

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**РАЗРАБОТКА ВЫСОКОНАГРУЖЕННОГО РАСПРЕДЕЛЕННОГО ПРИЛОЖЕНИЯ**

Работу выполнил А. В. Оврах

СОДЕРЖАНИЕ

Введение ................................................................................................................... 3

1. Свойства и подходы при разработке высоконагруженных распределѐнных систем .......................................................................................... 5
   1. Кластеры ......................................................................................................... 5
   2. Балансировка нагрузки .................................................................................. 7
   3. Интерактивность .......................................................................................... 11
   4. Кэширование ................................................................................................ 13
   5. Распараллеливание задач ............................................................................ 14
2. Разработка высоконагруженного web-приложения ..................................... 16
3. Реализация административной части проекта ................................................ 19
   1. База данных проекта .................................................................................... 19
   2. Взаимодействие с базой данных посредством технологий PHP ............ 21
4. Функциональность пользовательской части проекта ..................................... 28
   1. Описание интерфейса web проекта ............................................................ 28
   2. Описание главной страницы проекта ........................................................ 29
   3. Описание страницы чата реализуемого проекта ...................................... 33

Заключение ............................................................................................................ 40

Список использованных источников .................................................................. 42

ВВЕДЕНИЕ

Во все времена люди использовали и совершенствовали средства общения друг с другом. Они прошли путь от бумажных писем, телеграфа, до современных последних разработок в данной области. Сейчас, когда ритм жизни очень высокий и на живое общение между людьми остаѐтся совсем мало времени, на помощь человеку приходит интернет вместе с огромным количеством всевозможных средств, которые предоставляют возможность общаться людям, как далеко они бы не находились друг от друга.

К числу этих средств мы можем отнести всевозможные форумы, чаты, видеоконференции, клиенты для обмена короткими текстовыми сообщениями, такие как ICQ, QIP , IRQ и другие.

И все же возникает вопрос, как обеспечить всем пользователям отказоустойчивую работу приложений, когда их количество исчисляется сотнями тысяч, а то и миллионами. Вот тут уже и возникает речь о высоконагруженных приложениях, которые в свою очередь позволяют взаимодействовать с серверным приложением большому количеству пользователей.

Данная тематика больше всего затрагивает именно web—приложения, количество пользователей которых реально исчисляется миллионами человек, и чтобы приложения работали даже при пике нагрузки. В связи с этим, запросы к приложениям распределяются между серверами.

В последнее время все чаще говорят о высоконагруженных приложениях. Нельзя не заметить что это теперь очень популярная область знаний. Сам же термин ―высоконагруженная система‖ при этом не имеет в отрасли информационных технологий четкого определения.

Конечно же, высоконагруженное приложение в первую очередь является многопользовательским. То есть в один момент времени с ним работает более чем один человек. Сейчас, в эру стремительного развития Интернета, это тысячи и сотни тысяч человек.

Объектом исследования дипломной работы является разработка высоконагруженных web приложений в реальном времени.

Целью работы является разработка высоконагруженного web чата.

В процессе подготовки дипломной работы и поиска исходных кодов web чата, не было найдено ни одного бесплатного web чата, с доступной и удобной системой инсталляции, что не дало возможности использовать для обучения и наглядного примера ни один из скачанных web чатов. В разработанном же проекте предусмотрена удобная система инсталляции, автоматическое создание необходимых таблиц в базе данных и необходимая документация.

1. Свойства и подходы при разработке высоконагруженных распределѐнных систем

Сейчас проблема распределенных вычислений как никогда актуальна, в связи с появлением всемирной глобальной сети и с каждым годом увеличивающегося количества еѐ пользователей. Еще с конца прошлого века начали появляться поисковые системы, почтовые сервера, предоставляющие доступ большому количеству пользователей, что само по себе вызывает довольно большой поток трафика запросов к серверам. И так как только один сервер может обработать запросы только ограниченного количества пользователей, возникла необходимость объединения серверов в одно информационное пространство, предоставляющее услуги электронной почты или же иного сервиса. Это позволяет перераспределить нагрузку с одного сервера на все последующие, что в свою очередь уменьшит отклик на обработку запроса к серверу. Такие системы называют ещѐ кластерами.

Существует много определений, что такое кластер.

1.1 Кластеры

Кластер – это разновидность параллельной или распределенной системы, которая:

* состоит из нескольких связанных между собой компьютеров;  используется как единый унифицированный информационный ресурс; Существует несколько видов кластеров:
* отказоустойчивые;
* с балансировкой нагрузки;
* вычислительные;
* системы распределенных вычислений;

Отказоустойчивые кластеры создаются для высокой доступности сервиса. Гарантированное представление сервиса в случае отказа одного или нескольких компьютеров обеспечивается избыточностью числа узлов. Отказоустойчивые кластеры распределяются на три типа:

а) с холодным резервом или активный/пассивный. Активный узел выполняет запросы, а пассивный ждет его отказа, и включатся в работу когда таковой случится;

б) с горячим резервом или активный/активный. Все узлы выполняют запросы, в случае отказа одного нагрузка распределяется между оставшимися. Другими словами можно назвать это как кластер распределения нагрузки с поддержкой перераспределения запросов при отказе;

в) с модульной избыточностью. Применяется только в том случае, когда простой системы совершенно недопустим. Все узлы системы выполняют один и тот же запрос, из результатов выполнения берется любой;

Конкретная технология может сочетать в себе принципы любой комбинации.

Принцип действия кластеров распределения нагрузки строится на распределении запросов через один или несколько входных узлов, которые перенаправляют их на обработку в остальные вычислительные узлы. Изначально, цель такого кластера – это, прежде всего производительность, но в них часто используются и методы, повышающие надежность. Подобные конструкции называются ещѐ серверными фермами. Рассмотрим более подробно данный вид кластеров.

Основной задачей кластера распределения нагрузки является обеспечение оптимальной нагрузки серверов при минимально – необходимом времени отклика прикладных сервисов информационной системы во всех географических точках, где они должны быть доступны пользователям. Балансировка нагрузки достигается за счет нескольких параллельно установленных серверов. При этом технологии реализации балансировки могут отличаться.

1.2 Балансировка нагрузки

К приложениям, требующим балансировки, как правило относят наиболее типовые Web—серверы, серверы электронной почты и DNS—серверы, которые обслуживают сеть Интернет и корпоративные сети организаций. Помимо этого многие прикладные программы и инфраструктурные сервисы также могут поддерживать возможность балансировки, для чего они должны удовлетворять следующим требованиям:

* согласованность данных на всех параллельно—доступных серверах;
* целостность асинхронных и длительных по времени транзакций;
* поддержка приложением системной инфраструктуры балансировки нагрузки;

Согласованность данных — основная проблема, так как все серверы должны выдавать одинаковые ответы пользователям на один и тот же запрос. При этом, если данные изменились на одном сервере, то они должны быть реплицированы на остальные сервера. А в зависимости от требований к времени обновления реплик на серверах технологии реализации согласования данных могут сильно отличаться: от переноса данных оператором системы на диске до организации масштабной системы онлайн—кластеризации и репликации данных (как в поисковых Web—площадках). Фактически, данную задачу должно решать прикладное программное обеспечение, но ни в коем случае не «железо» (зачастую при закупке оборудования предприятием допускается такая ошибка).

Целостность транзакций — одна из наиболее сложных проблем при организации NLB—кластера. В NLB—кластерах возможны ситуации, в которых приложение, начиная работу с одним сервером кластера, завершает еѐ на другом сервере. Очевидно, что необходимо обеспечить целостность данных и корректность завершения транзакции.

Особое значение эта проблема имеет при организации сетей с балансировкой нагрузки и отказоустойчивостью маршрутизаторов и брандмауэров.

Инфраструктура балансировки сетевой нагрузки обеспечивает равномерное распределение заявок пользователей между серверами системы балансировки.

Различают следующие уровни реализации NLB—системы:

а) cетевой уровень. Поддерживается на уровне операционной системы и реализован в Windows Server Network Load Balancing. Суть подхода заключается в логическом объединении всех интерфейсов серверов под один IP—адрес в сети, при котором серверы договариваются между собой о порядке обслуживания запросов. Существует два способа реализации этого подхода:

* unicast mode.В случае unicast mode на всех серверах NLB—кластера

настраивается один виртуальный MAC—адрес, что позволяет организовать прием пакетов на все интерфейсы кластера. В этом случае узлы кластера не могут обмениваться информацией между собой с использованием таких интерфейсов. Для этого обычно организуется дополнительная административная сеть или сеть доступа к данным, в которую узлы подключаются дополнительными интерфейсами;

* multicast mode**.** В этом случае передача пакетов на узлы кластера

организуется посредством коммутатора, который должен поддерживать multicast mode. В этом случае на интерфейсах кластера задействуются штатные MAC—адреса и поэтому серверы могут взаимодействовать друг с другом, используя эти же интерфейсы;

б) уровень службы имен интернета DNS. Данная служба позволяет преобразовывать имена компьютеров вида "comp2.domain.net" в IP—адреса вида "192.168.200.201" и наоборот для коммуникаций по протоколу IP. Для балансировки нагрузки в службе DNS для одного имени типа

"comp2.domain.net" настраивается несколько IP—адресов, а также опция "round robin", которая устанавливает порядок выдачи адресов для DNS—запросов по кругу. В результате, каждый клиент, по мере выдачи запросов на одно имя на DNS—сервер, будет получать разные IP—адреса, а значит, подключаться к разным серверам;

в) уровень приложения. Данный подход реализуется в прикладном программном обеспечении и может не зависеть от инфраструктуры (уровней 1 и 2). Примерами таких приложений являются Active Directory, Microsoft

Exchange Server, WINS, Kaspersky Administration Kit, корпоративные распределенные системы учета и т.п., в которых реализация носит индивидуальный характер;

Вычислительные кластеры используются в частности в научных исследованиях. Для таких кластеров существенным показателем является высокая производительность процессора в операциях над числами с плавающей точкой и низкая латентность объединяющей сети, менее же существенна скорость операций ввода—вывода, которая в большей степени важна для баз данных и web—серверов. Данный вид кластеров позволяет уменьшать время расчѐтов в сравнении с одиночным компьютером, распределяя задания на параллельно выполняющиеся ветки, которые производят обмен данными по связывающей их сети.

Системы распределенных вычислений не принято считать кластерами, но их принципы в значительной степени с кластерной технологией. Главное отличие — низкая доступность каждого узла, то есть невозможность гарантировать его работу в заданный момент времени (узлы подключаются и отключаются в процессе работы), поэтому задача должна быть разбита на ряд независимых друг от друга процессов. Такая система, в отличие от кластеров, не похожа на единый компьютер, а служит упрощѐнным средством распределения вычислений. Нестабильность конфигурации, в таком случае, компенсируется большим числом узлов.

Устойчивая ассоциация высоконагруженных систем с большим количеством пользователей в индустрии информационных технологий появилась довольно давно. Ничего принципиально неверного в такой связи нет. Но если высокая нагрузка подразумевает большое количество пользователей, то большое количество пользователей совсем не обязательно подразумевает высоконагруженную систему.

Высоконагруженные системы являются системами распределенными, то есть работают более чем на одном сервере. Зачастую это десятки и сотни серверов. Требование распределенности вытекает из следующих причин:

* необходимости обрабатывать возрастающие объемы данных;
* необходимости ―живучести‖ системы в случаях отказа части серверов;

Сложно, скорее всего, сделать ошибку, если сказать что большинство высоконагруженных приложений являются Интернет—приложениями. А отличительной особенностью современного Интернета основанного на концепции Web 2.0 является тот факт, что сами пользователи генерируют данные, которые они сами же в итоге и потребляют. Это приводит к тому, что чем больше пользователей, тем больше потенциальный объем хранимых данных.

Требование обработки больших объемов данных может существенно осложнить жизнь. Под ―большим объемом‖ подразумевается такой объем информации, который не может эффективно обработать один компьютер. В большинстве случаев, это объем превышающий объем доступной на сервере оперативной памяти. То есть приходится тем или иным образом распределять данные между несколькими серверами, каждая из которых обрабатывает свой небольшой кусочек данных, но делает это эффективно, без page fault‘ов (не используя swap) и прочих неприятностей. Необходимость эффективной обработки данных диктуется другим очень важным качеством высоконагруженных систем, – интерактивностью, о котором будет сказано ниже.

Но большие объемы данных – это не все. Ко всему к этому хочется, чтобы система работала без остановок и перерывов. Но бывают такие моменты, что любое даже самое надежное оборудование иногда выходит из строя. Встает естественная задача обеспечения доступности системы в случаях отказа оборудования.

Тут и возникает потребность в области знания распределенных систем, эксплуатация которых редко бывает безоблачной, даже когда используются готовые решения. Тем не менее, распределенные системы, не смотря на сложность их разработки и поддержки, пожалуй, единственный подход позволяющий обеспечить вышеизложенные требования в полной мере.

Если приложение представляет хоть какой—то интерес, то даже если ничего не делать, аудитория пользователей будет расти просто с ростом аудитории интернета. Поэтому характерной чертой высоконагруженных систем является не просто большое количество пользователей, но и позитивная динамика количества пользователей.

В контексте реалий Web 2.0 растущее количество пользователей может привести к тому, что такую же позитивную динамику возможно иметь и по данным. Поэтому в контексте высоконагруженных систем корректней говорить не о большом, а о растущем количестве пользователей и данных.

1.3 Интерактивность

Интерактивность это одно из основополагающих качеств высоконагруженной системы. Интерактивность подразумевает, что пользователь после того как послал запрос сидит и ждет ответа, а люди как известно ждать не любят. При этом большинство интернет—приложений о которых мы говорим не являются критическими важными в жизни людей. Twitter, Flickr, Facebook и т.д. это все замечательно, но если они будут отвечать непомерно долго, это может отвлечь внимание пользователя. Жизнь пользователей от них не зависит, а это значит, что эти приложения должны занимать минимум времени пользователей. То есть отвечать за приемлемое время. Это контрастирует, например, с системами пакетной обработки данных, в которых время ответа системы вторично.

Из этого правила, безусловно есть исключения. Представим, что только что совершили покупку в Интернете, сообщив свою платежную информацию третьим лицам. Скорее всего, мы дождемся ответа системы, даже если она будет отвечать дольше минуты. Но, как известно исключение из правила лишний раз подтверждает само правило.

Качество интерактивности очень важно для понимания сути высоконагруженных приложений, потому что по выше обозначенным причинам разрабатывая такую систему мы должны быть уверены в том, что каждый раз когда приложение получает очередной запрос, у него должно быть достаточно свободных ресурсов (CPU, память, диск, сеть) для обработки запроса за приемлемое время.

Возможно, это и звучит тривиально, но именно данное требование приводит нас к основному посылу данной заметки, такой как — контроль за ресурсами, который является неотъемлемой частью работы с высоконагруженным проектом.

Необходимо также учитывать тот факт, что из—за позитивной динамики свободных ресурсов становится все меньше с течением времени. В этом заключается ―парадокс‖ высоконагруженных систем. Если случится так, что разработка высоконагруженного проекта будет заморожена (отправив всех разработчиков в бессрочный отпуск), то рано или поздно он перестанет работать.

Высоконагруженная система – это интерактивная распределенная система которая требует постоянного контроля за собственными ресурсами. В противном случае она перестает работать.

Если исходить из того что неотъемлемой частью высоконагруженного проекта является постоянный рост данных и аудитории, то становится понятно почему высоконагруженные проекты – это поголовно Интернет приложения.

В реальной жизни всегда есть некий предел, который не позволяет использовать систему все возрастающему количеству людей. В том же метрополитене человеку требуется некоторое время, чтобы пройти через турникет. Исходя из этого времени, а также общего количества турникетов можно достаточно точно рассчитать верхний предел возможной нагрузки. Выглядит очень невероятным чтобы за секунду через один турникет могло пройти 10 человек.

В сфере High Performance Computations приложения могут выполнять просто гигантское количество операций в единицу времени. Больше чем любое интернет—приложение. Но это количество зависит только от объема входных данных, а также алгоритма обработки (например, от точности моделирования, если речь идет о моделировании динамических систем). Тяжело придумать причину почему нагрузка на такую систему может вырасти сама по себе без вмешательства персонала ее сопровождающего.

1.4 Кэширование

Кэширование — это преобразование вычислений при помощи промежуточного буфера обмена с достаточно быстрым доступом, содержащий информацию к которой обращаются чаще всего. Существует несколько видов кэширования:

* Кэширование информации;
* Кэширование html узлов;
* Кэширование целиком html страниц;

Кэширование сейчас активно используется почти в любом Web—проекте, но необязательно в высоконагруженном. Для каждого ресурса критичным является, если пользователь ждѐт отклика от сервера очень долго. Таким образом ресурс отталкивает поток посетителей. Соответственно время отклика сервера следует минимизировать.

Решением как раз и является кэширование. Заранее все преобразованные вычисления заносятся в хранилище(кэш), которое обладает быстрым доступом к информации. И теперь вместо сложных запросов достаточно выполнить запрос к хранилищу, как указано на рисунке 1.

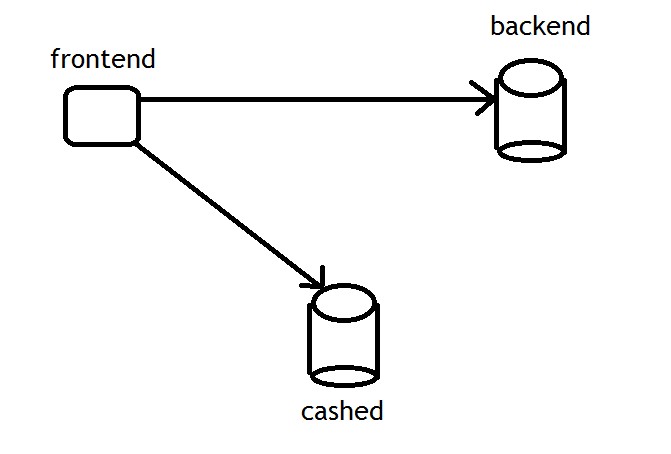


Рисунок 1 — Общая схема кэширования

Frontend это та часть проекта, которая формирует ответ пользователю. Пользователь хочет получить данные какой—либо выборки. Frontend посылает запрос к серверу хранилища cashed. Если нужные данные будут обнаружены, то работа прекращается. Иначе посылается запрос на более сложную и в то же время мощную базу данных backend.

1.5 Распараллеливание задач

Известным продуктом для распараллеливания задач является продукт Gearman.

Технология Gearman помогает решить проблему потери данных при сбое, при помощи постоянного хранилища. Алгоритм выглядит следующим образом:

а) при получении новой задачи, Gearman добавляет ее данные в

постоянное хранилище с уникальным идентификатором;

б) после обработки задачи сервером, Gearman удаляет ее данные из

постоянного хранилища используя уникальный идентификатор;

в) при загрузке, Gearman проверяет постоянное хранилище. Как только

будут обнаружены задачи, он загрузит их в очередь;

Уникальность заключается в том, что при любом сбое, выполненные задачи уже будут сохранены на диск и при новой загрузке они будут восстановлены. Таким образом можно защитить данные от потерь при сбое системы.

2 Разработка высоконагруженного web-приложения

Чат — это возможность обмениваться текстовыми сообщениями сразу с несколькими собеседниками.

Можно представить чат, как некоторую общую доску, на которой пишутся сообщения различными людьми. Кто—то задаѐт вопрос, кто—то отвечает, спорит и так далее. Никаких особых тем при этом не существует. Темы все время меняются. Каждое сообщение подписано. То есть, перед сообщением указано, кто из собеседников его отправил. Есть возможность общаться в общем чате с несколькими конкретными собеседниками, вмешиваться в другие разговоры, если, разумеется, позволяет тактичность.

Однако, есть возможность посылать персональные сообщения избранному собеседнику, и в данном случае это сообщение будет подсвечено как для того кто написал, так и для того кому оно адресовано.

В общем случае вид web чата представляет собой несколько блоков. В зависимости от того на основе какой технологии сделан web чат, блоками могут служить фреймы (кадры), модульные блоки либо окна браузера.

Блок сообщений чата — основной блок, в котором выводятся сообщения участников разговора. Этот блок обновляется через определѐнные интервалы времени, тем самым в нѐм постоянно отображаются последние сообщения пользователей.

Блок пользователей онлайн — в нѐм выводится список посетителей чата, находящихся в данный момент в чате. Этот блок также периодически обновляется, но интервал обновления может быть большим, по сравнению с блоком последних сообщений.

Блок отправки сообщений – блок, в котором находится поле для ввода вашего текста с управляющими кнопками.

Эти три блока являются основными для реализации web чата, но они могут быть дополнены другими блоками, для реализации тех или иных функциональных возможностей.

Структура чата разработанного проекта отображена на рисунке 2.

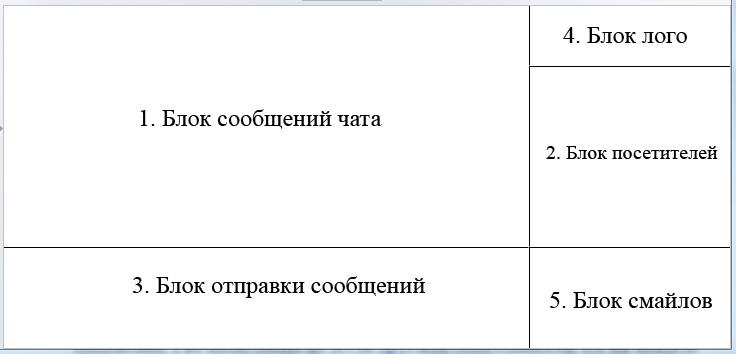


Рисунок 2 — Структура чата

а) блок сообщений чата. Основное окно, в котором отображаются последние сообщения пользователей. Для экономии трафика отображает только 15 последних сообщений. Количество отображаемых сообщений может быть изменено в конфигурационном файле config.php. Время обновления этого окна задаѐтся пользователем при входе в чат, по умолчанию равно 5 секундам;

б) блок посетителей – отображает ники посетителей, который находятся в данный момент онлайн. Обновляется через каждые 10 секунд. Время обновления может быть изменено в конфигурационном файле config.php;

в) блок отправки сообщений – В этом блоке происходит отправка пользователем сообщений в блок сообщений чата. Сообщение может быть адресным и общим. Адресное сообщение — если вы обращаетесь конкретно к кому—то из собеседников, а общее — ко всем участникам чата. И в том и в другом случае ваше сообщение читают все участники, но, если сообщение адресное, перед ним автоматически подставляется имя собеседника, к которому вы обращаетесь, и оно подсвечивается как для отправившего, так и для адресата сообщения. И в этом же блоке располагаются поле ввода сообщения, и поле ввода адресата сообщения, а также кнопка отправки;

г) блок лого– этот блок не относится к основным блокам чата. Он является одним из элементов оформления. В нѐм располагается небольшая картинка – логотип чата;

д) блок смайлов – В этом блоке располагаются смайлы;

Последние два блока не являются обязательными для web чатов, первый используется как элемент оформления, второй расширяет функциональные возможности нашего чата и позволяет пользователю использовать смайлы.

Схема проектирования базы данных указана на рисунке 3.

База данных включает две таблицы:

* данные о пользователях;
* информацию о сообщениях;

Создание таблиц, их изменение и добавление будет обрабатываться web—сервисом, написанный на языке программирования php.

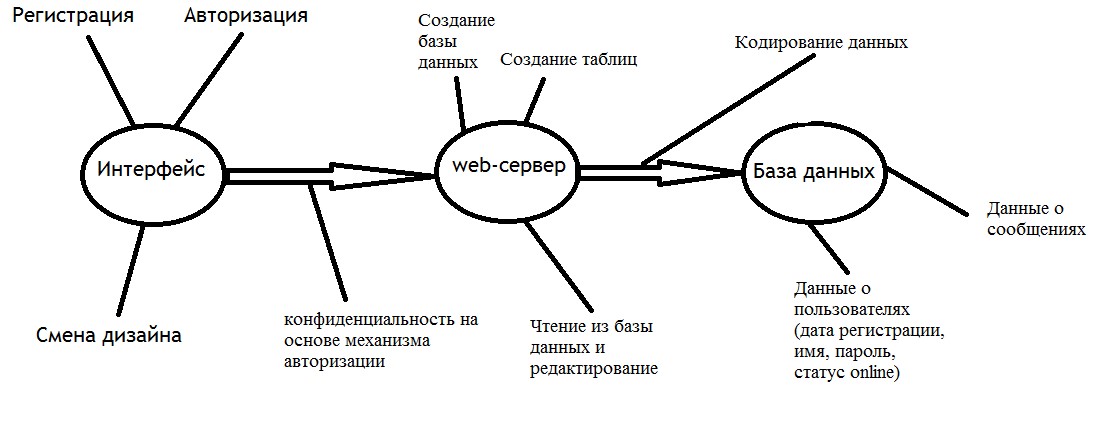


Рисунок 3 — Схема проекта

3 Реализация административной части проекта

Административная часть проекта защищена системой логирования построеной на базе механизма сессии. Логин администратора — admin, пароль администратора — admin. Основное назначение системы администрирования

— управление базой данных проекта.

3.1 База данных проекта

База данных приложения разработана в системе управления базами данных MySQL.

MySQL — свободная система управления базами данных (СУБД). MySQL является собственностью компании Sun Microsystems, осуществляющей разработку и поддержку приложения. Распространяется под GNU General Public License и под собственной коммерческой лицензией, на выбор. Помимо этого компания MySQL AB разрабатывает функциональность по заказу лицензионных пользователей, именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в LAMP. LAMP — акроним, обозначающий набор (комплекс) серверного программного обеспечения, широко используемый во Всемирной паутине.

LAMP назван по первым буквам входящих в его состав компонентов:

Linux — операционная система GNU/Linux;

Apache — веб—сервер;

MySQL — СУБД;

PHP — язык программирования, используемый для создания веб— приложений.

Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалѐнные клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

В web чате база данных используется для хранения информации о пользователях данного чата. Состоит из одной таблицы — user.

SQL запрос создания таблицы user приведен ниже:

CREATE TABLE `user` (

`id` int(11) NOT NULL auto\_increment,

`name` varchar(10) NOT NULL,

`pass` varchar(32) default NULL,

`online` smallint(1) NOT NULL default '0',

`time` bigint(15) default NULL,

`admin` int(2) NOT NULL default '0',

`block` int(1) NOT NULL default '0',

PRIMARY KEY (`id`),

UNIQUE KEY `name` (`name`)

) ENGINE=MyISAM AUTO\_INCREMENT=26 DEFAULT

CHARSET=cp1251 AUTO\_INCREMENT=26 ;

Если представить в виде таблицы, то каждая запись выглядит следующим образом на рисунке 4.



Рисунок 4 — Графическое представление записи в таблице

Поля:

а) id – содержит уникальный идентификационный номер пользователя, при добавлении нового пользователя в базу данных, автоматически увеличивается на единицу.

б) name – имя пользователя, максимум 10 символов.

в) pass – в этом поле содержится 32 битный хэш, полученный при зашифровке пароля пользователя с помощью криптографического алгоритма md5.

г) online – при авторизации пользователя это поле меняет своѐ значение на 1. При выходе пользователя посредством нажатия кнопки «Выход» меняется на 0. Позволяет отследить находится ли пользователь в режиме онлайн, при отображении списка активных пользователей.

д) time – в этом поле хранится дата и время последней активности пользователя в секундах, прошедших с начала «эпохи Unix» — полуночи 1 января 1970 года по Гринвичу. Этот формат данных принят в Unix как стандартный. Такое представление времени называется unix timestamp.

е) admin – данное поле показывает наделѐн ли текущий пользователь правами администратора, если наделѐн, то оно имеет значение 10, если нет – значение 0.

ж) block – данное поле показывает был ли пользователь заблокирован администратором, если был, то оно принимает значение 1, если нет – 0.

Заблоеированный пользователь не может осправлять сообщения.

3.2 Взаимодействие с базой данных посредством технологий PHP

В качестве серверной технологии при создании динамических страниц использовался язык программирования PHP. Все операции по взаимодействию с базой данных построенны на базе языка PHP. PHP — язык программирования, созданный для генерирования HTML—страниц на веб— сервере и работы с базами данных.

Для выполнения SQL запроса необходимо произвести следующие действия:

 соединение с сервером;

Прежде чем работать с базой данных, необходимо установить с ней сетевое соединение, а также провести авторизацию пользователя. Для этого служит функция mysql\_connect().

int mysql\_connect([string $host] [,string $user] [,string $password])

Функция mysql\_connect() устанавливает сетевое соединение с базой данных MySQL, расположенной на хосте $host (по умолчанию это localhost, т.е. текущий компьютер), и возвращает идентификатор открытого соединения. Вся дальнейшая работа ведѐтся именно с этим идентификатором. Также указывается имя пользователя $user и пароль — $password.

Соединение с MySQL — сервером будет автоматически закрыто по завершении работы сценария, либо же при вызове функции mysql\_close().

 подключение к базе данных; int mysql\_select\_db (string $dbname [,int $link\_identifier])

До того как послать первый запрос серверу MySQL, необходимо указать, с какой базой данных мы собираемся работать. Для этого и предназначена описываемая функция. Она уведомляет PHP, что в дальнейших операциях с соединением $link\_identifier будет использоваться база данных *$dbname.*

Использование этих функций и подключение к базе данных к таблице реализовано в отдельном файле — my\_sql\_base.php. Этот файл использует конфигурационный файл config.txt, в котором записаны данные для подключения к базе данных. В первой строчке – хост, во второй – имя пользователя, которому разрешено создание баз данных, добавление таблиц и изменение записей таблиц. В третьей строчке – пароль этого пользователя.

Использование конфигурационного txt файла облегчает инсталляцию чата на реальный интернет хостинг, предоставляющий возможность использования MySQL базы данных. На таких хостингах обычно выдаются имя пользователя и пароль, и такое простое и лѐгкое изменение конфигурации скрипта чата позволяет легко настроить web чат на удалѐнном хостинге.

<?php require\_once "lib/mysql\_qw.php";

$file2=file("config.txt");

$hostname = trim($file2[0]);

$username = trim($file2[1]);

$userpassword = trim($file2[2]);

$sql\_bd = @mysql\_connect($hostname,$username,$userpassword) or die("Could not connect to database.");

@mysql\_select\_db("chat");

?>

C помощью инструкции require\_once мы подключаем внешний файл lib/mysql\_qw.php в нашу программу, с проверкой на то, был ли этот файл подключѐн ранее, и если был, то не подключаем ещѐ раз. О файле lib/mysql\_qw.php подробнее будет рассказано в следующем подпункте, посвящѐнном выполнению запросов. Затем ассоциируется переменной файл config.txt и каждая его строчка присваевается различным переменным $hostname, $username, $userpassword. Затем, используя данные в этих переменных, устанавливаем соединение с сервером и проходим авторизацию с помощью описанной выше функции mysql\_connect. Следующей строчкой

подключаемся к базе данных – «chat».

Для выполнения SQL запроса используется функция mysql\_query(―SQL — запрос‖). Но так как она не поддерживает встроенное экранирование спецсимволов, и тем самым может стать брешью в защите разработанного web чата, была разработана альтернативная функция. Идея заключается в том, что вместо явного экранирования и вставки переменных в запрос на их место помещаются специальные маркеры (placeholders, «хранители места»), выглядящие как ?. Те же значения, которые будут подставлены вместо них, передаются отдельно, дополнительными параметрами. С использованием нашей функции mysql\_qw(), код которой будет описан чуть ниже, запрос к базе данных с использованием переменных мог бы выглядеть следующим образом:

mysql\_qw(‗DELETE FROM user WHERE name=?‘,$name);

таким образом, использование этой функции напоминает использование стандартной функции вывода sprintf() языка С++.

<?php ## Простейшая функция для работы с placeholder—ами.

// result—set mysql\_qw($connection\_id, $query, $arg1, $arg2, ...)

// — или —

// result—set mysql\_qw($query, $arg1, $arg2, ...)

// Функция выполняет запрос к MySQL через соединение, заданное как // $connection\_id (если не указано, то через последнее открытое).

// Параметр $query может содержать подстановочные знаки ?,

// вместо которых будут подставлены соответствующие значения

// аргументов $arg1, $arg2 и т.д. (по порядку), экранированные и

// заключенные в апострофы. function mysql\_qw() {

// Получаем все аргументы функции. $args = func\_get\_args();

// Если первый параметр имеет тип "ресурс", то это ID соединения.

$conn = null; if (is\_resource($args[0])) $conn = array\_shift($args);

// Формируем запрос по шаблону.

$query = call\_user\_func\_array("mysql\_make\_qw", $args);

// Вызываем SQL—функцию.

return $conn!==null? mysql\_query($query, $conn) : mysql\_query($query); }

// string mysql\_make\_qw($query, $arg1, $arg2, ...)

// Данная функция формирует SQL—запрос по шаблону $query,

// содержащему placeholder—ы. function mysql\_make\_qw() { $args = func\_get\_args();

// Получаем в $tmpl ССЫЛКУ на шаблон запроса.

$tmpl =& $args[0];

$tmpl = str\_replace("%", "%%", $tmpl);

$tmpl = str\_replace("?", "%s", $tmpl);

// После этого $args[0] также окажется измененным.

// Теперь экранируем все аргументы, кроме первого.

foreach ($args as $i=>$v) { if (!$i) continue; // это шаблон if (is\_int($v)) continue; // целые числа не нужно экранировать $args[$i] = "'".mysql\_escape\_string($v)."'";

}

// На всякий случай заполняем 20 последних аргументов

// недопустимыми

// значениями, чтобы в случае, если число "?" превышает количество // параметров, выдавалась ошибка SQL—запроса (поможет при

//отладке). for ($i=$c=count($args)—1; $i<$c+20; $i++) $args[$i+1] = "UNKNOWN\_PLACEHOLDER\_$i";

// Формируем SQL—запрос.

return call\_user\_func\_array("sprintf", $args);

}

?>

Извлечение набора записи из идентификатора рессурса, возвращенного функцией mysql\_query() производится при помощи функции mysql\_fetch\_assoc(). mysql\_fetch\_assoc — извлекает ряд результата как ассоциативный массив. Возвращает ассоциативный массив, соответствующий извлечѐнному ряду, или FALSE, если рядов больше нет.

Способ обработки выборки следующий:

* в цикле for;

// $result — идентификатор рессурса for (;$row = mysql\_fetch\_assoc($result);){

// обработка записей

}

* в цикле while;

// $result — идентификатор рессурса while ($row = mysql\_fetch\_assoc($result)){ // обработка записей

}

* в цикле do … while;

// $result — идентификатор рессурса do {

// обработка записей

}while ($row = mysql\_fetch\_assoc($result));

Запросы в базу данных используются в следующих случаях:

а) проверка есть ли пользователем с таким именем в базе данных, для этого формируется запрос на количество совпадений по имени, если количество больше 0, то пользователь с таким именем существует;

$sql\_count = @mysql\_qw('SELECT COUNT(\*) From user WHERE

name=?',trim($\_POST['login']));

б) регистрация;

При регистрации добавляется новая запись в базу данных.

@mysql\_qw('INSERT INTO user SET name=?, pass=?,

time=?',trim($\_POST['login']),md5(trim($\_POST['pass'])),time());

в) для вывода пользователей, которые находятся онлайн. Для этого ведѐтся выборка среди тех пользователей, у которых поле online=1 и поле time> time()—$min\_offline\*60. Последнее условие проверяет, зарегистрирована ли активность пользователя в течение последних $min\_offline минут. Если пользователь был не активен всѐ это время то, он считается оффлайн. Значение переменной $min\_offline задаѐтся в файле config.php и по умолчанию равно 20 минутам

$all\_online = @mysql\_qw('SELECT name FROM user WHERE (online=?) and (time>?)',1,(time()—$min\_offline\*60));

г) при авторизации изменяет значение поля online =1 и поля time, задавая

ему текущее время в формате timestamp

@mysql\_qw('UPDATE user SET online=?, time=? WHERE

name=?',1,time(),trim($\_POST['auth\_login']));

д) при нажатии на кнопку «Выход» изменяет значение поля online =0

@mysql\_qw('UPDATE user SET online=? WHERE

name=?',0,$\_SESSION['user']);

е) изменение поля time на текущее время, при отправки сообщения

@mysql\_qw('UPDATE user SET time=? WHERE

name=?',time(),$\_SESSION['user']);

ж) для проверки является ли активный пользователь заблокированным

@mysql\_qw('SELECT block From user WHERE

name=?',trim($\_SESSION['user']));

з) для проверки является ли активный пользователь администратором

$sql\_res = @mysql\_qw('SELECT admin From user WHERE

name=?',($\_SESSION['user']));

и) для удаления пользователя с заданным id из базы данных @mysql\_qw('DELETE From user WHERE id=?;',$\_POST['id']);

к) для блокировки пользователя с заданным id

@mysql\_qw('UPDATE user SET block=? WHERE id=?',1,$\_POST['id']);

л) для разблокировки пользователя с заданным id

@mysql\_qw('UPDATE user SET block=? WHERE id=?',0,$\_POST['id']);

м) для наделения пользователя правами администратора

@mysql\_qw('UPDATE user SET admin=? WHERE id=?',10,$\_POST['id']);

н) для снятия с пользователя прав администратора;

@mysql\_qw('UPDATE user SET admin=? WHERE id=?',0,$\_POST['id']);

п) для изменения пароля пользователя

@mysql\_qw('UPDATE user SET pass=? WHERE

id=?',trim(md5($\_POST['pass'])),$\_POST['id']);

1. Функциональность пользовательской части проекта

4.1 Описание интерфейса web проекта

При разработке дизайна проекта были использованны следующие специализированные программные пакеты обработки графики:

а) Adobe Photoshop CS 10 — растровый графический редактор, разработанный и распространяемый фирмой Adobe Systems. Этот продукт является лидером рынка в области коммерческих средств редактирования растровых изображений, и наиболее известным продуктом фирмы Adobe. Часто эту программу называют просто photoshop (фотошоп). В настоящее время Photoshop доступен на платформах Mac OS и Microsoft Windows [9, 10];

При разработке кода приложения были использованы программы Notepad++, NuSphere PhpED и Open Server.

б) Notepad++ это бесплатный редактор текстовых файлов (замена стандартного Блокнота) с поддержкой синтаксиса большого количества языков программирования, ориентирован для работы в операционной системе MS Windows. Использование данного редактора очень удобно при написании чистого кода отдельных кусков программного кода;

в) NuSphere PhpED — это профессиональная интегрированная среда обработки, предназначенная главным образом для того, чтобы формировать приложения с использованием баз данных и PHP, HTML, XML, CSS.

Устойчивый редактор кода, отличная система подсветки кода в языках PHP, XML, XHTML, HTML, CSS и JavaScript, мощный PHP отладчик, профилировщик и издатель — все в одном. Программа также включает интегрированную базу данных и клиенты CVS, сервисы SOAP, валидатор HTML и инструментальные средства Форматера Кода, поддержка работы с Smarty, редактор Unicode, SFTP поддерживается для безопасных загрузок и закачек, Telnet, SSH терминалы для удаленного администрирования, поддержка функциональных возможностей MySQL, Oracle, MSSQL SQLite, Interbase, поддержка анализа ошибок, графический интерфейс пользователя полностью настраиваемый, анализатор PHP кода;

г) Open Server — набор дистрибутивов и программная оболочка, созданный для отладки сайтов на локальной Windows — машине, без подключения к сети Интернет. Название пакета представляет собой сокращение от "джентльменский набор web — разработчика". Базовый пакет содержит: Apache2.2 , PHP5 с поддержкой GD и MySQL5, MySQL с поддержкой транзакций (mysqld — max), система управления виртуальными хостами, основанная на шаблонах, система управления запуском и завершением, phpMyAdmin — система управления MySQL через Web — интерфейс, ядро Perl без стандартных библиотек, эмулятор sendmail, инсталлятор;

4.2 Описание главной страницы проекта

Изображение интерфейса главной страницы web проекта приведено на рис 5.



Рисунок 5 — Изображение главной web страницы проекта

Главная страница состоит из пяти блоков и изображена на рисунке 6.

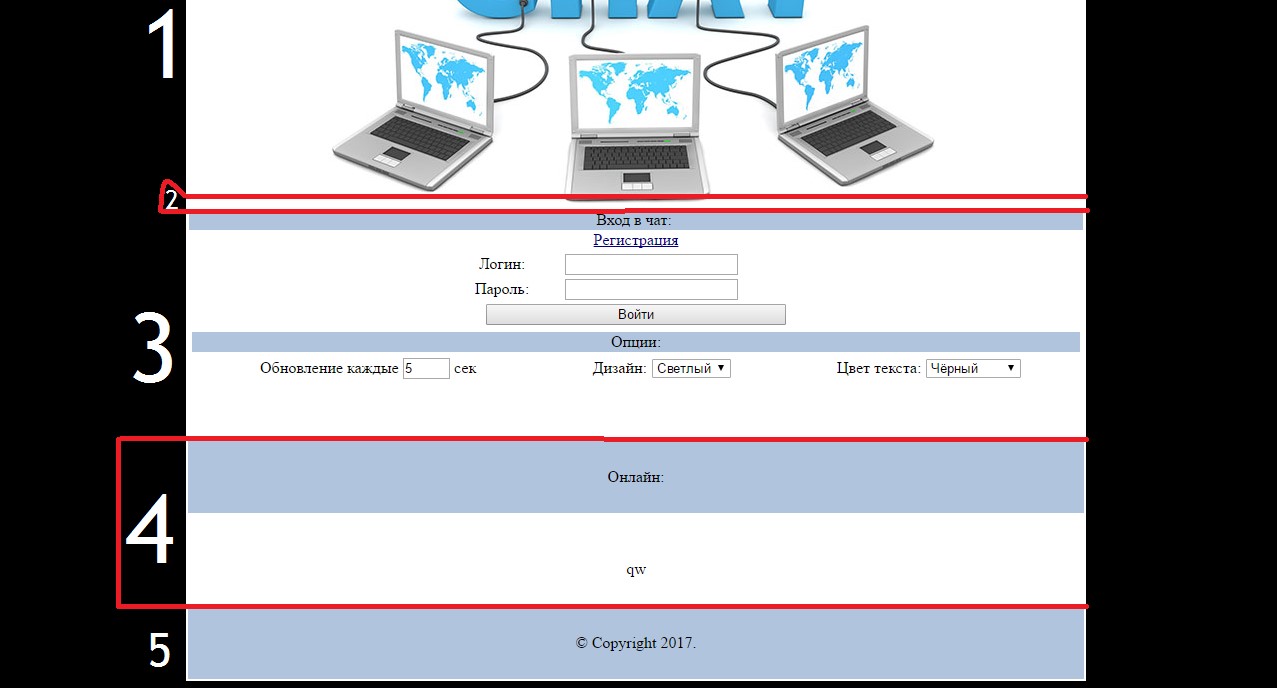


Рисунок 6 — Разделение по блокам

Скрипт index.php, который выводит главную страницу, универсален и используется ещѐ для нескольких целей. Внешний вид сгенерированной скриптом страницы зависит от параметра reg, передаваемого скрипту методом GET. В зависимости от значений этого параметра блоки меняют свой внешний вид. Рассмотрим подробнее все 5 блоков.

а) блок 1 — в котором находится шапка сайта, состоит из jpg файла с

графическим изображением шапки сайта. Не изменяется;

б) блок 2 — блок дополнительной информации, может находиться в

одном из трѐх возможных состояний:

* не содержит дополнительной информации;
* при значении ?reg=2 выводит следующую информацию, указанную

на рисунке 7;



Рисунок 7 — Сообщение блока дополнительной информации

Используется при успешной регистрации и редиректа пользователя на главную страницу, для авторизации.

* при значении ?reg=3 выводит следующую информацию, указанную на рисунке 8;



Рисунок 8 — Сообщение блока дополнительной информации

Используется при отсутствии активности пользователя в течение определѐнного времени. Время в минутах задаѐтся в файле config.php и по умолчанию равно 20 минутам.

в) блок 3 – блок активных действий. Может находиться в одном из двух

состояний:

1. форма авторизации и опций, как показано на рисунке 9;

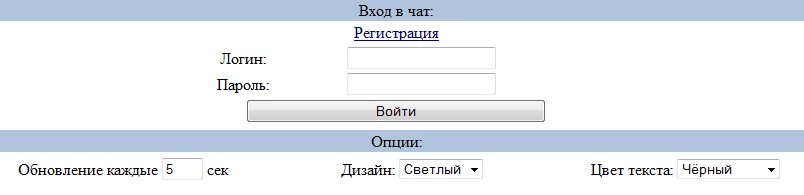


Рисунок 9 — Форма авторизации пользователей

При авторизации, происходит проверка на:

* + заполненность всех полей;
  + существование пользователя в базе данных;
  + соответствие имени пользователя и пароля;

При невыполнении хотя бы одного из этих условий выводиться соответствующее предупреждение.

В качестве опций, пользователь может настроить время обновления окна сообщений, выбрать один из двух сменных дизайнов, и выбрать цвет своих сообщений в окне сообщений.

1. при нажатии на ссылку «Регистрация» на главной странице выполняется тот же скрипт index.php, с переданным методом GET значением переменной reg, равным ?reg=1 и в блоке 3 выводится форма регистрации, показанная на рисунке 10;

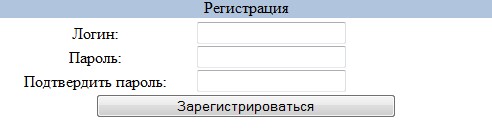


Рисунок 10 — Форма регистрации пользователей

При регистрации происходит проверка на:

* заполнение всех полей ввода;
* отсутствие пробелов в имени пользователя;
* отсутствие пользователя с таким именем в базе данных;
* совпадение содержимого поля ввода пароля, и содержимого поля ввода подтверждения пароля;

При невыполнении хотя бы одного из этих условий выводиться соответствующее предупреждение.

г) блок 4 – блок вывода списка пользователей, находящихся онлайн. Пользователь считается находящимся онлайн, если поле online записи в базе данных имеет значение 1. И кроме того поле time > time()—$min\_offline\*60.

Последнее условие проверяет, зарегистрирована ли активность пользователя в течение последних $min\_offline минут. Если пользователь был не активен всѐ это время то, он считается оффлайн. Значение переменной $min\_offline задаѐтся в файле config.php и по умолчанию равно 20 минутам;

Если ни один из пользователей не находится онлайн, то выводится надпись: Никого нет.

д) блок 5 – блок с указанием автора и разработчика реализуемого проекта, как на рисунке 11.



Рисунок 11 — Блок с указанием автора и разработчика

После прохождения авторизации пользователь попадает на страницу chat.php, которая отображает сам чат.

4.3 Описание страницы чата реализуемого проекта

При успешной авторизации пользователя, происходит редирект на страницу chat.php, которая первым делом проверяет, была ли осуществлена авторизация и в обратном случае перебрасывает пользователя на страницу index.php. Это сделано во избежание возможности ввода адреса страницы chat.php в адресной строке браузера, и тем самым обхода процедуры авторизации.

Скрипт chat.php подключает файл frames.php, структура которого была рассмотрена в главе 2 (Рис. 3.) и в зависимости от выбранного пользователем варианта дизайна выводит соответствующий вариант страницы. Каждый вариант дизайна – это определѐнный подключаемый CSS файл.

При выборе светлого дизайна подключается файл light.css, как показано на рисунке 12.



Рисунок 12 — Светлый дизайн чата

При выборе тѐмного дизайна подключается файл dark.css, как на рисунке

13.



Рисунок 13 — Тѐмный дизайн чата

Структура страницы следующая и изображена на рисунке 14.

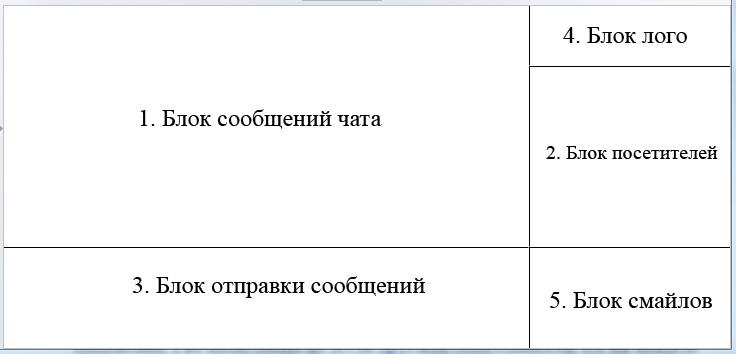


Рисунок 14 — Структура чата

Рассмотрим теперь каждый блок отдельно.

За блок сообщений чата отвечает скрипт text.php. Принцип его работы заключается в том, что с некоторой периодичностью, заданной пользователем при авторизации, он открывает файл chat.txt и построчно выводит его на страницу. Если в текущей обрабатываемой строке содержится имя данного пользователя, то сообщение в строке помещается внутрь тега <div>…</div>.

<div class=‖mess‖> Строка файла chat.txt</div>

Класс .mess в подключаемой таблице стилей характеризуется более тѐмным фоном текста, за счѐт чего возникает эффект подсветки строки сообщения, в случаях если сообщение было отправлено текущим пользователем, или адресовано текущему пользователю.

За блок посетителей онлайн отвечает скрипт users.php. Принцип его работы заключается в том, что с некоторой периодичностью, заданной в конфигурационном файле config.php и равной по умолчанию 10 секундам, выводится на странице список пользователей онлайн. Принцип построения списка пользователей описан в главе 4.2, в описании блока 4 главной страницы приложения.

При нажатии на любого из пользователей в этом списке запускается простенький javascript который добавляет имя пользователя в поле «Кому:» блока отправки сообщений.

Также данный скрипт проверяет, есть ли в списке пользователей онлайн имя данного пользователя, если его имени в списке нет, это означает, что пользователь не был активен определѐнный отрезок времени, заданный в файле config.php и по умолчанию равный 20 минутам. В этом случае происходит редирект на главную страницу, причѐм методом GET скрипту index.php передаѐтся значение переменной ?reg=3.

Ещѐ одна функция этого скрипта, это подсчѐт количества строк в файле chat.txt. Если количество строк больше величины заданной в файле config.php и по умолчанию равной 15, то скрипт переписывает данный файл, оставляя в нѐм только последние 15 записей.

За блок отправки сообщений отвечает скрипт send.php. Он проверяет заблокирован ли данный пользователь, если да, то выводится следующее сообщение, как изображено на рисунке 15.



Рисунок 15 — Сообщение о блокировке

Если пользователь не заблокирован, выводится форма со следующими элементами управления:

* поле ввода адресата;
* поле ввода сообщения;
* выпадающий список выбора текста сообщения;
* кнопка отправки сообщения;
* кнопка выхода из чата;
* ссылка на панель администратора (есть только если данный пользователь является администратором);

Эта форма изображена на рисунке 16.

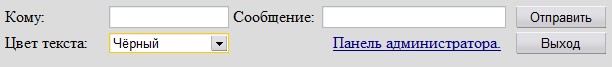


Рисунок 16 — Форма ввода сообщения

При нажатии на кнопку «Отправить» сообщение добавляется в файл chat.php, по следующей схеме:

[часы:мин:сек]<font color= ―цвет"> Адресат: Сообщение </font>

Причѐм данные полей ввода пропускаются через функцию htmlspecialchars(), которая преобразует специальные символы в HTML сущности, тем самым не позволяя пользователю ввести HTML код, который бы мог повлиять на внешний вид сообщения.

Если в тексте сообщения встречаются последовательности символов (1), (2),…,(16) они заменяются на <IMG SRC='smiles/1.gif'>, <IMG

SRC='smiles/2.gif'>,…, <IMG SRC='smiles/16.gif'>. Тем самым преобразуя эти последовательности в HTML—код для вывода на экран смайлов.

При нажатии на кнопку «Выход» запускается скрипт out.php, который изменяет значение поля online базы дынных на 0, после чего происходит редирект на главную страницу.

За содержимое этого блока отвечает файл logo.php. В зависимости от выбора варианта дизайна выводит светлое либо тѐмное лого.

За блок смайлов отвечает скрипт smiles.php. Он в цикле выводит в фрэйм 16 смайлов. Каждый смайл при нажатии добавляет в поле ввода сообщения последовательность символов, которая в последствии будет заменена на HTML—код для вывода на экран смайла.

За вывод страницы панели администратора отвечает скрипт admin.php. При загрузке этого скрипта, проверяется была ли пройдена авторизация и если да, то обладает ли авторизированный пользователь правами администратора.

Если нет, то происходит редирект на главную страницу.

Права администратора указаны на рисунке 17.

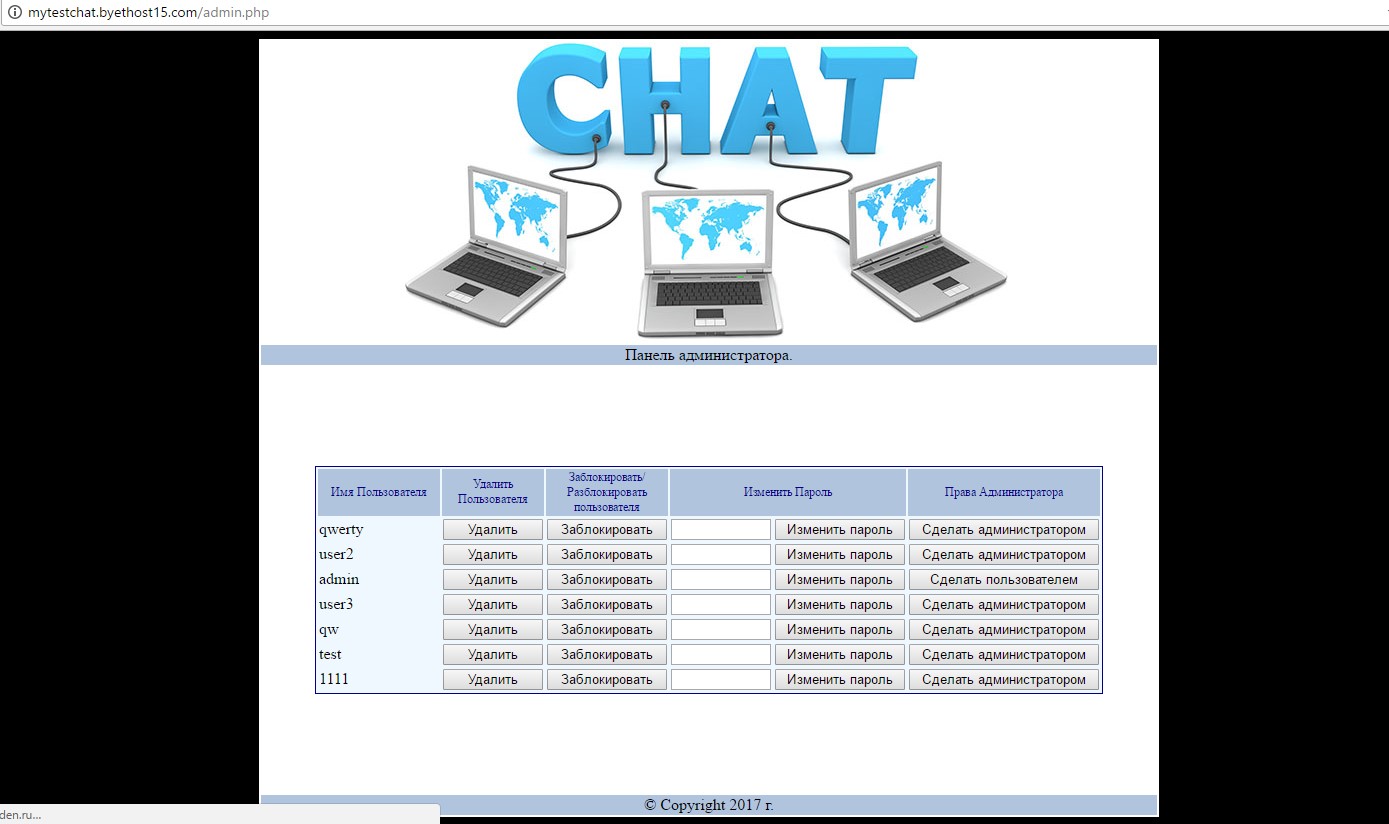


Рисунок 17 — Панель администратора

С помощью панели администратора можно:

* удалить пользователя из базы данных;
* заблокировать/разблокировать пользователя;
* изменить пароль пользователя;
* наделить пользователя правами администратора;
* снять с пользователя права администратора;

Для того чтобы разработанное приложение могло быть без проблем использовано на удалѐнном хостинге, необходимо чтобы оно обладало удобной системой инсталляции и конфигурирования, чего лишены большая часть бесплатных web чатов, доступных для скачивания в интернете.

Конфигурирование чата осуществляется путѐм редактирования файла config.php.

<?php

$user\_refresh = 10;

//Через сколько секунд обновляется список пользователей

$num\_lines = 15;

//Сколько сообщений хранится в памяти и выводится на экран

$min\_offline = 20;

// Если пользователь бездействует это количество минут, он //считается вышедшим из чата

?>

За инсталляцию чата отвечает скрипт install.php. Он использует конфигурационный файл config.txt. И сохраняет в нѐм данные введѐнные при инсталляции, эти данные затем используются для доступа к базе данных.

При нажатии на кнопку «Установить» скрипт с помощью SQL – запросов:

* создаѐт базу данных;
* входит в созданную базу данных;
* создаѐт в ней таблицу user, для хранения данных о пользователях;

ЗАКЛЮЧЕНИЕ

При работе над дипломной работой, автором работы был изучен теоретический материал, посвященный:

* свойствам и подходам при разработке высоконагруженных распределенных приложений;
* особенностям средств общения реального времени, в частности чатам;

При изучении материала по вышеуказанным пунктам были определенны следующие особенности чатов, обязательное существование следующих блоков:

* блок сообщений чата;
* блок посетителей онлайн;
* блок отправки сообщений;

Таким образом, на базе существующих особенностей при разработке проекта учитывались обязательные элементы, плюс были добавлены дополнительные блоки расширяющие функциональные возможности реализуемого проекта.

Для приобретения знаний при написании практической части были изучены следующие сетевые технологии:

а) язык гипертекстовой разметки HTML — на базе HTML был разработан

шаблон web проекта;

б) каскадные листы форматирования CSS — на базе CSS основан

механизм сменных дизайнов и общее форматирование текста и графики;

в) язык серверного программирования PHP — использовался при разработке функциональности проекта и создании динамических наборов данных;

г) система управления базами данных MySQL — база данных

приложения разработанна в MySQL;

Практически в работе разработан web чат. С точки зрения функциональности, проект разделен на 3 составляющие: пользовательскую, административную и инсталляционную.

В административной части администратор проекта может производить операции по управлению пользователями. Административная часть в целях безопасности закрыта механизмом авторизации.

Пользовательская часть проека обладает следующей функциональностью:

* конфиденциальность на основе механизма авторизации;
* возможность общения в реальном времени;
* возможность изменения цвета сообщений;
* возможность использования смайлов;
* возможность написания адресных сообщений;

Инсталляционная часть позволяет при минимальных усилиях добиться функционирования приложения на локальном либо удалѐнном сервере.

На текущий момент web чат находится на стадии тестирования и в скором времени будет запущен.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Петюшкин А. HTML экспресс курс — СПб.: БХВ—Петербург, 2003. —

258с.

1. Библиотека функций PHP [Электронный ресурс]. Статья, URL: http://www.phpclub.net (дата обращения: 23 апреля 2017).
2. Томсон Л. Разработка Web—приложений на РНР и MySQL: Пер. с англ./Лаура Томсон, Люк Веллинг — СПб.: ООО «ДиаСофтЮП», 2003. — 672 с.
3. Д.В.Котеров,А.Ф.Костарев. PHP5 в подлиннике — СПб.: БХВ— Петербург, 2005. — 1120с.
4. Высоконагруженные Интернет—проекты [Электронный ресурс]. Статья, URL: http://www.insight—it.ru/highload/voprosy—i—otvety/ (дата обращения: 22 марта 2017).
5. Архитектура высоких нагрузок [Электронный ресурс]. Статья, URL:

https://ruhighload.com/post/%D0% (дата обращения: 2 мая 2017).

1. Выбор технологий для большого и не очень большого веб—проекта [Электронный ресурс]. Статья, URL: https://habrahabr.ru/company/SECL\_GROUP/blog/315734/ (дата обращения:

21мая 2017).