Практическая работа (магистранты 1 года обучения, ИКМО-05-18)

Тема: Изучение инфраструктуры портальной системы управления web-контентом и информационных процессов, влияющих на ее функционирование

Цель: построение архитектуры программной системы, обеспечивающей управление web-контентом в составе интеграционной платформы

Управление контентом –коллективный процесс, участники которого имеют определенные роли:

– авторы создают и редактируют контент;

– редакторы контролируют стиль и подачу информации, а также занимаются переводом и локализацией;

– выпускающие редакторы отвечают за публикацию контента;

– администраторы управляют назначением прав доступа к контенту участников процесса и осуществляют другую поддержку;

– пользователи или потребители получают опубликованный контент.

Системы управления контентом и являются теми средствами, с помощью которых автоматизируется процесс управления контентом. Поскольку управление контентом состоит в обеспечении его жизненного цикла, системы управления контентом должны иметь адекватные средства для поддержки всех его стадий и обеспечения ролевого принципа коллективной работы.

Несмотря на то, что программное обеспечение для управления информацией строится на общих принципах, оно разделяется на ряд категорий, в соответствии с видами информации и спецификой их обработки. Различают:

– Document Management (DM) – управление документами;

– Web Content Management (WCM) – управление web-контентом;

– Knowledge Management (KM) – управление знаниями (сохранение и распространение в формализованном виде знаний от одних сотрудников предприятия другим);

– Digital Assets Management (DAM) – управление цифровыми активами (графической, мультимедийной информацией, файлами);

Все перечисленное может быть собрано в состав комплекса CMS – многопользовательской системы, взаимодействие с которой осуществляет несколько групп пользователей (авторы контента, потребители контента, администраторы), использующих различный функционал системы и создающих принципиально различную нагрузку на систему, существует проблема создания адекватной тестовой нагрузки.

CMS входит в состав гетерогенной системы, включающей компоненты, построенные с использованием различных программных платформ, а также унаследованные компоненты, механизмы работы компонентов с программными ресурсами могут отличаться. Также отличаются и средства предупреждения, диагностики и обработки ошибок, что приводит к различному поведению компонентов системы в случае возникновения утечек ресурсов и программных отказов. Это создает предпосылки к возникновению ошибок управления ресурсами, вызванных совместной работой компонентов.

Разработка CMS, как и любого другого программного продукта высокого уровня сложности, требует существенных усилий на отладку и тестирование. Большое значение в процессе разработки играют правильная постановка задачи, анализ требований к подлежащему реализации функционалу, сценариям и условиям использования, возможностям программных и технических платформ, ожидаемым нагрузкам. Во многом требования определяют рациональный выбор программных платформ и паттернов проектирования, позволяющих снизить трудозатраты. Не менее важной является организация итогового тестирования продукта,

позволяющего выявить часть программных ошибок и определить работоспособность продукта при расчетных нагрузках.

Компании уровня Google обладают значительными накопленными шаблонами нагрузок, которыми пользуются при тестировании внутренних разработок, однако такие возможности не доступны рядовым разработчикам.

Сложность организации нагрузки связана с тем, что крайне трудно предположить поведение будущих пользователей. Хотя число пользователей и поддается оценке, основная трудность состоит в определении их ожидаемой активности, т.е. частоты обращений к системе, смене уровней активности, различии уровней активности и характере перехода от одного уровня активности к другому.

Естественно, предлагаемые модели должны согласовываться с опытными данными, полученными при работе программного решения. Для этого рассматриваемая в настоящей работе CMS была оснащена средствами сбора характеризующих пользовательскую активность данных. В данном случае наличие собственного решения дало преимущество за счет возможности сбора внутренних компонентов и сбора всей интересующей информации.

CMS предоставляет доступ не только к собственному хранилищу, но и к внешним источникам информации, данные о состоянии и характеристики быстродействия которых системе не известны, и могут меняться в широких пределах. В частности, обращение к информационным источникам приводит к необходимости межпроцессного взаимодействия, в отличие от большинства других задач, работающих с объектами в памяти процесса web-сервера, что само по себе меняет величины быстродействия на порядки.

Так, например, у реляционных информационных источников, реализуемых, как правило, на основе сервера баз данных, можно наблюдать сложное поведение времени выполнения запроса в зависимости от внутреннего состояния,

Фундаментальными принципами CMS считаются:

1. использование для хранения контента специализированного хранилища;

2. поддержку коллективной работы пользователей, размещающих контент в этом хранилище.

В общем случае, архитектура системы управления контентом включает следующие подсистемы: хранилище контента, подсистему безопасности, подсистему производства контента и подсистему представления (рис. 1). В случае каркаса системы управления контентом эти же компоненты входят в состав ядра.

Функциональность систем CMS перечислена в [107, 108]. Далее эта функциональность будет рассмотрена применительно к рассматриваемым подсистемам.

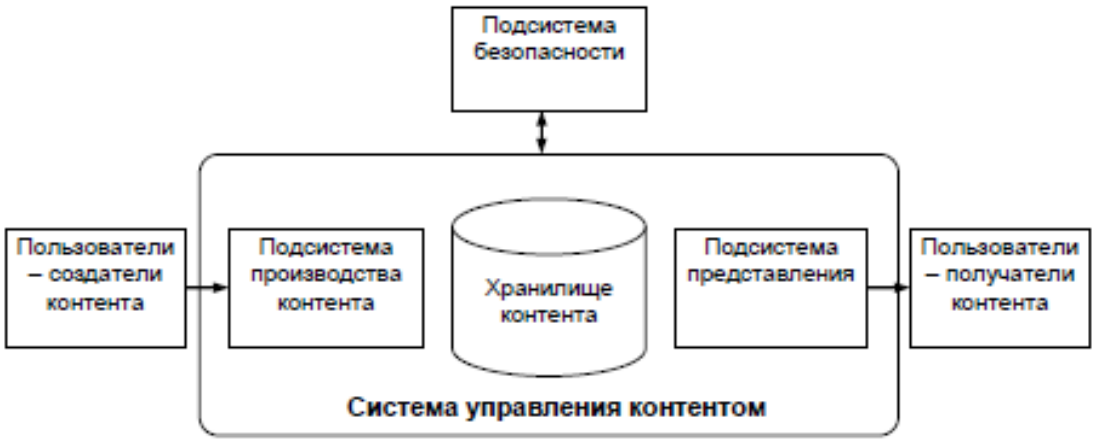

Рис. 1 Подсистема производства контента

Подсистема производства контента

Подсистема производства контента является основным механизмом для коллективной работы пользователей по созданию и модификации контента. Данная подсистема используется авторами, редакторами и администраторами контента.

Основная функциональность, реализуемая подсистемой производства контента:

Создание контента – помещение нового контента в систему, включая его индексирование, категоризацию, описание с помощью метаданных и назначение прав доступа.

WYSIWYG редактирование – использование для подготовки текста средств редактирования, позволяющих непосредственно наблюдать применяемое к тексту форматирование.

Преобразование контента – конвертирование различных типов контента во внутреннее представление системы (например, преобразование документов Word в HTML).

Извлечение контента – загрузка и преобразование данных и их информационной структуры, созданных в других системах.

Управление метаданными и таксономией – создание системы меток, используемых для категоризации контента, а также разработка правил категоризации.

Связывание контента – формирование и отслеживание целостности ссылок между элементами контента.

Документооборот – реализация необходимых бизнес-процессов по обработке контента (например, документ после его создания автором должен быть проверен редактором и одобрен главным редактором).

Планирование публикаций – возможность управления начальной датой и сроком публикации контента.

Интернационализация – обеспечение многонационального производства контента, включая многоязыковую поддержку и возможность учета специфики культур.

Размещение контента – процесс перевода подготовленного к публикации контента из среды разработки в среду доставки контента конечным пользователям.

Архивирование контента – поддержка правил хранения контента с истекшим сроком публикации. Большое количество устаревшего контента может существенно снижать производительность CMS, поэтому требуются специальные меры по разделению актуального и устаревшего контента.

Системные отчеты – обеспечение журналирования действий пользователей, истории модификации контента, протоколирование сбоев системы и т.д.

Подсистема производства контента может быть реализована как приложение с архитектурой «клиент-сервер» и как web-приложение. Первый вариант требует разработки и установки на рабочие места отдельного клиентского приложения, однако обеспечивает большую защищенность всей инфраструктуры подготовки данных и упрощает построение пользовательского интерфейса.

Большое значение при построении подсистемы имеет способ внутреннего представления (моделирования) контента в CMS. Используется два подхода к моделированию контента. При первом каждый тип контента моделируется как единый элемент (документ) с определенным набором атрибутов. При втором контент моделируется как контейнер, содержащий набор более простых информационных элементов (разделов, абзацев, ссылок, рисунков) . Второй подход лежит в основе компонентного управления контентом (Component Content Management, CCM).

Есть возможность повторного использования контента и, в частности, многоканальной публикации (один и тот же тип контента может использоваться для отображения в различных контекстах на web-сайте, для показа на мобильных и беспроводных устройствах, для синдикации и для создания печатной продукции типа брошюр или каталогов).

Для представления контента широкое применение получил язык XML (Extensible Markup Language), что связано с его удобством для представления и автоматизированной обработки структурированных данных. Ряд XML-схем

для представления контента был стандартизирован и получил широкое распространение.

Перспективным способом построения подсистемы производства контента является использование сервис-ориентированной архитектуры (SOA). Основной принцип этой архитектуры – обеспечение доступности всей внутренней программной архитектуры приложения через внешние интерфейсы (RPC, DCOM, web-сервисы), что позволяет организовать доступ одних приложений к данным и функциям других. Сервис-ориентированная архитектура опирается на такие стандарты как:

– Universal description, discovery and integration (UDDI);

– Web Services Description Language (WSDL);

– Simple Object Access Protocol (SOAP).

При этом походе подсистема производства контента и CMS в целом могут предоставлять функциональность другим прикладным системам организации, за счет чего расширяются возможности по управлению контентом.

В процессе эволюции систем управления контентом был выработан ряд протоколов и стандартов, применяемых при построении подсистемы производства контента:

– Web-based Distributed Authoring and Versioning (WebDAV) – протокол для поддержки коллективного редактирования и управления файлами на удаленных серверах. Протокол поддерживается многими приложениями (например, Microsoft Word), что позволяет интегрировать эти приложения с CMS, обладающими поддержкой протокола WebDAV, и использовать для создания и редактирования контента;

– Dublin Core– набор стандартов для описания web-ресурсов с помощью метаданных;

– Open Document Management API (ODMA) – стандарт для взаимодействия клиентов с системами управления документами (DMS);

– Docbook – XML схема для структурированного представления книг и статей технической направленности. В настоящее время практически вытеснен стандартом DITA;

– Darwin information typing architecture (DITA)– стандарт для структурированного представления контента с помощью основанного на XML языка, описывающий структуру, идентификацию, представление метаданных и ссылок, правила форматирования различных видов информации. Данный стандарт используется при реализации компонентного управления контентом.

Уровни программной архитектуры

При проектировании целесообразно выделить отдельный уровень архитектуры, обеспечивающий работу со всеми информационными источниками в терминах единой схемы данных. Также в отдельный уровень была выделена серверная бизнес-логика портала. Итоговая многозвенная клиент-серверная архитектура Информационного web-портала показана на рис. 2.

Архитектура содержит четыре уровня :

– клиентский, обеспечивающий взаимодействие пользователя с системой;

– бизнес-логики, реализующий алгоритмы работы с данными;

Организация метаданных

Каждый элемент контента при помещении в CMS получает набор атрибутов, образующих его метаданные. Атрибуты описывают тип данных, название элемента, дату его создания, категорию и другие характеристики в соответствии с решаемыми CMS задачами.

В разработанной системе существует два аспекта использования контента:

– с точки зрения посетителя сайта каждый элемент контента это страница, находящаяся в общей структуре сайта;

– с точки зрения создателя контента – каждый элемент контента это документ, находящийся в одной из папок хранилища.

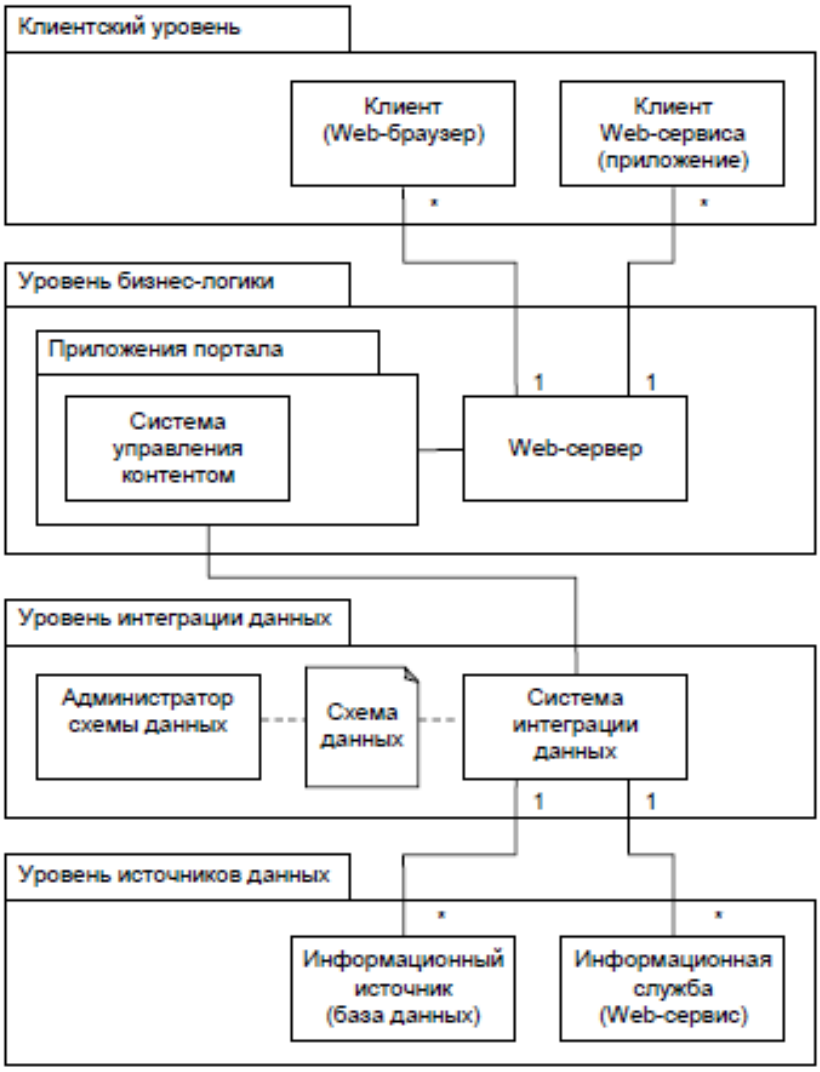

Рис.2. Многозвенная клиент-серверная архитектура web-портала

Чтобы придать элементу контента семантику страницы сайта, необходимо расширить его рядом атрибутов. В первую очередь, это URI, позволяющий однозначно идентифицировать страницу, содержащую конкретный элемент контента. Также должна быть определена связь страницы с разделом сайта, позволяющая задать положение страницы в структуре сайта. Для построения навигационного меню должно быть указано название страницы. Для правильного отображения контента должен быть

указан соответствующий шаблон представления. Указанные атрибуты образуют метаданные страницы.

В то же время элемент контента является документом в хранилище контента. Семантика документа выражается атрибутами «Автор», «Аннотация», «Дата документа», а также принадлежностью к определенному разделу классификатора.

В разработанной системе контент и метаданные хранятся раздельно.

Метаданные используются для формирования двух структур – структуры хранилища контента и структуры сайта.

Структура хранилища контента является инструментом для обеспечения удобства работы с большими массивами информации. Эта структура аналогична

Структура сайта — это основной способ упорядочения web-страниц и опубликованного на сайте контента. Она формируется и поддерживается редакторами структуры в соответствии с целевым назначением сайта.

Структуры хранилища контента и сайта содержат ссылки на одни и те же элементы контента, однако являются независимыми друг от друга. Взаимное отображение структур хранилища контента и сайта обеспечивается через файл отображения, содержащий ссылки на элементы контента и метаданные страниц (см. рис. 3).

Независимость структур хранилища контента и сайта обеспечивает ряд преимуществ по сравнению с CMS, в которых структуры хранилища контента и сайта совмещены. К ним относятся:

– разделение зон ответственности (создание нового контента и управление структурой сайта полностью разделены);

– возможность более удобной организации контента в хранилище;

– упрощение управления структурой сайта.

Поскольку CMS как и любая web-система выполняется на сервере, физический доступ пользователей к которому ограничен, основным средством взаимодействия с ней является web-интерфейс .

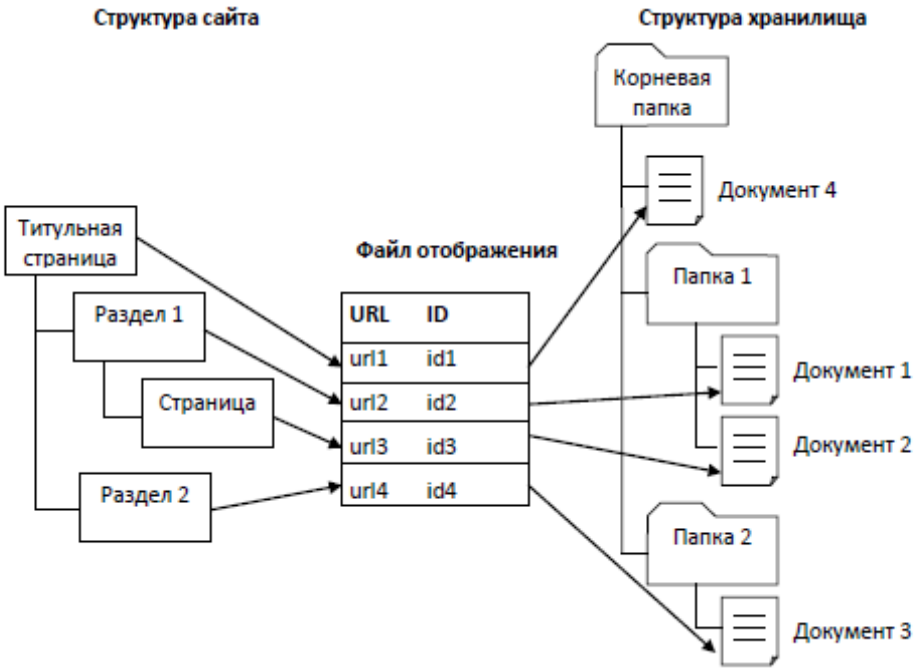

Рис.3. Связь структур сайта и хранилища

Функционирование web-интерфейса существенно отличается от функционирования интерфейса настольного приложения. В настольном приложении пользовательский интерфейс и обрабатываемые данные находятся в памяти одного компьютера на протяжении всего сеанса работы пользователя.

Информационный киоск – (information kiosk) – простая абонентская аудиовидеосистема публичного или персонального пользования, практически не управляемая самим пользователем в смысле конфигурирования архитектуры, каталогизации и даже видоизменения **контента**. В последние годы информационный киоск описывается как ИС киоска данных – весьма популярного и успешно применяемого типа хранения данных. Такие киоски представляют собой мини системы (ограниченного диапазона) **принятия решений (**http://www. *****/basic/kiosk. asp**),** то есть реализуют функции OLAP (Online Analytical Processing – оперативный анализ данных), намного реже – функции OLTP (Online Transactional Processing – оперативная

обработка **транзакций**). Киоски могут быть связаны в единое распределенное корпоративное хранилище данных (знаний) – «восходящий» или botton-up метод построения корпоративной ИС. Часто киоски строятся по так называемой звездообразной схеме, в которой на диаграмме отношений таблица фактов располагается в центре, а таблицы измерений – на исходящих из центра лучах, образующих рисунок звезды, причем важной особенностью корпоративной ИС как интеграции информационных киосков является аддитивность фактов (facts, continuosly valued), что позволяет суммировать их по всем измерениям и в процессе присоединения коиска к интегративной корпоративной ИС, построенной на основе киосков. В такой ИС кроме набора киосков обычно присутствуют репозитарий и управляющая (дирекционная) подсистема с функциями информационно-поисковой подсистемы.

ЗАДАНИЕ. Часть 1. На основе приведенного текста приведите рис 4. В полный вид (дополните пропущенные элементы), дайте развернутые ответы на вопросы.

В случае использования web-интерфейса обработка данных происходит на сервере, а отображение пользовательского интерфейса на компьютере пользователя. Действия пользователя при этом преобразуются в запросы по протоколу HTTP. Этот протокол не обеспечивает непрерывного характера взаимодействия с сервером и не позволяет определить действия, ранее выполненные в сеансе работы пользователя. Пользовательский интерфейс настольных приложений не может быть напрямую реализован средствами web-технологий. Он может быть только до некоторой степени сымитирован с помощью таких методик, как механизм пользовательских сессий, реализуемый сервером, файлы «cookie», передача состояния сеанса через скрытые поля форм или параметры web-запроса.

Пользовательские сессии – это механизм, реализуемый web-сервером для хранения состояния сеанса между последовательными запросами. Сервер создает специальный объект, идентификатор которого передается в заголовке или в строке параметров web-запроса. Объем данных сеанса ограничен объемом памяти web-сервера, механизм можно использовать при запрете приема файлов «cookie» в web-браузере. Однако данный механизм усложняет масштабирование web-системы, поскольку пользовательская сессия хранится на одном web-сервере. При необходимости одновременной работы нескольких web-серверов требуются дополнительные действия для обеспечения доступности данных сессии .

Файлы «cookie» являются частью протокола HTTP, поэтому их поддержка web-сервером всегда присутствует, а их применение не влияет на масштабирование web-системы. В то же время, файлы «cookie» ограничивают объем информации, используемой для хранения состояния сеанса. Помимо этого, прием файлов «cookie» может быть запрещен, либо заблокирован брандмауэрами. Тем не менее, использование файлов «cookie» распространено, и во многих случаях может быть достаточным решением .

Для сохранения состояния сеанса могут применяться *скрытые поля* HTML-форм. Этот метод пригоден только при работе с одной HTML-формой

и только для событий пользовательского интерфейса, приводящих к отправке формы на сервер (submit). При выполнении пользователем перехода по ссылке или при перенаправлении web-запроса на другой URL, содержимое скрытых полей не будет сохранено.

Метод используется в технологии Microsoft ASP.NET под названием «состояние отображения» (ViewState).

Состояние сеанса может быть сохранено как параметр URL web-запроса. Из-за ограниченной длины строки запроса (не более 2048 символов в большинстве web-браузеров), ограничений на используемые символы, а также возможного искажения передаваемых данных самим пользователем такой способ можно использовать лишь в некоторых случаях.

При реализации CMS, рассматриваемой в настоящей работе, была применена технология ASP.NET, содержащая программные решения для всех указанных методов. Благодаря этому стало возможно использовать в Информационном web-портале сочетание нескольких методов сохранения состояния сеанса. Из рассмотренных методов были исключены только пользовательские сессии из-за накладываемого ими ограничения на масштабирование web-системы.

Необходимость взаимодействия компонентов пользовательского интерфейса, например, при использовании для модификации данных нескольких окон, представляет собой еще одну проблему его реализации. Если несколько экземпляров web-браузера используется при модификации одних и тех же данных, изменения, выполняемые в одном из экземпляров web-браузера, недоступны в другом. Основным способом решения этой проблемы является имитация многооконных интерфейсов с помощью сценариев, однако уровни соответствия стандартам реализаций языков сценариев в web-браузерах различных производителей неодинаковы. При ограниченных ресурсах разработчики вынуждены ориентироваться на использование одного web-браузера. Так, административный интерфейс рассматриваемой CMS ориентирован на использование web-браузера Internet

Explorer. Очевидно, что это решение является компромиссным, и было принято исходя из ограниченного числа пользователей данного интерфейса.

На рис. 4 показана архитектура

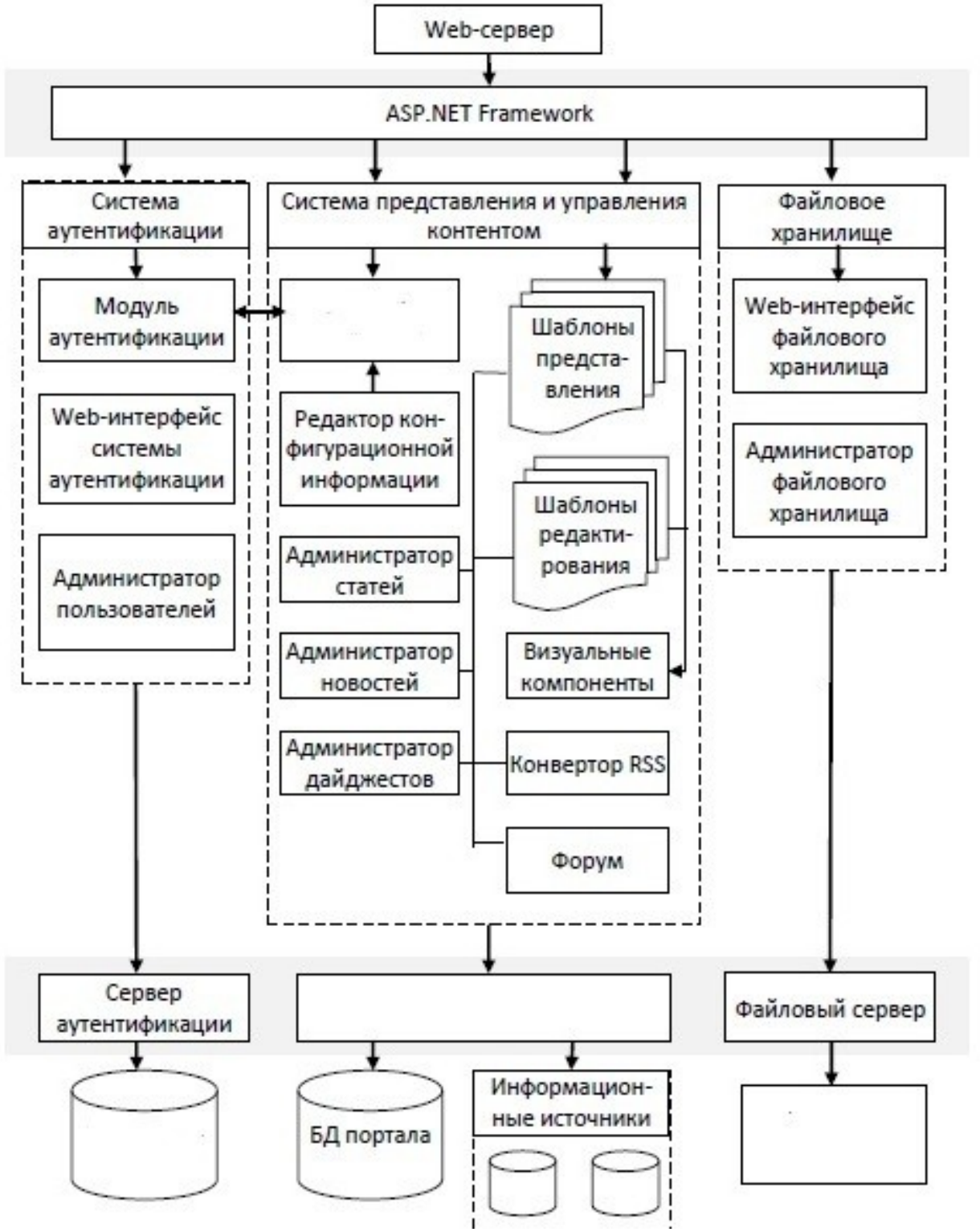

Рис.4 Архитектура –задание этой практической работы

CMS является web-приложением, выполняющимся на *web-сервере*, который обеспечивает прием запросов, генерируемых web-браузерами

пользователей, передачу запросов web-приложениям и возврат клиентам HTML-представления результатов запросов, сформированного web-приложениями. В качестве web-сервера в рассматриваемом решении используется Microsoft IIS.

представления и управления контентом и файлового хранилища являются приложениями ASP.NET, поэтому для обеспечения их работы на web-сервере установлена среда выполнения Microsoft .NET Framework 2.0.

Шаблоны представления предназначены для формирования HTML-представления требуемой web-страницы. Web-страница собирается из блоков (заголовка, меню, блоков новостей и контента, и т.п.), каждому из которых соответствует визуальный компонент, формирующий HTML-представление этого блока. Шаблон представления можно рассматривать как контейнер, содержащий требуемые компоненты, и объединяющий сформированные ими HTML-фрагменты. Свойства компонентов в программном коде шаблона могут переопределяться метаданными страницы, полученными от модуля конфигурационной информации. Каждый шаблон представления выполнен как обычная web-форма.

Визуальные компоненты портала формируют HTML-представление элементов контента на web-странице. Компоненты отвечают за преобразование входных данных, применение элементов стилевого оформления, генерацию событий пользовательского интерфейса. Шаблон представления предоставляет визуальным компонентам данные из различных источников (файла конфигурации, хранилища портала, внешних информационных источников). Взаимодействие с информационными источниками выполняется с помощью *сервера интеграции* [14]. В логической структуре отображения портала для каждой web-страницы задан набор метаданных. *Модуль работы с конфигурационной информацией* возвращает эти метаданные, используя адрес web-страницы. Затем код шаблона использует их для получения необходимых данных через сервер интеграции.

Поскольку URI (Uniform Resource Identifier) web-страниц не соответствуют физическим путям к шаблонам портала (URI являются виртуальными), требуется их преобразование, иногда обозначаемое термином URL rewriting . Модуль работы с конфигурационной информацией выполняет такое преобразование. Наличие преобразования путей делает их более наглядными и запоминающимися, упрощает их ввод с клавиатуры, облегчает индексирование портала поисковыми системами. Информация для преобразования путей, задающая соответствие URI и шаблонов представления, хранится в файле конфигурационной информации .

Компонент *«Редактор конфигурационной информации»* обеспечивает редактирование логической структуры отображения сайта. С помощью редактора производится создание метаданных страниц сайта портала, назначение страницам шаблонов отображения, размещение страниц в логической структуре отображения и их группировка в разделы. Эти данные хранятся в конфигурационном файле портала Content_Structure.xml.

Редактором поддерживается выполнение следующих действий :

– создание раздела;

– создание подраздела;

– удаление раздела;

– изменение позиции раздела в логической структуре отображения;

– добавление страницы к разделу;

– удаление страницы из раздела;

– изменение позиции страницы в логической структуре отображения;

– редактирование данных о разделе и страницах для выбранного языка представления;

– вызов шаблона редактирования для страницы раздела;

– вызов редактора командного запроса;

– управление разграничением доступа к странице (требование аутентификации пользователя, назначение списка ролей, имеющих доступ к странице).

Информация о разделе включает URI, текст пункта меню, описание раздела, перечень страниц раздела. Информация о странице включает URI, шаблон страницы, заголовок страницы, информацию о требуемых правах доступа для обращения к странице. Поддерживается хранение информации о разделах и страницах на нескольких различных языках

Списки поддерживаемых языков и зарегистрированных шаблонов не ограничены и сохраняется в XML-файле конфигурации редактора.

Редактирование параметров шаблонов представления обеспечивают *шаблоны редактирования*. Они предоставляют пользовательский интерфейс, позволяющий настроить внешний вид web-страницы путём указания настроек визуальных компонентов, а также обеспечивающий выбор и предварительный.

Система аутентификации идентифицирует пользователя, от имени которого инициирован текущий web-запрос. Портал рассчитан на работу с внешним сервером аутентификации, поэтому способ идентификации пользователя может меняться в зависимости от требований к порталу. Реализация, примененная на сайте www.ras.ru, использует т.н. «билет безопасности», выдаваемый пользователю в ходе процедуры регистрации при правильном вводе имени пользователя и пароля. Далее билет сохраняется в файле «cookie» и передается в последующих web-запросах. В начале каждого запроса валидность билета проверяется. Результаты проверки, а также сохраненная в билете информация используются модулем конфигурации для предоставления или запрета доступа пользователя к требуемому ресурсу.

Так как сервер аутентификации является приложением COM+, для взаимодействия с ним из среды .NET используется компонент интерфейса системы аутентификации. Данный компонент позволяет различным системам портала использовать функции сервера аутентификации, а также позволяет предоставлять услуги аутентификации другим web-серверам через web-сервис. Эта возможность позволяет интегрировать системы безопасности различных web-систем.

Компонент *«Администратор пользователей»* предназначен для управления регистрацией и правами пользователей портала. Сервер аутентификации поддерживает ролевую модель прав, при которой возможность пользователя выполнять определенные функции определяется назначенной ему ролью. Для удобства администрирования пользователей можно объединять в группы. При этом роли могут быть предоставлены как непосредственно пользователю, так и целой группе. Учетные данные пользователей, информация о группах и их членах и назначенных им ролях хранится в базе данных. Пользовательский интерфейс для редактирования этой информации и предоставляет компонент «Администратор пользователей».

Портал содержит компоненты *«Администратор статей», «Администратор новостей» и «Администратор дайджестов»*.

Компонент *«Конвертер RSS»* обеспечивает экспорт контента типа «новость» в формате RSS 2.0 для использования сторонними сайтами или программами RSS-агрегаторами

Компонент *«Форум»* реализует функции web-форума. Компонент содержит набор страниц для создания и просмотра тем и сообщений, выполнения административных задач. Поддерживаются как табличное так и иерархическое представление тем форума, стилевое оформление сообщений с помощью bulletin board кодов

Компонент «*Сервер интеграции данных»* представляет собой механизм, позволяющий скрыть различия в реализации источников данных от компонентов более высокого уровня. В роли источников могут выступать базы данных и службы, подключаемые с использованием различных протоколов и имеющие различные схемы данных. Сервер интеграции данных позволяет унифицировать работу с источниками данными других компонентов портала

Портальное хранилище контента использует реляционную базу данных, работающую под управлением Microsoft SQL Server. Работа с

хранилищем контента осуществляется посредством *сервера командных запросов*

Пользовательский интерфейс для доступа к файловому хранилищу портала реализует компонент *«Интерфейс файлового хранилища»*. Он обеспечивает просмотр содержимого папок, поиск и загрузку файлов. Добавление новых файлов, указание их атрибутов и редактирование структуры папок производится с помощью компонента *«Администратор файлового хранилища»*. Хранение файлов реализовано с помощью технологии FileStream. Метаданные файлов (тип, описание, источник) хранятся в базе данных

Одна из ключевых подсистем портала – это *Служба поиска*. Она обеспечивает полнотекстовый и атрибутный поиск во всех подключенных к порталу информационных источниках. Поскольку состав источников не известен заранее и может меняться в процессе эксплуатации портала, отсутствует возможность зафиксировать состав поисковых атрибутов и формат результатов поиска на этапе проектирования. Эту информацию служба поиска получает динамически из метаданных источников, указываемых в процессе их регистрации в сервере интеграции портала. При этом, для реализации полнотекстового поиска в схеме данных каждого источника должен быть реализован тип данных, наследующий от базового типа CSearchObject. При выполнении поиска служба вызывает все реализации данного типа, передавая им строку поиска, а затем отображает полученные результаты .

Для реализации атрибутного поиска, в схеме данных источника должно быть задано описание поисковой формы. В описании указывается тип данных, связь полей формы с параметрами поискового запроса, а также состав полей, отображаемых в результатах поиска. Служба поиска, используя данное описание, динамически формирует пользовательский интерфейс, отображает его пользователю, а затем выполняет поисковый запрос и отображает результаты поиска для выбранного источника.

2 часть задания

ПЕРЕРИСУЙТЕ архитектуру Рис.4 Архитектура –задание этой практической работы

С учетом добавления схематично средств обеспечения аутопоэзиса инфраструктуры портальной системы управления web-контентом и информационных процессов, влияющих на ее функционирование

Например:

На схеме рис .5 показаны ключевые объекты, по отношению к которым должны применяться средства обеспечения аутопоэзиса системы:

– Входящая информация.

– Процесс обработки информации.

– Процесс хранения информации.

– Исходящая информация.

Рис.5. Схема применения средств обеспечения живучести в технологическом процессе функционирования условной системы

Аутопоэзис придает условной системе определенных действий на всех этапах ее жизненного цикла. На этапе создания разрабатываются специальные средства обеспечения аутопоэзиса, органически интегрируемые в систему.

Ответьте на контрольные вопросы

(письменно и развернуто)

1 Что означает термин «контент»

2 Приведите схему жизненного цикла контента

3 Что такое «производство контента»

4 Что такое информационный морфизм

5 Что такое эмерджентность носителя знаний в ИС

6. Что такое информационный консорциум (картель)

7 Что такое энтальпия ИС

8 Система управления контентом Microsoft IIS

9. Модель Эдгара Кодда FASMI

10. Кто сформулировал основные принципы работы CMS

11. Понятие информационный ценоз

12. Что затрудняет разработку web-браузеров и требует создания разных версий сценариев?

Далее вопросы в зависимости от вашего варианта (номер в журнале)

Номер варианта	Перечень вопросов, на которые нужно ответить развернуто
1	1-12,14,34,53,28
2	1-12,15,36,55,27
3	1-12,13,35,41,26
4	1-12,16,37,42,25
5	1-12,18,36,43,24
6	1-12,15,38,44,23
7	1-12,14,37,45,22
8	1-12,19,39,46,21
9	1-12,21,41,47,20
10	1-12,20,40,41,19
11	1-12,22,42,47,18
12	1-12,21,41,46,17
13	1-12,23,44,45,16
14	1-12,22,43,44,15

15	1-12,24,45,43,14
16	1-12,26,47,42,13
17	1-12,28,46,41,29
18	1-12,27,48,47,30
19	1-12,29,50,46,31
20	1-12,31,49,45,32
21	1-12,30,52,44,33
22	1-12,32,51,43,34
23	1-12,33,54,42,35

13 Протокол для распространения контента между системами

14 Формат для синдикации контента

15 Прикладной протокол для синдикации

16 Формат архивирования и редактирования данных

17 Особенность и функционирование Software Configuration Management (SCM)

18 Особенность и функционирование Digital Rights Management (DRM)

19 Особенность и функционирование Records Management (RM)

20 Особенность и функционирование Learning Management (LM)

21 Особенность и функционирование Product Data Management (PDM)

22 Особенность и функционирование Open Management Group (UML)

23 Особенность и функционирование World Wide Web Consortium (HTML)

24 Особенность и функционирование World Wide Web Consortium (CSS)

25 Особенность и функционирование World Wide Web Consortium (XML)

26 Особенность и функционирование World Wide Web Consortium (WSDL)

27 Особенность и функционирование World Wide Web Consortium (SOAP)

28 Особенность и функционирование Open Management Group (PKI)

29 Особенность и функционирование European Computer Manufacturers Association (JavaScript)

30 Особенность и функционирование Internet Engineering Task Force (RFC)

31 Функционирование ассоциации OSCOM (Open Source Content Management)

32 Функционирование аналитической службы CMSWatch американской компании CMSWorks,

33 Функционирование австралийская аналитическая служба Step Two Designs,

34 Функционирование американская аналитическая компания ContentHere,

35 Функционирование интернет журнал CMSWire

36 Особенность и функционирование Dropbox (почта, облачные диски, средства для совместной работы над документами).

37 Особенность и функционирование ITSM систем

38 Особенность и функционирование ServiceDesk-системы

39 Особенность и функционирование ServiceNow

40 Особенность и функционирование ITSM365

41 Особенность и функционирование ZenDesk

42 Особенность и функционирование CRM-системы («Мегаплан»)

43 Особенность и функционирование CMS-системы

44 Особенность и функционирование почтовых систем (в основном MS Exchange)

45 Особенность и функционирование служебных сообщений и e-mail-маркетинга (Unisender)

46 Особенность и функционирование служебных сообщений и e-mail-маркетинга Mailchimp)

47 Особенность и функционирование служебных сообщений и e-mail-маркетинга (Mandrill)

48 Особенность и функционирование Information and Content Exchange Protocol (ICE)

49 Особенность и функционирование Really Simple Syndication (RSS)

50 Особенность и функционирование Atom

51 Особенность и функционирование JSR

52 Особенность и функционирование Automated Server Pages (ASP);

53 Особенность и функционирование PHP: Hypertext Preprocessor (PHP);

54 Особенность и функционирование Java Server Pages (JSP);

55 Особенность и функционирование ColdFusion;

56 Особенность и функционирование ASP.NET.

Оформленный отчет должен содержать:

Титульный лист

Лист 2 (цель, тема, сведения о ПК на котором выполнялась работа)

Теоретическая часть (не повторять текст, представленный в задании)

Задание часть 1

Рисунок 4

Задание часть 2

Рисунок на основе рис 4

Вывод

Ответы на контрольные вопросы (16) в каждом варианте

Список источников