Министерство образования Республики Беларусь

Белорусский Государственный Университет

Факультет радиофизики и физической электроники

Кафедра кибернетики

Курсовая работа на тему:

«**Разработка алгоритмов и программных средств подсистемы документооборота системы управления содержанием информационного сервера»**

Выполнил:

студент Басалаев Н.В.

курс 3, семестр 6-ой.

Руководитель:

Стрикелев Д.А.

Минск

2004

**Содержание:**

[Введение 3](#_Toc73844944)

[Глава 1. Теоретические основы разработки информационного сервера 6](#_Toc73844945)

[1.1. Web-технологии как основа доставки информации в информационной системе 6](#_Toc73844946)

[1.2. Архитектура информационного сервера 15](#_Toc73844947)

[1.3. Принципы организации документооборота на информационном сервере 18](#_Toc73844948)

[1.4. Средства создания информационных серверов 24](#_Toc73844949)

[Глава 2. Подсистема организации документооборота «InfoBeacon» 29](#_Toc73844950)

[2.1. Архитектура и функциональность подсистемы 29](#_Toc73844951)

[2.2. Организация политики безопасности в рамках подсистемы 33](#_Toc73844952)

[2.3. Компоненты подсистемы и схема хранения данных 34](#_Toc73844953)

[Заключение 40](#_Toc73844954)

[Список литературы 41](#_Toc73844955)

[Приложение 1. Листинги SQL-запросов по созданию таблиц 42](#_Toc73844956)

[Приложение 2. Листинги основных PHP скриптов 44](#_Toc73844957)

# Введение

Последние десять лет ознаменовались фантастическим развитием Internet и новых способов общения между людьми. На переднем крае этого явления находится World Wide Web (WWW). Ежедневно в этой новой коммуникационной среде открываются тысячи новых сайтов, а потребителям предлагаются новые виды услуг. Часто можно услышать, что в Internet можно найти все. Остается только вопрос: «Где именно?». Создание сайтов, наполненных гигантскими объемами информации отвечает на этот вопрос. Так что появление такого определения как «информационный сервер» является вполне естественным.

Естественно, что информация на таких серверах должна быть достаточно упорядоченной, иначе пропадает вся информативная ценность. Но ведь до того, как информация станет упорядоченной и дойдет до конечного потребителя, она должна пройти довольно жесткий отбор, редактирование. С этой целью и разрабатываются системы управления содержанием ИС, с функциональными возможностями значительно различающимися в зависимости от области их применения.

В зависимости от ситуации, возможности ИС могут включать в себя операции по подготовке и обработке документов, обмена документами, автоматизированного формирования одних документов на основе других, их автоматизированной обработке. Зачастую ИС позволяет связать участников территориально-распределенной организационной структуры в единое поле деятельности и обеспечить согласованную, последовательную, непротиворечивую и оперативную работу. Почти всегда от ИС требуется предоставлять средства не только для технологического управления и администрирования системой, но и решения, позволяющие обеспечить необходимый уровень защиты конфиденциальных данных от несанкционированного доступа, нежелательного просмотра или удаления, а также от ошибок безответственных и неблагонадежных пользователей.

 Нередко встречаются проблемные ситуации, когда на CMS ИС налагаются дополнительные требования: в том случае, если процесс публикации данных либо долгий и многоэтапный (онлайн-издания, различные каталоги Internet адресов, онлайн-магазины и т.д.), либо требования к достоверности и корректности информации крайне высоки (корпоративные сайты известных компаний, марок, и т.п.), необходима многократная проверка и утверждение публикуемых данных разными участниками, управляющими документопотоками, что требует включения в состав CMS подсистемы документооборота. [2]

Основными задачами подобной подсистемы являются разделения процессов создания, редактирования и оформления документов между пользователями системы, что позволяет более эффективно управлять цифровой интеллектуальной собственностью организации.

Если еще одно десятилетие назад можно было задуматься над управлением бумажным документооборотом, то сейчас та же проблема стоит в электронной плоскости. Только уже на гораздо более остром уровне. Не зря же наш век называется «информационным».

Вообще, CMS сама по себе является системой документооборота, т.к. CMS (Content Management System) – система динамического обновление и редактирования содержания Интернет-сайта или любой другой информационной системы. Система, созданная с использованием CMS - это, прежде всего, гораздо более эффективный инструмент для бизнеса компании, чем статично сверстанный сайт. Информации становится больше, а управлять сайтом становится проще.

CMS снижают стоимость создания сайтов и их поддержки. Основными функциями систем являются разработка, доставка контента (наполнения, содержания) и управление сайтом. Несомненным плюсом системы управления содержанием является снижение стоимости администрирования вообще и поддержки сайта в частности. Это происходит, благодаря снижению потерь времени на поиски документов, пресечению дублирования и ошибок. Часто CMS создают для пользователей, которые мало знакомы с разработкой сайтов. Используя CMS они могут получить возможность создать и администрировать собственный сайт, не отличающийся по своим возможностям от сайтов, выполненных профессиональными разработчиками.

Системы управления содержимым исключительно полезны для Web-сайтов, на которых содержание поддерживается более чем одним автором, либо сопровождение осуществляет нетехнический персонал, либо содержимое и графическое оформление разрабатываются различными людьми и даже отделами.

Таким образом, разработка алгоритмов и программных средств подсистемы документооборота системы управления содержанием информационного сервера представляется крайне актуальной темой исследования и разработки.

# Глава 1. Теоретические основы разработки информационного сервера

# 1.1. Web-технологии как основа доставки информации в информационной системе

World Wide Web - система для доступа к гипертекстовой и гипермедиа-информации. Изначально проект WWW зародился в CERN, европейском центре физики высоких энергий в 1990, но со временем перерос рамки сообщества ученых-физиков. Первые программы, демонстрирующие работу системы, были закончены в 1992 году для компьютера NeXT. За несколько лет, прошедших с тех пор, система WWW совершила победоносное шествие практически по всем операционным платформам, включая самые примитивные (MS-DOS).

В связи с отсутствием возможности дать строгое определение World Wide Web следует обратить внимание на генетическую связь этой системы с информационно-поисковыми системами и глобальными сетями. По существу, Web представляет собой результат применения возможностей доступа к территориально распределенной информации для создания глобальных гипертекстовых и мультимедиа информационно-поисковых систем. Возможности доступа к территориально-распределенной информации обеспечивает для Web всемирная сеть Internet. Наследуя базовые черты информационно-поисковых систем, web-система в основном развивается как хранилище слабоструктурированной, разноплановой и часто несогласованной информации и тем отличается от баз данных, где информация структурирована и взаимосвязана.

Web представляет собой сеть узлов, содержащих гипермедиа-документы и связи, позволяющие из одного документа ссылаться на другие, размещенные как на том же узле, так и на других.

С самого момента своего рождения Web была определена в качестве технологии-посредника для связывания различных типов информационных ресурсов. При этом HTML-страницы играли роль цемента всей этой информационной конструкции. Это давало возможность быстро наращивать информационную емкость за счет конвертации информационных массивов в формат Web или их подключения серверам Web через программы-шлюзы.

Сама технология была построена по схеме «клиент-сервер», неориентированной на постоянное соединение.

Технология клиент-сервер является реализацией распределенной обработки данных. В системе архитектуры клиент-сервер обработка данных разделена между компьютером-клиентом и компьютером-сервером, связь между которыми происходит по сети. Это разделение процессов обработки данных основано на группировании функций. Как правило, компьютер-сервер баз данных выделяется для выполнения операций с базами данных, а компьютер-клиент выполняет прикладные программы.

Основная функция компьютера-клиента состоит в выполнении приложения (интерфейса с пользователем и логики представления) и осуществлении связи с сервером, когда этого требует приложение. Компьютер-клиент может быть как простой машиной типа персонального компьютера (ПК) с процессором 286 и операционной системой DOS, так и мощной рабочей станцией с многозадачной и многопользовательской операционной системой типа UNIX. Таким образом, выбор компьютера, операционной системы, оперативной и дисковой памяти, другого оборудования определяется требованиями приложения. В качестве программы-клиента обычно выступает браузер.

Как следует уже из самого термина, главная функция компьютера-сервера заключается в обслуживании потребностей клиента. Одно из важных требований к серверу - операционная система, в среде которой размещен сервер, должна быть многозадачной (и, желательно, но не обязательно, многопользовательской). Сервером, как правило, выступает программа-сервер протокола обмена гипертекстовой информацией HTTP, которая отвечает на запросы клиентов.

Преимущества технологии клиент-сервер:

* независимость от платформ: доступ к разнородным сетевым средам, в состав которых входят компьютеры разных типов с различными операционными системами;
* большее число пользователей;
* относительно низкие затраты на внедрение и эксплуатацию;
* высокая способность к интеграции существующих информационных ресурсов;
* повышение уровня эффективности использования оборудования;
* прикладные программные средства доступны с любого рабочего места, имеющего соответствующие права доступа;
* минимальный состав программно-технических средств на клиентском рабочем месте (теоретически необходима лишь программа просмотра - браузер и общесистемное программное обеспечение);
* минимальные затраты на настройку и сопровождение клиентских рабочих мест, что позволяет реализовывать системы с тысячами пользователей.

 Hypertext Transfer Protocol (HTTP) - это протокол, который клиенты и серверы WWW используют для общения между собой. Он, по сути дела, является основой Web.

Рассмотрим основные принципы работы HTTP .

Все HTTP-транзакции имеют один общий формат. Каждый запрос клиента и ответ сервера состоит из трех частей: строки запроса (ответа), раздела заголовка и тела.

Сервер отвечает на запрос клиента. Если запрос клиента успешен, то сервер посылает затребованные данные. Это может быть копия файла или документ сформированный "на лету". Если запрос клиента удовлетворить нельзя, то сервер передает дополнительные данные в виде удобного для человека разъяснения причин, по которым сервер не смог выполнить запрос.

В HTTP 1.0 после передачи сервером затребованных данных следует разъединение с клиентом и транзакция завершается. В HTTP 1.1 сервер по умолчанию не разрывает соединение, и клиент может посылать другие запросы. Это позволяет сэкономить время и затраты клиента, которому не приходится заново соединяться с тем же сервером. Таким образом, в HTTP 1.1 транзакция может циклически повторяться, пока клиент или сервер не закроет соединение явно.

С технологией HTTP неразрывно связано такое понятие как URL. Эта аббревиатура расшифровывается как Uniform Resource Locator, что можно вольно перевести, как "единый указатель на ресурс". Практически, это адрес документа.

Типичный для URL вид:

 *протокол://полное.имя.машины.или.адрес:порт/путь*

Здесь «*протокол*» принимает значения:

http - передача гипертекста; ftp - протокол передачи файлов; telnet - терминальный доступ; news - новости Usenet; file - для доступа к локальным файлам.

Параметр «*порт»* можно не указывать и тогда подразумевается порт, стандартный для данного протокола. Для ftp используются порты 20 и 21, для http - 80, для telnet - 23, для gopher - 70, news - 119 и т.д.

Параметр «*путь»* специфичен для каждого протокола, например, для ftp - это путь в файловой системе. Похожий смысл имеет этот параметр и для других протоколов.

Согласование типов передаваемых в рамках www документов производится с помощью заголовков, которыми обмениваются навигатор и WWW-сервер. Весь комплекс этих заголовков известен как MIME, Multipurpose Internet Mail Extensions. Навигатор (браузер) должен знать, какого типа документ он получает, ведь он должен его интерпретировать, показывать и вообще что-то с ним делать.

Браузер предоставляют пользователю возможность указывать внешние программы-интерпретаторы для разных типов документов.

Чудесной находкой, позволившей открыть множеству людей доступ к Интернет, была концепция гипертекста, предложенная Теодором Хольмом Нельсоном. Именно Нельсон считается отцом идеи гипертекста в том виде, в котором он сейчас существует.

Гипертекст - это обычный текст, содержащий ссылки как на собственные фрагменты, так и на другие тексты. Рассказывая о том, что послужило прообразом для этого изобретения, Нельсон вспоминает отрывок из одного очерка Ванневара Буша, написанного в 1945 году: «Работа человеческой мысли построена на принципе ассоциаций. Анализируя какое-либо понятие или элемент, она непременно стремится поставить ему в соответствие какой-нибудь другой знакомый образ, подсказываемый ассоциацией мыслей, и это соответствие устанавливается благодаря трудноуловимой паутине связей, формируемых клетками человеческого мозга». Спроецировав эту идею о работе мозга одного человека на компьютерную сеть, охватывающую весь мир, Нельсон посеял семена явления, которое впоследствии переросло во «Всемирную Паутину».

Идея гипертекста была простой, элегантной и великолепной. Но успех идеи определялся наличием сети. Если сеть есть, гипертекст невероятно полезен. Он тогда становится ключевым механизмом. Дело в том, что при наличии сети тексты, связанные друг с другом ссылками, можно размещать на различных, территориально удаленных компьютерах, и создавать и редактировать тексты могут разные люди. Таким образом, создается «паутина» взаимосвязанных текстов, способная стать гигантским информационным хранилищем.

В 1988 году проект гипертекстовой системы Xanadu Теодора Нельсона обрел источник финансирования у Джона Уокера, основателя Autodesk. Тогда Уокер пророчески заявил: «В 1964 году Xanadu была мечтой одиночки. В 1980 году - общей целью небольшой группы талантливых технологов. В 1989 году она станет продуктом. А в 1995 году она начнет переделывать мир». Все оказалось даже ближе к истине, чем Уокер мог вообразить.

В основе web-технологий лежит простая идея - HTML-страницы не обязаны быть статичными и храниться в готовом виде. Ничто не мешает формировать их динамически в ответ на запрос пользователя. Если для этого используется отдельное приложение, которое запускается www-сервером, это CGI (Common Gateway Interface). Создать CGI-приложение несложно. В то время как www-сервер занимается управлением правами доступа, обработкой поступающих запросов, передачей данных клиенту и пр., от программы CGI требуется всего лишь вывести HTML-страницу в стандартный поток вывода. При этом она может быть написана на C++, Perl, Php присоединяться к базам данных или другим ресурсам и выполняться очень быстро. Данные запроса передаются в CGI-приложения через переменные окружения или через стандартный ввод. В настоящее время генерация HTML с помощью CGI, будь то скомпилированная программа или интерпретируемый perl-скрипт, распространено чрезвычайно широко. Однако использование CGI имеет и недостатки. Например, при сильной загрузке www-сервера. В течение одной секунды он должен обслужить 100 запросов пользователей. Это означает одновременный запуск 100 CGI-приложений. С точки зрения операционной системы создание нового процесса трудоемкая процедура, как, впрочем, и поддержание его в работоспособном состоянии. Для запуска программы операционная система создает специальные структуры внутри ядра, выделяет память под сегменты задачи, загружает данные приложения с диска и связывает его с динамическими библиотеками. После завершения работы приложения необходимо освобождать все занятые им ресурсы. Нельзя забывать и про время инициализации приложения. В случае, когда идет работа с базой данных, время инициализации - это время установления соединения с сервером БД, и это соединение не всегда выполняется быстро (требуется установить канал связи, проверить права доступа и пр.) В ситуации, когда сервер БД загружен, это время будет еще больше. Технология CGI проста и удобна, но ее следует использовать в том случае, когда время отклика не критично (генерация отчетов и пр.) и когда запросы для CGI-приложений поступают не очень часто (раз в 10-60 секунд). Что же делать, если необходим динамический HTML, но ресурсы на CGI тратить не хочется?

Выход был найден и здесь - чтобы обойтись без запуска отдельного приложения, его нужно встроить в www-сервер. Поскольку заранее неизвестно, что именно будет делать это приложение, следует встроить в www-сервер интерпретатор удобного для обработки данных языка. Такое решение позволит каждый запрос обрабатывать отдельным потоком сервера, а не отдельным процессом операционной системы. Это увеличивает время отклика и снижает нагрузку на процессор. Программы, которые обрабатывает этот интерпретатор, часто называют «скриптами» (Scripts). В решении внедрить интерпретатор в www-сервер таятся как плюсы, так и минусы - интерпретация скриптов позволяет вносить в них изменения и немедленно видеть результат, но выполняются они медленнее скомпилированной программы. К тому же ошибка в CGI-приложении никак не повлияет на устойчивость www-сервера, а вот ошибка во встроенном интерпретаторе, скорее всего, будет для сервера фатальна. Как показала практика, плюсы все же перевесили. Основной задачей, возлагаемой на скрипты, стало взаимодействие с БД, а здесь основные задержки возникают, как правило, на сервере БД. К тому же отсутствие необходимости в компиляции необычайно удобно при отладке интернет-приложений. Это важно, так как в стадии отладки эти приложения пребывают вплоть до того момента, когда их заменяют другими.

В стане UNIX изначально получил широкое распространение интерпретатор языка PERL (Practical Expression Regular Language). В названии таится главный смысл – «обработка регулярных выражений», то есть специальных выражений, которые обрабатывают строки по шаблонам. Perl позволяет лаконично описывать и решать сложные задачи хранения и синтаксического разбора строк произвольной длины. Для интернет-задач, которые, в основном, оперируют текстом, это вполне удобный инструмент. Интерпретатор Perl был встроен в основной www-сервер операционной системы Linux Apache. Однако, Perl придуман был изначально не для Internet, и отсюда вытекали все его недостатки, например, неудобства при организации доступа к базам данных. Да и далеко не прост PERL в обучении! Более разумной альтернативой является PHP (Personal Home Page). PHP был придуман в 1994 году для расширения возможностей «домашней» страницы. Вначале умел он не очень много - понимал простейший язык и всего несколько макросов. Позднее PHP получил развитие и в настоящее время это интерпретатор мощного C-подобного языка, который встраивается в www-сервер Apache. Огромная часть интернет-приложений под UNIX- семейством создается на PHP. Существует он и под Windows, но особой популярностью здесь не пользуется. Для создания работоспособной версии веб-сервера со встроенным PHP придется немного потрудиться. Вначале необходимо получить исходный текст www-сервера и PHP-интерпретатора. Затем установить библиотеки для работы с СУБД, которая Вам необходима. Наконец, скомпилировать, связать (слинковать, как-то некрасиво, хотя более точно) это друг с другом и заняться редактированием настроечных файлов.

 К недостаткам PHP (версии 3) сам автор относит снижение производительности на больших проектах, отсутствие поддержки сессий, малое кол-во подключаемых модулей (хотя это вопрос времени). В настоящее время уже существует PHP 4, которая решила большинство этих проблем (например, отсутствие сессий) и буквально в этом году должен появиться PHP5.

Именно указанные технологии на основе общих принципов построения сети Internet и, в особенности, на базе системы протоколов TCP/IP сделали возможным функционирование WWW. Следует обратить внимание на тот факт, что, общаясь с web, пользователь в каждый конкретный момент времени устанавливает связь только с одним web-узлом. Таким образом, взаимодействие пользователя с web всегда укладывается в схему клиент-сервер, несмотря на то, что серверы, т.е. web-узлы, могут сменяться даже во время одного сеанса, а управляет этой сменой узлов пользователь (клиент) с помощью активации ссылок в изображении просматриваемого документа.

Рассмотрим процесс миграции информационной системы из традиционной технологической схемы локального окружения в Web-технологию.

Традиционная схема представляет из себя:

1) интерфейс пользователя

2) ядро системы

3) информационный массив

4) интерфейс администратора

5) утилиты администратора

С точки зрения Web-технологии интерфейс пользователя - это браузер, который взаимодействует с ядром через http-сервер. Таким образом происходит первый этап декомпозиции традиционной информационной системы в Web.

Второй шаг - это возможность использования браузера в качестве интерфейса администратора. Здесь возникают вопросы разграничения доступа и актуализации информации в базах данных системы.

Следующий шаг - распределение нагрузки по нескольким серверам, а также использование кэширования на серверах-посредниках.

Пока декомпозиции подвергалась связка "конечный пользователь-ядро". Можно провести декомпозицию и на стороне сервера. Первым таким шагом является применение CGI при доступе к ресурсам. Сервер становится посредником между браузером и сервером ресурса.

Основные проблемы Web-технологии - это вопросы отсутствия реального сеанса работы с сервером и безопасность.

Для поддержки сеанса в Web применяется спецификация Cookie. Идея состоит в том, чтобы передавать от клиента на сервер и обратно информацию о пользователе и его действиях, которая привязывается по типу информационного ресурса и времени.

# 1.2. Архитектура информационного сервера

Первоначально системы такого уровня базировались на классической двухуровневой клиент-серверной архитектуре (рис. 1).

Транзакция

Запрос

Система управления базами данных

Сервер базы данных

Программа-браузер

*Рис. 1. Двухуровневая клиент-серверная архитектура*

Данная клиент-серверная архитектура характеризуется наличием двух взаимодействующих самостоятельных модулей – программы-браузера и сервера базы данных, в качестве которого может выступать Microsoft SQL Server, Oracle, Sybase, MySQL и другие. Сервер БД отвечает за хранение, управление и целостность данных, а также обеспечивает возможность одновременного доступа нескольких пользователей. Клиентская часть представлена так называемым “толстым” клиентом, то есть приложением, на котором сконцентрированы основные правила работы системы и расположен пользовательский интерфейс программы. При всей простоте построения такой архитектуры, она обладает множеством недостатков, наиболее существенные из которых - это высокие требования к сетевым ресурсам и пропускной способности сети. Кроме того, при большом количестве «клиентов» возрастают требования к аппаратному обеспечению сервера БД, а это, как известно, самый дорогостоящий узел в любой информационной системе.

Как видим, минусов у такой архитектуры достаточно, а решение тривиально - нужно отделить логику от клиентской части и СУБД, выделив ее в отдельный слой. Так и поступили разработчики и следующим шагом развития клиент-серверной архитектуры стало внедрение среднего уровня, реализующего задачи управления механизмами доступа к БД (рис. 2).

Программа-браузер

Сервер приложений

(СУБД)

Сервер базы данных

Сервер базы данных

*Рис. 2. Трехуровневая клиент-серверная архитектура*

Плюсы данной архитектуры очевидны. Благодаря концентрации логики на сервере приложений, стало возможно подключать различные БД. Теперь, сервер базы данных освобожден от задач распараллеливания работы между различными пользователями, что существенно снижает его аппаратные требования. Также снизились требования к клиентским машинам за счет выполнения ресурсоемких операций сервером приложений и решающих теперь только задачи визуализации данных. Именно поэтому такую схему построения информационных систем часто называют архитектурой “тонкого” клиента.

Но, тем не менее, узким местом, как и в двухуровневой клиент-серверной архитектуре, остаются повышенные требования к пропускной способности сети, что в свою очередь накладывает жесткие ограничения на использование таких систем в сетях с неустойчивой связью и малой пропускной способностью (Internet, GPRS, мобильная связь).

Существует еще один важный момент использования систем, построенных на такой архитектуре. Уровень «клиентов», в целом обладающий огромной вычислительной мощностью, на самом деле простаивает, занимаясь лишь выводом информации на экран пользователя. Так почему бы не использовать этот потенциал в работе всей системы?

Более 95 % данных, используемых в документообороте, могут быть размещены на одном персональном компьютере, обеспечив возможность его независимой работы. Поток исправлений и дополнений, создаваемый на этом компьютере, ничтожен по сравнению с объемом данных, используемых при этом. Поэтому если хранить непрерывно используемые данные на самих компьютерах, и организовать обмен между ними исправлениями и дополнениями к хранящимся данным, то суммарный передаваемый трафик резко снизиться. Это позволяет понизить требования к каналам связи между компьютерами и чаще использовать асинхронную связь, и благодаря этому создавать надежно функционирующие распределенные информационные системы, использующие для связи отдельных элементов неустойчивую связь типа Интернета, мобильную связь, коммерческие спутниковые каналы. А минимизация трафика между элементами сделает вполне доступной стоимость эксплуатации такой связи. Конечно, реализация такой системы не элементарна, и требует решения ряда проблем, одна из которых своевременная синхронизация данных.

Построенные на основе данной архитектуры системы будут обладать надежностью, безопасностью информации и высокой скоростью вычислений, что от них в первую очередь и требуется.

# 1.3. Принципы организации документооборота на информационном сервере

Документооборот - это регламентированная технологическая схема и процесс движения документов по установленным пунктам обработки для выполнения необходимых операций с ними.

Документооборот можно трактовать и как статическую структуру пунктов прохождения, и как динамический процесс прохождения до­кументов одновременно. Число пунктов обработки документов, ско­рость перемещения документов между ними и оперативность выпол­нения в каждом пункте необходимого набора операций определяют степень совершенства технологической линии обработки документов и эффективность обеспечения аппарата управления полезной и досто­верной информацией.

Эффективная работа с документами невозможна без систематизации и классификации. Для систематизации документов используется номенклатура дел - перечень наименований дел, оформленный в установленном порядке. Номенклатура дел является классификационным справочником и используется для построения информационно-поисковой системы. Дело представляет собой первичный комплекс документов, сгруппированных по определенному признаку.

Наиболее крупной единицей систематизированной информации является информационный ресурс - организованная совокупность документированной информации, включающая базы данных и знаний, другие массивы информации в информационных системах. В настоя­щее время, с увеличением объема документированной информации существенно возрастает роль автоматизированных информационно-поисковых сис­тем. Они становятся неотъемлемой частью современных систем доку­ментооборота. Перенесенные на электронные носители информаци­онные ресурсы с помощью средств вычислительной техники и телекоммуникаций приобретают качественно новое состояние и становят­ся доступными для оперативного воспроизводства необходимой ин­формации и превращаются в важнейший фактор эффективного функ­ционирования организации.

Не всегда различают понятия электронный документ и электронное сообщение. Иногда термин электронный документ понимается как файл данных, сформированных, например, текстовым процессором или электронной таблицей, т.е. электронное сообщение. Это допустимо, если не вносится путаница и подмена понятий. Между тем к передаче и обработке электронных сообщений и электронных документов предъявляются разные требования. Электронное сообщение может свободно редактироваться и в измененном виде передаваться далее для после дующей обработки. Электронный документ, как и документ бумажный, изменяться не может, если он наделен юридической силой. Это требование определяет различие между компьютерными системами обработки электронных сообщений и электронных документов.

Жизненный цикл документа в системе документооборота включает следующие основные стадии работы с документом: создание; исполнение; хранение; уничтожение документа

На различных стадиях пользователи системы выполняют над документом множество действий. Совокупность действий, которые осуществляет пользователь на всех стадиях жизненного цикла документа, определяет его роль и функции в документообороте. В ряде случаев понятия роли и функции участника документооборота совпадают.

Пользователь системы как участник документооборота играет в определенный момент времени одну из заранее определенных ролей. Он может иметь одну роль или несколько в зависимости от характера работы над текущим документом и установленного регламента. Например, сотрудник управления выступает в роли автора одних документов, ответственного исполнителя - других, рецензента - третьих, и так далее.

Каждый конкретный документопоток предполагает соответствующее распределение ролей. Практически всегда в соответствие с ролью ставится определенный набор действий и процедур, содержание которых зависит от технологии документооборота. Состав действий для одной и той же роли может изменяться в зависимости от типа обрабатываемого документа.

Для каждого действия определяется система элементарных технологических операций, составляющих содержание действия. Этот набор операций может изменяться в зависимости от параметров и характеристик технологической линии обработки документов.

Автоматизированная система документооборота должна поддерживать на качественно новом уровне следующие процессы:

* создание и генерацию информации;
* сбор, накопление, обработку, хранение и передачу информации;
* поиск, распространение и использование информации.

Такая система является компонентом интегрированной информационной среды как совокупности информационных ресурсов, телекоммуникационной среды и в целом информационной инфраструктуры. Заметим, что мы не будем разграничивать понятия автоматизированной системы обработки документов и автоматизированной системы управления документооборотом, поскольку в современных информационных средах их функции часто объединены.

Типовая компьютерная среда включает следующие основные компоненты:

* средства разработки сообщений;
* корпоративную информационную систему;
* систему документооборота и обработки сообщений.

Проанализируем состав и назначение каждого компонента.

**Средства разработки сообщений**

К средствам разработки сообщений относятся разнообразные инструментальные системы, при помощи которых создаются электронные сообщения. Типичными средствами этой категории являются текстовые редакторы и электронные таблицы. Все создаваемые этими средствами файлы имеют статус электронных сообщений.

**Информационная система**

Информационная система включает средства накопления и обработки больших массивов однотипной информации, базы данных и приложений на их основе.

**Система документооборота и обработки сообщений**

 В системе документооборота и обработки сообщений, как было уже указано, следует разделять электронный документ и электронное сообщение. В соответствии с этим разделяют системы электронного документооборота и обмена электронными сообщениями (по составу операций они могут быть близки). Поступающие файлы электронных сообщений должны преобразовываться в формат электронных документов. Последние формируются непосредственно в среде системы документооборота при помощи встроенных специализированных средств.

Между компонентами системы выделяют следующие интерфейсы:

1. Интерфейс между средствами разработки сообщений и информационной системой – определяет способы получения средствами разработки сообщений информации из баз данных информационной системы.
2. Интерфейс между средствами разработки сообщений и системой документооборота и обработки сообщений. Определяет способы взаимодействия средств разработки сообщений, с одной стороны, и системы документооборота - с другой. Передаются проекты документов и копии документов, имеющие статус электронных сообщений, а также сопровождающие их комментарии, замечания и т.д.
3. Интерфейс между информационной системой и системой документооборота. Сообщения формируются в информационной системе при помощи генераторов отчетов и передаются в систему документооборота. Этот интерфейс имеет смысл и в том случае, если рассматривать систему документооборота как составную часть корпоративной информационной системы.

Рассмотрим некоторые основные принципы и правила автоматизированного документооборота.

Система автоматизированного документооборота должна прежде всего осуществлять эффективную обработку и перемещение электронных документов, создавая тем самым физическую среду для информации.

При разработке или выборе технологии обработки документов и управления документопотоками необходимо учитывать ограничения технического характера, связанные:

* с соблюдением стандартов;
* с защитой электронного документа и его реквизитов;
* с обработкой информации в графическом виде.

Международными организациями по стандартизации (ISO, CCIT, ЕСМА) проводится большая работа по унификации электронных документов. В работах по созданию этих стандартов решаются не только технические проблемы согласования различных структурных характеристик документов. Главное заключается в том, что разрабатывается общая модель документа, которая является основой выработки нескольких взаимоувязанных стандартов.

Проблема защиты электронного документа и его реквизитов определяется требованием неизменяемости документа после придания ему юридической силы. Содержимое файла документа и его реквизиты могут быть защищены от несанкционированного изменения путем разработки специальной структуры файла и включения в его состав средств электронной подписи. Такой способ защиты должен быть надлежащим образом сертифицирован, законодательно регламентирован и признан всеми участниками документооборота.

Проблема обработки информации в графическом виде возникает в отношении сканированных образов документов, поскольку выходная информация сканера создается в виде графического файла. Текст в графическом файле не может быть обработан типовыми средствами такими, как текстовые процессоры или СУБД. Кроме того, графические файлы имеют большие размеры. В настоящее время существуют компьютерные системы, позволяющие обрабатывать большие объемы графической информации за приемлемое время, однако из-за высоких требований к их характеристикам стоимость таких систем достаточно высока.

С позиций повышения эффективности и снижения ресурсоемкости сегодня наибольший интерес представляет решение оптимизационных задач, которые существенно сдерживают процесс перехода управления документооборотом на качественно новый уровень.

Оптимизационный характер ряда задач построения систем документооборота вытекает из основных технологических требований к документообороту. [3]

*Таблица 1*

**Требования, предъявляемые к документообороту**

|  |
| --- |
| Прохождение /документов кратчайшим путем |
| Избирательность распределения документов между руководителями и специалистами |
| Обусловленность перемещения документов деловой необходимостью |
| Единообразие маршрута движения и состава технологических операций для массовых категорий документов |
| Однократность выполнения каждой операции |

#  1.4. Средства создания информационных серверов

PHP — это серверный (или серверной стороны) язык сценариев, разработанный специально для Web. В HTML-страницу можно внедрить PHP-код, который будет вы­полняться при каждом ее посещении. PHP-код интерпретируется Web-сервером и генерирует HTML-код или другой вывод, наблюдаемый посетителями страницы.

Разработка РНР была начата в 1994 году и вначале осуществлялась одним чело­веком, Расмусом Лердорфом (Rasmus Lerdorf). Этот язык был принят рядом та­лантливых людей и претерпел три основных редакции, пока не стал широко ис­пользуемым и зрелым продуктом, с которым мы имеем дело в настоящее время. По состоянию на октябрь 2002 года он использовался в более чем девяти миллионах доменов разбросанных по всему миру, причем их число неуклонно растет. Текущее юличество доменов, в которых используется РНР, можно посмотреть по адресу *http://www.php.net/usage.php*.

PHP - это продукт с открытым исходным кодом, то есть вы имеете доступ к исходному коду. Его можно использовать, изменять и свободно распространять другим пользователям или организациям.

Первоначально РНР являлось сокращением от «Personal Home Page» («Персональная домашняя страница»), но затем это название было изменено в соответствии с соглашением по рекурсивному именованию GNU (GNU = Gnu's Not Unix) и теперь означает «РНР Hypertext Preprocessor» («Гипертекстовый препроцессор РНР»).

В настоящее время основной версией РНР является четвертая. Адрес домашней страницы РНР выглядит как *http //www.php.net*.

MySQL - очень быстрая, надежная система управления реляционными базами данных (СУРБД). Вообще говоря, база данных позволяет эффективно хранить, искать, сортировать и получать данные. Сервер MySQL управляет доступом к данным, позволяя работать с ними одновременно нескольким пользователям, обеспечивает быстрый доступ к данным и гарантирует предоставление доступа только пользователям, имеющим на это право. Следовательно, MySQL является многопользовательским, многопоточным сервером. Он применяет SQL (Structured Query Language - язык структурированных запросов), используемый по всему миру стандартный язык запросов в базы данных. MySQL появился на рынке в 1996 году, но его разработка началась еще в 1979 году.

В настоящее время пакет MySQL доступен в соответствии с лицензией Open Source, но в случае необходимости можно получить и коммерческие лицензии.

Приступая к созданию информационного сайта, можно использовать множество различных продуктов.

Возникает необходимость в выборе следующих компонентов:

* оборудование для Web-сервера;
* операционная система;
* программное обеспечение Web-сервера;
* система управления базами данных;
* язык программирования или создания сценариев.

Выбор некоторых из этих компонентов будет зависеть от уже произведенных выборов. Например, не все операционные системы могут работать на любом оборудовании, не все языки создания сценариев могут обеспечить подключение ко всем базам данных и т. д.

Одно из замечательных свойств РНР состоит в том, что он доступен для операционной системы Microsoft Windows, для многих версий UNIX и выполняется на любых полнофункциональных Web-серверах. Система MySQL обладает такой же степенью гибкости.

В число конкурентов РНР входят Perl, Active Server Pages (ASP) от Microsoft, Java Server Pages (JSP) и Allaire ColdFusion.

PHP обладает множеством преимуществ по сравнению с этими продуктами которых наиболее значительными являются:

* высокая производительность;
* наличие интерфейсов ко многим различным системам баз данных;
* встроенные библиотеки для выполнения многих общих задач, связанных с Web;
* низкая стоимость;
* простота изучения и использования;
* переносимость;
* доступность исходного кода.

Система РНР исключительно эффективна. Используя единственный недорогой сервер, можно обслуживать миллионы обращений в день. Результаты тестирования, опубликованные компанией Zend Technologies на своем официальном сайте *http://www.zend.com*, подтверждают более высокую производительность РНР по сравнению с конкурирующими продуктами.

РНР обладает встроенной возможностью подключения ко многим системам управления базами данных. В дополнение к MySQL, помимо прочих, можно непосредственно подключаться к базам данных PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase и Sybase.

Используя стандарт открытого интерфейса связи с базами данных (Open Database Connectivity Standard— ODBC), можно подключаться к любой базе данных, для которой существует ODBC-драйвер. Это распространяется на продукты Microsoft и множества других компаний.

Поскольку РНР был разработан для использования в Web, он имеет множество встроенных функций для выполнения большого разнообразия полезных задач, связанных с Web. С его помощью можно на лету генерировать GIF-изображения, подключаться к другим сетевым службам, отправлять сообщения электронной почты, работать с cookie-наборами и генерировать PDF-документы — и все это при помощи всего нескольких строк кода.

Пакет РНР является бесплатным. Наиболее новую версию можно в любой момент, причем совершенно бесплатно, выгрузить из сайта по адресу *http://www.php.net*.

Синтаксис РНР основан на других языках программирования, в первую очередь С и Perl.

Пакет PHP можно использовать под управлением множества различных операционных систем. PHP-код можно разрабатывать в среде таких бесплатных Unix-подобных операционных систем, как Linux и FreeBSD, коммерческих версий Unix типа Solaris и IRIX или различных версий Microsoft Windows.

Как правило, программы будут работать без каких-либо изменений в различных средах, в которых установлен пакет РНР.

Пользователь имеет доступ к исходному коду РНР. В отличие от коммерческих закрытых программных продуктов, если нужно что-либо изменить или добавить в этом языке, то это всегда можно сделать.

Не следует дожидаться, пока компания-изготовитель выпустит пакет исправлений. Нет необходимости беспокоиться о том, что изготовитель покинет рынок или перестанет поддерживать продукт.

К конкурентам MySQL, помимо прочих, относятся PostgreSQL, Microsoft SQL Server и Oracle.

MySQL обладает многими преимуществами, в том числе высокой производительностью, низкой стоимостью, простотой конфигурирования и изучения, переносимостью и доступностью исходного кода.

MySQL, вне всяких сомнений, работает исключительно быстро. Результаты сравнительных тестов производительности, выполненных компанией-изготовителем, можно посмотреть на странице по адресу *http://web.raysql.com/benchmark.html*. Многие из этих сравнительных тестов показывают, что MySQL работает на порядок быстрее конкурирующих продуктов.

Пакет MySQL доступен бесплатно в соответствии с лицензией на программное обеспечение с открытым исходным кодом или, если это необходимо для приложения, за небольшую сумму можно приобрести коммерческую лицензию.

В большинстве современных баз данных используется язык SQL. Если ранее вы работали с другими СУРБД, переход к этой системе не должен вызывать какие-либо затруднения. Установка MySQL столь же проста, как и установка многих аналогичных продуктов.

MySQL может использоваться в среде многих UNIX-подобных систем, а также в среде Microsoft Windows.

Как и в случае РНР, исходный код MySQL можно свободно выгружать и изменять. [1]

# Глава 2. Подсистема организации документооборота «InfoBeacon»

# 2.1. Архитектура и функциональность подсистемы

Давайте предположим, что в группу Web-разработчиков онлайн-издания некоторой компании входят хороший администратор-оформитель, редакторы тематических разделов и авторы статей. Сайт содержит регулярно обновляемые страницы новостей со всего мира, новостей и обзоров из сферы высоких технологий, спортивной и музыкальной рубрик. Главная страница отображает заголовки новейших статей по каждой из четырех категорий.

В этом онлайн-издании администратор-оформитель обеспечивает привлекательный вид содержимого, а также контролирует содержание материалов и безопасность системы в целом. С другой, стороны авторы пишут замечательные статьи, но не умеют толком оформлять свои работы для представления в Web.

Задача состоит в том, чтобы позволить каждому сконцентрироваться на своей работе и объединить результаты усилий, чтобы получилась оперативная служба новостей.

Основные требования, которые будут предъявляться к этой системе:

* увеличение продуктивности работы, позволив авторам сконцентрироваться на статьях, а дизайнеру - на оформлении;
* позволить редактору просматривать статьи и выбирать те, которые будут публиковаться;
* создать единообразный внешний вид сайта с использованием шаблонов страниц;
* предоставить авторам доступ только к предназначенным для них областям сайта;
* предотвратить изменение актуального содержимого.

Первым делом, необходимо продумать способ ввода содержимого в систему. Для этого существует три возможности.

1. FTP

Авторам и дизайнерам можно предоставить FTP-доступ к областям Web-сервера. Это позволит им загружать на сервер файлы со своих локальных компьютеров. Для загружаемых файлов потребуется выработать строгий стандарт именования (который позволит четко идентифицировать принадлежность изображений к тем или иным статьям). С другой стороны, можно воспользоваться основанной на Web системой, которая будет решать упомянутые задачи отдельно от загрузки файлов через FTP.

Использование FTP порождает проблему, связанную с выдачей полномочий. Необходимая степень гибкости не дает возможность применять FTP-протокол для предоставления пользователям функциональности по передаче файлов.

2. Метод загрузки файлов

HTTP-протокол предоставляет метод загрузки файлов при помощи Web-браузера. Язык РНР позволяет решать эту задачу очень просто и эффективно.

Кроме того, метод загрузки файлов дает возможность хранить текст в базе данных вместо файлов. Для этого выполняется чтение во временный файл и сохранение его содержимого в базе данных, а не копирование в другую область файловой системы.

3. Интерактивное редактирование

Пользователи должны иметь возможность, создавать и редактировать документы без задействования FTP-протокола либо другого метода загрузки файлов. Вместо этого авторам, например, можно предоставить большое текстовое поле, в котором они смогут редактировать содержимое своих статей.

Несмотря на относительную простоту этого метода, он зачастую оказывается весьма эффективным. Web-браузер не предоставляет каких-либо возможностей по редактированию текста, кроме лишь функций копирования и вставки, за реализацию которых отвечает операционная система. Однако, когда требуется внести лишь небольшие изменения, скажем, исправить орфографическую ошибку, при помощи подобных функций подобное осуществляется достаточно быстро. [1]

Как и при методе загрузки файлов, данные формы можно записать в файл либо сохранить в базе данных.

В нашем проекте реализуется как раз последний метод ввода содержимого в систему.

Итак, после того, как статья введена, ее можно отправить на редактирование. Для этого выбирается тематика статьи (в нашей системе это новости, компьютеры, спорт и музыка) и после этого материал отсылается соответствующему редактору. При входе редактора в систему у него появляется список статей, предназначенных для редактирования. Он может начать редактирование, написать сообщение автору статьи либо написать свою статью. При написании статью в чужой раздел у него будут права обычного автора на эту статью. Кстати, статью может редактировать и автор, пока редактор не нажмет ссылку «Готово». После нажатия этой ссылки статья становится недоступной ни для редактора, ни для автора. Она переходит в распоряжение администратора. Он может ее редактировать, удалить либо, что наиболее вероятно, опубликовать ее на сайте.

Из рис.3 видно, что на протяжении своего маршрута документ попадает к трем участника процесса документооборота. То есть весь процесс обработки документа можно условно разделить на три этапа: создание, редактирование и публикация. Однако это разделение, как и названия этапов, достаточно условны потому, что границы между этими этапами не такие конкретные, как может показаться с первого взгляда. Например, администратор имеет доступ к документу еще на этапе редактирования. Он может следить за ходом редактирования, а также имеет право сам править или даже удалить документ. На схеме это никак не указано, так как в общем-то это дополнительная функция системы и принципиально ничего не изменяет в архитектуре системы. Она только дает больше прав администратору.

1

Автор статьи

(создание)

3

Администратор сайта

(публикация)

2

Редактор раздела

(редактирование)

*Рис. 3. Маршруты документов в системе*

На рис. 4 показано, как эта подсистема маршрутизации документа вписывается в общую логическую структуру информационного сервера.

Редактирование документа

Взаимная переадресация

 Обмен сообщениями

Схемы отображения документов

Подсистема типизации документов

Подсистема маршрутизации документов

Query

(Запрос)

Query

Пользователь

Web-сервер

Сервер приложений

Transaction

(Транзакция)

Сервер БД

*Рис. 4. Описание логической структуры информационного сервера*

# 2.2. Организация политики безопасности в рамках подсистемы

При успешной аутентификации пользователя (сравнение данных, введенных пользователем для входа, с данными, введенными при регистрации, которые хранятся в базе данных), выводится список собственных статей, их состояние на данный момент, новые статьи для редактирования (если вошедший пользователь является редактором) и личные сообщения. Если пользователь еще не вводил данные для входа в систему, то отображается только форма входной регистрации.

После входа автора в систему переменной сеанса auth\_user присваивается значение. Информация, введенная в форме входной регистрации, передается в сценарий login.php, который сравнивает имя пользователя и пароль с соответствующим значениями базы данных. В случае успешного входа пользователь перемещается настраницу, на которой он пребывал ранее, с помощью значения глобальной переменной HTTP\_REFERER. Это означает, что сценарий, входа в систему может вызываться из любой страницы приложения.

Затем автор приветствуется с использованием его имени и ему предоставляете возможность выхода из системы. Эта ссылка всегда отображается в верхней част страницы stories.php, что позволяет легко выйти из системы в любой момент. Сценарий logout. php просто сбрасывает значение переменной auth\_user. [1]

При попытке передачи сценарию редактирования идентификатора статьи для изменения, этот же сценарий проверяет наличие прав доступа к этой статье, формируя запрос к базе данных с использованием имени пользователя вошедшего автора или редактора. Это позволяет надежно разграничить права доступа как между авторами, так и между редакторами.

# 2.3. Компоненты подсистемы и схема хранения данных

В качестве статьи будем рассматривать документ состоящий из заголовка, текста и изображения. Такого рода документы вполне можно считать структурированными.

Чем выше степень структуризации документа, тем проще его разбить на составляющие, которые будут храниться в базе данных. Преимущество такого подхода состоит в возможности единообразного структурированного представления всех документов.

В качестве примера возьмем статью новостей. Заголовок будет храниться в своем поле отдельно от текста. Изображение, по самой своей природе, является отдельным компонентом документа.

Поскольку заголовок является отдельным элементом, для его отображения можно задать стандартный шрифт и стиль, а также легко отделить заголовок от остальной части статьи, сформировав главную страницу заголовков.

Другой подход применительно к крупным документам предполагает организацию отдельных абзацев в соответствие с отношением "один ко многим". Другими словами, каждый абзац будет храниться в отдельной строке базы данных и иметь связь с идентификатором главного документа. Такой вид динамической структуры документа по­зволит формировать страницу содержания для каждого документа и отображать каждый раздел независимо либо же отображать документ целиком.

На начальном этапе необходимо принять чрезвычайно важное решение относительно метода хранения содержимого после его загрузки в систему. Поскольку вместе с текстом хранятся и метаданные, благоразумно поместить текстовую часть содержимого в базу данных. Несмотря на то что система MySQL способна хранить мультимедиа-данные все же лучше держать загружаемые изображения в файловой системе. Использование больших двоичных объектов (BLOB) в базе данных MySQL может привести к снижию производительности приложения.

В базе данных будут храниться лишь имена файлов изображений. Дескриптор <IMG SRC> может напрямую ссылаться на каталог графических файлов стандартным образом.

Когда объем данных велик, очень важно оптимизировать их хранение. Подобно тому, как эффективность базы данных зависит от правильно выбранной индексации, файловая тема существенно выигрывает от хорошо продуманной системы каталогов. [1]

В нашем проекте сайта новостей страницы отображаются в простом, тем не менее, структурированном формате. Каждая страница содержит набор статей, сформированных одним и тем же образом. Прежде всего, заголовок выводится крупным шрифтом, слева внизу отображается фотография, а справа – собственно заголовок статьи. Страница целиком содержится в стандартном шаблоне страниц, что является предпосылкой непротиворечивого оформления сайта.

Подобный вид компоновки исключительно популярен. Какая бы ни отображалась страница, сначала выводите заголовок, затем текст и, наконец, нижний колонтитул.

Реализация сайта с шаблонами заголовка и нижнего колонтитула позволяет легко и просто изменять оформление сайта, внося требуемые модификации только в фай­лы шаблонов.

Возможно, авторы будут дополнять статьи фотографиями, которые они сделали самостоятельно. Нам необходимо достигнуть единообразия оформления, но что произойдет, когда какой-то автор загрузит крупное изображение высокого качества, а другой автор - небольшую картинку качества пиктограммы?

Базируясь на предположении, что изображения, в основном, представляют собой фотографии, можно ограничиться лишь форматом JPEG и для манипулирования графикой воспользоваться соответствующими PHP-функциями.

В нашем проекте существует довольно простой сценарий resize\_image.php, который на лету изменяет размер изображения, в результате чего оно может выводиться с использованием дескриптора <IMG SRC>. Этот

сценарий принимает три параметра, в число которых входит имя файла изображения, максимальная ширина и высота в точках. Не стоит полагать, что если указан максимальный размер 200 х 200, то изображение будет масштабировано в соответствии с этими значениями. Наоборот, масштаб изображения будет пропорционально уменьшен таким образом, чтобы заданные максимальные размерыне превышались. Например, изображение размером 400 х 300 будет уменьшено до размера 200x150. В результате максимально точно сохраняются пропорции изобра­жения.

Динамическое (то есть, на лету) изменение размеров изображения на сервере гораздо предпочтительнее, нежели простое задание атрибутов HEIGHT и WIDTH в дескрипторе <IMG>. Размер большого изображения с высоким разрешением может составить несколько мегабайт. Если же картинку уменьшить до приемлемых размеров, то размер может оказаться менее 100 Кб. Следовательно, нет надобности загружать на сервер файл большого размера, а затем предлагать браузеру изменить размеры изображения.

Ключевой аспект операции изменения размеров состоит в вычислении новых параметров ширины и высоты. При этом определяется соотношение между реальными и максимальными размерами. Параметры $max\_width и $max\_height можно пере­дать в одной строке запроса, в противном случае будут задействованы стандартные значения, определенные вначале сценария.

Если изображение уже меньше заданных размеров, его ширина и высота остаются неизменными.

Для хранения данных, необходимых для работы системы используется 6 таблиц: keywords, messages, pages, stories, writer\_permissions, writers.

Keywords служит для хранения ключевых слов для каждой статьи, что позволяет более эффективно найти статью с помощью функции поиска.

*Таблица 2. Структура таблицы keywords базы данных*

|  |  |
| --- | --- |
| **Поле** | **Тип** |
| story  | int(11) |
| keyword | varchar(32) |
| weight | int(11) |

Поле story – это индивидуальный номер статьи, по которому она однозначно определяется, keyword – ключевое слово, weight – вес (цена) слова (изменяется от 1 до 10). Чем больше вес слова, тем больше релевантность этой статьи к запросу поиска. Во втором столбце указан тип данных для каждого поля, а в скобках максимальная длина значения поля.

Таблица messages применяется для хранения сообщений, которыми могут обмениваться пользователи системы.

*Таблица 3. Структура таблицы messages базы данных*

|  |  |
| --- | --- |
| **Поле** | **Тип** |
| message\_id  | int(11) |
| from\_user  | varchar(32) |
| to\_user | varchar(16) |
| body  | text |
| read  | char(1) |
| date  | int(11) |

Единственное, что хотелось бы отметить в этой таблице – это наличие поля read, которое может принимать значения 1 либо 0. При создании сообщения этому поля присваивается значение 0, а после того, как тот пользователь, которому оно адресовано, открывает страницу с сообщениями ему присваивается 1. Это служит для того, чтобы как-то выделить новые непрочитанные сообщения.

В таблице pages есть всего 2 поля: это название раздела и краткое его описание.

А вот таблица stories наиболее крупная из всех – она служит собственно для хранения статей и является основой всей системы.

*Таблица 4. Структура таблицы stories базы данных*

|  |  |
| --- | --- |
| **Поле** | **Тип** |
| id  | int(11) |
| writer  | varchar(16) |
| page  | varchar(16) |
| headline  | text |
| story\_text  | text |
| picture | text |
| created | int(11) |
| modified | int(11) |
| published | int(11) |
| for\_admin  | char(1) |

Итак, поле id – индивидуальный номер статьи, writer – автор статьи, page – к какому тематическому разделу относится данная статья, headline – заголовок, story\_text – основной текст, picture – адрес файла с изображением. Следующие 3 поля предназначены для работы с датой, соответственно создания, редактирования и публикации документа. Последнее поле по своей функциональности похоже на поле read из таблицы messages: оно может принимать 0 или 1. Поле принимает значение 1, когда редактор раздела нажимает ссылку «Готово», которое служит для отправки статьи администратору. То есть, когда в этом поле стоит 1, то уже не автор, не редактор не имею доступа к этому документу – он уже предназначен для проверки администратором.

Таблица writer\_permissions служит для присвоения редакторам прав управления тем или иным разделам. Кстати, редакторы определяются в последней таблице writers последним полем editor.

*Таблица 5. Структура таблицы writers базы данных*

|  |  |
| --- | --- |
| **Поле** | **Тип** |
| username | varchar(16) |
| password | varchar(16) |
| full\_name | text |
| editor  | char(1) |

Подчеркнутые поля в приведенных таблицах означают, что данное поле является индексом всей таблицы.

#

# Заключение

Итак, в своей курсовой работе я постарался собрать воедино наиболее важную и актуальную информацию по разработке информационных систем вообще и подсистему документооборота в частности.

Были исследованы и проанализированы основные принципы создания ИС, ее структура и функциональность, взаимодействие основных компонентов.

Также были рассмотрены новейшие и наиболее перспективные Web-технологии, которые уже сегодня с успехом использует при создании и обслуживании информационных серверов, содержащих в себе гигантские объемы информации.

Была разработана, создана и протестирована собственная подсистема документооборота, в которой реализовались основные принципы построения и взаимодействия ИС, а также основная функциональность работы.

# Список литературы

1. **Веллинг Л., Томасон Л.** Разработка Web-приложений с помощью PHP и MySQL – М.: Издательский дом «Вильямс», – 2003. – 800с.: ил.
2. **Гилмор В.** PHP 4. Учебный курс – СПб.: «Питер», – 2001. – 352с.: ил.
3. **Курбацкий А. Н.** Автоматизация обработки документов - Мн.: БГУ, 1999. - 221с.: ил.

# Приложение 1. Листинги SQL-запросов по созданию таблиц

*Листинг 1. SQL-запрос создания таблиц базы данных*

# БД : `content`

#

# --------------------------------------------------------

#

# Структура таблицы `keywords`

#

CREATE TABLE `keywords` (

 `story` int(11) NOT NULL default '0',

 `keyword` varchar(32) NOT NULL default '',

 `weight` int(11) NOT NULL default '0'

) TYPE=MyISAM;

# --------------------------------------------------------

#

# Структура таблицы `messages`

#

CREATE TABLE `messages` (

 `message\_id` int(11) NOT NULL auto\_increment,

 `from\_user` varchar(16) NOT NULL default '',

 `to\_user` varchar(16) NOT NULL default '',

 `body` text,

 `read` char(1) NOT NULL default '0',

 `date` int(11) default NULL,

 PRIMARY KEY (`message\_id`)

) TYPE=MyISAM AUTO\_INCREMENT=20 ;

# --------------------------------------------------------

#

# Структура таблицы `pages`

#

CREATE TABLE `pages` (

 `code` varchar(16) NOT NULL default '',

 `description` text,

 PRIMARY KEY (`code`)

) TYPE=MyISAM;

# --------------------------------------------------------

#

# Структура таблицы `stories`

#

CREATE TABLE `stories` (

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

 `writer` varchar(16) NOT NULL default '',

 `page` varchar(16) NOT NULL default '',

 `headline` text,

 `story\_text` text,

 `picture` text,

 `created` int(11) default NULL,

 `modified` int(11) default NULL,

 `published` int(11) default NULL,

 `for\_admin` char(1) NOT NULL default '0',

 PRIMARY KEY (`id`)

) TYPE=MyISAM AUTO\_INCREMENT=65 ;

#

# Структура таблицы `writer\_permissions`

#

CREATE TABLE `writer\_permissions` (

 `writer` varchar(16) NOT NULL default '',

 `page` varchar(16) NOT NULL default ''

) TYPE=MyISAM;

# --------------------------------------------------------

#

# Структура таблицы `writers`

#

CREATE TABLE `writers` (

 `username` varchar(16) NOT NULL default '',

 `password` varchar(16) NOT NULL default '',

 `full\_name` text,

 `editor` char(1) NOT NULL default '0',

 PRIMARY KEY (`username`)

) TYPE=MyISAM;

# Приложение 2. Листинги основных PHP скриптов

*Листинг 2. Файл stories.php (основной файл системы)*

<?php

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Подключение библиотеки с функциями

include('include\_fns.php');

include('header.php');

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Проверка входа в систему

if (!check\_auth\_user()) {

?>

<form action="login.php" method="post">

<table border="0">

<tr>

 <td>Если Вы новый пользователь, то зарегистрироваться можно <a href=add\_new\_user.php>тут</a>.</td>

</tr>

</table>

<br>

<table border="0">

<tr>

 <td>Вход:</td>

</tr>

<tr>

 <td>Логин:</td>

 <td><input size="16" name="username"></td>

</tr>

<tr>

 <td>Пароль:</td>

 <td><input size="16" type="password" name="password"></td>

</tr>

</table>

<input type="submit" value="Пуск!">

</form>

<?

}

else {

 $conn = db\_connect();

 $w = get\_writer\_record($HTTP\_SESSION\_VARS['auth\_user']);

 print 'Привет, '.$w['full\_name'];

 print ' (<a href="change\_passwd\_form.php">Поменять пароль</a>)';

 print ' ';

 print ' (<a href="logout.php">Выход</a>)';

 print '<p>';

 $sql = 'select \* from stories where writer = \''.

 $HTTP\_SESSION\_VARS['auth\_user'].'\' order by created desc';

 $result = mysql\_query($sql, $conn);

 print 'Ваши статьи: ';

 print mysql\_num\_rows($result);

 print ' (<a href="story\_add.php">Добавить</a>)';

 print '</p><br /><br />';

 if (mysql\_num\_rows($result)) {

 print '<table border=1>';

 print '<tr><th>Заголовок:</th><th>Раздел:</th>';

 print '<th>Дата создания:</th><th>Последнее изменение:</th><th>Действия:</th></tr>';

 while ($qry = mysql\_fetch\_array($result)) {

 print '<tr>';

 print '<td>';

 print $qry['headline'];

 print '</td>';

 print '<td>';

 print $qry['page'];

 print '</td>';

 print '<td>';

 print date('M d, Y H:i', $qry['created']);

 print '</td>';

 print '<td>';

 print date('M d, Y H:i', $qry['modified']);

 print '</td>';

 print '<td>';

 if ($qry['published'])

 print '[Напечатано '.date('M d, H:i', $qry['published']).']';

 else {

 if ($qry['for\_admin'])

 print 'Ожидает проверки администратором...';

 else {

 print '[<a href="story.php?story='.$qry['id'].'">Ред.</a>] ';

 print '[<a href="delete\_story.php?story='.$qry['id'].'">Удал.</a>] ';

 print '[<a href="keywords.php?story='.$qry['id'].'">Слова</a>]';

 }

 }

 print '</td>';

 print '</tr>';

 }

 print '</table>';

 }

print ('<table>');

print ('<br>');

 print ' (<a href="message\_add.php">Написать сообщение</a>)';

print ('</table>');

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Меню редактора

$conn = db\_connect();

$sql = 'select \* from writers where username = \''.

 $HTTP\_SESSION\_VARS['auth\_user'].'\' ';

$result = mysql\_query($sql, $conn);

$qry = mysql\_fetch\_array($result);

if ($qry['editor'])

{

 editor\_menu ($HTTP\_SESSION\_VARS['auth\_user']);

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Меню сообщений

$conn = db\_connect();

$sql = 'SELECT \* FROM messages WHERE to\_user = \''. $HTTP\_SESSION\_VARS['auth\_user'].'\' order by date desc ';

$result = mysql\_query($sql, $conn);

$num\_results = mysql\_num\_rows($result);

if ($qry)

{

if ($num\_results) {

 print ('<table>');

 print ('<br><br> -------------------------- <br>');

 print ('<b>Раздел сообщений:</b>');

 print ('</table>');

 print '<table border=1>';

 print '<tr><th>От:</th><th>Текст:</th>';

 print '<th>Дата:</th><th>Действия:</th></tr>';

 for ($i=0; $i < $num\_results; $i++) {

 $qry = mysql\_fetch\_array($result);

 print '<tr>';

 print '<td>';

 print $qry['from\_user'];

 print '</td>';

 print '<td>';

 print $qry['body'];

 print '</td>';

 print '<td>';

 print date('M d, H:i, Y', $qry['date']);

 print '</td>';

 print '<td>';

 print '[<a href="message\_add.php?to\_user='.$qry['from\_user'].'">Ответить</a>] ';

 print '[<a href="delete\_message.php?message\_id='.$qry['message\_id'].'">Удал.</a>] ';

 print '</td>';

 print '<td>';

 if (!$qry['read'])

 print ('NEW!');

 print '</td>';

 print '</tr>';

 }

 $conn = db\_connect();

 $sql = 'UPDATE `messages` SET `read` = \'1\' WHERE `to\_user` = \''. $HTTP\_SESSION\_VARS['auth\_user'].'\' ';

 $result = mysql\_query($sql, $conn);

 print '</table>';

 }

}

}

include('footer.php');

?>