МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

Доска объявлений

Курсовая работа

09.03.04 Программная инженерия

Допущен к защите

Заказчик _____ *В.С. Тарасов* __.__.20__

Обучающийся _____*М.Е. Новоточинов, 3 курс*

Воронеж 2020

Содержание

1. Постановка и анализ задачи

Постановка задачи: разработать приложение для организации электронной доски объявлений.

Исходные данные к работе: Windows 10, интернет-сервер, наличие БД, доступ к приложению через WEB-интерфейс. Требуется обеспечить механизм регистрации и авторизации пользователей, возможность добавления, разделения по темам каталога, даньнейшего редактирования и удаления объявлений.

Приложение должно взаимодействовать с пользователями, то есть быть интерактивным, поэтому должно быть написано на одном из языков программирования web-сценариев, например PHP. Очевидно, что приложению требуется оперировать с большими массивами данных, поэтому для их надежного хранения потребуется база данных, поскольку случай хранения информации непосредственно в файлах на сервере не является надежным и безопасным из-за проблем со множественным доступом. Приложение устанавливается на интернет-сервере, доступ пользователей к приложению осуществляется через web-интерфейс с помощью браузера, например Google Chrome. Таким образом подразумевается, что взаимодействие с приложением будет вестись только через HTTP-протокол.

Структура интерфейса электронной доски объявлений должна быть понятна для нового пользователя, в то же время необходимо позаботиться об наборе функциональных средств, обеспечивающих удобство работы с набором объявлений. Для этого разделим все объявления на рубрики и организуем их отображение в виде логического дерева разделов и подразделов каталога. Поскольку у каждого объявления есть определенный тип («спрос», «обмен», «предложение» и т.п.), нужно позаботиться о том чтобы пользователь мог просматривать объявления только нужного ему типа.

В целях обеспечения порядка и безопасности объявления могут добавлять только зарегистрированные пользователи. При регистрации у

пользователя запрашивается логин, пароль и e-mail. Для того чтобы исключить хранение паролей пользователей в базе данных в открытом виде, нужно предусмотреть их шифрование (хэширование).

Зарегистрированные пользователи проходят процедуру авторизации, в которой у них запрашивается логин и пароль. Так как используется протокол HTTP, все отправляемые данные идут от пользователя к серверу в открытом виде.

Таким образом, возникает понятие «сессии пользователя» - при успешной авторизации на данного пользователя открывается так называемая сессия, пользователю средствами языка программирования сообщается только указатель на эту запись. При выполнении каких-либо операций с данными пользователь «возвращает» этот указатель приложению, которое сначала проверяет, есть ли у пользователя по полученному указателю открытая сессия и только потом выполняет требуемые действия.

Каждому зарегистрированному пользователю выделяется свой личный аккаунт, из которого он может добавлять, удалять и редактировать уже отправленные объявления.

Необходимо предусмотреть режим администрирования с возможностью редактирования основных параметров электронной доски объявлений, таких как название, число объявлений, отображаемых на одной странице.

Администратор может создавать и удалять разделы и подразделы каталога доски объявлений, просматривать список зарегистрированных пользователей и удалять пользователей.

Режим модерирования предназначен для удаления любого объявления из каталога. Любой зарегистрированный пользователь может являться модератором доски объявлений, если администратор наделит его соответствующими правами доступа.

2. Разработка схемы данных

Для хранения данных в разрабатываемом приложении используется БД MySQL, формат данных которой принято представлять в табличной форме. Структурную схему базы данных электронной доски объявлений можно представить в виде набора из восьми таблиц, информация в каждой из которых группируется по смысловому и функциональному назначению и хранится в различных полях. Таким образом, приложение с помощью SQL-запросов обращается к БД только к нужным таблицам и полям и затем выполняет различные операции с полученными данными. Благодаря такому механизму достигается увеличение скорости обмена данными между приложением и БД.

Рассмотрим назначение и структуру таблиц, используемых в проекте:

1. Таблица **OPTIONS**.

Статическая таблица, предназначена для хранения основных параметров электронной доски объявлений, состоит из трёх полей:

id – порядковым номер записи, тип поля smallint (допустимое значение до 32767), ключевое.

value – значение параметра, тип text.

name – название параметра, тип поля text (до 65535 символов)

В данной версии проекта в таблице содержится шесть записей, которые заносятся при инсталляции. Содержание записей поля name: название BBS, число отображаемых на одной странице объявлений, максимальное время жизни объявления, рассылка объявлений по почте, удаление объявлений по истечению времени жизни, максимальный размер объявления.

При необходимости администратор может изменить любое из первоначальных значений поля value.

2. Таблица **ACTION**.

Статическая таблица, содержит тип объявлений, состоит всего из двух полей:

action – название типа объявления, тип поля text.

id – порядковым номер записи, тип поля smallint, ключевое.

Содержание записей поля action: предложение, спрос, обмен, аренда, прочее. Значения полей задаются автоматически в процессе инсталляции и в последующем времени не изменяются.

3. Таблица **SUBJECT**.

Предназначена для хранения индекса основного каталога (содержит информацию о структуре разделов и подразделов, образующих каталог)

id – идентификатор раздела или подраздела, тип поля int (значение до 2 147 483 647), ключевое. Для того чтобы данный идентификатор был уникальным (с неповторяющимися значениями), полю назначен дополнительный тип auto_increment. При первоначальном создании таблицы значение этого поля равно единице, при добавлении новой записи его значение автоматически инкрементируется. Поскольку удаление записей из таблицы не влияет на значение этого счётчика, мы получаем уникальность идентификатора записи. Для оптимизации поиска по таблице устанавливаем тип поля index.

topic – значение идентификатора раздела каталога, тип поля int, ключевое. Если данная запись описывает не подраздел, а корневой раздел каталога, то значение поля равно 0.

name – название раздела или подраздела каталога, тип поля text.

Поскольку администратор каталога может добавлять и удалять разделы и подразделы, число записей в этой таблице непостоянно.

4. Таблица **USERS**.

Содержит информацию о зарегистрированных пользователях.

id – идентификатор пользователя, тип поля int, ключевое, auto_increment, index.

login – логин пользователя, тип поля text.

password – 32-х символьный хэш-код пароля пользователя, тип поля text.

contact – контактная информация пользователя, тип поля text.

access – права доступа, тип поля smallint. 0 - администратор, 1 – пользователь, 2 – модератор.

5. Таблица **MESSAGES**.

Предназначена для хранения текста объявлений и их параметров, заданных отправителем.

id – идентификатор объявления, тип поля int, ключевое, auto_increment, index.

topic_id – значение идентификатора подраздела каталога, тип поля int, index.

user_id – значение идентификатора отправителя (пользователя), тип поля int.

action_id – значение идентификатора типа объявления, тип поля smallint.

time – дата написания объявления, тип поля text.

time_live – время удаления объявления в UNIX-формате, тип поля bigint.

text – текст объявления, тип поля text.

6. Таблица **SESSIONS**.

Содержит значения идентификаторов сессий авторизованных пользователей, используется для реализации механизма безопасной аутентификации.

user_id – значение идентификатора пользователя в таблице USERS, тип поля int.

sid – идентификатор сессии авторизованного пользователя, тип поля text.

time – значение времени в UNIX-формате, по истечению которого не продленная сессия будет автоматически удалена из таблицы, тип поля bigint.

Связи между полями таблиц приведены на рис. 2.1 структурной схемы данных. Ключевые поля-идентификаторы помечены знаком “*”, типы используемых связей: “один ко многим” и “один к одному”.

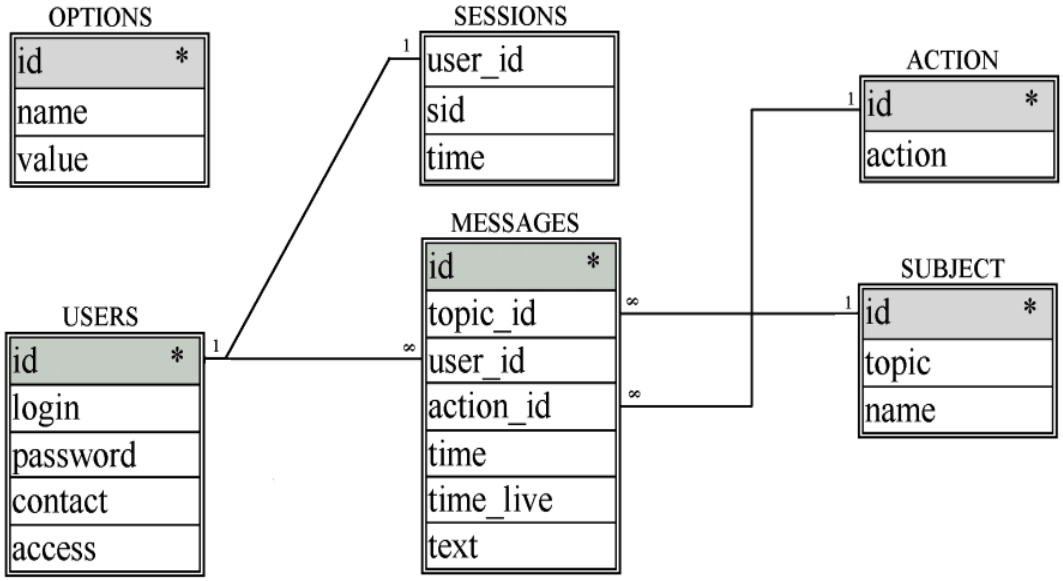

Рис. 2.1 - Схема данных

3. Разработка схемы программы

Рассмотрим основные задачи и требования, предъявляемые к разрабатываемому приложению на уровне организации WEB-интерфейса:

1) интерфейс отображение разделов каталога и объявлений,

2) интерфейс аккаунта пользователя,

3) интерфейс аккаунта модератора,

4) интерфейс аккаунта администратора,

5) интерфейс авторизации и регистрации пользователей.

На функциональном уровне:

1) первоначальная инсталляция приложения на сервере,

2) соединение с БД MySQL,

3) инициализация основных параметров каталога,

4) проверка на корректность значений переменных, принимаемых от пользователя,

5) вывод разделов и подразделов каталога, а также объявлений,

6) регистрация новых пользователей,

7) авторизация пользователей,

8) аутентификация пользователей с помощью механизма сессий и проверка прав доступа,

9) добавление, редактирование и удаление объявлений,

10) редактирование основных параметров приложения,

11) добавление и удаление пользователей,

12) установка и снятие прав доступа с пользователей,

13) автоматическое удаление объявлений по истечению срока жизни, удаление не продлённых пользовательских сессий, удаление не активированных адресов почтовой рассылки.

На рис. 3.1 показана общая схема приложения и взаимодействие между его основными частями.

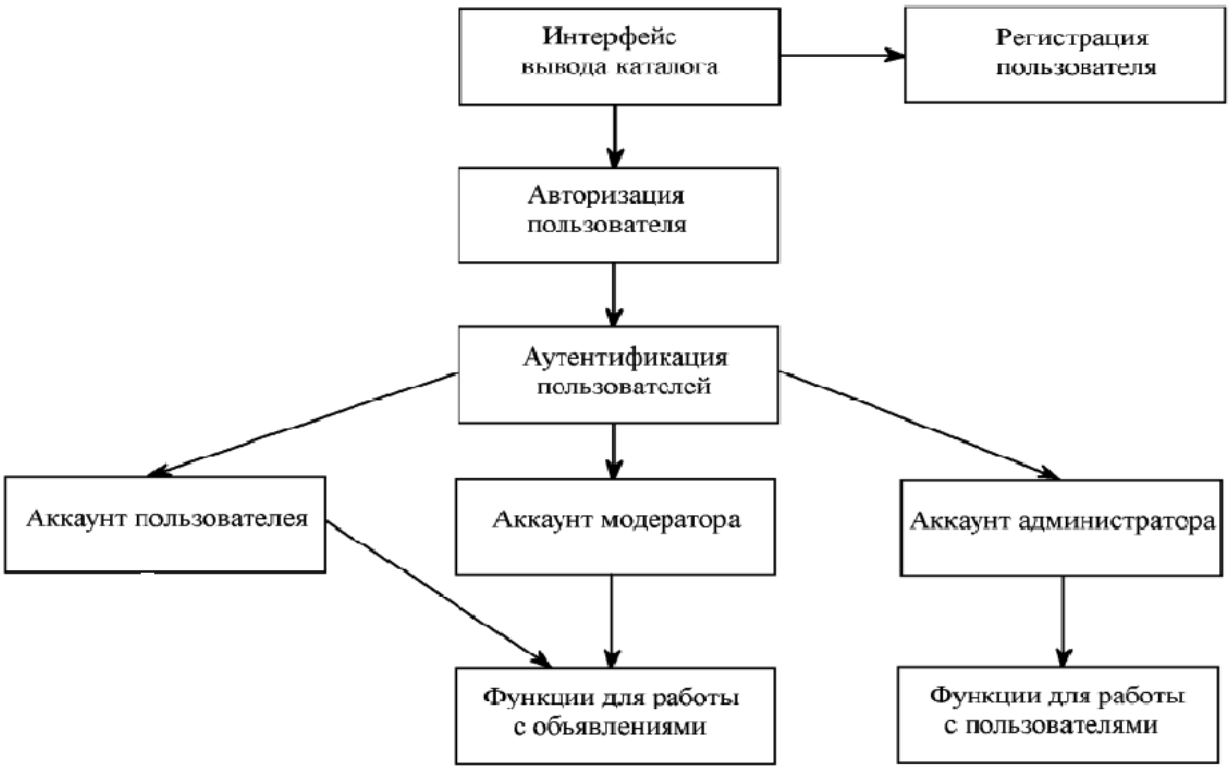

Рис. 3.1 - Функциональная структура программы

Таким образом, проект целесообразно реализовать в виде нескольких функциональных модулей, каждый из которых будет выполнять определённую задачу:

1) модуль инсталляции (с отображением интерфейса),

2) модуль соединения с БД MySQL,

3) модуль отображения разделов каталога и объявлений (с отображением интерфейса),

4) модуль регистрации новых пользователей (с отображением интерфейса),

5) модуль авторизации пользователей (с отображением интерфейса),

6) модуль аутентификации пользователей, основанном на механизме сессий,

7) модуль реализации аккаунта пользователя (с отображением интерфейса),

8) модуль реализации аккаунта администратора (с отображением интерфейса),

9) модуль реализации аккаунта модератора (с отображением интерфейса)

4. Разработка алгоритмов

1) Инсталяция.

Инсталляция подразумевает первоначальную установку приложения на интернет-сервер и создание аккаунта администратора. После инсталляции администратор переходит в свой аккаунт и добавляет разделы и подразделы в основной каталог.

Модуль-инсталлятор выполняет следующие действия:

а) создает новую БД или удаляет все таблицы в текущей БД, использующиеся приложением, если они уже были созданы;

в) создает таблицы с указанием всех необходимых полей и типов;

г) добавляет записи со значением базовых параметров в таблицы ACTION и OPTIONS.

г) запрашивает логин и пароль администратора, добавляет запись в таблицу USERS и установить права администратора.

2) Отображения разделов и подразделов.

Для отображения списка разделов необходимо послать запрос на получение всех значений поля name из таблицы SUBJECT, где значение идентификатора раздела каталога topic = 0, и вывести результат на экран.

Для отображения списка подразделов используется точно такой же метод, только значение идентификатора раздела каталога topic сравнивается с текущим идентификатором раздела.

Для того чтобы записи сортировались в алфавитном порядке, все запросы дополняются директивой «ORDER BY name ASC». Если в текущем подразделе каталога присутствуют объявления нескольких типов, выводится вспомогательный список-фильтр с перечислением найденных типов, тогда пользователь может выбирать тот или иной тип объявлений, который должен выводиться на всех разделах и подразделах каталога.

3) Алгоритм отображения объявлений с применением фильтра на тип объявлений.

Для отображения объявлений берутся все значения полей таблицы MESSAGES, в которой значение поля topic_id совпадает со значением текущего идентификатора раздела каталога. Поскольку при отображении объявлений необходимо выводить ещё и контактную информацию об отправителе, а также тип объявления, в запрос вводятся дополнительные условия: значение поля user_id таблицы MESSAGES должно совпадать со значением идентификатора пользователя id таблицы USERS, а значение поля action_id – со значением поля id таблицы перечисления типов объявлений ACTION. Если пользователь применил фильтр на тип объявлений, запрос дополняется ещё одним условием: значение поля action_id должно соответствовать значению фильтра. Сортировка записей производится по

полю id в порядке убывания, в результате чего вверху WEB-страницы каталога отображаются последние добавленные объявления.

4) Ограничение числа объявлений, выводимых на одной WEB-странице.

Поскольку число объявлений, удовлетворяющих вышеприведённому запросу может быть слишком велико, для того чтобы не перегружать выводимые WEB-страницы используется алгоритм ограничения числа объявлений и формирование так называемой линейки – ссылок на страницы, содержащие результаты запроса. Для этого с помощью директивы COUNT предварительно подсчитывается количество строк в результирующем запросе, причём приложение запрашивает у БД именно число строк в нужном запросе, а не результат запроса, тем самым экономя время обмена данными и трафик (в случае если БД находится на другом удалённом сервере). По полученному значению определяется, нужно ли формировать линейку или нет. Если нужно, запрос дополняется директивой LIMIT с перечислением диапазона выводимых значений в результирующем запросе. Диапазон представляет собой некоторое начальное значение указателя на «окно» и максимальный размер этого «окна». Таким образом, результат запроса делится на «окна» и не перегружает выводимые страницы. Пользователь может выбирать для просмотра то или иное «окно», при этом само содержание запроса не изменяется, меняются лишь границы диапазона.

5) Передача значения фильтра при переходах по ссылкам каталога.

Включив фильтр на тип объявлений, пользователь может перейти с текущей страницы каталога на какую-либо другую, причём во время таких переходов приложение должно «знать», что фильтр был включен. Самый оптимальный способ для сохранения значения фильтра в случае программирования на языке PHP – это использование механизма сессий.

6) Регистрация новых пользователей.

Регистрация пользователя происходит в следующей последовательности: на запрос программы пользователь вводит логин, пароль и контактную информацию о себе или об организации, которую он

представляет на доске объявлений. Для удобства работы администратора со списком логинов пользователей проводится проверка на допустимый набор символов, состоящий только из английских букв и цифр. Для того чтобы исключить регистрацию пользователей с одинаковыми логинами, необходимо сделать проверку по полю login таблицы USERS. После вышеперечисленных проверок вычисляется хэш-код пароля пользователя. Хэш-код представляет собой 32-х символьную последовательность, вычисленную по алгоритму «MD5» (разработка корпорации RSA Data Security, [3]). Полученная последовательность является уникальной для символьной строки пароля пользователя, т.е. вероятность того, что два одинаковых пароля дадут одинаковый хэш-код, стремится к нулю. Основное преимущество кодирования по алгоритму MD5 – это невозможность обратного восстановления первоначальной символьной строки (пароля) по полученному хэш-коду. Вся полученная информация добавляется в поля login и password таблицы USERS, значение поля access устанавливается равным 1, что означает «права доступа на уровне пользователя».

7) Авторизация пользователей.

Пользователю выводится форма для запроса логина и пароля, после ввода значений вычисляется хэш-код пароля и посылается запрос к таблице USERS. Если у пользователя есть учётная запись, соответствующая этим значениям, то он считается авторизованным. Прочитав значение поля id, мы получим идентификатор пользователя. Затем необходимо позаботиться об информации для модуля аутентификации, для этого используется таблица сессий SESSION. Поле userr_id этой таблицы содержит значения идентификаторов пользователей с открытыми сессиями. Поскольку на один и тот же логин может быть открыта только одна сессия, сначала из таблицы удаляется запись с полученным ранее значением идентификатором авторизованного пользователя. Далее случайным образом генерируется 20-символьная строка, которую обозначим как «код сессии авторизованного пользователя». Именно эта строка, а не хэш-код пароля пользователя (или тем

более его истинный пароль) впоследствии будет передаваться между приложением на сервере и браузером клиента. Код сессии заносится в поле sid. В целях безопасности, для того чтобы открытые сессии имели ограниченный во времени срок действия, каждой сессии назначается время жизни time, равное текущему значению времени на сервере плюс 60 минут. В случае, если пользователь ввёл ошибочные данные и его учётная запись не была обнаружена, ему предлагается либо заново пройти процедуру авторизации, либо зарегистрировать новую учётную запись.

8) Аутентификация пользователей.

Обращение к данным идёт через запросы к таблице сессий SESSIONS и таблице пользователей USERS. Сначала идёт попытка считать 20-х символьный хэш-код сессии авторизованного пользователя, передаваемый между приложением и авторизованным клиентом. Затем из таблицы SESSIONS удаляются все сессии с просроченным временем жизни, если таковые имеются. Для этого текущее значение времени на сервере сравнивается со значением поля time.

Из таблицы SESSIONS получаем значение id той записи, у которой значение sid совпадает с хэш-кодом сессии авторизованного пользователя, после чего из таблицы USERS выбирается учётная запись с полученным идентификатором пользователя id. Эта учётная запись содержит идентификатор пользователя, логин, контактную информацию и права доступа, которая в дальнейшем используется различными модулями, требующими аутентификацию.

Далее продлевается срок жизни сессии: обновляется значение поля time.

В том случае, если 20-х символьный хэш-код сессии авторизованного пользователя считать не удалось, или такая сессия была не найдена в таблице SESSIONS, то аутентификация не возможна и пользователь считается неавторизованным.

Если пользователь закрывает сессию самостоятельно, происходит немедленное удаление записи из таблицы SESSIONS.

9) Добавление новых объявлений.

Сначала идёт проверка максимально допустимой длины объявления, значение которой хранится в одной из записи таблицы OPTIONS. Объявления короче пяти символов также считаются недопустимыми. Поскольку сообщение может содержать недопустимые HTML-тэги, происходит замена таких тэгов на их «безопасные» эквиваленты.

Из таблицы MESSAGES удаляются сообщения, у которых истёк срок жизни, затем добавляется новая запись со значением полей topic_id, равному идентификатору текущего подраздела каталога доски объявлений в таблице SUBJECT; userr_id, указывающим на идентификатор пользователя, отправившего объявление в таблице USERS; action_id, указывающим на идентификатор типа сообщения в таблице ACTION; time, равный значению текущей даты; time_live, обозначающим срок жизни объявления, и text, содержащий текст самого объявления.

В случае невыполнения условий проверок объявление в БД не заносится.

5. Программная реализация

Для реализации проекта был выбран широко распространенный на сегодняшний день язык PHP – язык описания сценариев, которые встраиваются непосредственно в гипертекстовые HTML-файлы и исполняются на Web-сервере, что в значительной степени упрощает написание готовых проектов, сокращает время разработки структуры визуального отображения выводимой информации. Программа на PHP заключается в теги <?, ?>, а интерпретатор обрабатывает команды между этой парой тегов и формирует окончательный результат, передающийся на локальную машину.

PHP поддерживает множество реляционных баз данных, в том числе Oracle. Тем не менее немало скриптов на PHP используют сравнительно небольшую и компактную СУБД MySQL, совместимую со стандартом ANSI SQL и обеспечивающую высокую производительность. MySQL является реляционной СУБД и в SQL-запросах позволяет связывать таблицы по общим полям, поддерживает индексы, автоинкрементные поля, а также множество функций для преобразования данных. К MySQL разработаны самые разнообразные надстройки, предоставляющие графический или Web-интерфейс для манипуляции данными - создания таблиц, добавления и редактирования в них записей, отбора нужных строк. Например, с помощью системы phpMyAdmin, написанной целиком на PHP, можно подготовить структуру таблиц, ввести начальные значения вручную или из текстового файла и проверить работоспособность SQL-запросов, использующихся в проекте.

Работа с базой данных

Последовательность подключения к базе данных и управления табличными данными традиционна – сначала устанавливается связь, потом выдается запрос и обрабатывается результат. Для подключения к базе данных нужны три параметра: имя хост-узла, имя пользователя и пароль. Определив

эти параметры, можно установить соединение с БД при помощи функции MYSQL_CONNECT(), затем функцией MYSQL_SELECT_DB() выбирается нужная база данных (см. Приложение А). В случае ошибки соединения или выбора БД выводится соответствующее сообщение.

Получить значения отдельных полей позволяет функция MYSQL_FETCH_ARRAY(), которая возвращает ассоциативный массив с данными (см. Приложение В).

Для получения данных используется функция MYSQL_QUERY(), в которой указывается необходимый SQL-запрос, передаваемый MySQL.

Для получения количества строк в результате запроса служит функция MYSQL_NUM_ROWS(), которая используется для определения наличия в таблице той или иной записи (например, учетной записи пользователя).

Составление SQL-запросов

Чтобы начать работу с MySQL прежде всего требуется создать базу данных, с которой мы будем взаимодействовать. Это делается запросом “CREATE database имя_БД”.

Для создания таблицы используется запрос “CREATE table имя_таблицы” с перечислением полей и их типов (см. Приложение В). Для добавления новой записи в таблицу – “INSERT INTO имя_таблицы VALUES (перечисление_значений)”, для вывода данных “SELECT перечисление_полей FROM имя_таблицы WHERE условие ”.

Важной особенностью является возможность дополнения запросов следующими функциональными директивами: лимит выдаваемых строк “LIMIT”, сортировка данных в порядке убывания/возрастания значения поля “ORDER BY название_поля DESC/ASC”, подсчёт строк в запросе “COUNT (*)”.

Для редактирования значения поля используется запрос “UPDATE имя_таблицы SET имя_поля = значение поля WHERE условие”, для удаления “DELETE FROM имя_таблицы WHERE условие”. Практическая реализация вышеперечисленных запросов в коде приведена в Приложении.

Обработка данных, полученных от пользователя

Для проверки строк данных, полученных от пользователя на допустимые значения символов, используется функция ereg() и составленное соответствующим образом регулярное выражение, которым проверяется строка, передаваемая этой функции. Например, регулярное выражение “[^0-9]” эквивалентно по смыслу фразе “проверка на наличие в строке цифровых символов”. Регулярными выражениями проверяются все цифровые значения переменных, а так же строковые переменные логина, пароля и e-mail’а.

Шифрование паролей, вычисление хэш-кода символьной строки

Существует множество способов зашифровать или вычислить хэш-код какой-либо символьной строки, язык PHP предоставляет для этого целый

набор функций (crypt, crc32 и др.). Поскольку мы уже ранее определились с алгоритмом MD5, будем использовать функцию md5().

Формирование интерфейса

Формирование интерфейса происходит стандартным образом на языке гипертекстовой разметки HTML. Для увеличения скорости загрузки страниц каталога электронной доски объявлений весь дизайн интерфейса выполнен в текстовом виде. Автору удалось минимизировать HTML-код путем исключения избыточности (ненужных символов, HTML-тэгов, замена однотипных участков текста на вызов функции JavaScript, содержащий шаблон с этим текстом и т.п.), которая на сегодняшний день весьма распространена в Web-дизайне.

6. Описание программы

Разрабатываемое интернет-приложение предназначено для организации интерактивной многофункциональной электронной доски объявлений. Приложение устанавливается на сервере, поддерживающим выполнение PHP-скриптов не ниже третьей версии. Для работы приложения с данными необходима СУБД MySQL версии 14.0 или выше, которая в целях максимальной производительности должна быть установлена на том же сервере, что и интерпретатор языка PHP, однако это условие не является обязательным.

Для установки приложения необходимо по FTP-протоколу или любым другим образом скопировать файлы с поставляемого дистрибутива в каталог выделенного сервера, прописать название БД, логин и пароль для доступа (выдаются администратором интернет-сервера) в модуле tunes.php и запустить инсталлятор install.exe. После ввода регистрационной информации администратора электронная доска объявлений сразу же готова к работе.

Просмотр объявлений осуществляется через интуитивно-понятный WEB-интерфейс.

Для удобства пользователя, все объявления разделены на тематики, и содержатся в упорядоченном порядке в разделах каталога. Каждому объявлению соответствует тип – «предложение», «спрос», «обмен», «аренда», «прочее». Пользователь может включить фильтр на отображение объявлений только нужного ему типа.

Объявления могут добавлять только зарегистрированные пользователи. При регистрации у пользователя запрашивается пароль, логин и контактная информация. После регистрации пользователь может работать в своем аккаунте, редактировать уже отправленные объявления, удалять и добавлять.

Предусмотрен режим администрирования с возможностью редактирования основных параметров электронной доски объявлений, таких как число объявлений отображаемых на одной странице, название.

Администратор может просматривать список зарегистрированных пользователей, а так же удалять пользователей и создавать и удалять разделы и подразделы каталога доски объявлений.

Режим модерирования предназначен для удаления любого объявления из каталога. Любой зарегистрированный пользователь может являться модератором доски объявлений, если администратор наделит его соответствующими правами доступа.

Авторизация пользователей основана на безопасном алгоритме аутентификации, который обеспечивает надежную защиту от взлома паролей пользователей для приложений подобного уровня и имеет простой интерфейс.

7. Контрольный пример

Интерфейс приложения выглядит следующим образом:

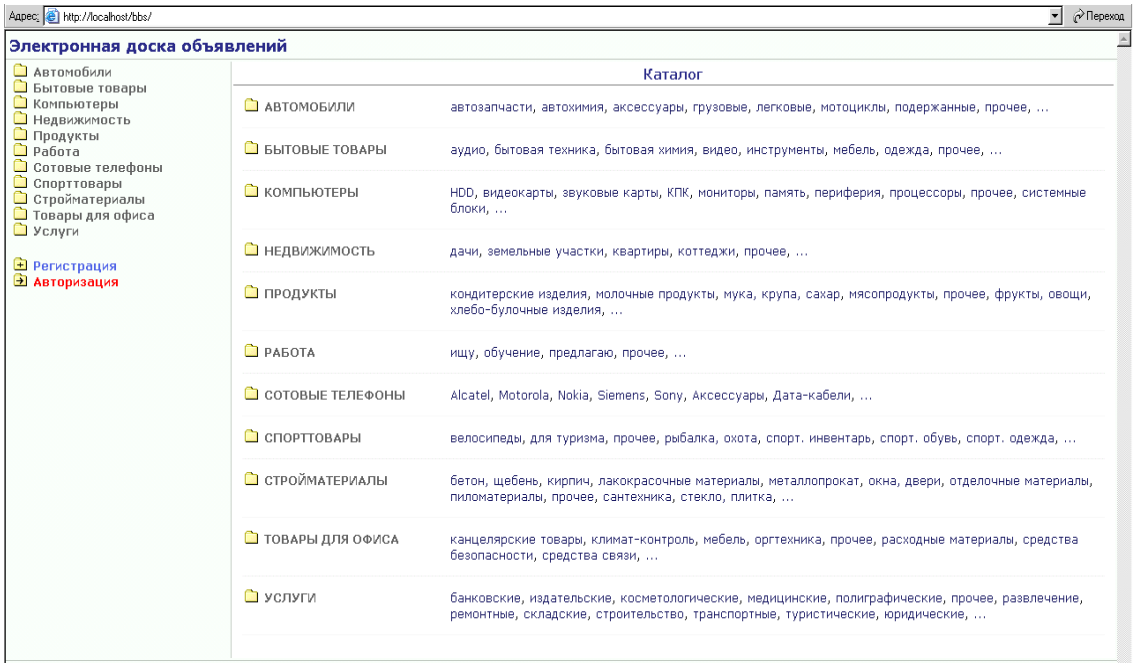

Рис. 8.1 - Вывод разделов и подразделов основного каталога электронной доски объявлений

Рассмотрим процесс регистрации нового пользователя, например *belka*:

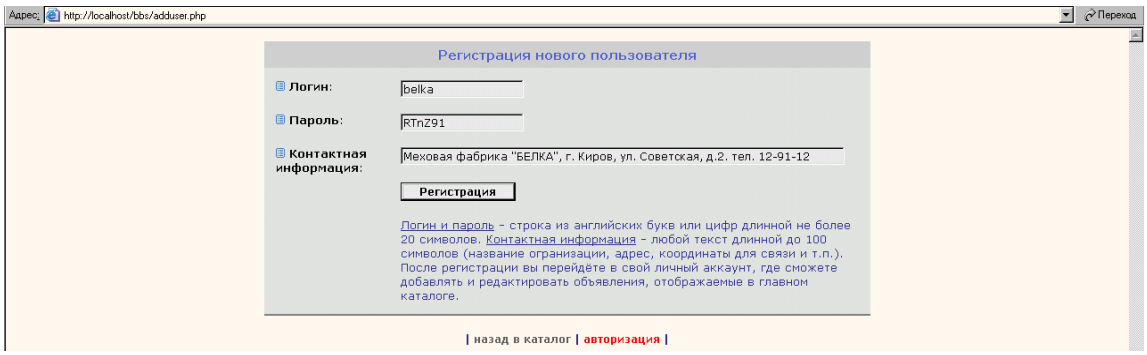

Рис. 8.2 - Регистрация нового пользователя *belka*

Рассмотрим процесс авторизации зарегистрированного пользователя *belka*:

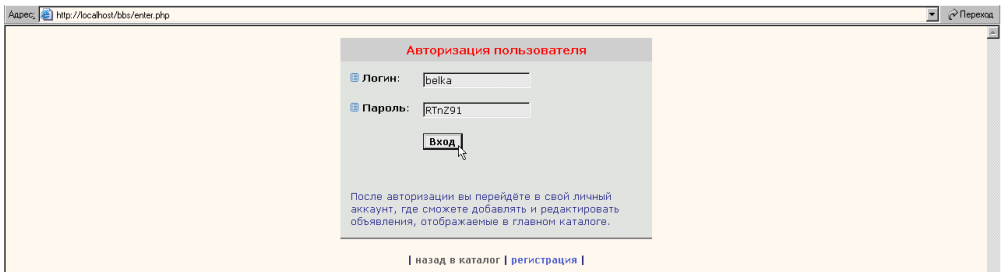

Рис. 8.3 - Авторизация пользователя

После успешной аутентификации пользователь продолжает работу в своем аккаунте:

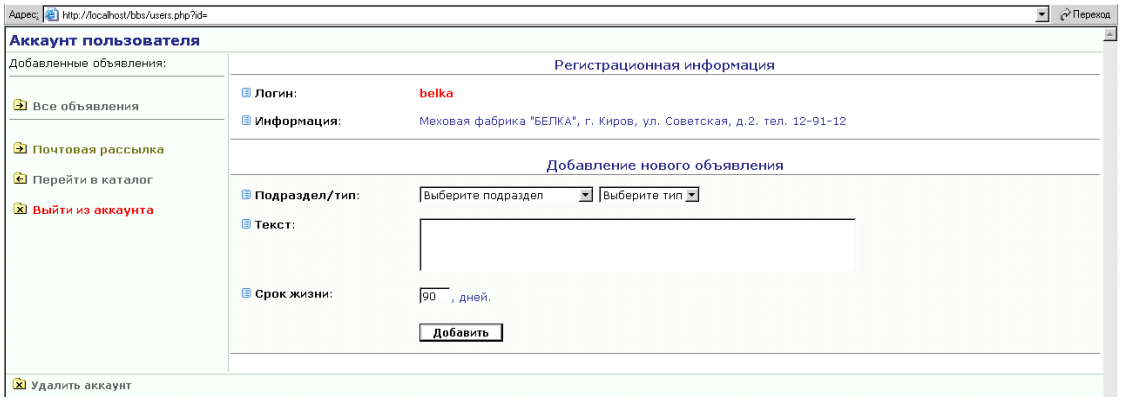

Рис. 8.4 - Аккаунт пользователя

Рассмотрим пример добавления нового объявления:

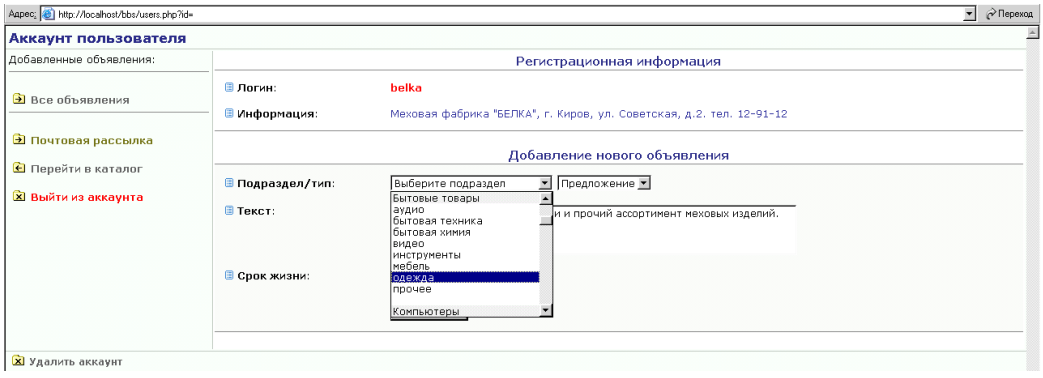

Рис. 8.5 - Добавление нового объявления: выбор подраздела каталога

8. Заключение

Таким образом, поставленная задача была выполнена. Мной было разработано интерактивное интернет-приложение, сочетающее в себе работу с базой данных MySQL, практическую реализацию механизма регистрации, авторизации пользователей. Также в работе был разработан и применен алгоритм аутентификации, который обеспечивает надежную защиту от взлома паролей пользователей для приложений подобного уровня.

Взаимодействие приложения с пользователем было выполнено в виде простого и интуитивно-понятного WEB-интерфейса.

9. Список литературы

1. Programming PHP. Расмус Лердорф, 2002 г.
2. MySQL по максимуму. Петр Зайцев, Берон Шварц, Вадим Ткаченко.
3. Wikipedia MD5 https://ru.wikipedia.org/wiki/MD5
4. PHP Manual.
5. M. Kabir. Secure PHP Development.