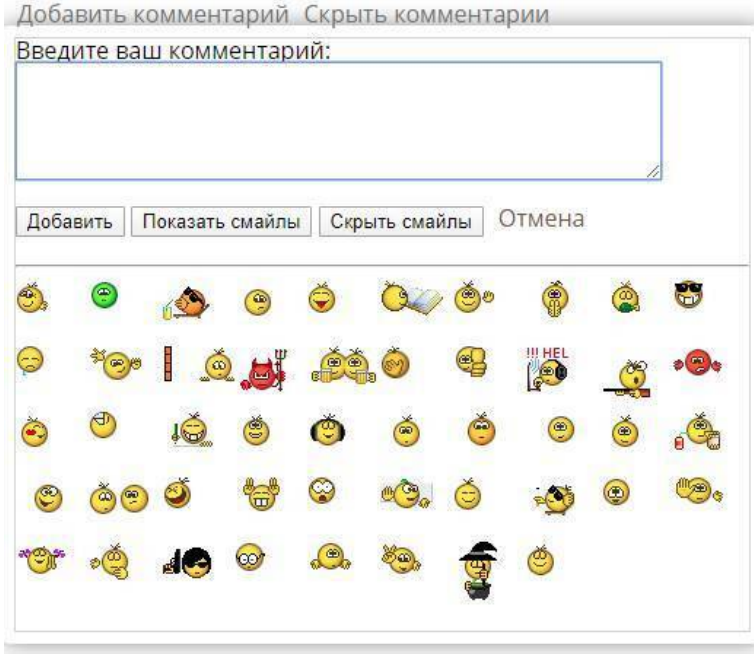


Рис.28 Форма добавления комментариев

В форме пишем комментарий, добавляем при желании смайлы.

Яндекс | Управление

Не защиш | euclid.com/ch...

Управление пользователями

Добавить нового пользователя

 # 1 dimon

admin

Редактировать | Удалить

Вернуться

Рис. 29 Страница управления пользователями.

На странице управления пользователями можно добавлять, редактировать и удалять пользователей.

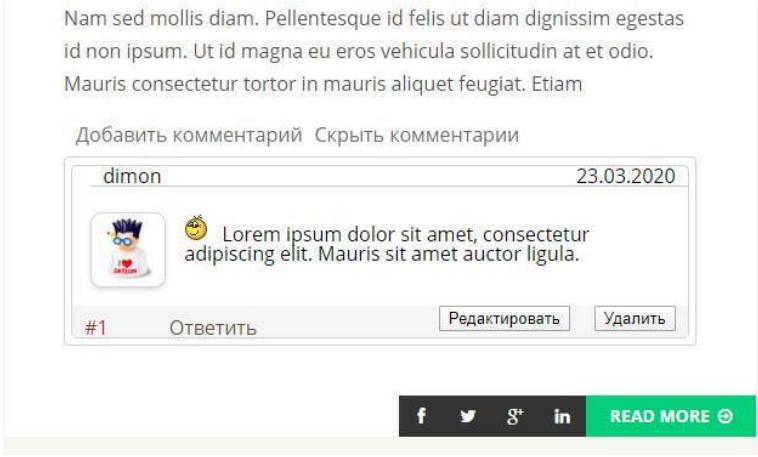


Рис. 30 Добавленный комментарий

На рис. 30 скриншот с новым добавленным комментарием.

если мы хотим разместить комментарии еще на какой- либо странице, делаем следующее:

1. Открываем для этой страницы доступ к переменной $_SESSION разместив в заголовке страницы, следующий код:

<?php if (session_id() == '') {session_start();} ?>

2. Вставляем файл с функциями проверки пользователя access.php

3. После него вставляем контроллер 3:

<?php

include_once $_SERVER['DOCUMENT_ROOT'].'/chat/admin/access.php';

include_once $_SERVER['DOCUMENT_ROOT'].'/chat/say_controller.php'

?>

В листинге ТЕСТ 2 приведен пример для вставки комментариев на страницу blog.html.

Листинг ТЕСТ 2. файл blog.html Путь: euclid.com/blog.html

```
<?php if (session_id() == '') {
session_start();
}?>
<!DOCTYPE html">
<html lang="en">
<head>
…//код блока
</head>

<body class="single single-post postid-49 single-format-standard">
…//код блока
```

<!– Start Header –>

Integer auctor, mauris vel consequat viverra, nibh arcu elementum odio, ut varius arcu sapien vitae ligula. Fusce erat metus, cursus nec felis eget, vulputate vulputate turpis. Nulla iaculis venenatis magna, lobortis egestas magna faucibus mollis. Quisque molestie turpis dolor, blandit convallis elit pellentesque eu. Nam sit amet enim a est congue vestibulum eget id leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Phasellus sit amet ipsum eros. Etiam faucibus sapien turpis, vitae sagittis tellus faucibus quis.

<?php

include_once $_SERVER['DOCUMENT_ROOT'].'/chat/admin/access.php';

include_once $_SERVER['DOCUMENT_ROOT'].'/chat/say_controller.php'?>

<!– Post Tags –>

…//код блока

<!– End Footer –>

</body>

</html>

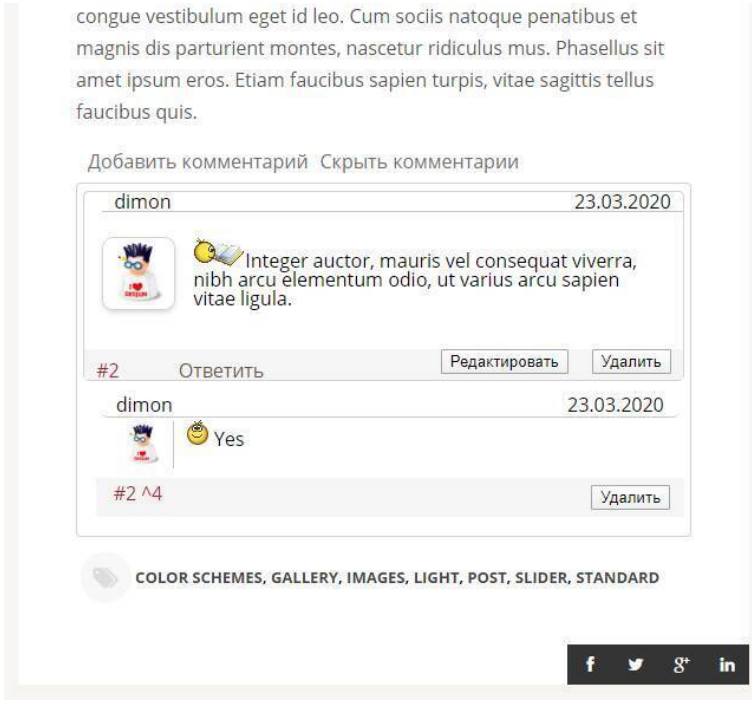


Рис. 31 Комментарии на странице blog.html

Как мы видим, комментарии можно будет оставлять на любых дополнительных страницах сайта. На каждой странице будут выводиться комментарии только для этой страницы.

Заключение

Ну вот и все, скомпиллирован код, позволяющий оставлять коментарии на любом сайте. Как можно заметить, в итоге получилась мини CMS. Если добавить в административный раздел модуля возможность формирования страниц из блоков типа header, footer, menu, content и т.д., возможность оформления материалов при помощи редакторов типа CKEditor и уделить внимание безопасности обработки передаваемых данных, то получится полноценная CMS. Можно применить классы и использовать принципы MVC, шаблоны. Но это уже выходит за рамки практикума, поэтому если вам интересна тема программирования сайтов, можете заняться этим увлекательным занятием самостоятельно.

Список источников

Кевин Янк., PHP и MySQL. От новичка к профессионалу. М.: Эксмо, 2013. – 384 с.

Д. В. Котеров, И. В. Симдянов. , PHP 7. СПб.: БХВ-Петербург, 2016. – 1088 с.

Сайт Михаила Русакова . https://myrusakov.ru/

Сайт Евгения Попова https://ruseller.com/

Сайт Эрика Байгузина https://bayguzin.ru/

Сайт PHPFAQ http://phpfaq.ru/

Сайт PHP Group https://www.php.net/