ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Автономная некоммерческая организация науки и образования

«Институт компьютинга»

УТВЕРЖДАЮ

Директор АНО «Институт компьютинга»

_____ /А.И. Миков/

«__._.»_____2008 г.

м.п.

РЕКЛАМНО-ТЕХНИЧЕСКОЕ ОПИСАНИЕ

Система создания порталов WebMETAS

.31569113.00007-01 99 01

Листов 96

Разработчики:

_____/Хлызов А.В./

_____/ Рыжкова Е.А./

_____/ Лядова Л.Н./

14.08.2008

Пермь 2008

1. Функциональное назначение программы, область ее применения, ее ограничения

1.1. Назначение программы

Комплекс программ предназначен для реализации порталов на основе CASE-технологии METAS.

WebMETAS – это Web-приложение, которое обеспечивает средства создания порталов, их динамической настройки на потребности различных групп пользователей, а именно предоставляет возможности:

- *администратору* – настройки структуры портала, набора доступных конечным пользователям модулей, общего интерфейса пользователя и дизайна Web-форм редактирования объектов портала;
- *конечным пользователям* – просмотра и наполнения ресурсов портала.

Основными отличиями разработки, составляющими *научную новизну*, по сравнению с существующими системами являются:

- возможность создания приложения *в пошаговом режиме* (описание составляющих основу портала объектов и взаимосвязей между ними, параметров их отображения, навигации, адаптации под требования отдельных пользователей и т.д.) *без привлечения разработчиков*;
- *единый интерфейс пользователя для работы со всеми объектами портала*, включая объекты, соответствующие ресурсам портала, настройкам его модулей, параметров адаптации интерфейса и т.д;
- возможность *простого расширения функциональности* за счет введения новых типов ресурсов портала, параметров их отображения и взаимосвязей между ресурсами *без программирования и перезапуска системы*;
- возможность более сложного расширения функциональности за счет включения в портал новых модулей, разработанных внешними разработчиками или самостоятельно, *во время его работы* без необходимости останова.

Приложение реализовано в рамках создания CASE-системы METAS и основано на возможностях технологии METAS, позволяющей создавать динамически адаптируемые информационные системы (ИС), управляемые метаданными. CASE-технология METAS обеспечивает разработчиков средствами реструктуризации данных, генерации и настройки пользовательского интерфейса, средствами репортинга и управления бизнес-процессами, средствами защиты.

Реализация перечисленных особенностей приложения WebMETAS возможна за счет введения *нового уровня метаданных* в дополнение к существующим основным уровням (физическому, логическому и презентационному), описывающим все объекты портала, его структуру и интерфейс пользователя. Дополнительные метаданные предназначены для задания *Web-представления объектов портала, его навигационных структур и настройки под конкретного пользователя*. Новые метаданные, так же как и использующиеся другими компонентами системы, применяются *в режиме интерпретации в процессе эксплуатации портала*, что обеспечивает динамическую настройку на условия эксплуатации, адаптацию к требованиям пользователей и позволяет сделать интерфейс максимально удобным для пользователя.

Интерфейс нельзя считать достаточно удобным, если он не может адаптироваться под потребности пользователя, меняющиеся условия эксплуатации, изменения в структуре портала и т.д. CASE-технология METAS имеет средства для динамической реструктуризации баз данных (БД) информационных систем, созданных на ее основе, а также имеет определенные возможности по расширению функциональности. Однако остается нереализованной адаптация за счет настройки интерфейса и расширения функциональности для Web-составляющей портала.

В первой версии WebMETAS был реализован Web-интерфейс к системам, создаваемым на основе технологии METAS, для удаленных пользователей этих систем. Основным недостатком этой версии было то, что данный интерфейс представлял собой лишь аналог Windows-интерфейса ИС, что ограничивало возможности приложения. Он не отвечал стандартам и требованиям, которые предъявляются сейчас к удобству использования и функциональности порталов. Требования, которым удовлетворяет вторая версия, были сформулированы на основе анализа существующих средств создания порталов, систем управления контентом и средств генерации Web-приложений.

Вторая версия приложения WebMETAS может быть использована как для создания порталов, так и для реализации удаленного доступа к информационным системам. Предложенный подход достаточно универсален и позволяет работать с любыми системами и любыми предметными областями, описанными в соответствии с технологией METAS.

Приложение ориентировано на создание корпоративных порталов и информационных систем. Их отличительной особенностью от обычных сайтов является высокий уровень адаптации к изменениям предметной области, условий эксплуатации, требований к функциональности и интерфейса. Каждое подобное изменение без наличия высокоуровневых средств генерации Web-

приложений неизбежно ведет к значительным временным затратам на доработку портала. Применение предложенного при реализации WebMETAS подхода позволяет значительно сократить сроки первоначальной разработки порталов, введения их в эксплуатацию и подстройки под новые условия функционирования.

На текущий момент приложение WebMETAS представляет собой законченное приложение, обеспечивающее основные функции порталов по просмотру и редактированию ресурсов, навигации по ним, адаптации и настройки интерфейса, разграничения доступа и поиска.

В перспективе планируется усовершенствование его возможностей. В частности, ведется работа по увеличению набора поддерживаемых типов данных. Запланировано обеспечение поддержки бизнес-процессов. Также необходима интеграция с подсистемами разграничения доступа пользователей информационных систем, доступных через портал. Другим направлением дальнейшей работы является увеличение степени удобства интерфейса пользователя, а именно введение новых методов адаптации интерфейса на основе текущей цели работы пользователя и его уровня осведомленности о ресурсах портала. Также возможно создание системы дистанционного обучения на основе данной разработки. Для этого необходимо расширить модель пользователя для хранения и обработки уровня его знаний. Знания пользователя также можно использовать при настройке интерфейса.

Приложение прошло апробацию при разработке Web-сайта кафедры математического обеспечения вычислительных систем, а также в процессе опытной эксплуатации информационной системы «Образование Пермской области».

1.2. Область применения программы

Данное приложение может быть использовано *для создания корпоративных порталов и Web-доступа к информационным системам*, разработанным на основе CASE-технологии METAS.

Порталы находят свое применение в сфере обслуживания массовой аудитории пользователей Internet (например, Yahoo!, Lycos, Excite, Rambler), а также в коммерческом секторе для обслуживания клиентов и партнеров и для удовлетворения потребностей (в т.ч. организации удаленных и мобильных рабочих мест) сотрудников [1, 2, 3]. В связи с ростом потребности в быстром создании порталов и облегчении их поддержки появились коммерческие продукты и технологии, ориентированные как на автоматизацию процесса программирования Web-страниц, так и на создание сайтов без привлечения Web-

разработчиков [2]. В качестве примеров последних можно привести WebSphere Portal Server и Oracle Portal, обладающих большой гибкостью и широким набором функций. Помимо подобных продуктов на рынке средств создания порталов появился ряд разработок, представляющих готовые решения. В них входит стандартный набор модулей, достаточный для большинства сайтов, что позволяет за короткий период времени ввести портал в эксплуатацию. Обычно портал состоит из модуля новостей, модуля документов, форума, модуля ссылок, модуля поиска, модуля аутентификации и др.

Комплекс программ WebMETAS предназначен для «заполнения ниши» между дорогостоящими коммерческими продуктами от корпораций типа IBM и Oracle и готовыми решениями. Созданные программные средства, с одной стороны, должны иметь стоимость, сравнимую со стоимостью готовых решений, а с другой – иметь возможности расширения функциональности, характерные для корпоративных продуктов.

1.3. Ограничения использования программы

Приложение WebMETAS является составной частью CASE-системы METAS. Программное ядро CASE-системы METAS (MDK METAS) позволяет настраиваться на различные программные платформы, работать под управлением различных операционных систем Microsoft, использовать для создания информационных систем различные реляционные СУБД и источники данных, для которых существуют драйверы ODBC.

Для функционирования runtime-компонентов необходимо установить .NET Framework (распространяется бесплатно), драйверы ODBC (при установке операционной системы) и СУБД (можно использовать, в частности, Microsoft SQL Server Express, которая распространяется бесплатно), сервер IIS. В основе портала лежит технология Microsoft ASP.NET 2.0. В данной разработке использовались следующие компоненты ASP.NET: встроенная модель безопасности, аутентификации и авторизации пользователей (Security Model), привязка к данным (Data Binding), Web-части (Web Parts), пользовательские элементы управления (User Controls).

Возможности масштабирования системы определяются возможностями настройки на различные платформы.

Ограничения использования системы определяются только требованиями лицензионной чистоты и требованиями, предъявляемыми перечисленными выше программными средствами, применяемыми как для разработки ИС, так и для организации ее функционирования.

2. Техническое описание программы

При реализации приложения использована разработанная авторами модель портала. Ниже описаны основные принципы разработки комплекса программ WebMETAS, модель, лежащая в основе его реализации, и созданный на ее основе программный комплекс.

2.1. Общие принципы разработки

Сформулируем основные требования к порталам, которые будут создаваться при помощи приложений WebMETAS. На основе этих требований разработана общая архитектура приложения, реализованы входящие в комплекс программные компоненты.

Требования к порталам

Сначала перечислим *функции,* которые встречаются *в большинстве порталов*:

- обслуживание большого числа пользователей;
- обеспечение защиты хранящейся информации;
- разбивка хранимой информации на категории;
- персонализация;
- поиск и навигация;
- автоматизация коллективной работы;
- интеграция.

Поскольку портал предоставляет средства для создания и редактирования своих ресурсов, то его можно рассматривать как *систему управления контентом* (CMS). На основе анализа возможностей существующих CMS [39] можно добавить требования, которые не были перечислены ранее:

- Возможность изменения дизайна и структуры.
- Система документооборота с возможностью публикации.
- Наличие визуального редактора для редактирования наполнения портала.

Кроме того, практически все порталы, а также сайты, создаваемые при помощи CMS, имеют *минимальный набор модулей*. К ним относятся:

- модуль новостей,
- модуль документооборота,
- модуль голосований,
- форум,
- статистика посещений портала,
- поиск по порталу.

Кроме того, современный портал должен подстраиваться под потребности пользователя, иметь возможность адаптации средств навигации. На основе анализа *методов адаптации* мы выделили два из них: *глобальное руководство* и *индивидуализация информации*.

Сформулируем теперь *требования к интерфейсу*, которые можно реализовать при помощи этих методов:

- Возможность просмотра ранее посещенных страниц портала.
- Возможность просмотра закладок для страниц, отмеченных пользователем.
- Возможность просмотра релевантных текущей цели и содержимому изучаемого материала страниц.

Связь с уровнями метаданных METAS

Разработка базируется на CASE-технологии METAS. Вполне логично связать ресурсы портала с объектами логического уровня METAS, т.е. *каждый ресурс (страница) портала представляется отдельным объектом логического уровня*. Это позволяет создавать классы ресурсов, такие как класс новостей, документов, сообщений форумов и т.д., а сами новости, документы и сообщения являются экземплярами этих классов. Кроме того, объекты METAS используются для хранения параметров модулей (портлетов) и для служебных целей.

Таким образом, задается соответствие «1:1» в связке страница портала – объект METAS. Однако для того чтобы была возможность *отображать объект в качестве страницы*, необходимо ввести дополнительный атрибут объекта, хранящий HTML-текст этой страницы, который и будет отображаться в окне браузера.

Как правило, портал состоит не только из обычных HTML-страниц, содержащих какую-либо информацию, но также *включает различные формы для выполнения операций*. В своей основе операции есть манипуляции над объектами портала и отображение результата этих операций пользователю. В этом случае недостаточно хранить в дополнительном атрибуте объекта HTML-текст страницы, поскольку помимо отображения содержания нужно выполнять обработку событий на странице, посредством которых реализуются операции. С этой целью для каждого объекта возможно переопределение способа отображения в браузере. Это реализуется за счет включения в портал специальных форм для каждого такого объекта. При обращении к объекту портала пользователю возвращается не страница с текстом из HTML-атрибута объекта, а именно такая, специально запрограммированная для этого объекта форма.

Кроме отображения информации необходимо предоставить пользователю возможность *изменять информацию об объектах*. Для этого нужны Web-формы просмотра-редактирования атрибутов этих объектов, подобные формам Windows-интерфейса. Пользователь при обращении к объекту при условии наличия соответствующих прав должен иметь возможность переключаться из режима отображения в режим редактирования объекта. В этом случае в браузере пользователя должна отображаться форма редактирования этого объекта. Информация о параметрах отображения формы, наборе и типах элементов управления для отображения значений атрибутов объектов берется из презентационного уровня метаданных METAS.

Защита информации

Важным вопросом является обеспечение защиты информации и разграничения доступа пользователей. Защита необходима для предотвращения несанкционированного доступа к информации пользователями, а разграничение доступа – для возможности персонализации и организации рабочих мест пользователей. С этой целью вводится компонент защиты, который позволяет задавать права пользователей на доступ к информации и контролирует его.

Создаваемая модель защиты предназначена для разграничения прав пользователей на просмотр, редактирование, удаление и создание объектов системы. Поскольку категорий пользователей в системе немного, достаточно одного уровня иерархии ролей. Это позволяет назначать одинаковые права для пользователей, выполняющих схожие операции над объектами системы, что облегчает работу администратора. Так как портал должен поддерживать возможность персонализации и обеспечения личным рабочим местом, нужно иметь возможность назначения прав пользователям индивидуально.

Портлеты

Портал имеет модульную структуру. Это означает, что ресурсы портала следует отображать и обрабатывать по-разному. Например, работа с новостями принципиально отличается от работы в форуме. В данной работе модульная структура реализуется при помощи технологии портлетов.

Портлеты – это многоразовые компоненты, обеспечивающие доступ к некоторым внешним данным (например, образовательным данным) и предоставляющие пользователю некоторые специфические услуги. По сути, работа портала заключается во взаимодействии портлетов портала между собой.

Портлеты *должны иметь одинаковый программный интерфейс*, для того чтобы можно было расширять функциональность портала за счет подключения новых портлетов и экспорта портлетов из одного портала в другой.

Соответственно можно выделить *портлеты для новостей, документов, форумов, голосований и поиска*.

Кроме того, *возможности по адаптации интерфейса также можно реализовать в виде портлетов*. Поэтому к списку добавятся портлеты закладок, часто используемых ресурсов и релевантных ссылок.

Следует поподробнее рассмотреть *портлет релевантных ссылок*, поскольку он требует введения новых связей между объектами, а именно – связей по релевантности. Как правило, многие объекты портала связаны логически между собой, представляют собой информацию об одних и тех же сущностях, процессах или явлениях, но с различных точек зрения. Поэтому с целью предоставить пользователю возможность перехода между этими объектами вводится связь по релевантности между ними. Также возможны ситуации, когда объекты не могут быть связаны между собой непосредственно, но имеют схожую тематику. Для того чтобы можно было переключаться между такими объектами, для каждого объекта задается набор ключевых слов. Таким образом, объекты, имеющие одинаковые ключевые слова, оказываются связанными между собой неявно. Соответственно в портале должны быть реализованы функции для поиска связанных с текущим объектом сущностей как по связям по релевантности, так и по ключевым словам.

Кроме задания неявных связей между объектами, ключевые слова используются при поиске объектов.

Ключевые слова также используются для задания цели работы пользователя. Ключевые слова, взятые из текущей цели, вносят свой вклад в определение релевантных объектов.

Для работы *портлетов закладок* и *часто используемых ресурсов* необходимо хранить соответствующую информацию для каждого пользователя отдельно. Поэтому необходимо ввести модель пользователя. Об этом будет рассказано позднее.

Мы рассмотрели механизмы адаптации интерфейса, реализуемые и выполняемые системой автономно. Однако необходимо также предоставить пользователю определенные возможности *настройки своего рабочего места*. Это реализуется за счет выбора используемых пользователем портлетов и задания их параметров посредством редактирования соответствующих объектов.

2.2. Модель портала и основные алгоритмы

Для определения внутренней структуры портала и алгоритмов его работы необходимо формализовать задачу и построить математическую модель портала.

Введем формальную модель портала. Она включает в себя следующие подмодели:

- логическая модель;
- презентационная модель;
- модель Web-представления;
- модель защиты;
- модель пользователя.

Модель портала опирается на математическую модель системы METAS [40, 41]. Поэтому необходимо описать ее составляющие, а именно логический и презентационный уровни. Затем на их основе будет описана модель портала.

Логический уровень

Опишем граф логической модели G_l. Вершинами графа логической модели являются сущности предметной области; между сущностями существуют связи, которые представляются направленными дугами в графе логической модели:

$$G_l = (V_l, E_l)$$
$$V_l = \{e_1, e_2, \ldots, e_n\}, \text{где } n \in N$$
$$E_l = \{r_1, r_2, \ldots, r_m\}, \text{где } m \in N$$
$$r_i = (e_j, e_k), \text{где } i = \overline{1,m};\ j,k = \overline{1,n}$$

Любая вершина $e \in V_l$ имеет набор атрибутов $Attr(e) = \{a_0, a_1, \ldots, a_n\}, n \in N$. Атрибут a_0 является ключевым атрибутом сущности ($a_0 = key(e)$).

Любой неключевой атрибут a_i, где $i = \overline{1,n}$, может быть либо собственным атрибутом сущности (множество таких атрибутов – $Attr_{own}(e)$), либо атрибутом, реализующим связь с родительской сущностью, т.е. внешним атрибутом ($Attr_{parent}(e)$). Ключевой атрибут также относится к собственным атрибутам.

Все собственные атрибуты имеют тип $Type(a_i)$, который задает множество возможных значений атрибута и операции, применимые к этим значениям, а следовательно, и способ отображения и ввода значений атрибута.

Каждый атрибут a_i связи с родительской сущностью $e' \in V_l$ соответствует какой-либо входящей в вершину e, но не двунаправленной дуге $(e', e) \in E_l$ ($rel(a_i) = (e', e)$). Тип такого атрибута совпадает с типом ключевого атрибута сущности-родителя e'.

Презентационный уровень модели METAS

Опишем граф презентационной модели. Каждая вершина графа презентационной модели G_{pr} – это форма, соответствующая некоторой сущности; дуги между вершинами – возможные переходы между формами (соответствуют дугам в графе логической модели); дуги являются направленными, направление задается от рассматриваемой формы к формам, которые можно вызвать с текущей формы:

$$G_{pr} = (V_{pr}, E_{pr})$$
$$V_{pr} = \{ f_1, f_2, ..., f_n \}, \text{где } n \in N$$
$$E_{pr} = \{ r_1, r_2, ..., r_m \}, \text{где } m \in N$$
$$r_i = (f_j, f_k), \text{где } i = \overline{1,m};\ j,k = \overline{1,n}$$

Здесь пара (f_j, f_k) упорядочена. Это означает, что граф G_{pr} является ориентированным, т.е. существование возможности перехода от вершины f_j к вершине f_k не означает наличия обратного перехода от f_k к f_j.

Вершина-форма $f \in G_{pr}$ соответствует некоторой сущности $e \in G_l$. Эту сущность назовем главной сущностью формы (записываем $ME(f) = e$). Таким образом, получаем:

$$(\forall f \in G_{pr})(\exists e \in G_l : ME(f) = e).$$

Дуги, входящие в вершину-сущность e в графе G_l, т.е. дуги для перехода к родительским сущностям в связи типа «1:M», «0..1:M», «0..1:1», включаются в граф презентационной модели (кроме двунаправленных дуг). Любая дуга в графе G_{pr} соответствует некоторой дуге в графе G_l:

$$(\forall r_i = (f_j, f_k) \in E_{pr})(\exists r_i = (e_k, e_j) \in E_l : ME(f_j) = e_j, ME(f_k) = e_k).$$

Вершина f графа G_{pr} включает в себя набор элементов управления $AttrCtrl(f) = \{ac_1, ..., ac_n\}$, соответствующих атрибутам сущности $e = ME(f)$. Ключевой атрибут не попадает в презентационную модель.

Каждый элемент управления имеет тип, определяемый по типу соответствующего атрибута или родительской связи.

Модель Web-представления

Модель Web-представления описывает пользовательский интерфейс портала, способ отображения в нем информации о сущностях логического уровня. Поэтому она базируется на логической модели METAS. Web-представление основывается также на презентационной модели.

Граф модели Web-представления

Опишем граф модели Web-представления. Вершинами графа являются блоки пользовательского интерфейса портала, а дуги между вершинами – возможные переходы между ними. Блоками пользовательского интерфейса служат Web-страницы главного окна, в котором отображается страница, соответствующая некоторому объекту портала, а также страницы компонентов быстрого доступа, предназначенные для упрощения навигации пользователя по ресурсам портала. Переходы возможны либо от страницы компонента быстрого доступа к странице портала, либо между страницами портала. В случае перехода в главном окне отображается страница соответствующего объекта портала.

Таким образом, граф Web-представления – это пара $G_{web} = (V_{web}, L_{web})$, где

- $V_{web} = \{h_1, h_2, ..., h_n\}, n \in N$ – множество вершин графа. Каждая вершина h_i представляет собой блок пользовательского интерфейса;
- $L_{web} = \{l_1, l_2, ..., l_m\}, m \in N, l_i = (h_j, h_k), i = \overline{1,m}; j,k = \overline{1,n}$ – множество дуг графа. Каждая дуга соответствует возможному переходу между блоками.

Граф G_{web} является ориентированным, поскольку возможность перехода по дуге еще не означает возможности перехода в обратном направлении. Поэтому дуги графа G_{web} являются направленными.

Переход между блоками характеризуется набором параметров $Params(l) = (Com(l), Arg_1, Arg_2, ..., Arg_n), n \in N$. $Com(l)$ задает операцию, которую должна выполнить страница, на которую осуществляется переход. Примерами команд являются стандартные операции просмотра (R), создания (C), редактирования (E) и удаления (D), а также операции, специфичные для сущности, на страницу которой выполняется переход. $Args_i, i = 1..n$ – это параметры, необходимые для выполнения операции над сущностью.

Поскольку в портале должна присутствовать функциональность по редактированию его объектов, нужно предоставлять пользователю возможность открывать формы редактирования.

Определим два подмножества V_{web}^{pr} и V_{web}^{f} множества V_{web}, причем $V_{web} = V_{web}^{pr} \cup V_{web}^{f}, V_{web}^{pr} \cap V_{web}^{f} = \varnothing$.

Подмножество V_{web}^{pr} представляет собой совокупность вершин, предназначенных для отображения Web-страниц портала. Каждая вершина $h \in V_{web}^{pr}$ может отображаться двумя способами в браузере пользователя в режиме просмотра: в виде стандартной страницы, содержимое которой хранится в поле $Content(h)$, или в виде заданной Web-формы $WebForm(h)$.

Подмножество V_{web}^{f} представляет собой совокупность вершин Web-форм

редактирования объектов. Web-формы редактирования являются аналогами форм редактирования презентационной модели с тем лишь отличием, что они используются для отображения в Web-браузере клиента.

Взаимное отображение графов Web-представления и логической модели

Зададим отображения графа Web-представления и графа логической модели. Они необходимы для генерации Web-интерфейса по логической модели.

Одной вершине графа логической модели может соответствовать 2 вершины графа Web-представления. Такая ситуация может возникнуть в случае, когда для объекта существуют и Web-страница для отображения html-контента, и его Web-форма редактирования. Поэтому необходимо задать отображения для получения соответствующих объекту Web-страницы и Web-формы редактирования.

Определим функциональное отображение $MT_{pr} : V_l \to V_{web}^{pr}$ или $MT_{pr}(e_j) = h_i$, где

- $e_j \in V_l$ – вершина графа G_l, представляющая объект логической модели;
- $h_i \in V_{web}^{pr}$ – вершина графа G_{web}, представляющая блок пользовательского интерфейса.

Определим также функциональное отображение $MT_f : V_l \to V_{web}^{f}$ или $MT_f(e_j) = h_k$, где

- $e_j \in V_l$ – вершина графа G_l, представляющая объект логической модели;
- $h_k \in V_{web}^{f}$ – вершина графа G_{web}, представляющая Web-форму редактирования.

Следует отметить, что оба эти отображения не являются сюрьективными. Это означает, что могут существовать объекты, для которых нет соответствующей Web-формы редактирования или соответствующей Web-страницы портала.

Поскольку на Web-страницах портала отображается информация, которая извлекается из объектов портала, необходимо определить связь графа Web-представления с графом логической модели.

Определим функциональное отображение $MH : V_{web} \to V_l$ или $MH(h_i) = e_j$, где

- $h_i \in V_{web}$ – вершина графа G_{web}, представляющая блок пользовательского интерфейса или Web-форму редактирования;
- $e_j \in V_l$ – вершина графа G_l, представляющая объект логической модели.

Данное отображение аналогично функции *ME*, определенной в презентационной модели и задающей соответствие объекта форме редактирования. Отметим, что данное отображение является сюрьективным:

$$(\forall h_i \in V_{web})(\exists e_j \in V_l) : MH(h_i) = e_j .$$

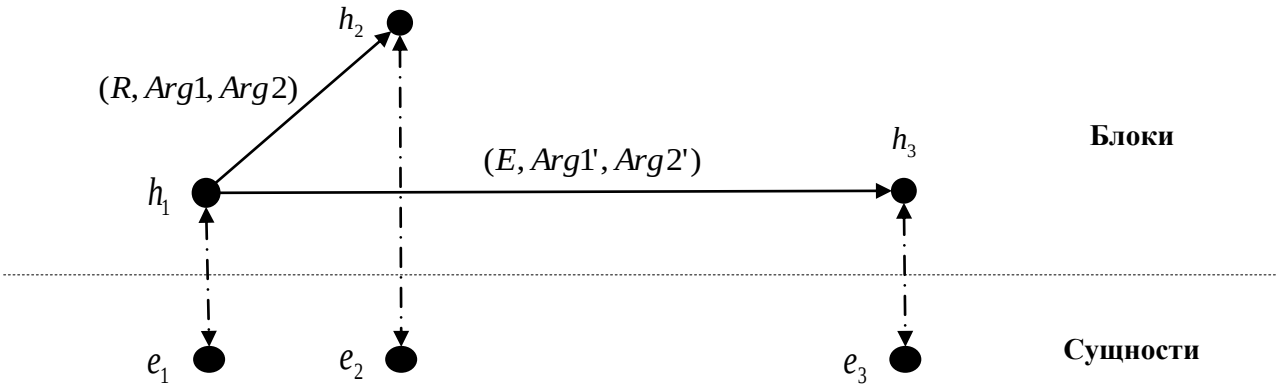

Рис. 2.1. Пример графа модели Web-представления и его связи с графом логической модели

На рис. 2.1 представлен пример отображения вершин графа Web-представления и графа логической модели.

Отображение графа Web-представления на презентационную модель

Web-формы редактирования объектов должны быть аналогичны формам редактирования Windows-интерфейса, поэтому нужно генерировать Web-формы с учетом параметров отображения форм редактирования, которые задаются в презентационной модели. Для этого нужно связать граф Web-представления с графом презентационной модели и описать способ настройки Web-форм редактирования на основе параметров форм редактирования презентационного уровня.

Определим функциональное отображение $MF : V_{web}^{f} \to V_{pr}$ или $MF(h_i) = f_j$, где

- $h_i \in V_{web}^{f}$ – вершина графа G_{web}, представляющая Web-форму редактирования объекта;
- $f_j \in V_{pr}$ – вершина графа G_{pr}, представляющая форму редактирования объекта.

При наличии такого отображения становится возможным извлекать настройки для вычисления расположения атрибутов объекта на Web-форме из презентационной модели.

Аналогично презентационной модели необходимо ввести взаимосвязь между дугами логической модели и дугами модели Web-представления.

Дуги, входящие в вершину-сущность *e* в графе G_l, т.е. дуги для перехода к родительским сущностям в связи типа «1:M», «0..1:M», «0..1:1», включаются в граф модели Web-представления (кроме двунаправленных дуг). Однако, в отличие от графа презентационной модели, не любая дуга в графе G_{web} соответствует некоторой дуге в графе G_l. В данном случае любая дуга между вершинами только подмножества V_{web}^{f} имеет соответствующую дугу в графе G_l:

$$(\forall l_i = (h_j, h_k) \in L_{web} \mid h_j, h_k \in V_{web}^{f})(\exists r_i = (e_k, e_j) \in E_l : MH(h_j) = e_j, MH(h_k) = e_k).$$

Причем операция перехода *Com* в этом случае всегда является операцией чтения:

$$(\forall l_i = (h_j, h_k) \in L_{web} \mid h_j, h_k \in V_{web}^{f}) : Com(l_i) = R$$

Кроме того, следует ввести связь между вершинами подмножеств V_{web}^{pr} и V_{web}^{f}, поскольку пользователю при просмотре любой Web-страницы должна предоставляться возможность редактирования информации об объекте, связанном с данной страницей, при условии наличия у пользователя соответствующих прав на редактирование.

Для этого определим функциональное отображение $MC : V_{web}^{pr} \to V_{web}^{f}$ или $MC(h_i) = h_j$, где

- $h_i \in V_{web}^{pr}$ – вершина графа G_{web}, представляющая Web-страницу объекта;
- $h_j \in V_{web}^{f}$ – вершина графа G_{pr}, представляющая Web-форму редактирования объекта.

Для каждой вершины $h_i \in V_{web}^{pr}$ всегда существует переход к соответствующей вершине $h_j = MC(h_i) \in V_{web}^{f}$, причем операция этого перехода – операция чтения (сначала пользователю предоставляется возможность только просмотра атрибутов объекта):

$$(\forall h_i \in V_{web}^{pr})(\exists h_j \in V_{web}^{f})(\exists l_k = (h_i, h_j) \mid Com(l_k) = R).$$

Кроме того, всегда существует переход в обратном направлении:

$$(\forall h_j \in V_{web}^{f})(\exists h_i \in V_{web}^{pr})(\exists l_k = (h_j, h_i) \mid Com(l_k) = R).$$

На Web-форме редактирования всегда имеется возможность перехода в режим редактирования и обратно в режим просмотра:

$$(\forall h_i \in V_{web}^{f})(\exists l_j = (h_i, h_i) \mid Com(l_j) = E)(\exists l_k = (h_i, h_i) \mid Com(l_k) = R).$$

Портлеты

Портал включает в себя модули (портлеты). Каждый модуль использует определенный набор сущностей и реализует определенный набор функций, выполняемых над этими объектами. Поэтому необходимо знать для каждого модуля, какие сущности необходимы для его корректной работы.

Множество модулей представляется в виде: $M = \{m_1, m_2, ..., m_k\}$, $k \in N$. Для каждого модуля известен набор сущностей, которые необходимы для функционирования модуля (в частности, для хранения параметров модуля и его компонентов быстрого доступа, параметров Web-форм, представления Web-ресурсов): $EM(m) = \{e_1, e_2, ..., e_n\}, m \in M, n \in \overline{1, |V_l|}$. Для каждой сущности модуля хранится способ использования этой сущности. Всего имеется три способа использования сущностей. Первый способ (*I* – от слова *Inner*) используется в задачах, не связанных с непосредственным отображением интерфейса. Второй способ (*P* – от слова *Page*) соответствует использованию сущности в качестве главной сущности для отображения страницы портала. Третий способ (*Q* – от выражения *Quick Access*) используется для хранения настроек компонентов быстрого доступа (рис. 2.2).

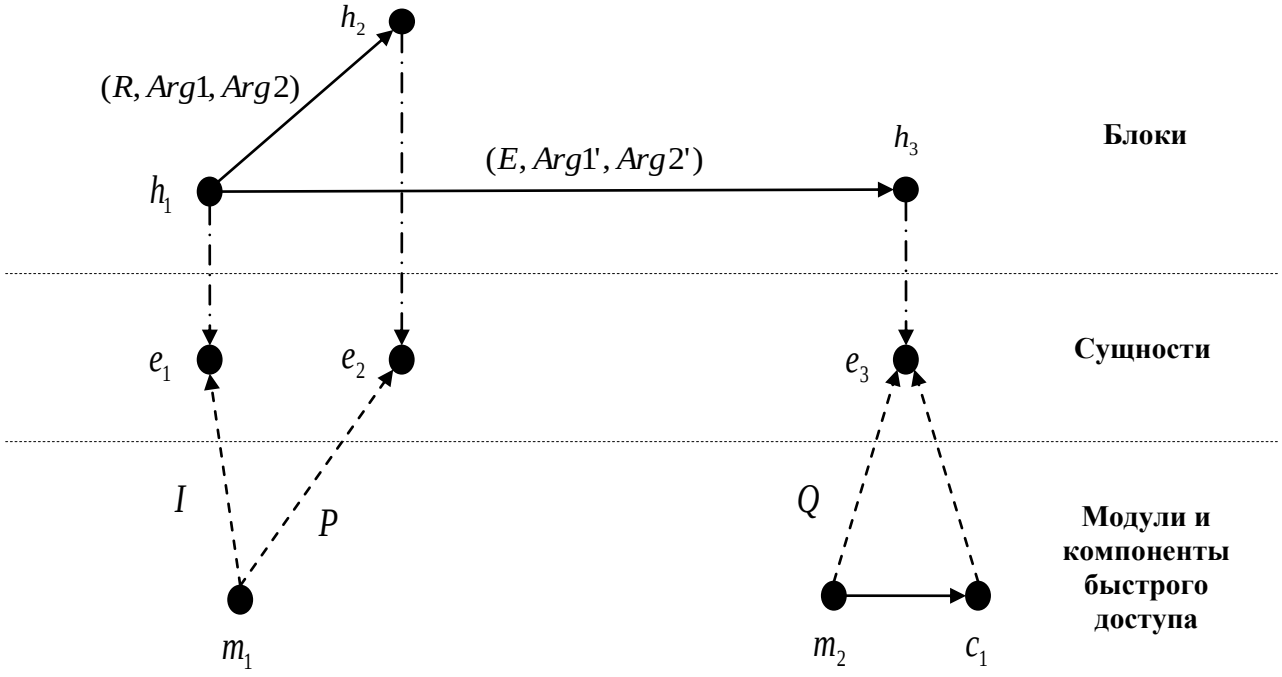

Рис. 2.2. Пример связей между блоками, сущностями и модулями

Определим функциональное отображение $KE: V_l \times M \to \{I, P, Q\}$ или $KE(e_i, m) = k_j$, где

– $e_i \in V_l$ – вершина графа G_l, представляющая сущность портала;

– $m \in M$ – модуль, в котором используется сущность e_i;

– $k_j \in \{I, P, Q\}$ – способ использования сущности e_i модулем m.

Определим свойства этого отображения.

Свойство 1. Для каждой сущности, используемой в модуле, задан способ ее использования:

$$(\forall m \in M)(\forall e \in V_l \mid e \in EM(m))(\exists KE(e,m)) \quad (1)$$

Свойство 2. Сущности, которые используются в задачах, не связанных с непосредственным отображением интерфейса, или используются для хранения настроек компонентов быстрого доступа, не имеют связанных с ними Web-страниц:

$$(\forall m \in M)(\forall e \in V_l \mid KE(e,m) = I \vee KE(e,m) = Q)(\neg\exists h \in V_{web}^{pr} \mid MT_{pr}(e) = h) \quad (2)$$

Свойство 3. Каждая сущность, которая используется в качестве главной сущности, имеет связанную с ней Web-страницу:

$$(\forall m \in M)(\forall e \in V_l \mid KE(e,m) = P)(\exists h \in V_{web}^{pr} \mid MT_{pr}(e) = h) \quad (3)$$

Свойство 4. Каждая сущность, которая используется в качестве главной сущности или используется для хранения настроек компонентов быстрого доступа, имеет связанную с ней Web-форму редактирования:

$$(\forall m \in M)(\forall e \in V_l \mid KE(e,m) = P \vee KE(e,m) = Q)(\exists h \in V_{web}^{f} \mid MT_{f}(e) = h) \quad (4)$$

Обозначим множество компонентов быстрого доступа модуля как $C(m) = \{c_1, c_2, ..., c_n\}, m \in M, n \in N$, где n – количество компонентов модуля. Каждому компоненту c соответствует некоторая сущность e для хранения его настроек по умолчанию. Назовем такую сущность e главной сущностью компонента c.

Определим функциональное отображение $MC : C \to V_l$ как функцию $MC(c_i) = e_j$, где

– $c \in C(m), m \in M$ – компонент модуля m портала;

– $e_j \in V_l$ – вершина графа G_l, представляющая главную сущность компонента c.

Свойство 5. Главная сущность каждого компонента быстрого доступа используется для хранения его настроек по умолчанию (способ использования равен Q):

$$(\forall m \in M)(\forall c \in C(m)) : KE(MC(c), m) = Q \quad (5)$$

Ключевые слова

В данной работе логическая модель расширяется множеством ключевых слов сущностей.

Определим множество ключевых слов портала $W = \{w_1, w_2, ..., w_n\}, n \in N$.

Для реализации поиска информации и определения релевантности одних сущностей портала другим для каждой сущности хранится набор ключевых слов и выражений, описывающих информацию, которая хранится в этой сущности.

Определим функциональное отображение $KeyWords : V_l \to WE$, задающее множества ключевых слов для сущностей, или $KeyWords(e_i) = we_j$, где

- $WE = \{(we_{i1}, we_{i2}, ..., we_{in}) \mid i_1, i_2, .., i_n \in \overline{1, |W|}, n \in N\}$ – множество наборов ключевых слов;
- $e_i \in V_l$ – вершина графа G_l, представляющая сущность портала;
- $we_j \in WE$ – набор ключевых слов сущности e_i.

Для определения степени релевантности одних объектов другим и ускорения поиска необходимо также хранить для каждого ключевого слова и выражения набор сущностей, для которых данное слово или выражение является ключевым.

Определим функциональное отображение $KeyWordEntities : W \to VE$, задающее множества сущностей для ключевых слов, или $KeyWordEntities(w_i) = ve_j$, где

- $VE = \{(ve_{i1}, ve_{i2}, ..., ve_{in}) \mid i_1, i_2, .., i_n \in \overline{1, |V_l|}, n \in N\}$ – множество наборов сущностей;
- $w_i \in W$ – ключевое слово;
- $ve_j \in VE$ – набор сущностей, которым соответствует ключевое слово *w*.

Определим свойства этих отображений.

Свойство 1. Каждая сущность попадает в набор сущностей каждого своего ключевого слова:

$$(\forall e \in V_l)(\forall w \in KeyWords(e)) : e \in KeyWordEntities(w) \quad (6)$$

Свойство 2. Каждое ключевое слово принадлежит наборам ключевых слов сущностей своего набора сущностей:

$$(\forall w \in W)(\forall e \in KeyWordEntities(w)) : w \in KeyWords(e) \quad (7)$$

Данный способ позволяет связать сущности между собой неявно. Это означает, что релевантными в рамках такого подхода являются сущности, которые имеют общие ключевые слова. Однако возможны ситуации, когда

имеются сущности, связанные с данной сущностью непосредственно. При этом релевантные сущности могут не иметь общих ключевых слов. В этом случае нужно задавать связи релевантности явно.

С этой целью вводится множество связей между сущностями, которые определяют релевантность. Обозначим его как $E_{rel} = \{z_1,...,z_p\}, p \in N,\ z_i = (e_j, e_k),\ i = \overline{1,p},\ j,k = \overline{1,n}$ (рис. 2.3). Релевантность одних сущностей другим устанавливается создающим эти сущности пользователем, который задает таким образом связи релевантности между ними.

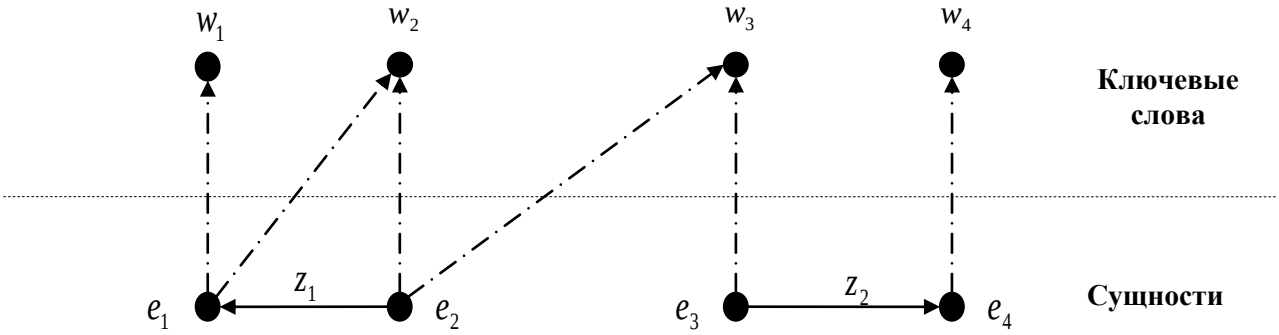

Рис. 2.3. Пример сущностей, их ключевых слов и связей между ним

Таким образом, при помощи явных и неявных (через ключевые слова) связей между сущностями реализуется поиск релевантных объектов и поиск объектов вообще.

Модель защиты

Модель защиты включает в себя три составляющие: объекты защиты, субъекты защиты и права на доступ субъектов к объектам.

Объект защиты – это находящаяся в ИС совокупность данных, которая может содержать подлежащие защите сведения. В качестве защищаемого объекта в модели защиты можно рассматривать данные на разных уровнях:

- атрибутов записей таблицы базы данных,
- таблицы базы данных,
- объектов портала,
- страниц портала.

Защита на уровне атрибутов записей представляется наиболее гибкой, однако это приведет к чрезмерному количеству объектов, что усложнит администрирование системы. Защита на уровне таблиц не даст возможности контролировать доступ на уровне конкретных объектов. Защита на уровне страниц портала невозможна, поскольку многие страницы генерируются динамически. Поэтому выберем защиту на уровне *объектов портала*.

Субъектами защиты являются пользователи портала. Однако для

упрощения администрирования вводят еще один вид субъектов – группы пользователей. Задание прав доступа для групп пользователей позволяет избежать многократного задания одинаковых прав на одни и те же объекты разным пользователям.

Права доступа – это совокупность правил, регламентирующих порядок и условия доступа субъектов к объектам защиты. Права доступа определяют набор действий (например, чтение, запись, выполнение), разрешенных для выполнения субъектам над объектами данных. В качестве операций, для которых требуется контроль доступа, выберем

- чтение,
- создание,
- изменение,
- удаление,
- передача прав редактирования.

При этом разделим права на две группы: права на отдельные объекты (чтение, редактирование, удаление) и права на типы объектов (создание, доступ, передача прав редактирования).

Обозначения

Используем следующие обозначения:

A – конечный алфавит;

A^* – множество слов конечной длины в алфавите A;

S – множество субъектов, $S = R \cup U$;

O – множество объектов;

T – множество типов объектов;

R – множество групп, $R \subseteq S$;

U – множество пользователей, $U \subseteq S$;

P – множество видов прав доступа к конкретному объекту, $P = \{read, edit, delete\}$;

TP – множество видов прав доступа к типу объектов, $TP = \{access, create, transfer\}$;

rights: $S \times O \to 2^{|P|}$ – функция, возвращающая для пары (s, o) в качестве результата множество прав доступа субъекта $s \in S$ к объекту $o \in O$;

typerights: $S \times T \to 2^{|TP|}$ – функция, возвращающая для пары (s, t) в качестве результата множество прав, определяющих, может ли субъект $s \in S$ получать доступ, создавать объекты данного типа $t \in T$ и передавать права на редактирование объектов этого типа другим пользователям и группам;

allrights: $U \times O \to 2^{|P|}$ – функция, возвращающая для пары (u, o) в качестве результата множество суммарных прав доступа пользователя $u \in U$ к объекту $o \in O$;

alltyperights: $U \times T \to 2^{|TP|}$ – функция, возвращающая для пары (*u, t*) в качестве результата множество прав, определяющих, может ли пользователь $u \in U$ с учетом вхождения в группы получать доступ, создавать объекты данного типа $t \in T$ и передавать права на редактирование объектов этого типа другим пользователям и группам;

groups: $U \to 2^R$ – функция, возвращающая для каждого пользователя в качестве результата множество групп, членом которых он является;

owners: $O \to 2^U$ – функция, возвращающая для каждого объекта в качестве результата множество пользователей, являющихся его владельцами.

name: $S \to A^*$ – функция, возвращающая для каждого субъекта в качестве результата его имя.

public: $T \to \{true, false\}$ – функция, возвращающая в качестве результата истинностное значение, показывающее, являются ли объекты этого типа общедоступными.

Права субъектов

Формализуем основные свойства субъектов в рамках математической модели защиты.

Свойство 1. У каждого субъекта имеется имя:

$$(\forall s \in S)(\exists a \in A^* \mid name(s) = a) \tag{8}$$

Свойство 2. Имена у субъектов не могут повторяться:

$$(\forall s1, s2 \in S)(\neg \exists a \in A^* \mid name(s1) = a \,\&\, name(s2) = a) \tag{9}$$

Каждый пользователь может входить в несколько групп. Вхождение пользователя в группы определяется функцией groups.

Свойство 3. Существуют, по крайней мере, две группы – администраторы и гости:

$$(\exists r1, r2 \in R \mid name(r1) = "guests" \& name(r2) = "ad\ \min istrators") \tag{10}$$

Другие группы добавляются при помощи Windows-приложения.

В системе различаются права на типы объектов и права на объекты.

Права на типы задают возможность доступа к объектам определенного типа и их создания. Права на типы можно получить при помощи функции *typerights*.

Права на объекты включают права на чтение, редактирование и удаление конкретного объекта. Права на объекты задаются для субъектов.

Свойство 4. У каждого объекта всегда есть владелец:

$$(\forall o \in O)(\exists u \in U \mid u \in owners(o)) \tag{11}$$

Свойство 5. Владельцами объекта могут быть признаны только создатель объекта и администраторы:

$$(\forall o \in O)(\neg\exists u1, u2 \in U \mid u1 \in owners(o) \,\&\, u2 \in owners(o) \,\&\, (\neg\exists r \in R \mid name(r) = "admin istrators" \,\&\, (r \in groups(u1) \vee r \in groups(u2)))) \quad (12)$$

Свойство 6. Владельцы объекта обладают правами чтения, редактирования и удаления этого объекта:

$$(\forall u \in U)(\forall o \in O \mid u \in owners(o)) : \{read, edit, delete\} \subseteq rights(u, o) \quad (13)$$

Свойство 7. Помимо владельцев объекта правами на чтение, редактирование и удаление объекта могут обладать и другие пользователи. Это возможно в случае, когда владелец объекта задает соответствующие права для других пользователей. Другими словами, если пользователь, не являющийся владельцем объекта, имеет право на чтение, редактирование или удаление, то владелец объекта имеет право на передачу прав редактирования:

$$(\forall o \in O)(t \in T \mid t = Type(o))(\forall u1 \in U \mid u1 \notin owners(o) \,\&\, rights(u1, o) \cap P \neq \varnothing)$$
$$(\forall u2 \in U \mid u \in owners(o)) : transfer \in typerights(u2, t) \quad (14)$$

Свойство 8. К группе гостей принадлежат как зарегистрированные пользователи системы, так и незарегистрированные. Вхождение незарегистрированных пользователей в группу «Гости» обусловлено необходимостью отображения некоторой общей для всех пользователей информации (объектов), которая должна отображаться при первом открытии страницы WebMETAS:

$$(\forall u \in U)(\exists r \in R \mid name(r) = "guests" \,\&\, r \in groups(u)) \quad (15)$$

Свойство 9. Пользователи группы «Гости», не являющиеся членами других групп, имеют доступ только к общей информации. При этом после создания какого-либо объекта гость не имеет возможности его дальнейшего редактирования и удаления:

$$(r \in R \mid name(r) = "guests"))(\forall t \in T \mid public(t) = true) : access \in typerights(r, t) \quad (16)$$

Свойство 10. Группа «Администраторы» предоставляет всем пользователям этой группы иметь права на просмотр и создание объектов всех типов, поскольку каждый администратор должен иметь возможность управления содержимым системы. Другими словами, все пользователи группы «Администраторы» принадлежат ко всем группам:

$$(r \in R \mid name(r) = "admin istrators")(\forall t \in T) : \{access, create\} \subseteq typerights(r, t)) \quad (17)$$

Свойство 11. Группа «Администраторы» предоставляет всем пользователям этой группы иметь права на чтение, редактирование и удаление всех объектов, поскольку каждый администратор должен иметь возможность управления содержимым системы. Т.е. можно сказать, что все пользователи группы

«Администраторы» являются владельцами всех объектов:

$$(\forall u \in U \mid (\exists r \in R \mid name(r) = "admin istrators" \& r \in groups(u)))$$
$$(\forall o \in O)(u \in owners(o)) \quad (18)$$

Свойство 12. Права каждого пользователя на доступ к конкретным объектам складываются из назначенных ему собственных прав и из прав всех групп, в которые он входит. При этом пользователь имеет право на чтение, редактирование и удаление объектов, в случае, если он либо имеет соответствующее собственное право, либо подобное право имеет хотя бы одна из групп, в которые он входит:

$$(\forall o \in O)(\forall u \in U)(\forall r_i \in groups(u) \mid i = \overline{1,n}, n = \mid groups(u) \mid):$$
$$allrights(u,o) = rights(u,o) \vee rights(r_1,o) \vee ... \vee rights(r_n,o) \quad (19)$$

Свойство 13. Права каждого пользователя на доступ к объектам определенных типов складываются из назначенных ему собственных прав и из прав всех групп, в которые он входит. Пользователь имеет право на доступ и создание объектов соответствующего типа, в случае, если он либо имеет соответствующее собственное право, либо подобное право имеет хотя бы одна из групп, в которые он входит:

$$(\forall t \in T)(\forall u \in U)(\forall r_i \in groups(u) \mid i = \overline{1,n}, n = \mid groups(u) \mid):$$
$$alltyperig hts(u,t) = typerights(u,t) \vee typerights(r_1,t) \vee ... \vee typerights(r_n,t) \quad (20)$$

Замечание. При реализации можно представлять значения функций *rights*, *allrights*, *typerights* и *alltyperights* в виде битовых строк, где каждому праву (*read*, *edit*, *delete* – для объектов; *access*, *create* – для типов объектов) ставится в соответствие константа (0 – нет права, 1 – есть право) в соответствующей этому праву позиции в строке. Выполнение операция объединения прав (вычисления значений функций *allrights* и *alltyperights*) сводится к побитовому сложению (операция ИЛИ) битовых строк, представляющих права пользователя и права всех групп, в которые он входит.

Граф прав доступа

В качестве математической модели защиты выберем граф прав доступа. Альтернативными моделями могли бы быть дискреционная (матрица прав доступа) или мандатная модель защиты. Однако им присущи некоторые существенные недостатки.

К минусам *дискреционной модели* относятся:

- статичность определенных в ней правил разграничения доступа;
- отсутствие механизмов слежения за безопасностью потоков информации;

– быстрый рост сложности администрирования при увеличении количества пользователей или объектов.

К минусам *мандатной модели* можно отнести:

– реализация сложна и требует значительных ресурсов системы;
– сложности с изменением категории пользователя и соответствующих прав на объекты;
– несоответствие классификации по уровням конфиденциальности характеру хранимой и обрабатываемой информации в коммерческих организациях.

Используя же *графовую модель* (расширение ролевой модели), можно значительно сократить по сравнению с другими моделями сложность создания и администрирования системы защиты, обеспечив при этом необходимую гибкость назначения прав и контроля доступа субъектов к объектам на их основе. Гибкость можно обеспечить за счет введения специфических типов вершин и дуг графа и реализации на графе определенных алгоритмов защиты (например, наследования прав доступа [42, 43]).

Будем рассматривать ориентированный помеченный граф $G_{acc} = (V_{acc}, E_{acc})$, где

V_{acc} – множество вершин графа, $V_{acc} = S \cup O \cup T$;

E_{acc} – множество дуг графа.

Вершину, задающую объект системы, обозначим *object-vertice*, вершину, представляющую пользователя или группу, – *subject-vertice*, а вершину, представляющую тип объекта, – *type-vertice*.

На графе будем представлять объекты символом ●, субъекты – символом ■, типы объектов – символом ◆.

В графе определим *четыре вида дуг*:

– субъектная дуга;
– дуга права на доступ к типу;
– дуга права на доступ к объекту;
– дуга владения.

Субъектная дуга соединяет две вершины-субъекта, соответствующих пользователю и группе (см. рис. 2.4). Этот вид дуги задает членство пользователя в группе. Дуга направлена от вершины $u_i \in U$, представляющей пользователя, к вершине $r_i \in R$, представляющей группу. Дуга помечается символом *m*.

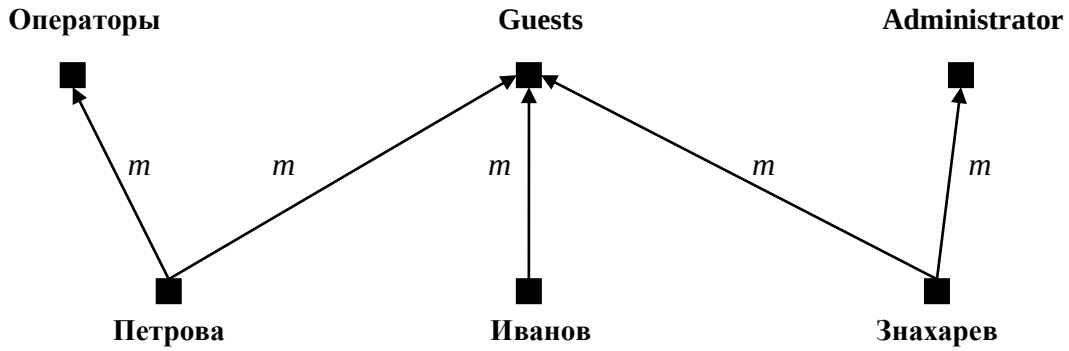

Рис. 2.4. Граф с субъектными дугами

Дуга *права на доступ к типу* соединяет две вершины, одна из которых является субъектом, а другая – типом объекта (рис. 2.5). Этот вид дуги задает права доступа и создания субъекта к объектам определенного типа. У дуги может быть две пометки: *a* (право на доступ) и *c* (право на создание). Дуги, у которых отсутствуют обе пометки, на графе не представляются. Дуга направлена от субъекта к типу.

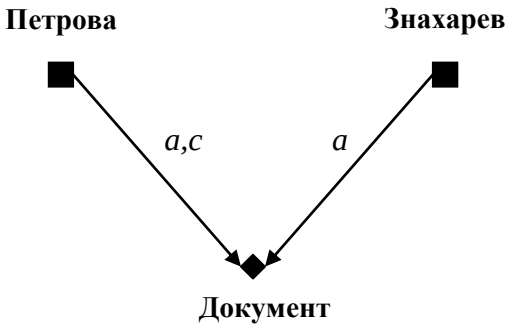

Рис. 2.5. Граф с дугами прав на доступ к типу

Дуга права на доступ к объекту соединяет две вершины, одна из которых является субъектом, а другая – объектом (рис. 2.6). Этот вид дуги задает права чтения, редактирования или удаления субъектом объекта. У дуги может быть три пометки: *r* (право на чтение), *e* (право на редактирование) и *d* (право на удаление). Дуги, у которых отсутствуют пометки, на графе не представляются. Дуга направлена от субъекта к объекту.

Дуга владения соединяет вершины, представляющие пользователя и объект (рис. 2.7). Дуга задает право пользователя владения объектом. Дуга помечается символом *O*. Дуга направлена от пользователя к объекту.

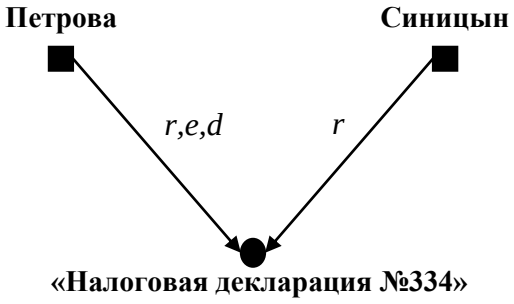

Рис. 2.6. Граф с дугами прав на доступ к объекту

Рис. 2.7. Граф с дугами владения

Модель пользователя

Данная модель необходима для предоставления пользователю возможности настраивать интерфейс в соответствии со своими нуждами, а также для того, чтобы система могла адаптировать интерфейс под текущего пользователя с целью повышения эффективности и удобства его работы с системой.

Адаптация интерфейса заключается в следующем:

- адаптация представления, а именно рабочего места пользователя (выбор необходимых модулей и задание расположения компонентов быстрого доступа этих модулей). Настройку рабочего места производит сам пользователь;
- адаптация навигации (предоставление пользователю возможности быстрой навигации к ресурсам, которые могут ему понадобиться). Настройка навигации производится системой.

Рассмотрим представление в модели пользователя настроек первой категории.

Представим набор выбранных пользователем модулей системы как множество $M'(u) \subseteq M$, где M – множество всех доступных модулей системы. Для каждого модуля m известен список компонентов быстрого доступа $C(m)$. Для каждого такого компонента $c \in C(m)$ хранится специальная сущность $e = MC(c)$, хранящая настройки этого компонента по умолчанию. При регистрации пользователя эта сущность копируется в модель пользователя для того, чтобы пользователь мог поменять эти настройки. В работе портала эта сущность будет использоваться вместо сущности-оригинала. Тогда можно определить функцию $Settings(u, c) \in V_l$, которая для данного пользователя и данного компонента возвращает сущность, хранимую в модели пользователя и содержащую настройки пользователя этого компонента.

Теперь рассмотрим настройки навигации.

В модели пользователя хранятся последняя посещенная страница портала h_{last}. Эта страница будет отображаться пользователю при очередном входе в портал. Определим функцию получения последней посещенной страницы пользователя $LastPage(u) \in V_{web}$.

Помимо последней посещенной страницы ведется список наиболее часто используемых страниц $H_{freq}(u) = \{(h_{i1}, fr_{i1}), (h_{i2}, fr_{i2}), (h_{ik}, fr_{ik}) \mid i1, ..., ik = \overline{1,n}\}$, где n – количество страниц портала, fr – число переходов на соответствующую страницу, осуществленных пользователем. Этот список используется модулем часто используемых ресурсов для предоставления быстрого доступа к тем объектам портала, к которым пользователь чаще всего обращался и для которых высока вероятность повторного обращения к ним пользователем. Обозначим функцию получения значения частоты использования пользователем *u* страницы *h* за *Freq*(*u*, *h*).

Поскольку представление блоков интерфейса для компонентов быстрого доступа зависит от текущего состояния модели пользователя и от настроек компонентов, то необходимо описать генерацию этих блоков на основе модели и настроек.

Определим функциональное отображение $Block : U \times C \to V_{web}$ или $Block(u_i, c_j) = h_k$, где

- $u_i \in U$ – пользователь, для которого генерируется Web-представление компонента c_j;
- $c_j \in C(m), m \in M'(u)$ – компонент быстрого доступа модуля *m*, Web-представление которого генерируется для пользователя;
- $h_k \in V_{web}$ – блок пользовательского интерфейса компонента c_j, настроенный в соответствии с моделью пользователя u_i.

Эта функция позволяет получить Web-представление компонента быстрого доступа для пользователя на основе настроек этого компонента и текущего состояния модели пользователя.

Следует отметить, что генерация Web-представления для компонентов быстрого доступа была бы неэффективной, если бы производилась при каждом обращении пользователя к ресурсам портала. Это объясняется тем, что после достаточно длительного периода работы в портале изменения в модели пользователя становятся крайне редкими (для модуля часто используемых ресурсов становится устоявшейся статистика частоты обращения к ресурсам портала, модуль поиска вообще не зависит от модели пользователя и не меняется, если сохраняются одни и те же значения его настроек). Поэтому необходимо ввести кэширование блоков интерфейса для компонентов быстрого доступа.

Определим функциональное отображение $BlockChanged : U \times C \to \{false, true\}$ или $BlockChanged(u_i, c_j) \in \{true, false\}$, где

- $u_i \in U$ – пользователь, для которого генерируется Web-представление компонента c_j;
- $c_j \in C(m), m \in M'(u)$ – компонент быстрого доступа модуля *m*, Web-представление которого генерируется для пользователя;
- *true, false* – значения функции. Значение *true* означает, что Web-представление компонента поменялось, значение *false* означает, что оно не поменялось.

Эта функция позволяет определить, поменялось ли Web-представление компонента. Способ вычисления значения функции определяется конкретным модулем портала. Если Web-представление поменялось, то получить блок интерфейса для компонента можно при помощи функции *Block*. В случае же, если Web-представление не поменялось, нужно иметь функцию, позволяющую получить кэшированное представление для компонента.

Для этого определим функциональное отображение $CachedBlock : U \times C \to V_{web}$ или $CachedBlock(u_i, c_j) = h_k$, где

- $u_i \in U$ – пользователь, для которого генерируется Web-представление компонента c_j;
- $c_j \in C(m), m \in M'(u)$ – компонент быстрого доступа модуля *m*, Web-представление которого генерируется для пользователя;
- $h_k \in V_{web}$ – кэшированный блок пользовательского интерфейса компонента c_j, настроенный в соответствии с моделью пользователя u_i.

Определим теперь свойства этих отображений.

Свойство 1: Если пользователь еще ни разу не обратился к странице портала с расположенным на ней компонентом быстрого доступа, то для этого компонента нет кэшированного блока интерфейса:

$$(\forall u \in U)(\forall m \in M'(u))(\forall c \in C(m) \mid Freq(u, Block(u,c)) = 0)(\neg\exists CachedBlock(u,c)) \quad (21)$$

Свойство 2: Если пользователь еще ни разу не обратился к странице портала с расположенным на ней компонентом быстрого доступа, то значение функции *BlockChanged* равно *false*:

$$\begin{aligned}&(\forall u \in U)(\forall m \in M'(u))(\forall c \in C(m) \mid Freq(u, Block(u,c)) = 0):\\ &BlockChanged(u,c) = false\end{aligned} \quad (22)$$

Свойство 3: Если нет изменений в модели пользователя, связанных с компонентом быстрого доступа, используемым этим пользователем, то блок пользовательского интерфейса совпадает со своей кэшированной версией:

$$\begin{aligned}&(\forall u \in U)(\forall m \in M'(u))(\forall c \in C(m) \mid BlockChanged(u,c) = false):\\ &Block(u,c) = CachedBlock(u,c)\end{aligned} \quad (23)$$

Алгоритмы модели Web-представления портала

Опишем наиболее важные алгоритмы, выполняющиеся в портале, в терминах введенной математической модели.

Алгоритм 1. *Переход по дуге графа Web-представления*

Данный алгоритм применяется при переходе по ссылке с одной страницы на другую для контроля прав доступа к странице, изменения модели пользователя и настройке компонентов быстрого доступа на основе новой страницы и текущего состояния модели пользователя.

Задача: перейти по дуге $l = (h_1, h_2)$, $Params(l) = (Com, Arg_1, Arg_2, ..., Arg_n)$ при текущем пользователе *u*.

Шаги:

1. Получить сущность $e_2 = MH(h_2)$.
2. Проверить права пользователя на выполнение операции *Com*.
   a. Если *Com = C*, то проверить, есть ли у пользователя права на создание сущностей типа $Type(e_2)$, т.е. выполняется ли условие $(Type(e_2), create) \in alltyperights(u)$.
   b. Если *Com = R*, *Com = E* или *Com = D*, то проверить, есть ли у пользователя соответствующее право на доступ к сущности e_2, т.е. выполняется ли условие: $(e_2, Com) \in allrights(u)$.
   c. Если условие выполняется, то перейти на *шаг* 3, иначе сообщить пользователю о невозможности обращения к сущности.

3. Выполнить команду *Com* с параметрами $Arg_1, Arg_2, ..., Arg_n$.
4. Сохранить страницу h_2 как последнюю страницу пользователя $LastPage(u)$.
5. Обновить статистику посещения этой страницы пользователем: $fr_2 = fr_2 + 1$ для $(h_2, fr_2) \in H_{freq}(u)$, если она уже занесена в $H_{freq}(u)$, иначе добавить $(h_2, 1)$ в $H_{freq}(u)$.
6. Настроить компоненты быстрого доступа.
   a. Получить список отображаемых пользователем модулей $M'(u)$.
   b. Для каждого модуля $m \in M'(u)$ выполнить *шаг* 6.*c*.
   c. Все компоненты $c \in C(m)$ обновить в соответствии с настройками $Settings(u, c)$ и текущей страницей h_2.
   d. Обновить значение кэша блоков для тех компонентов, для которых изменения в модели пользователя и настройках привели к изменению Web-представления.
   e. Все компоненты $c \in C(m)$ расположить на странице и обновить в соответствии с настройками $Settings(u, c)$ и текущей страницей h_2.
7. Настроить главное окно Web-страницы. Если для h_2 не существует $WebForm(h_2)$, то поместить в окно содержимое $Content(h_2)$, иначе поместить в окно Web-форму $WebForm(h_2)$.
8. Отобразить Web-страницу пользователю.

Псевдокод:

```
WebPage PageTransition( l = (h1, h2) ∈ E_web , u ∈ U )
Begin
    e2 ← MH(h2) ; // Шаг 1
    HasRights ← CheckRights(u, e2, Com) ; // Шаги 2.a-2.c
    If HasRights = False then
        Return ErrorPage  // Шаг 2.c
    Else Begin
            Execute(Com, Params(l)); // Шаг 3
            LastPage(u) ← h2 ; // Шаг 4
            If  Freq(u, h2) = Nil then // Шаг 5
                H_freq(u) ← H_freq(u) ∪ (h2, 1)
            Else
                Freq(u, h2) ← Freq(u, h2) + 1;
            For Each m ∈ M'(u) Do // Шаг 6.a
                For Each c ∈ C(m) Do
                Begin
                    UpdateModule(c, Settings(u, c), h2); // Шаг 6.c
```

```
                    If BlockChanged(u, c) then
                        CachedBlock(u,c) ← Block(u,c); // Шаг 6.d
                    Place(CachedBlock(u, c), h2);        // Шаг 6.e
                End;
                If WebForm(h2) = Nil then  // Шаг 7
                    Return Content(h2)
                Else
                    Return WebForm(h2);
            End;
        End;

        Function CheckRights(u ∈ U , e ∈ V1 , Com)
        Begin
            HasRights ← False ;
            If  Com = C  And  create ∈ alltyperig hts(u,Type(e))
            then  HasRights ← True ; // Шаг 2.b
            If  (Com = R Or Com = E  Or  Com = D) And  Com ∈ allrights(u,e)
            then  HasRights ← True ; // Шаг 2.c
            Return HasRights;
        End
```

***Анализ сложности*:** Пусть $N = |\mathrm{M}|$ – количество модулей портала, $K = \sum_{m \in M} |C(m)|$ – общее количество компонентов быстрого доступа. В худшем случае пользователь использует все доступные модули. Для простоты будем считать, что все вызываемые функции, не описанные в приведенном псевдокоде, имеют сложность $O(1)$. Функция *CheckRights* имеет сложность $O(1)$, поскольку проверка наличия прав у пользователя на тип и на объект будет сводиться к сложности обращения к конкретной записи таблицы базы данных. Так как таблицы прав проиндексированы по обоим полям (одно для пользователя, а другое для типа в одном случае и для объекта в другом), то сложность операции будет значительно меньше остальных действий в процедуре. Все остальные действий процедуры CheckRights имеют сложность $O(1)$. В функции *PageTransition* в худшем случае просматриваются все модули портала и все компоненты быстрого доступа. Тогда сложность алгоритма составляет $O(N + K)$.

***Оптимизация*:** не требуется.

Алгоритм 2. *Экспорт модуля из одной системы в другую*

Данный алгоритм применяется при экспорте модуля из одного портала в другой. Экспорт может быть необходим администратору для того, чтобы

перенести функциональность модуля между порталами, созданными на основе одной технологии. При этом выполняется проверка возможности переноса и копирование всех сущностей и связей между ними, необходимых для модуля (подграфов логической, презентационной моделей и модели Web-представления), в случае, если перенос можно осуществить.

Задача: перенести модуль $m \in M$ из портала P_1 в портал P_2.

Шаги:

1. Зарегистрировать модуль *m* в портале P_2 и перенести все настройки модуля в P_2.
2. Перенести используемые модулем сущности. Для каждой сущности *e* $e \in EM(m)$ выполнить *шаги* 1.a-1g.
   a. Если типа сущности нет в портале P_2, то добавить тип и его поля в модель логического уровня.
   b. Если тип сущности есть в портале P_2, то проверить, есть ли у этого типа все поля типа $Type(e)$ (обозначим множество всех полей типа *t* за $F(t) = \{f\}$).
   c. Если каких-то полей не хватает, то выдать пользователю сообщение о невозможности переноса модуля и завершить алгоритм. Если все поля присутствуют, то продолжить выполнение алгоритма.
   d. Если сущности нет в портале P_2, то добавить ее в модель логического уровня и добавить значение способа использования сущности в модель Web-представления для модуля *m*.
   e. Если сущность есть в портале P_2 (назовем ее *e'*), то выполнить копирование всех значений полей из *e* в *e'* и обновить способ использования сущности в модели Web-представления для модуля *m*.
   f. Перенести все настройки вершины презентационной модели, соответствующей сущности *e*.
   g. Если сущность e используется для отображения страниц или для отображения компонентов быстрого доступа ($KE(e) \in \{Q, P\}$) и имеется Web-форма $WebForm(e)$, то перенести в портал форму $WebForm(e)$.
3. Перенести связи между типами сущностей, которых не хватает в портале P_2 для модуля *m*.
4. Перенести структуру вершин дерева объектов для модуля *m* из портала P_1 в портал P_2.

Псевдокод:

```
Procedure CopyModule( m ∈ M(P1) , P1 , P2 );
Begin
    RegisterModule(m, P2 ); // Шаг 1
```

CopySettings(m, P_2);

For Each $e \in EM(m)$ Do

Begin

$AllFieldsExist \leftarrow True$

If $\neg\exists t \in T(P_2) \mid Equal(Type(e), t)$ then // Шаг 2.a

Begin

t' = CopyType(t);

CopyTypeFields(t, t');

$T(P_2) \leftarrow T(P_2) \cup \{t'\}$;

End

Else

Begin

$AllFieldsExist \leftarrow True$;

For Each $f \in F(Type(e))$ Do // Шаг 2.b

If $\neg\exists f' \in F(t) \mid Equal(f', f)$ then

Begin

ShowError("Поля" +f +" нет в типе"+ t);

// Шаг 2.c

$AllFieldsExist \leftarrow False$;

End;

End;

If AllFieldsExist = True then

Begin

If $\neg\exists e' \in V_l(P_2) \mid Equal(e, e')$ then

Begin

e' = CopyEntity(e); // Шаг 2.d

$V_l(P_2) \leftarrow V_l(P_2) \cup \{e'\}$;

End

Else CopyFields(e, e'); // Шаг 2.e

$k(e') \leftarrow k(e)$;

CopyPresentSettings(e, e'); // Шаг 2.f

If $KE(e) \in \{Q, P\}$ & $WebForm(e)$ <> Nil

then $WebForm(e') \leftarrow WebForm(e)$; // Шаг 2.g

End;

End;

CopyLinks(P_1, P_2, m); // Шаг 3

CopyTreeStruct(P_1, P_2, m); // Шаг 4

End;

***Анализ сложности*:** Пусть $N = |EM(m)|$ – количество используемых в модуле сущностей, $F = \sum_{e \in EM(m)} |Attr(e)|$ – суммарное количество полей этих сущностей, $E = |\{r_k \mid r_k = (e_i, e_j) \in E_l; e_i, e_j \in EM(m)\}|$ – количество связей между сущностями, T – количество вершин в поддереве дерева объектов, соответствующем данному модулю. В худшем случае не возникает ошибок при копировании сущностей. Тогда в функции *CopyModule* просматриваются все сущности модуля, и для каждой сущности выполняется сначала проверка всех полей, а затем копирование всех полей. Функция *CopyLinks* имеет сложность $O(E)$, функция *CopyTreeStruct* имеет сложность $O(T)$. Для простоты будем считать, что все остальные вызываемые функции, не описанные в приведенном псевдокоде, имеют сложность $O(1)$. Тогда сложность алгоритма составляет $O(N + F + E + T)$.

***Оптимизация*:** Возможна в шагах 2.a-2.c. Можно проверять не для каждой сущности соответствие полей типа в разных порталах, а сначала сформировать на основе сущностей модуля список типов этих сущностей и проверять наличие полей по этому списку. Тогда сложность этой проверки уменьшится с $O(F = \sum_{e \in EM(m)} |Attr(e)|)$ до $O(F' = \sum_{t \in T(P_1) | \exists e \in EM(M), Type(e) = t} |Attr(e)|)$, и общая сложность алгоритма составит $O(N + F' + E + T)$.

Алгоритм 3. *Поиск сущностей по ключевым словам*

Данный алгоритм применяется в модуле поиска для поиска объектов портала по ключевым словам.

Задача: найти совокупность сущностей и их мер релевантности $V_{rel} = \{e, rel\} \in V_l \times N$, соответствующих ключевым словам $W' = \{w\}$ (считается, что все слова попарно не совпадают).

Шаги:

1. Изначально V_{rel} пусто.
2. Для каждого ключевого слова $w \in W'$ получить множество $KeyWordEntities(w)$. Для каждой сущности $e \in KeyWordEntities(w)$ выполнить *шаги* 2.a-2.b.
   a. Если для сущности *e* уже существует $(e, rel) \in V_{rel}$, то обновить меру релевантности $rel = rel + \frac{1}{|W'|}$, где $|W'|$ – количество слов во множестве *W'*.
   b. Если для сущности *e* еще не существует $(e, rel) \in V_{rel}$, то добавить $(e, \frac{1}{|W'|})$ во множество V_{rel}.

3. Отсортировать пары $(e, rel) \in V_{rel}$ по убыванию меры релевантности *rel* и отобрать первые *k* элементов, где $k \in Attr(Settings(u, c))$, *u* – текущий пользователь, *c* – модуль поиска (*k* обозначает максимальное количество релевантных заданному набору слов *W'* сущностей, выдаваемых процедурой поиска).

Псевдокод:

Function FindEntities($W' = \{w\}$*)* $\subset V_l \times N$ *;*
Begin
$V_{rel} \leftarrow \varnothing$ *; // Шаг 1*
For Each $w \in W'$ *Do*
For Each $e \in KeyWordEntities(w)$ *Do*
Begin
If $\exists (e, rel) \in V_{rel}$
then $rel \leftarrow rel + \frac{1}{|W'|}$ *// Шаг 2.a*
Else $V_{rel} \leftarrow V_{rel} \cup (e, \frac{1}{|W'|})$ *; // Шаг 2.b*
End;
RelevanceSort(V_{rel}*); // Шаг 3*
$k \leftarrow GetMaxEntityCount(Attr(Settings(u, c)))$ *;*
$V_{top} \leftarrow GetTop(V_{rel}, k)$ *;*
Return V_{top}*;*
End;

***Анализ сложности*:** Пусть $M = |W'|$ – количество ключевых слов, $N = |\{e_i \in V_l \mid KeyWords(e_i) \cap W' \neq \varnothing\}|$ – количество сущностей портала, у которых хотя бы одно ключевое слово принадлежит множеству W'. Тогда сложность основного цикла составляет $O(M \times N)$. Функция *RelevanceSort* в лучшем случае реализована с использованием быстрой сортировки. В этом случае она имеет сложность $O(N \times \log N)$. Функция *GetTop* имеет сложность $O(k)$. Тогда сложность алгоритма составляет $O(N \times M + N \times \log N + k)$.

***Оптимизация*:** Возможна за счет того, чтобы сначала считать меру релевантности для каждой сущности в целых числах, а потом нормировать делением на $M = |W'|$. Тогда вместо быстрой сортировки можно использовать сортировку подсчетом, запоминая для каждого значения меры релевантности список сущностей с этим значением меры. Тогда сложность сортировки уменьшится с $O(N \times \log N)$ до $O(N + M)$, и общая сложность алгоритма составит $O(N \times M + N + M + k)$.

***Реализация*:** Поскольку разработка ведется на языке C# платформы Microsoft .NET 2.0, то будем рассматривать особенности реализации алгоритмов именно на этом языке. Во-первых, для быстрого получения по ключевому слову совокупности соответствующих слову сущностей нужно использовать словарь (generic-класс Dictionary, у которого в качестве типа ключа выступает тип ключевого слова, а в качестве значения – тип множества сущностей). Словари в C# основаны на хэшировании. Это означает, что сложность операций по добавлению и поиску элементов по ключу составляет $O(1)$. Также словарь можно использовать для хранения для сущности степени ее релевантности во множестве V_{rel}. Типом ключа этого словаря будет тип сущности, а типом значения релевантности – любой из вещественных типов C#.

Алгоритм 4. *Поиск релевантных сущностей для заданной сущности*

Данный алгоритм применяется в модуле релевантных ссылок для поиска объектов, релевантных объекту, соответствующему открытой пользователем странице портала, и добавления на страницу ссылок на эти объекты.

Задача: Найти совокупность сущностей и их мер релевантности $V_{rel} = \{e, rel\} \in V_l \times N$ для заданной сущности e_0.

Шаги:

1. Изначально V_{rel} пусто.
2. Для каждой сущности $e \in V_l \mid (e_0, e) \in E_{rel}$ добавить пару $(e,1)$ во множество V_{rel}.
3. Запустить алгоритм поиска по ключевым словам (*алгоритм* 3) на основе ключевых слов сущности $KeyWords(e_0)$.
4. Для каждой пары (*e, rel*), полученной на *шаге* 2, при условии, что сущность *e* еще не включена во множество V_{rel}, добавить эту пару во множество V_{rel}.
5. Отсортировать пары $(e, rel) \in V_{rel}$ по убыванию меры релевантности *rel* и отобрать первые *k* элементов, где $k \in Attr(Settings(u, c))$, u – текущий пользователь, c – модуль релевантных ссылок (*k* обозначает максимальное количество сущностей, релевантных заданной, выдаваемых процедурой поиска).

Псевдокод:

Function FindRelativeEntities($e_0 \in V_l$ *)* $\subset V_l \times N$ *;*

Begin

$V_{rel} \leftarrow \varnothing$ *; // Шаг 1*

For Each $e \in V_l \mid (e_0, e) \in E_{rel}$ *Do // Шаг 2*

$V_{rel} \leftarrow V_{rel} \cup (e,1)$ *;*

$V_{rel}' \leftarrow FindEntitiies(KeyWords(e_0))$ *; // Шаг 3*

For Each $(e, rel) \in V_{rel}' | \neg \exists (e, rel') \in V_{rel}$ *Do // Шаг 4*

$V_{rel}' \leftarrow V_{rel}' \cup (e, rel)$

RelevanceSort(V_{rel}*); // Шаг 5*

$k \leftarrow GetMaxEntityCount(Attr(Settings(u, c)))$;

$V_{top} \leftarrow GetTop(V_{rel}, k)$;

Return V_{top};

End;

***Анализ сложности*:** Пусть $M = | KeyWords(e_0) |$ – количество ключевых слов сущности e_0, $N = | \{ e_i \in V_l \mid KeyWords(e_i) \cap W' \neq \varnothing \vee \exists (e_0, e_i) \in E_{rel} \} |$ – количество сущностей портала, релевантных сущности e_0. Тогда сложность цикла в *шаге* 2 составляет $O(N)$. Функция *FindEntities* имеет сложность $O(N \times M + N \times \log N + k')$, где k' – количество сущностей, возвращаемых этой функцией. Сложность цикла в *шаге* 4 составляет $O(N)$. Функция *RelevanceSort* в лучшем случае реализована с использованием быстрой сортировки. В этом случае она имеет сложность $O(N \times \log N)$. Функция *GetTop* имеет сложность $O(k)$. Тогда сложность алгоритма составляет $O(2 \times N + N \times M + N \times \log N + k + k'$.

***Оптимизация*:** Если применять оптимизацию, описанную в алгоритме поиска по ключевым словам, то сложность данного алгоритма составит $O(3 \times N + N \times M + M + k + k')$.

***Реализация*:** Аналогично алгоритму поиска сущностей по ключевым словам нужно использовать словари для хранения степеней релевантности для найденных сущностей.

2.3. Структура программного продукта

Портал имеет многоуровневую архитектуру (рис. 2.8). Можно выделить следующие уровни портала:

- Web-интерфейс пользователя – включает в себя основную страницу портала *default.aspx* и ее компоненты (Web-части) для доступа к отдельным модулям портала;
- логика приложения – состоит из отдельных модулей портала, каждый из которых имеет свое назначение;
- подсистема доступа к данным (WebMDK) и подсистема безопасности;
- базы данных портала.

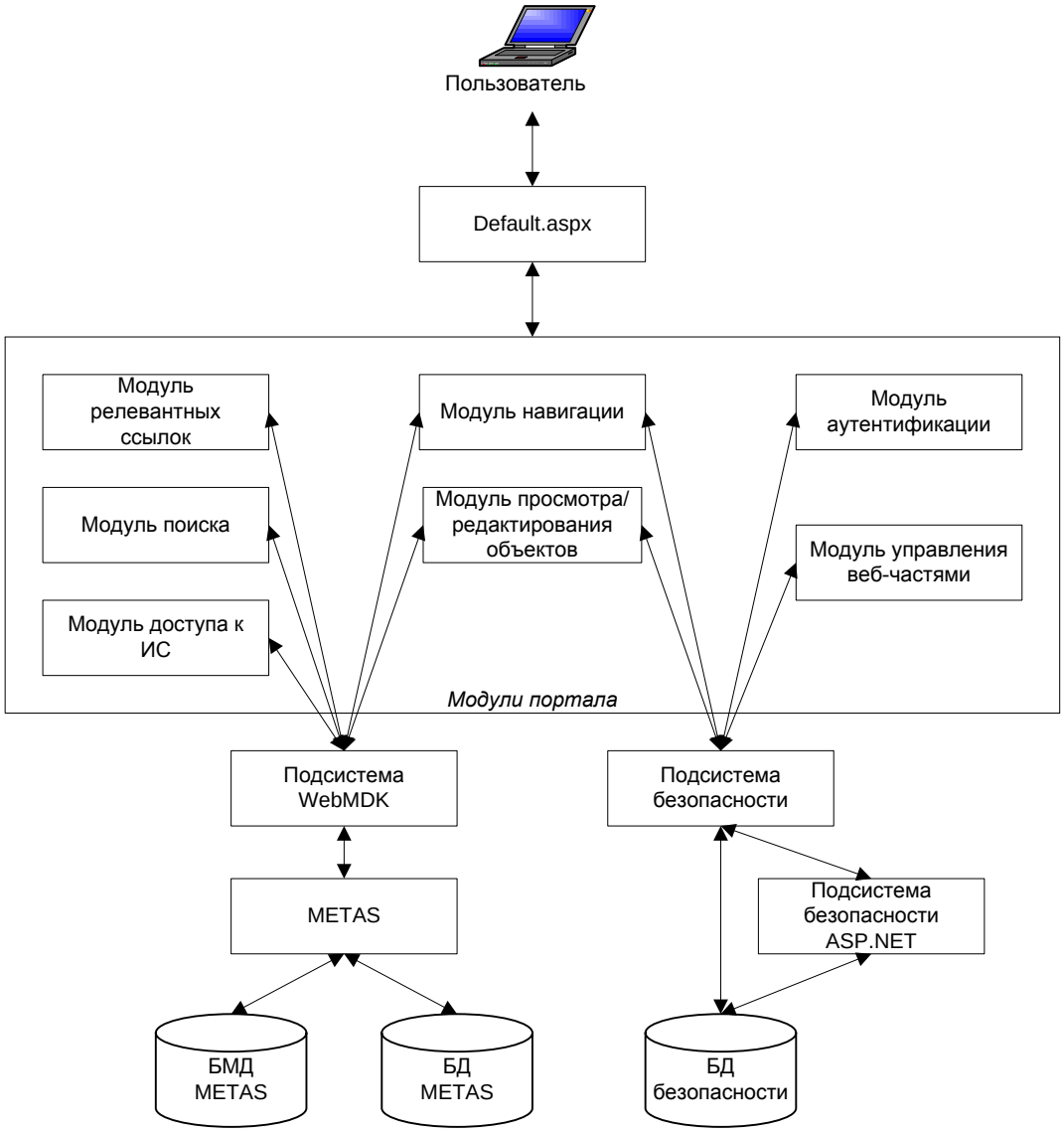

Рис. 2.8. Многоуровневая архитектура портала

Рассмотрим *упрощенный алгоритм работы портала*:

1. Web-пользователь посылает запрос порталу на выполнение какой-либо операции (например, получение каких-либо данных).
2. Сервер IIS принимает запрос и загружает страницу портала *default.aspx*.
3. Страница *default.aspx* обрабатывает запрос и обращается к модулям портала для выполнения операции, запрошенной пользователем.
4. Каждый модуль решает свою задачу обработки запроса (отображение данных, карты сайта, формирование релевантных ссылок, поиск и т.д.). При этом модуль проверяет права пользователя на выполнение операции при помощи подсистемы безопасности и получает данные через подсистему WebMDK.

5. Подсистема WebMDK вызывает методы логической модели METAS для получения объектов, необходимых для выполнения операции.
6. Логическая модель METAS и подсистема безопасности обращаются к базам данных для формирования необходимых объектов и проверки прав пользователя.
7. Страница *default.aspx* и составляющие ее Web-части на основе полученных данных формируют HTML-страницу в ответ на запрос пользователя.
8. Сервер IIS пересылает HTML-страницу пользователю.

В основе портала лежит технология Microsoft ASP.NET 2.0. По сравнению с предыдущей версией ASP.NET 1.1, появилось большое количество готовых компонентов, облегчающих создание Web-сайтов на основе этой технологии.

В данной разработке использовались следующие компоненты ASP.NET: встроенная модель безопасности, аутентификации и авторизации пользователей (Security Model), привязка к данным (Data Binding), Web-части (Web Parts), пользовательские элементы управления (User Controls).

Web Parts (веб части) – это набор элементов управления, предназначенный для создания Web-сайтов. Эта технология предоставляет следующие возможности:

- создание областей страницы (технология позволяет структурировать страницу посредством задания областей, называемых зонами Web-частей (Web Part Zones)),
- настройка областей страницы (пользователь может выбрать, какие элементы (Web-части) нужно отображать на странице, и задать их параметры отображения и поведения без участия разработчика),
- сохранение настроек пользователя (ASP.NET автоматически сохраняет внешний вид страницы, настроенный пользователем).

Разработанный портал имеет единственную страницу *default.aspx*. Страница состоит из Web-частей, основные из которых представлены на рис. 2.9.

Модульная структура страницы позволяет автоматически сохранять внешний вид при изменении ее содержимого и настраивать ее в соответствии с текущими параметрами отображения и содержания.

Единственность страницы портала позволяет обеспечить полный контроль над выполняемыми в портале операциями. Каждая ссылка в портале задается операцией, которую нужно выполнить при переходе на страницу, и ее параметрами. Перед отображением страницы пользователю производится проверка прав пользователя на выполнение соответствующей операции, а также ее корректность. Если данную операцию может выполнить текущий

пользователь, то интерфейс страницы настраивается под нее. В противном случае пользователю предоставляется информация о невозможности выполнения операции с описанием причины.

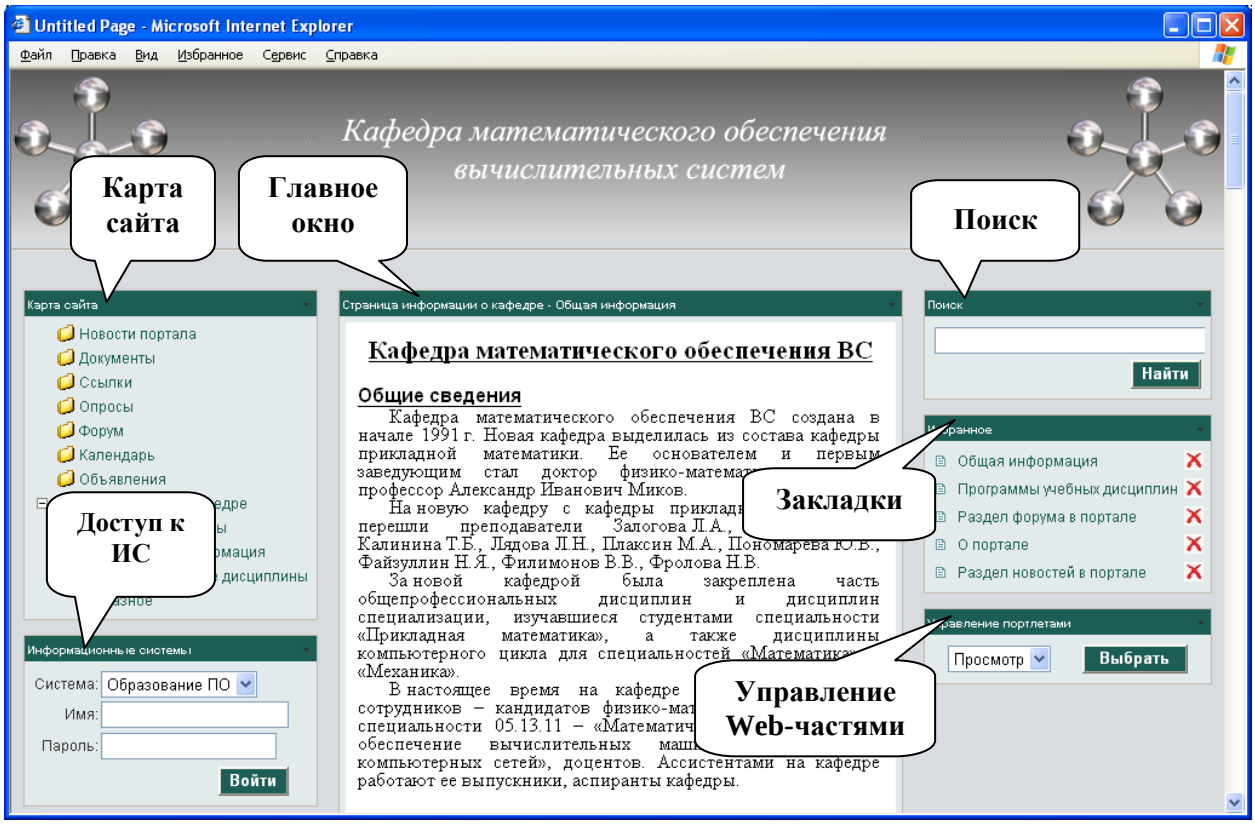

Рис. 2.9. Внешний вид главной страницы портала

Такой подход к формированию ссылок позволяет также обеспечить ведение журнала выполняемых операций, единообразно настраивать внешний вид всех элементов страницы и сохранять его между сеансами для каждого пользователя.

Логическая структура страницы в общем виде представлена на рис. 2.10.

Страница включает в себя набор Web-частей, расположенных в трех зонах: левой зоне, центральной зоне (главном окне) и правой зоне.

Каждая Web-часть наследует от базового класса WebPart. При загрузке страницы Web-часть принимает ее параметры, переданные в адресной строке, и в зависимости от них настраивает свой внешний вид и внешний вид составляющих ее компонентов (элементов управления).

Разработанные на данный момент Web-части включают в себя по одному пользовательскому элементу управления, реализующему отображение требуемой в данный момент функциональности.

Для всех Web-частей, кроме Web-части главного окна, задан фиксированный элемент управления.

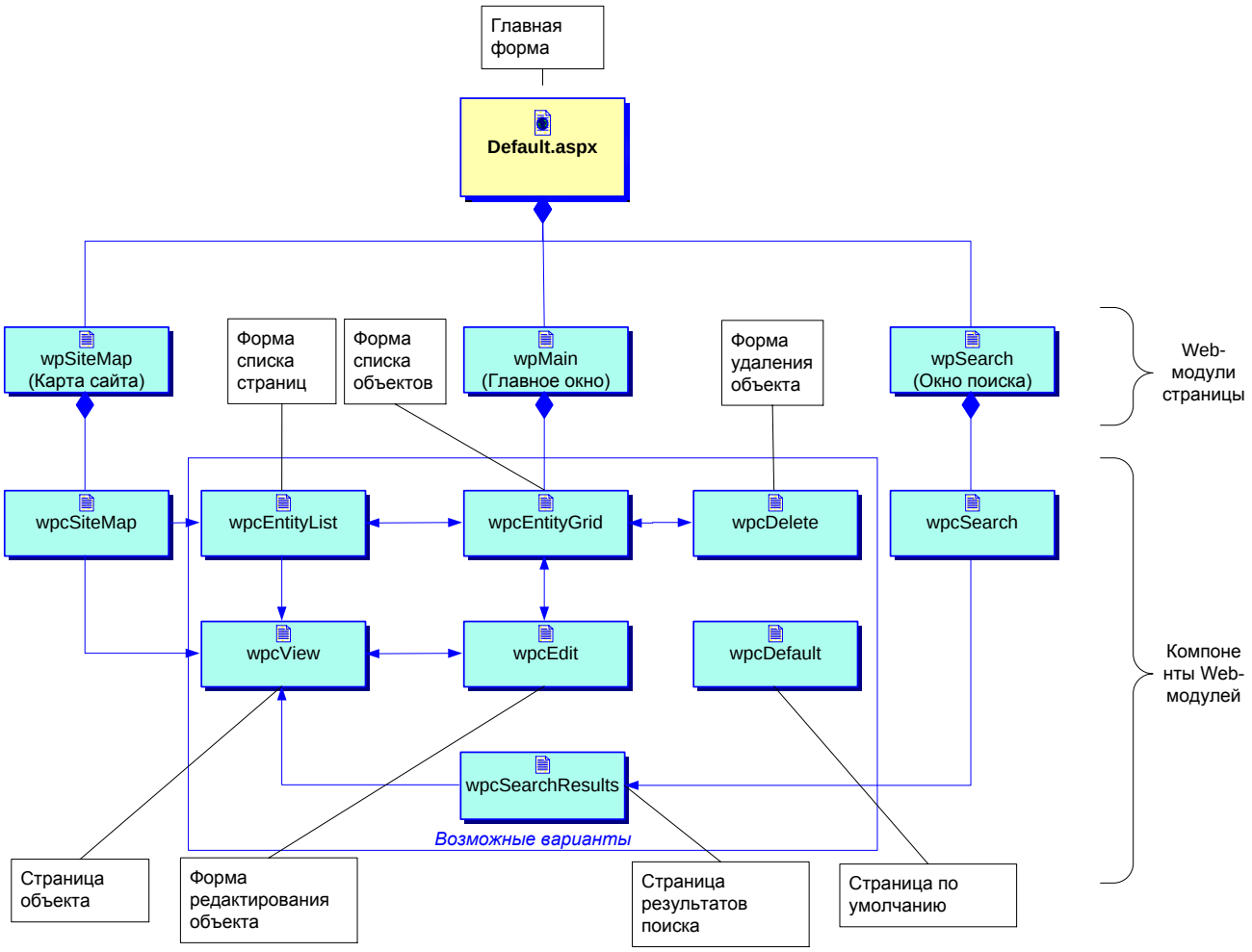

Рис. 2.10. Логическая структура главной страницы

Главное окно служит для отображения всех ресурсов и сервисов портала. Оно выбирает нужный элемент управления на основе текущей операции (см. табл. 2.1).

Помимо Web-части главного окна имеется ряд других Web-частей (см. табл. 2.2).

Технология WebParts тесно связана с подсистемой безопасности ASP.NET. Поэтому перейдем к ее описанию и способу связи с ней разработанной модели безопасности портала.

Таблица 2.1. Основные операции портала и соответствующие им элементы управления главного окна

Операция	Отображаемый элемент	Функция элемента
Logout, ISLogout	wpcDefault	Отображение информации при первом входе в портал, выходе пользователя из своего рабочего места или выходе из информационной системы
View	wpcView	Просмотр объекта портала
New, Edit	wpcEdit	Создание нового или редактирование существующего объекта портала
List	wpcEntityList	Просмотр списка страниц объектов
Grid	wpcEntityGrid	Просмотр таблицы однотипных объектов с возможностями по созданию, редактированию и удалению отдельных объектов
Delete	wpcDelete	Подтверждение удаления объекта
Deny	wpcDeny	Информирование пользователя о нехватке прав для выполнения выбранной операции
Search	wpcSearchResults	Отображение результатов поиска
ISLogin	wpcISDefault	Отображение информации при первом входе в информационную систему

Таблица 2.2. Web-части портала

Web-часть	Название	Назначение
wpMain	Главное окно	Отображение основной информации портала
wpSiteMap	Карта сайта (дерево объектов)	Навигация по порталу и информационным системам. Отображает ссылки на основные объекты портала и систем
wpSearch	Модуль поиска	Поиск по порталу на основе ключевых слов
wpReferences	Релевантные ссылки	Отображение ссылок на релевантные текущему объекту ресурсы портала
wpLogin	Модуль входа для пользователей портала	Вход зарегистрированных пользователей в портал
wpISLogin	Модуль входа для пользователей информационных систем	Вход зарегистрированных пользователей указанных в конфигурационном файле информационных систем, совместимых с порталом
wpPortletManager	Модуль управления Web-частями	Выбор совокупности и настройка внешнего вида, расположения и параметров Web-частей портала
wpBookmarks	Модуль закладок пользователя («Избранное»)	Отображение сохраненных пользователем ссылок на объекты и списки объектов портала

Подсистема безопасности ASP.NET

Подсистема безопасности ASP.NET включает в себя классы для аутентификации и авторизации пользователей, а также для управления аутентифицированными пользователями в приложении. К тому же, сама платформа .NET предоставляет набор классов, позволяющих обеспечить конфиденциальность и целостность информации посредством шифрования и цифровых подписей.

Модель безопасности ASP.NET основана на концепции привратников (gatekeepers). Модель привратников предполагает, что защищенное приложение всегда использует больше механизмов обеспечения безопасности, чем необходимо. Каждый механизм реализован в виде привратника, который отвечает за соблюдение некоторых условий безопасности системы. Если один из привратников терпит неудачу при обнаружении нарушения безопасности, атакующему пользователю или приложению придется иметь дело со следующим в цепочке привратников. Чем больше привратников в приложении, тем сложнее будет атакующему достичь злонамеренной цели.

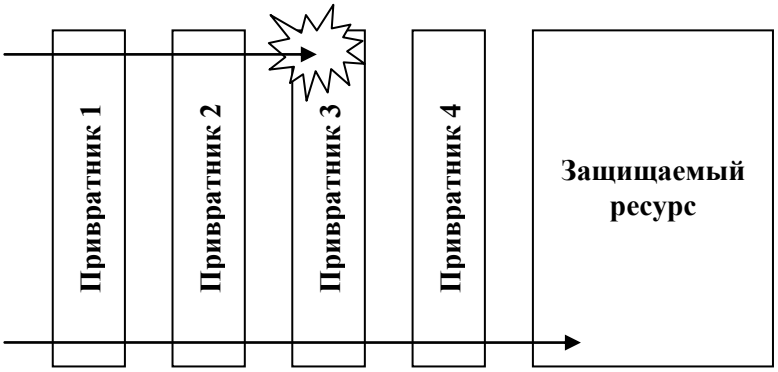

Рис. 2.11. Цепочка привратников

На рис. 2.11 представлена цепочка привратников. В конце цепочки находится защищаемый ресурс. К нему можно получить доступ только в том случае, если каждый привратник его разрешает. Если хотя бы один привратник запрещает доступ, пользователю или приложению, инициировавшему запрос, возвращается ошибка безопасности.

ASP.NET реализует следующие *механизмы обеспечения безопасности*:

- аутентификация – позволяет идентифицировать личность пользователя системы и определить ее подлинность;
- авторизация – позволяет определить права и ограничения аутентифицированного пользователя системы на основе ролей;
- конфиденциальность – позволяет обеспечить невозможность просмотра данных, передаваемых по сети или находящихся в хранилище (например, в БД), неавторизованными пользователями или приложениями на основе

шифрования;

- целостность – позволяет обеспечить невозможность изменения данных при пересылке между клиентом и сервером неавторизованными пользователями или приложениями.

Опишем теперь процесс работы подсистемы безопасности ASP.NET при доступе к страницам Web-приложения.

По умолчанию, неаутентифицированные пользователи могут обращаться к любой странице ASP.NET. Однако если пользователь обращается к странице, запрещающей анонимный доступ, выполняется следующий алгоритм:

1. Запрос на страницу посылается серверу. Поскольку личность пользователя еще не известна в этот момент, пользователю предлагается ввести имя и пароль.
2. Пользователь предоставляет свои данные (имя и пароль), которые затем верифицируются самим приложением или сервером IIS.
3. Если предоставленные пользователем данные достоверны, то пользователю разрешается доступ к странице. Если же нет, то пользователю предлагается ввести данные снова или же ему передается сообщение о запрещении доступа к странице.

Когда пользователь обращается к защищенной странице, которая разрешает доступ только определенным пользователям или пользователям определенных ролей, дополнительно выполняется соответствующая проверка прав пользователя на доступ к странице.

В общем виде схема работы механизма работы подсистемы безопасности представлена на рис. 2.12.

Взаимодействие с подсистемой безопасности ASP.NET

При первом запуске Web-приложения ASP.NET создает набор таблиц для управления пользователями и ролями (см. табл. 2.3).

Разработанная в рамках данной работы модель безопасности использует эти таблицы для хранения информации о пользователях и ролях. Такой подход имеет ряд преимуществ:

- единая модель безопасности – не нужно реализовывать механизмы синхронизации моделей безопасности портала и ASP.NET;
- высокая надежность модели безопасности, тщательно проработанной и протестированной разработчиками ASP.NET;
- наличие готовой объектной модели – отсутствует необходимость реализации типичных для порталов операций по идентификации пользователей, их регистрации и хранения персональных данных.

Рис. 2.12. Обработка запроса на доступ к защищенной странице

Таблица 2.3. Таблицы управления пользователями и ролями ASP.NET

Название	Описание
aspnet_Membership	Таблица данных о пользователях. Хранит идентификаторы, имена, хеши паролей, адреса электронной почты и другие предоставленные пользователем данные, а также дату последней активности пользователя
aspnet_Paths	Таблица персонализируемых ресурсов. Хранит идентификаторы и адресные строки персонализируемых ресурсов, запрошенных пользователями
aspnet_PersonalizationPerUser	Таблица персонализации. Хранит для каждого пользователя настройки всех персонализированных ресурсов, запрошенных пользователем
aspnet_Roles	Таблица ролей. Хранит идентификаторы, названия и описания ролей
aspnet_UsersInRoles	Таблица вхождения пользователей в роли

Технология WebParts использует таблицу персонализации для хранения настроек Web-частей каждой персонализируемой страницы для каждого пользователя портала. Для каждой Web-части страницы сохраняются параметры расположения и скрытия, а также параметры, заданные разработчиками портала.

Поскольку каждая страница портала формируется динамически на основе метаданных, то для сохранения внешнего вида страницы в таблице персонализации введены дополнительные параметры Web-частей: название и атрибуты текущей операции. Например, для компонента редактирования объекта сохраняются название операции (Edit) и идентификаторы сущности редактируемого объекта и самого объекта.

При каждом обращении пользователя к странице *default.aspx* ее настройки сохраняются в таблице персонализации. Это позволяет при выходе из портала в любой момент времени обеспечить восстановление последнего состояния страницы для пользователя при его следующем входе.

Подсистема безопасности портала

Для хранения прав пользователей на доступ к объектам используются таблицы EntityRights и EntityTypeRights (рис. 2.13).

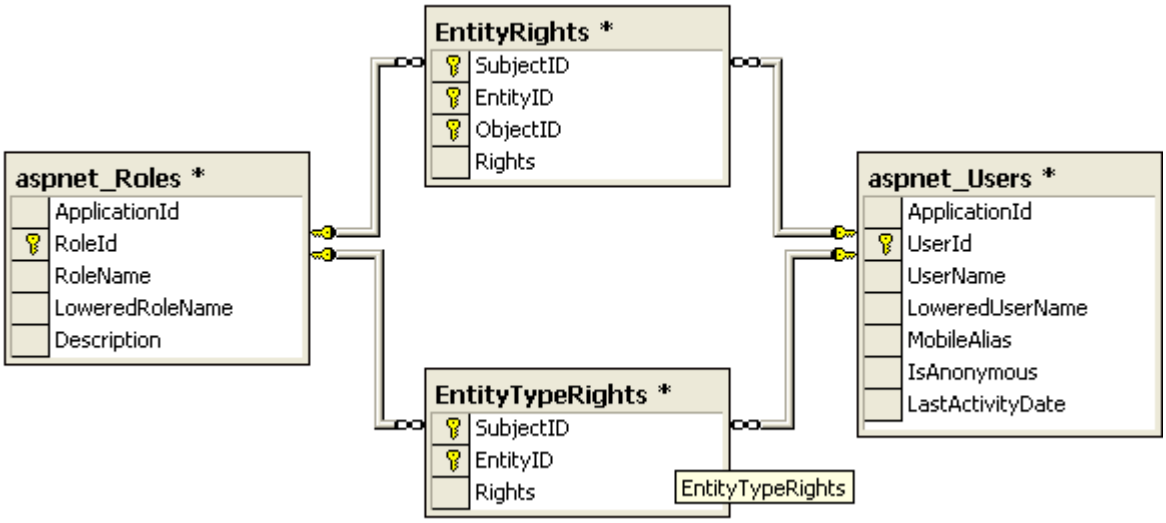

Рис. 2.13. Связь таблиц прав пользователей с таблицами ASP.NET

Таблица *EntityRights* используется для хранения прав пользователей на выполнение операций над отдельными объектами, а таблица *EntityTypeRights* – для хранения прав пользователей в отношении типов объектов (сущностей). Обе таблицы имеют поле *SubjectID* типа *uniqueidentifier*. Тип *uniqueidentifier* (уникальный идентификатор) предоставляет в качестве значений строки размером 32 байта. Данный тип позволяет генерировать новые значения, вероятность совпадения которых с существующими близка к нулю. Поэтому

достаточно одного поля *SubjectID* в каждой таблице прав для хранения идентификатора субъекта, который может быть либо ролью (в этом случае идентификатор служит внешним ключом, ссылающимся на таблицу ролей), либо пользователем (в этом случае идентификатор ссылается на таблицу пользователей).

Таблица *EntityTypeRights*, кроме поля *SubjectID*, имеет поле *EntityID* для хранения идентификатора сущности (типа объекта), для которой задаются права, и *Rights* для хранения самих прав. Значения поля *Rights* представляются в виде целых чисел, однако каждое число несет информацию сразу о трех правах. Нулевой бит числа показывает, может ли субъект просматривать список объектов типа, первый бит – может ли субъект создавать объекты данного типа, а второй бит – может ли субъект назначать права на доступ к объекту другим пользователям или ролям. Такой способ представления прав в упакованном виде обеспечивает минимальный размер таблицы и большую скорость выполнения операций выборки данных и их обновления.

Таблица *EntityRights* имеет поля *EntityID, ObjectID* и *Rights* соответственно для хранения идентификаторов сущности и объекта, а также прав на выполнение операций субъектом над этим объектом. Права задаются также в упакованном виде и назначаются для операций просмотра, редактирования и удаления объекта.

Права пользователя при обращении к ресурсам портала складываются из прав, назначенных только ему, и прав всех ролей, в которые он входит.

В случае если для субъекта нет информации в соответствующей таблице о разрешении на выполнение определенной операции над определенной сущностью или объектом, считается, что субъект соответствующего права не имеет, т.е. действует принцип «что не разрешено, то запрещено».

Классы подсистемы безопасности портала представляют объектно-ориентированную прослойку между базой данных и интерфейсом пользователя как Windows-, так и Web-приложения.

В портале для работы с правами субъектов имеются *три основных класса*:

- WebSecuritySystem – класс подсистемы безопасности портала;
- EntityRights – класс для получения и сохранения прав субъектов на доступ к конкретным объектам;
- EntityTypeRights – класс для получения и сохранения прав субъектов на доступ к сущностям.

Методы и свойства данных классов представлены на рис. 2.14.

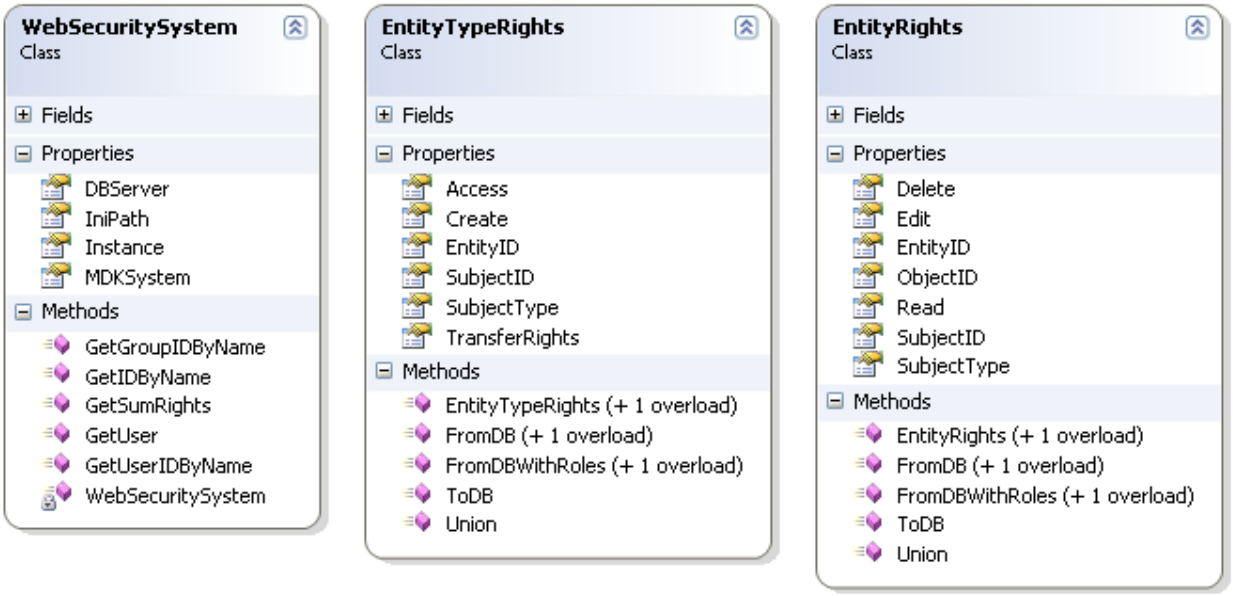

Рис. 2.14. Классы подсистемы безопасности портала

Класс *WebSecuritySystem* инициализируется строкой соединения с БД безопасности портала и объектом MDKSystem для доступа к информации о сущностях и объектах портала через систему METAS. Он предоставляет возможности для получения текущего пользователя портала и получения идентификаторов пользователей и ролей по их имени. Текущим пользователем портала считается аутентифицированный подсистемой безопасности ASP.NET пользователь или в случае анонимного доступа пользователь портала *Guest* (гость). Для пользователя Guest отдельно задаются права для общедоступных ресурсов портала. Это позволяет анонимным пользователям просматривать некоторые объекты портала (новости, форум и т.д.) без необходимости аутентификации.

Класс *WebSecuritySystem* реализует паттерн *Singleton*. Это означает, что в портале существует лишь один объект данного класса. Это обеспечивает единую точку доступа к БД безопасности и препятствует нарушению целостности информации о защите портала и его ресурсов.

Подсистема безопасности также включает в себя Windows-компонент для добавления и удаления пользователей и ролей портала и назначения их прав. Компонент может запускаться отдельно или же в составе Windows-интерфейса системы METAS. Для запуска в составе METAS используется класс *WebSecComponent*, который реализует интерфейс *MDK.Components.IComponent* и переопределяет методы для своей инициализации.

Подсистема WebMDK

Подсистема WebMDK служит в качестве прослойки между порталом и системой METAS (классом MDKSystem).

Класс MDKSystem загружает компоненты системы METAS и выполняет инициализацию всех ее моделей на основе переданных ему строк соединений с базами данных и метаданных, а также предоставляет доступ ко всем моделям посредством своих полей.

Однако этот класс реализует паттерн Singleton, что не позволяет создавать несколько его экземпляров. Поэтому подсистема WebMDK расширяет этот класс, ослабляя данное ограничение, а именно разрешает создавать только один объект в пределах сессии пользователя.

Подсистема WebMDK также выполняет синхронизацию доступа к системе METAS, блокируя одновременные обращения к ней посредством критической секции.

В функции подсистемы WebMDK также входит подключение к информационным системам METAS. С точки зрения METAS как портал, так и информационные системы имеют одну и ту же структуру баз данных и метаданных. Чтобы выполнить подключение к ИС, подсистема WebMDK перезагружает объект класса *MDKSystem*, хранящийся в сессии пользователя, передавая ему строки соединений ИС, извлекаемых из конфигурационного файла портала.

Практически все модули портала тесно связаны с пользовательским интерфейсом. Поэтому сначала опишем требования, предъявляемые к современному интерфейсу пользователя, а затем опишем модули, которые его реализуют.

Требования, предъявляемые к пользовательскому интерфейсу современных ИС

Пользовательский интерфейс (User Interface, интерфейс пользователя) объединяет в себе все элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением [44]. Графический интерфейс пользователя (Graphics User Interface – GUI) является обязательным компонентом большинства современных программных продуктов, ориентированных на работу конечного пользователя. Наиболее часто графический интерфейс реализуется в интерактивном режиме работы пользователя для программных продуктов, функционирующих в среде Windows, и строится в виде системы спускающихся меню с использованием в качестве средства манипуляции мыши и клавиатуры [45]. Работа пользователя

осуществляется с экранными формами, содержащими объекты управления, панели инструментов с пиктограммами режимов и команд обработки.

Такой тип пользовательского интерфейса принято обозначать аббревиатурой WIMP (Windows – Icons – Menus – Pointing device), что отражает задействованные в нем интерактивные элементы – окна, пиктограммы, меню и позиционирующее устройство (обычно мышь). Именно интерфейсы этого типа, завоевавшие популярность вместе с Macintosh в 1984 году и позднее скопированные, в частности, в Windows для персональных компьютеров, доминируют и по сей день [46].

Современный интерфейс, предоставляемый Web-браузерами удаленным пользователям, во многом схож с Windows-интерфейсом. Поэтому ниже перечисленные требования применимы к Web-интерфейсу и должны выполняться во всех современных Web-приложениях.

Основные требования, предъявляемые к пользовательскому интерфейсу, описываются стандартом ISO/DIS 9241-14. Интерфейс пользователя должен обладать следующими семью свойствами:

- *Соответствие задачам*, решаемым пользователем. Интерфейс соответствует задаче, если для выполнения своей работы пользователю не приходится решать проблемы, не обусловленные характеристиками задачи.
- *Легкость использования*. Интерфейс легок в использовании, если каждый шаг диалога понятен пользователю, либо объясняется по его запросу. Не должно возникать тупиковых ситуаций, вызывающих замешательство пользователя.
- *Управляемость*. Организация интерфейса позволяет пользователю в течение всей работы управлять диалогом для достижения поставленной цели. В системе не должны возникать ситуации, когда программа, выполняя собственные расчеты, не реагирует на действия пользователя.
- *Соответствие ожиданиям* пользователя. Интерфейс должен соответствовать предыдущему опыту или образованию пользователя.
- *Устойчивость к ошибкам*. Интерфейс терпим к ошибкам, если несмотря на очевидные ошибочные действия пользователя требуемый результат может быть достигнут без дополнительных усилий пользователя или с минимальными корректирующими действиями. Не должно возникать фатальных ошибок. Если ошибка возникает, должно выдаваться развёрнутое сообщение, объясняющее ошибку.
- Пригодность к *индивидуализации*. Интерфейс индивидуализируемый (или гибкий), если допускает адаптацию к индивидуальным требованиям и профессиональной подготовке пользователя.
- *Легкость изучения* интерфейса. Интерфейс отвечает требованиям

легкости обучения, если он обеспечивает последовательное обучение пользователя за минимальное время с минимальными усилиями со стороны пользователя.

Поскольку портал должен обеспечивать возможность доступа к ИС, требования, предъявляемые к интерфейсу ИС, должны быть выполнены при разработке портала.

Помимо вышеперечисленного, с учетом специфики технологии METAS, а также характера ИС, построенных на основе этой технологии, следует сформулировать дополнительное требование к средствам разработки интерфейса: возможность динамической настройки при изменении структуры данных и расширении функциональности системы.

Генерация пользовательского интерфейса

Модуль навигации и модуль отображения и редактирования объектов обеспечивают доступ к основным функциям портала. Поскольку структура и сущности портала полностью описываются метаданными, то карта сайта и формы просмотра и редактирования объектов генерируются «на лету». Этим данные модули отличаются от остальных.

Задача разработки пользовательского интерфейса приложения заключается в отображении внутренней структуры объектов ИС на уровень представления пользователя о предметной области. Управление интерфейсом пользователя, описанное в данной работе, опирается на метаданные презентационного уровня, которые строятся на основе логического уровня ИС [47]. Метаданные уровня представления предназначены для автоматического создания экранных форм, с помощью которых пользователь может просматривать, вводить и редактировать данные.

Для адаптации презентационной модели к требованиям Web-интерфейса введена расширяющая ее модель Web-представления.

Модуль навигации

Модуль навигации и соответствующая ему Web-часть предназначены для формирования карты сайта для портала и дерева объектов для ИС, представляющих структуру портала и ИС в виде дерева ссылок на их ресурсы и объекты (см. рис. 2.15).

Карта сайта настраивается на основе данных о правах пользователей на просмотр объектов портала. Загружаются только те вершины, по отношению к объектам которых пользователь имеет право просмотра. Поскольку такая карта сайта имеет свою структуру для каждого пользователя, в сессии пользователя хранится ссылка на нее.

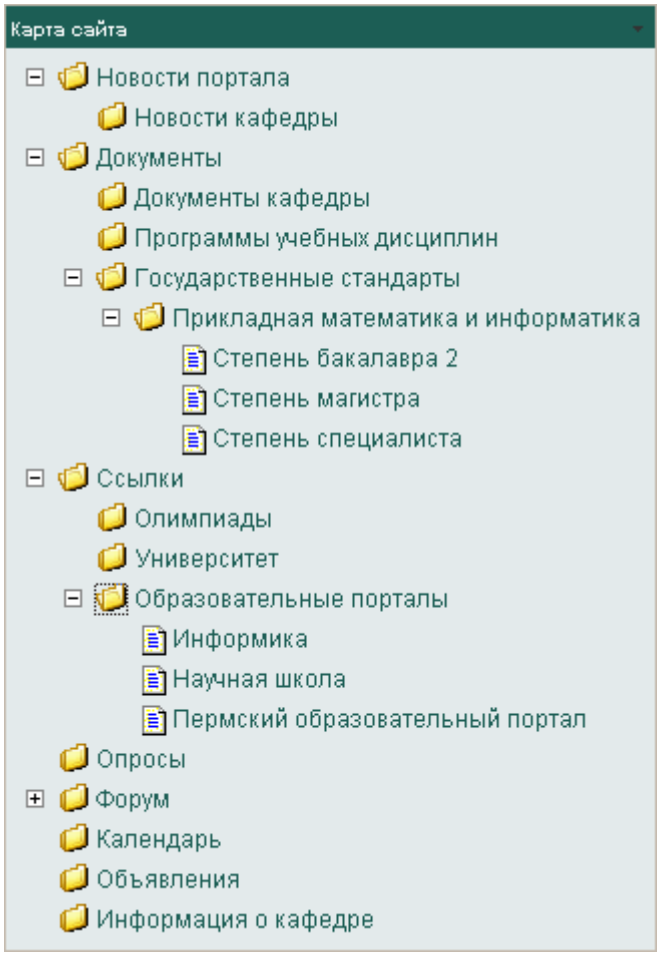

Рис. 2.15. Карта сайта

Заполнение дерева объектов выполняется динамически. Это означает, что дочерние вершины загружаются только в момент раскрытия родительской вершины. Это позволяет снизить трафик в сети и обеспечить более быстрое функционирование приложения. Данная функциональность основана на технологии AJAX, которая предоставляет возможность загружать элементы страницы без ее полной перезагрузки. В данном случае при раскрытии вершины загружается только та часть страницы, которая отвечает за отображение дочерних вершин. При этом выполняется обращение к серверу, который на основе идентификатора и параметров родительской вершины формирует HTML-код представления дочерних вершин.

Загрузка дочерних вершин реализуется в классе *SubTreeView* процедурой *LoadTreeNodes.*

Все основные операции по инициализации и загрузки вершин карты сайта выполняются классом *TreeViewNodeManager*.

TreeViewNodeManager имеет следующие методы:

– *InitTreeView* – реализует начальную загрузку дерева (карты сайта), заполняя только корневые вершины дерева.

– *AddNodeType* – реализует добавление в дерево вершины на основе данных о родительской вершине, параметрах вершины (путь к ней в дереве объектов) и текущем пользователе портала, а также настройку последовательности действий при ее выборе (более подробное описание алгоритма загрузки вершины дерева рис. 2.16).
– *SqlSelectForTreeView* – формирует запрос к БД на выборку вершин из нее на основе данных о родительской вершине и типе загружаемых вершин.

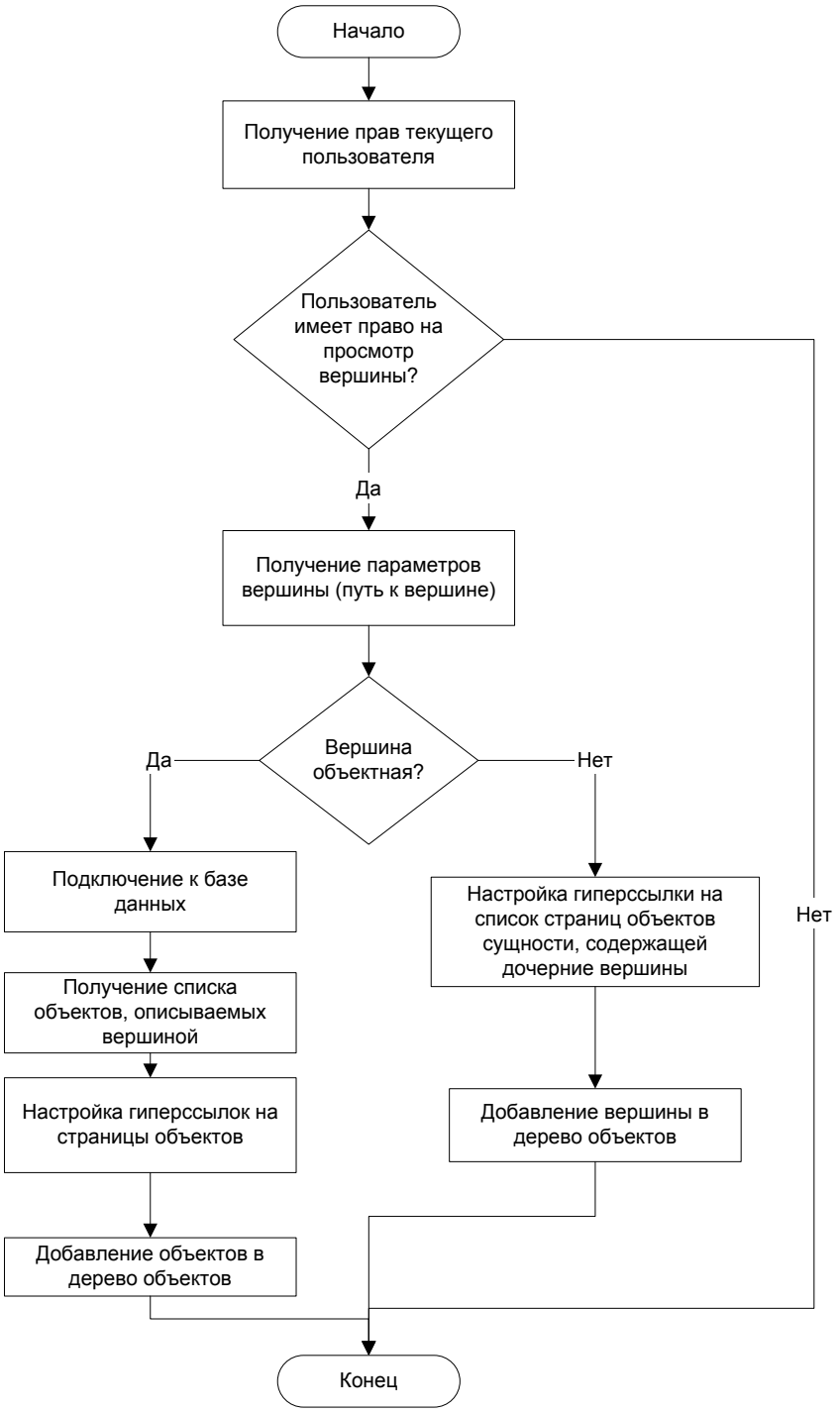

Рис. 2.16. Алгоритм работы загрузки вершины дерева

При выборе вершины дерева объектов в главном окне портала будет выдана информация, соответствующая настроенной гиперссылке вершины. Если вершина представляла объект, то под информацией понимается страница или форма просмотра этого объекта. Если вершина – папка, то информация будет представлять собой список соответствующих папке однотипных объектов портала.

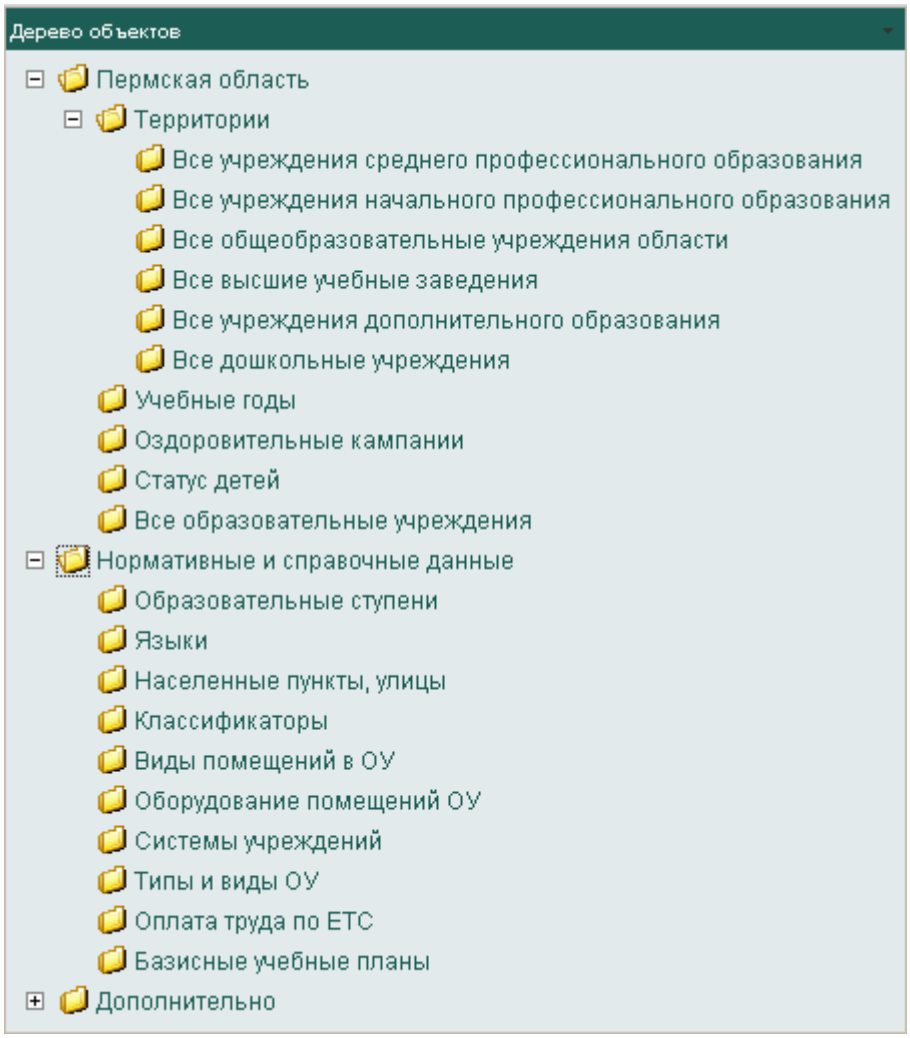

Рис. 2.17. Дерево объектов ИС «Образование Пермской области»

При входе пользователя в информационную систему или выходе из нее карта сайта перенастраивается в соответствии с метаданными той системы, которая загружается в данный момент. На рис. 2.17 представлена перенастроенная карта сайта, представляющая собой дерево объектов информационной системы «Образование Пермской области».

Модуль просмотра и редактирования объектов

Данный модуль предназначен для выполнения операций просмотра, изменения и удаления объектов портала или ИС, а также генерации интерфейса главного окна для этих операций.

Модуль реализует обработку следующих типов операций:

- просмотр списка страниц объектов некоторой сущности (операция «List»),
- просмотр списка объектов некоторой сущности (операция «Grid»),
- просмотр страницы некоторого объекта (операция «View»),
- создание нового объекта некоторой сущности (операция «New»),
- редактирование атрибутов некоторого объекта (операция «Edit»),
- удаление некоторого объекта (операция «Delete»).

Для каждой операции имеется отдельный элемент управления, загружаемый в Web-часть главного окна, который реализует эту операцию.

Модель Web-представления к стандартному списку атрибутов объекта добавляет два новых: списковое представление и представление просмотра объекта (страница объекта).

Страница объекта предназначена для отображения объекта в режиме просмотра. Например, для объекта сущности «Новость» страница может представлять собой текст этой новости. Для объекта сущности «Тема форума» страницей будет служить форма, на которой будет отображаться список сообщений текущей темы с возможностями добавления новых сообщений и редактирования или удаления старых.

Списковое представление предназначено для описания объекта при его просмотре в списке объектов одной сущности. Для объекта сущности «Документ» списковым представлением может быть краткое описание данного документа, для объекта сущности «Новость» – краткое содержание новости и т.д.

Для хранения этих атрибутов используется отдельная таблица в БД под названием «Web pages». Данная таблица включает в себя следующие поля:

- *EntityID* – идентификатор сущности объекта;
- *ObjectID* – идентификатор объекта;
- *IsCustomControl* – определяет, будет ли использоваться при просмотре объекта стандартная страница просмотра, наполнение которой задается полем PlainHTML или отдельный элемент управления, заданный полем *CustomControl*;
- *CustomControl* – путь к элементу управления относительно корневой

директории портала, который нужно использовать при просмотре объекта;

- *PlainHTML* – HTML-содержимое страницы просмотра;
- *ListHTML* – списковое представление объекта.

Помимо возможности задавать произвольные компоненты для просмотра объектов, можно задавать произвольные компоненты для просмотра списка объектов. Для этого предназначена таблица «Web lists». Она включает в себя следующие поля:

- *EntityID* – идентификатор сущности;
- *IsCustomControl* – определяет, будет ли использоваться при просмотре списка объектов данной сущности стандартный список объектов или отдельный элемент управления, заданный полем Custom Control;
- *CustomControl* – путь к элементу управления относительно корневой директории портала, который нужно использовать при просмотре списка объектов данной сущности.

Следует сказать, что данные нововведения предназначены только для объектов портала. После входа в ИС через портал команда «View» заменяется командой «Edit» в режиме чтения, а вместо команды «List» используется команда «Grid».

При выполнении каждой из таких операций проверяются права пользователя на нее. Для операции просмотра проверяется наличие права на чтение объекта, для операции редактирования – изменения, для операции просмотра списка страниц и списка объектов – доступа к сущности. Кроме того, при отображении списка из него исключаются страницы, которые пользователь не может просматривать. К тому же при отображении таблицы объектов (команда «Grid») при отсутствии прав пользователя на создание, изменение или удаления конкретных объектов становятся недоступными соответствующие кнопки формы.

Изменить списковое представление и представление просмотра можно на форме редактирования объекта, однако данная операция доступна только пользователям, обладающим правом редактирования объекта.

Пользователю предоставляются базовые возможности по редактированию и форматированию HTML-текста, аналогичные встроенным в текстовый редактор Microsoft Word (рис. 2.18), а именно операции «Вырезать», «Копировать», «Вставить», выбора полужирного, курсива или подчеркнутого шрифта для выделенного текста, различные способы выравнивания, вставка изображений и т.д.

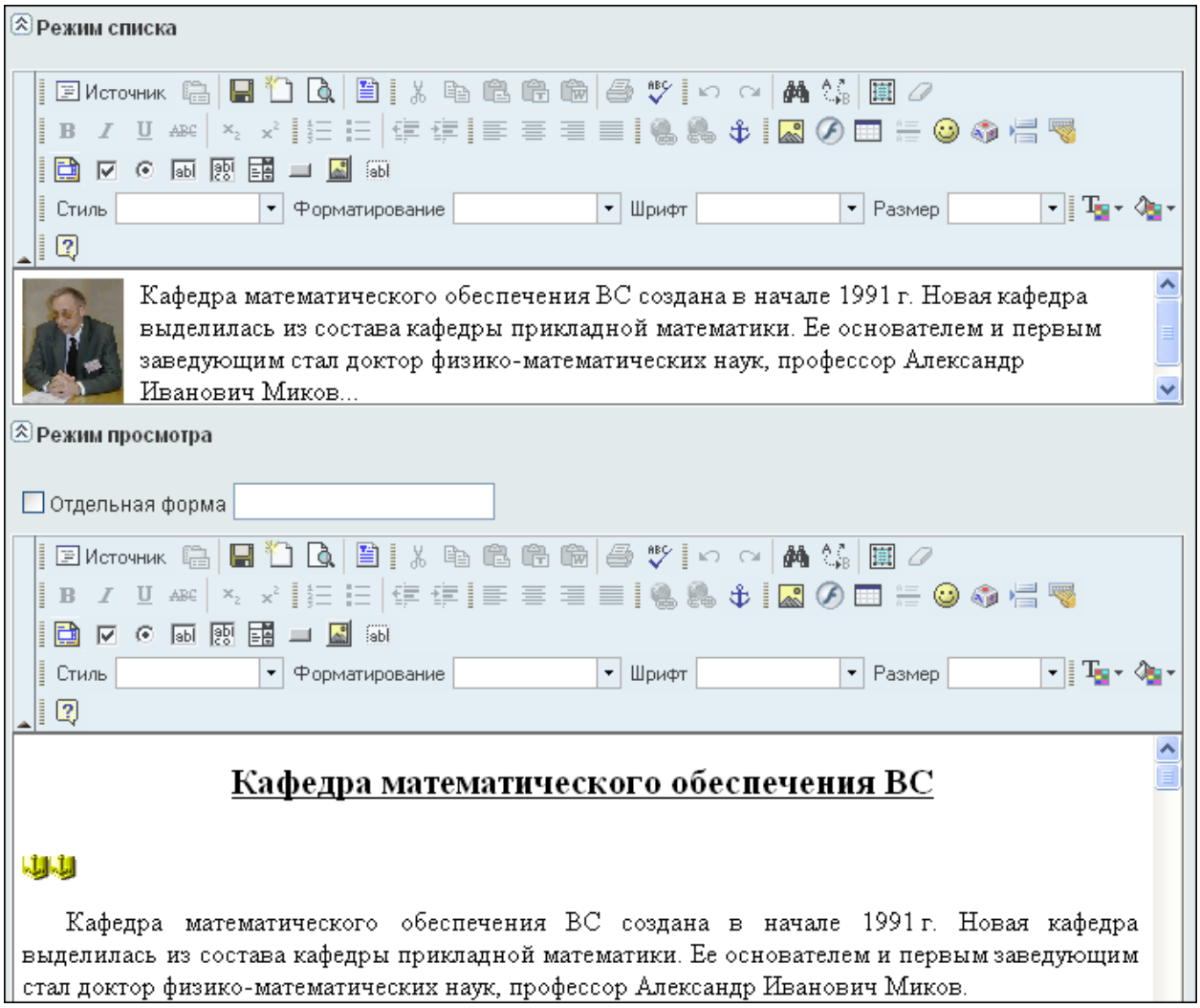

Рис. 2.18. Редактирование представлений режимов списка и просмотра

В случае если пользователю не хватает прав для выполнения операции, она подменяется командой «Deny», которая перенаправляет его на страницу с сообщением «У Вас нет прав на доступ к данной странице».

Кроме того, в режиме просмотра при наличии у пользователя соответствующих прав, на странице отображается ссылка для перехода в режим редактирования, а в режиме списка страниц – для перехода в режим таблицы объектов.

Использование технологии привязки к данным

Элементы управления для представления списка объектов, редактирования отдельных объектов, а также некоторые специально разработанные элементы управления, используемые в режиме просмотра, применяют для отображения данных и выполнения действий над ними технологию привязки к данным (Data Binding).

Привязка к данным позволяет присоединять объекты данных к одному или нескольким элементам управления на форме, которые затем автоматически будут отображать данные. Это избавляет разработчика от необходимости реализовывать логику по обработке данных, заданию значений элементов управления и обновления данных на основе введенных пользователем значений.

Кроме того, с появлением ASP.NET 2.0 стали доступны новые элементы управления «Data Source Controls». Такой элемент управления позволяет декларативно задать связь между aspx-страницей и источником данных (БД или произвольным компонентом доступа к данным). Настроив такой элемент, можно присоединить его к другим элементам управления на странице при ее создании в дизайнере. Остальные детали привязки к данным ASP.NET выполнит автоматически.

Такой подход позволяет достигнуть отделения представления данных (элементов управления) от логики приложения.

В данной работе технология Data Binding используется в основном для просмотра списков различных объектов (не только объектов METAS). При этом используется стандартный компонент GridView, который позволяет задать элемент источника данных (Data Source Control) и настроить внешний вид при помощи стандартных свойств или при помощи шаблонов. Компонент GridView, помимо отображения данных в виде таблицы, может генерировать дополнительные ячейки, в которых располагаются кнопки для выделения, редактирования и удаления строк таблицы. Кроме того, при помощи шаблонов GridView удается реализовать добавление строк в таблицу.

При выборке, создании, редактировании и удалении объектов компонент обращается за выполнением операций к указанному в нем источнику данных. После выполнения команд, изменяющих итоговую таблицу, данные запрашиваются заново.

Кроме того, возможен просмотр таблицы в постраничном режиме и сортировки данных по отдельным полям.

Компонент GridView, в частности, используется для отображения списка страниц объектов. При этом в качестве источника данных используется класс EntityListDB, который предоставляет компоненту следующие методы:

- *EntityListCount* – метод получения количества отображаемых объектов, который используется для разбивки итоговой таблицы на страницы;
- *GetEntityList* – метод получения списка объектов.

Каждый метод требует передачи в качестве параметров идентификатора сущности отображаемых объектов, параметры вершины, к которой привязаны объекты, и настройки фильтрации. Кроме того, метод GetEntityList получает на вход номер первого отображаемого объекта (при разбивке таблицы на страницы)

и максимальное количество объектов, отображаемых на текущей странице таблицы. При подсчете общего количества отображаемых объектов и получении списка объектов учитываются права пользователя на просмотр отдельных объектов (запрещенные для просмотра объекты в список не попадают).

Генерация форм редактирования объектов

При генерации Web-форм редактирования (при выполнении операции Edit) аналогично Windows-формам используются метаданные презентационного уровня.

Метаданные презентационного уровня задают соответствие между формами редактирования объектов и отображенными на них сущностями. При этом на одной и той же форме возможно представление сразу нескольких сущностей. Отображаемые сущности включают главную сущность формы и, возможно, родительские по отношению к ней. Родительские сущности распределяются по уровням. Родителями 0-го уровня считаются те сущности, связи с которыми непосредственно указаны в главной сущности формы. Родители 1-го уровня – это непосредственные родители сущностей 0-го уровня. В общем случае, родители *n*-го уровня – это непосредственные родители сущностей (*n*–1)-го уровня.

Отображение сущностей на форме редактирования подразумевает генерацию элементов управления, соответствующих атрибутам представляемых сущностей. При этом тип элемента управления зависит от типа атрибута. Если тип атрибута – текстовый, то типом элемента управления будет *TextBox*. Если тип атрибута – число месяца, то в качестве типа элемента управления будет выступать *DateTimePicker*. Представление родителей может выполняться двумя способами:

- отображение родителя в списке однотипных объектов, задаваемых своим презентационным полем,
- отображение всех родительских атрибутов по отдельности.

Распределение элементов управления на форме, соответствующих атрибутам сущностей, выполняется между несколькими типами группирующих элементов управления. В первую очередь каждый элемент привязывается к странице данных формы. Затем для него задается группа данных на странице. Между страницами на форме существует отношение порядка. Упорядоченные страницы отображаются в направлении слева направо.

Используемые классы

В данной работе используются следующие классы логического уровня метаданных:

- *Entity* – описание сущности предметной области;
- *Attribute* – атрибут сущности;
- *EntityRelation* – отношение между сущностями предметной области;
- *EntityObject* – экземпляр сущности, объект данных предметной области. Используется для загрузки данных и отражает состояние какой-либо записи в таблице БД;
- *AttributeValue* – значение атрибута сущности в объекте данных;
- *LogicalModel* – логическая модель метаданных.

В данной работе используются следующие классы презентационной модели для представления *экранных форм* при вводе, модификации и просмотре данных:

- *EntityForm* – описание экранной формы;
- *DataPage* – страница данных на форме;
- *DataGroup* – группа данных на странице формы;
- *AttributeControl* – описание элемента управления, соответствующего атрибуту;
- *ShownEntity* – элемент, соответствующий отображенной на форме сущности;
- *ShownEntityRelation* – связь отображенных на форме элементов *ShownEntity*.

Класс *EntityForm* содержит ссылку на сущность, для которой строится форма, и информацию для отображения формы на экране пользователя (размеры и местоположение формы, заголовок, режим последнего открытия формы). Содержит коллекцию элементов *DataPage* расположенных на форме.

Классы *ShownEntity* и *ShownEntityRelation* представляют логическую структуру формы EntityForm с точки зрения отображенных на ней сущностей и взаимосвязей между ними. Класс ShownEntity содержит ссылки на объект *Entity* и объект *EntityRelation*, которые определяют, какая сущность входит в структуру формы и какое отношение связывает указанную сущность с сущностью, являющейся родительской в дереве структуры формы *EntityForm*. Класс *ShownEntityRelation* ссылается на следующий в дереве объект *ShowEntity*.

Класс *DataPage* определяет страницу данных (вкладки формы) при визуализации этой формы. Содержит коллекцию элементов *DataGroup*, расположенных на описываемой странице данных. Каждый объект этого класса всегда имеет в этой коллекции хотя бы один элемент *DataGroup*, который

соответствует невидимой пользователем группе данных. Этой вспомогательной группе принадлежат элементы управления, которые пользователь считает расположенными прямо на странице данных.

Класс *DataGroup* определяет информацию для отображения группы элементов управления на форме (какой-либо ее странице) при визуализации этой формы, содержит коллекцию описаний элементов управления AttributeControl.

Класс *AttributeControl* содержит информацию о внешнем виде элемента управления, представляющего на форме атрибут сущности или связь с другой сущностью, и о связанном с элементом управления заголовке. Ссылается на атрибут сущности и/или связь сущностей, которые представляет на форме объект *AttributeControl*. Также содержит ссылку на объект *ShownEntity*, которому принадлежит элемент управления. Элементы управления для родительских атрибутов и атрибутов, значения которых выбираются из списков-справочников, используется поле *SortListValue*, определяющее необходимость сортировки значений списка.

Форма редактирования

В момент обращения к форме редактирования к отображаемым сущностям привязываются объект главной сущности и соответствующие связям главной сущности родительские объекты. Затем на форму загружаются элементы управления и ссылки на родительские и дочерние объекты. Ссылки на родительские и дочерние объекты добавляются только в случае, если текущий пользователь имеет право на просмотр соответствующих объектов.

Пример формы редактирования объекта можно увидеть на рис. 2.19.

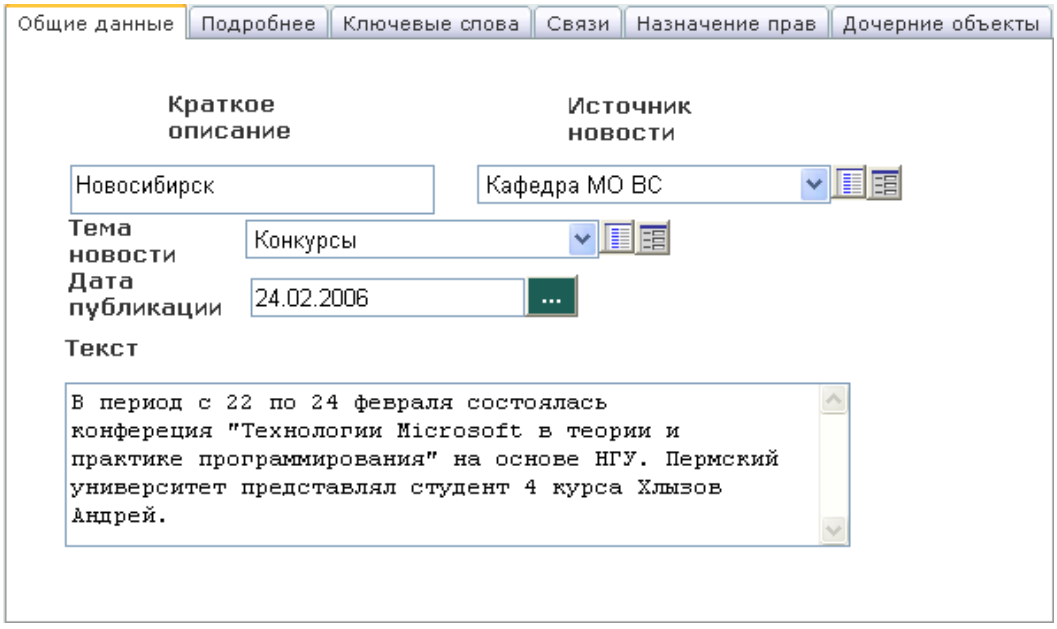

Рис. 2.19. Форма редактирования объекта сущности «Новость»

При редактировании объектов выполняется синхронизация атрибутов отображаемых объектов, связи между которыми задаются метаданными логической модели, и проверка ограничений на значения атрибутов.

Редактирование родительских атрибутов, за исключением выбора родителя из списка, разрешено только в том случае, если текущий пользователь является владельцем родительского объекта.

Более подробный алгоритм инициализации формы приведен на рис. 2.20.

При сохранении объекта сначала фиксируются изменения в родительских объектах, а затем уже в объекте главной сущности. Таким образом, сначала сохраняются объекты n-го уровня, затем (*n*–1)-го, …, 0-го уровня и в конце объект главной сущности формы. Алгоритм сохранения представлен на рис. 2.21.

Алгоритм сохранения родительских объектов отличается от вышеприведенного лишь тем, что в случае связи «1:M» вне зависимости от типа операции будет произведено добавление родительского объекта в БД.

Иерархия элементов управления

Как было сказано выше, каждому типу атрибута ставится в соответствие тип элемента управления (типы элементов управления для ввода текстовой информации, целых и вещественных чисел, логических значений и т.д.). Тип элемента управления описывается классом *AttributeControl* презентационной модели. Этот класс задает тип и расположение элемента управления на форме, его ширину и длину, а также расположение, длину, ширину и название заголовка.

Поскольку в распоряжении разработчика ИС находится ограниченный набор типов и, соответственно, элементов управления для ввода значений этих типов, а требования к ИС постоянно меняются, необходим гибкий механизм определения новых типов (например, были реализованы типы элементов управления «Классификатор» и «Файл»).

Общие для всех элементов управления свойства и методы задаются в классе WMC_*BaseControl*, от которого элементы наследуют (рис. 2.22). Ниже перечислены свойства данного класса:

- система защиты,
- текущий пользователь,
- атрибут чтения,
- список всех элементов управления формы,
- объект-владелец атрибута,
- экземпляр класса AttributeControl, описывающий внешний вид элемента.

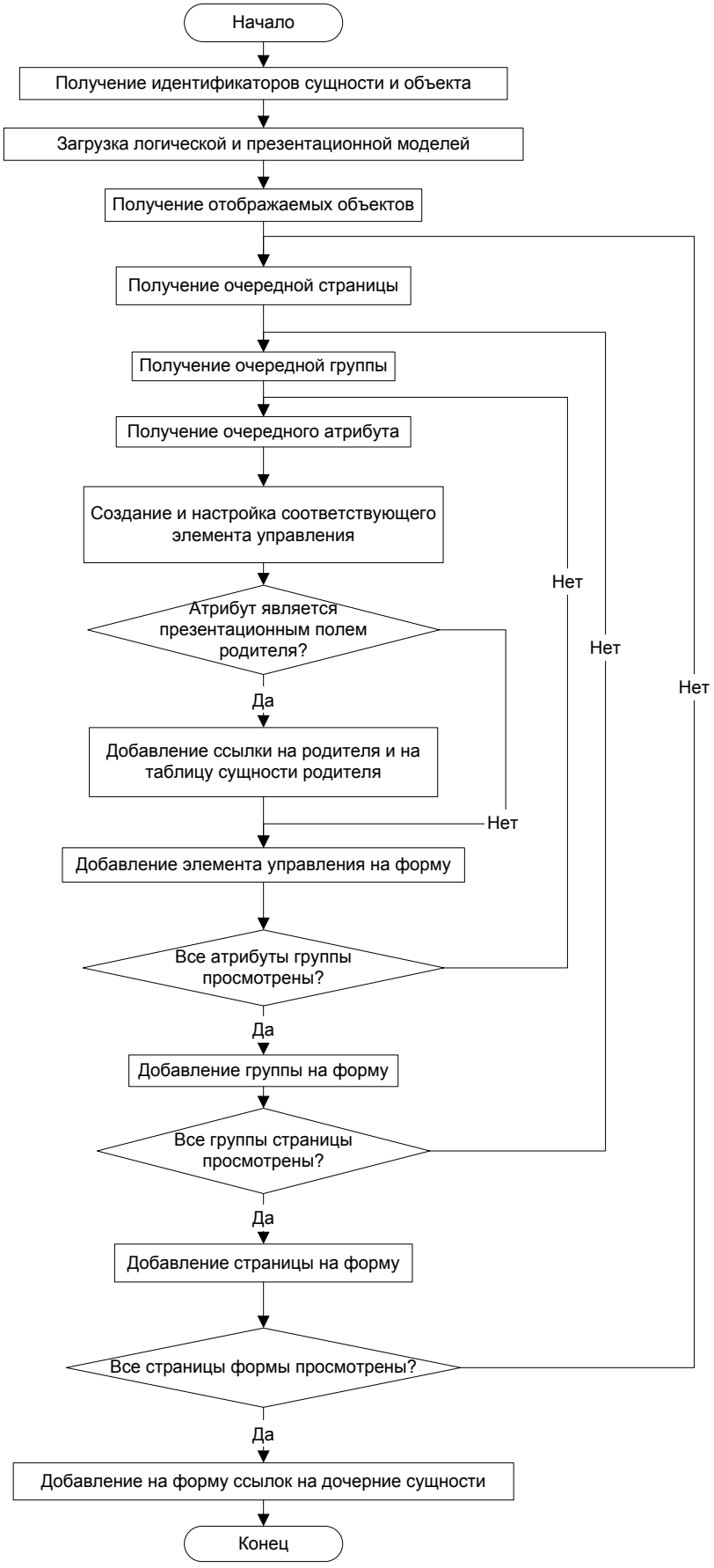

Рис. 2.20. Алгоритм инициализации формы

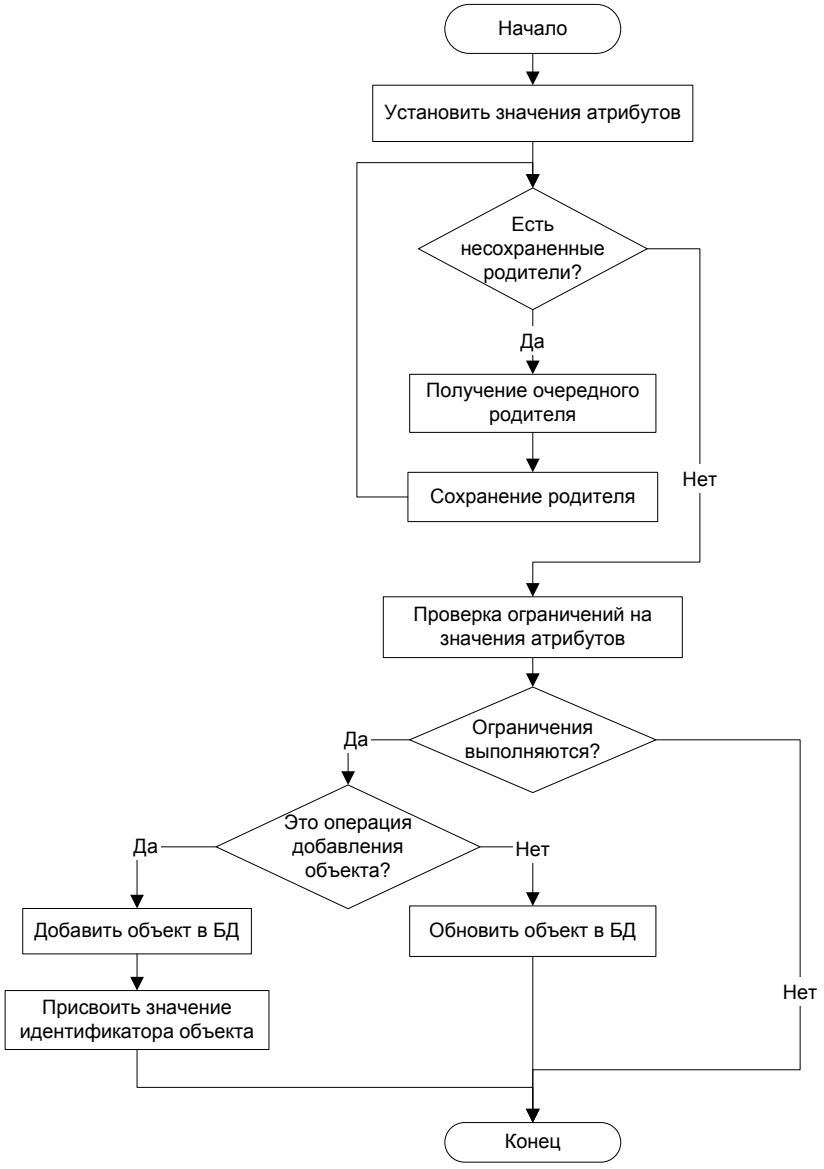

Рис. 2.21. Алгоритм сохранения главной сущности

Общими методами элементов управления, заданными в классе *WMC*_BaseControl, являются:

- инициализация,
- установка значения элемента управления,
- установка значения атрибута объекта,
- получение значения атрибута объекта.

Данные методы переопределяются в наследующих классах в целях преобразования передаваемых значений к соответствующему типу.

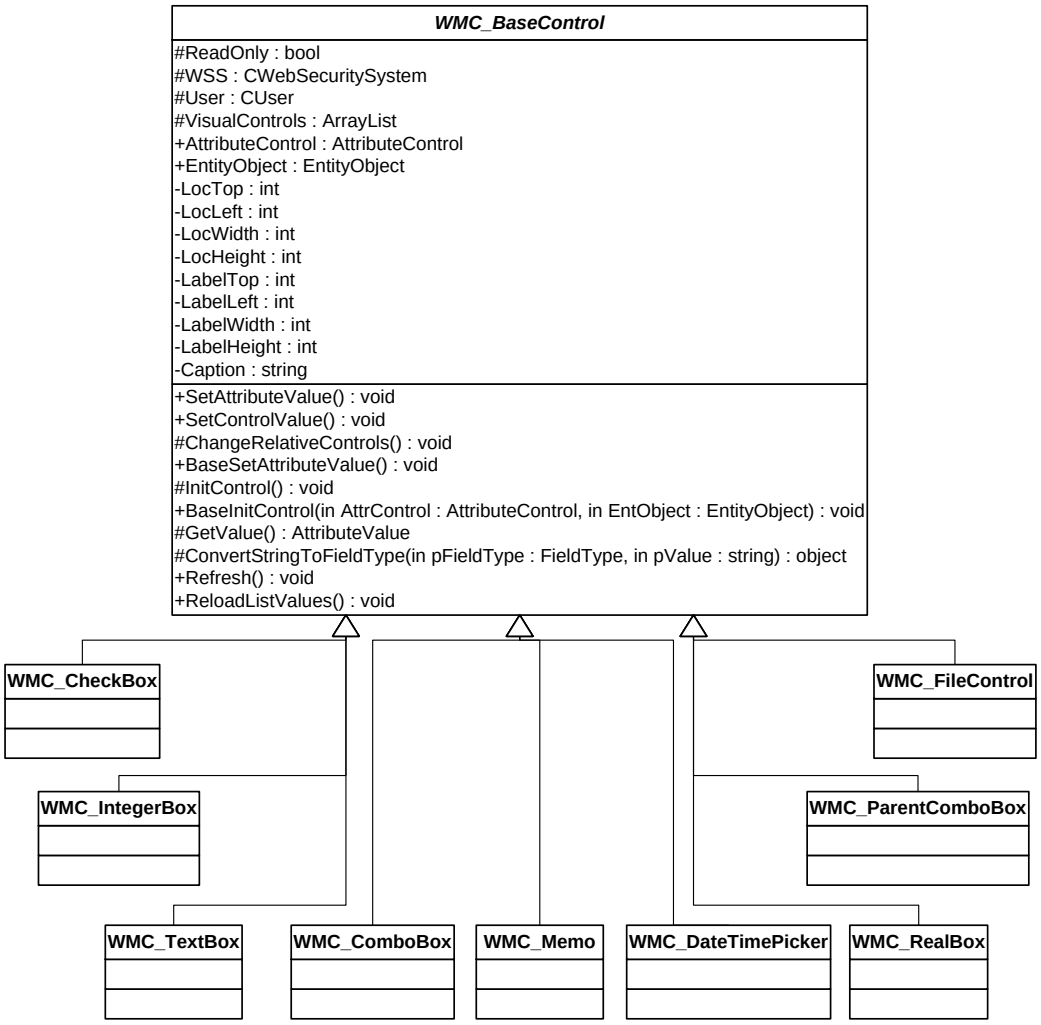

Рис. 2.22. Иерархия элементов управления

В процессе инициализации формы в зависимости от типа, указанного в AttributeControl, выбирается пользовательский элемент управления, производится его динамическая загрузка и создание экземпляра. Затем выполняется его инициализация, преобразование к необходимому типу и установка значения элемента управления. Инициализация элемента управления включает задание расположения, длины, ширины, блокирование, если форма была открыта в режиме только для чтения. В случае если атрибут является презентационным полем родителя, то при наличии соответствующих разрешений пользователя размещаются ссылки на форму редактирования родителя и таблицу его сущности.

При сохранении производится преобразование введенных значений к соответствующему типу данных, установка флага изменения значения и изменение значений связанных элементов управления.

Расширение возможностей интерфейса за счет создания новых элементов управления

На уровне интерфейса реализованы элементы, которые являются стандартными и для Windows-, и для Web-интерфейса (например, поле ввода, список и пр.). Для организации работы со сложно структурированными данными разработаны новые элементы управления (например, для работы с иерархическими классификаторами, файлами и пр.). В данной работе реализована поддержка типа «Файл».

Файл представляет собой тип данных атрибута некоторой сущности. В качестве такого атрибута сущности может выступать любой файл. Файл хранится как текстовое поле. Для занесения в базу данных файл сначала преобразуется в соответствии с форматом SOAP, а затем записывается в поле Value атрибута сущности.

Над файлом можно выполнять следующие операции: изменение, удаление, просмотр, сделать архивным (неизменяемым).

При инициализации данного элемента управления определяется, прикреплен ли к сущности какой-нибудь файл. Если файла нет, то в поле для отображения имени файла выводится «Нет документа». Если файл есть, то в зависимости от того, является ли он архивным, становятся доступными те или иные операции над ним. Архивный файл можно только просматривать.

Перед просмотром файл сохраняется на сервере в каталоге временных файлов. Затем осуществляется переход на сохраненный файл.

При инициализации элемента управления, изменении или удалении файла содержимое файла запоминается во ViewState для фиксирования изменений, поскольку при перезагрузке страницы ASP.NET не сохраняет созданные во время предыдущей загрузки объекты.

Модуль поиска

Данный модуль предназначен для поиска объектов портала по ключевым словам. Пользователь вводит в строку поиска, расположенной в отдельной Web-части, ключевые слова искомых объектов, и после нажатия кнопки поиска в главном окне выдаются все найденные объекты. Для каждого объекта отводится одна строчка, в которой указывается значение презентационного выражения объекта в качестве заголовка, являющегося ссылкой его просмотр, и списковое представление объекта (рис. 2.23).

Рис. 2.23. Результат работы модуля поиска

Для вывода результатов поиск используется вышеописанный компонент привязки к данным *GridView*.В качестве источника данных для него выступает класс *SearchDB*, который содержит методы *SearchItemsCount* для получения общего количества найденных объектов и *GetSearchItems* для получения списка отображаемых в текущей странице результата поиска.

На уровне БД для хранения ключевых слов используются две таблицы. В таблице «Keywords» хранятся ключевые слова и их идентификаторы. В таблице «Entity keywords» задается связь объектов портала и ключевых слов (для этого таблица включает три поля: *EntityId* и *ObjectId* для обозначения объекта и *WordId* – для обозначения ключевого слова). При добавлении ключевых слов отслеживание уникальности слов в таблице без учета регистра, т.е. гарантируется, что для каждого ключевого слова существует ровно одна запись в таблице и ему соответствует единственное значение ключевого поля. Это позволяет свести к минимуму размер таблицы и повысить скорость поиска объектов.

Добавить или удалить ключевые слова объекта можно на его форме редактирования, однако эти операции могут выполнять только те пользователи,

которые обладают соответствующим правом редактирования.

При поиске объекты сортируются по убыванию степени релевантности ключевым словам, т.е. выше оказываются те объекты, для которых было найдено больше соответствий между ключевыми словами, введенными пользователем в строке поиска, и ключевыми словами, указанными в объекте.

Следует сказать, что для поиска доступны только те объекты, по отношению к которым текущий пользователь имеет право просмотра.

Модуль релевантных ссылок

Данный модуль относится к модулям адаптации навигации. Он реализует метод глобального руководства, назначение которого состоит в том, чтобы помочь пользователю сориентироваться в окружающем его гипермедиа пространстве.

Релевантная ссылка – это ссылка на объект, который явно или неявно имеет отношение к открытому в главном окне объекту. Например, для объекта сущности «Кафедра» релевантными могли бы быть ссылки на объекты сущности «Преподаваемая дисциплина» или «Направление научной деятельности».

Для хранения релевантных ссылок в БД используется таблица «ObjectReferences» с полями *SourceEntID*, *SourceObjID*, *DestEntID* и *DestObjID*. Первые два поля отвечают за представление объекта (идентификаторы сущности и объекта), для которого назначаются ссылки, а вторые два – за представление объекта, к которому она ведет.

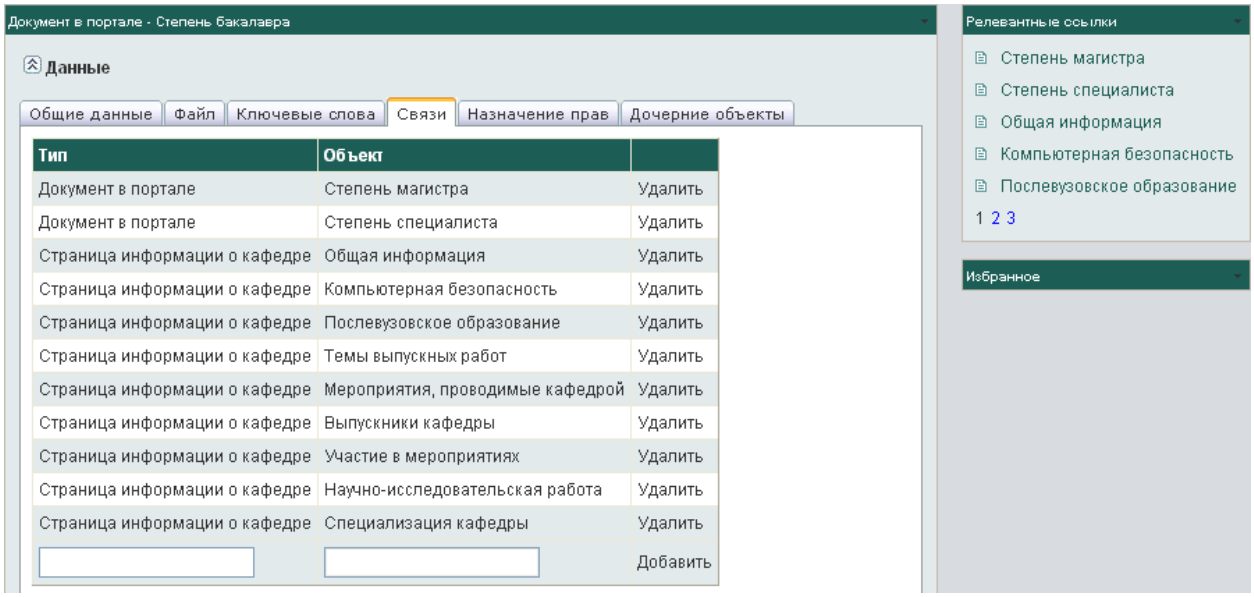

Рис. 2.24. Назначение релевантных ссылок и Web-часть для перехода по ним

Релевантные ссылки для объекта назначаются на вкладке «Связи» его формы редактирования (рис. 2.24). Добавление ссылки выполняется в режиме предикативного ввода. Сначала вводится название сущности релевантного объекта, а затем презентационное выражение самого объекта. Предикативный ввод подразумевает предложение пользователю выбрать один из доступных вариантов (название сущности или объекта) по уже введенным первым буквам названия. После ввода обоих параметров выполняется поиск идентификаторов сущности и объекта для добавления релевантной ссылки.

Для перехода по релевантным ссылкам открытого в данный момент времени объекта служит Web-часть «Релевантные ссылки» (рис.2.24).

Для отображения и назначения ссылок на форме редактирования и представления их на Web-части перехода к ним используется компонент GridView, источником данных для которого является объект класса ObjectReferenceDB. Данный класс имеет следующие методы:

- GetRefCount (метод получения общего количества релевантных ссылок объекта),
- GetObjectReferences (метод получения релевантных ссылок объекта текущей страницы GridView),
- InsertObjectReference (метод добавления релевантной ссылки объекта),
- DeleteObjectReference (метод удаления релевантной ссылки объекта).

Предикативный ввод осуществляется при помощи обращения к выделенному Web-сервису, который при вводе сущности выдает список названий доступных текущему пользователю сущностей, а при вводе объекта выдает список названий всех доступных текущему пользователю объектов уже введенной сущности.

Модуль закладок

Модуль закладок, так же, как и модуль релевантных ссылок, предназначен для адаптации навигации. Он отличатся тем, что не предоставляет пользователю автоматически список релевантных объектов, а позволяет пользователю самому ввести ссылки на объекты, к которым ему нужно обращаться время от времени. Данная функциональность напоминает закладки браузеров (например, Internet Explorer, Opera или Mozilla), однако закладки назначаются только для объектов портала.

Закладка хранит в себе операцию и атрибуты этой операции. Например, закладка просмотра объекта включает в себя название операции «View», а также соответствующие идентификаторы сущности и объекта. Закладка списка страниц объектов представляется операцией «List», идентификатором сущности отображаемых объектов, условиями фильтрации и т.д.

Для хранения закладок в БД используется таблица «Bookmarks». Она содержит следующие поля:

– *Id* (уникальный идентификатор закладки),
– *UserId* (идентификатор пользователя, добавившего закладку),
– *Type* (операция закладки),
– *Attributes* (атрибуты операции закладки),
– *Title* (заголовок закладки, в случае операции *Edit* или *View* – презентационное выражение объекта).

Добавить страницу портала в закладки можно при помощи специальной кнопки меню Web-части главного окна под названием «Добавить в избранное». Для страниц, уже добавленных в закладки, эта кнопка недоступна.

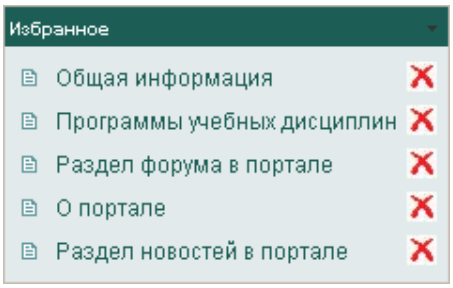

Рис. 2.25. Представление Web-части «Избранное»

Перейти по ссылкам, сохраненным в закладках, или удалить закладку из этого списка можно при помощи Web-части «Избранное». Для отображения и манипулирования списком закладок Web-часть использует компонент GridView, источником которого является объект класса BookmarkDB. Данный класс поддерживает методы для получения количества всех закладок текущего пользователя, получения закладок для текущей страницы GridView, добавления и удаления закладок, а также проверки, является ли заданная страница закладкой.

Модуль управления Web-частями

Данный модуль предназначен для управления составом отображаемых Web-частей, их расположения и параметров. Параметрами могут быть, к примеру, количество отображаемых закладок в Web-части «Избранное», глубина раскрытия карты сайта и т.д.

Модуль практически полностью реализуется за счет стандартных возможностей ASP.NET по управлению Web-частями. С точки зрения функциональности самой Web-части реализован выбор одного из трех режимов просмотра страницы *default.aspx*:

– просмотр,
– дизайн,

– каталог.

Режим «Просмотр» – обычный способ отображения страницы. В нем присутствуют простейшие возможности по работе с Web-частями, а именно их закрытие и сворачивание/разворачивание.

Режим «Дизайн» отличается от предыдущего возможностью изменения расположения Web-частей в браузере (2.26). При выборе этого режима рамкой выделяются три зоны расположения Web-частей – левая, центральная и правая. При помощи перетаскивания мышью, ухватившись за заголовок формы, пользователь может перемещать Web-части из одной зоны в другую, выбирая при этом позицию Web-части в зоне.

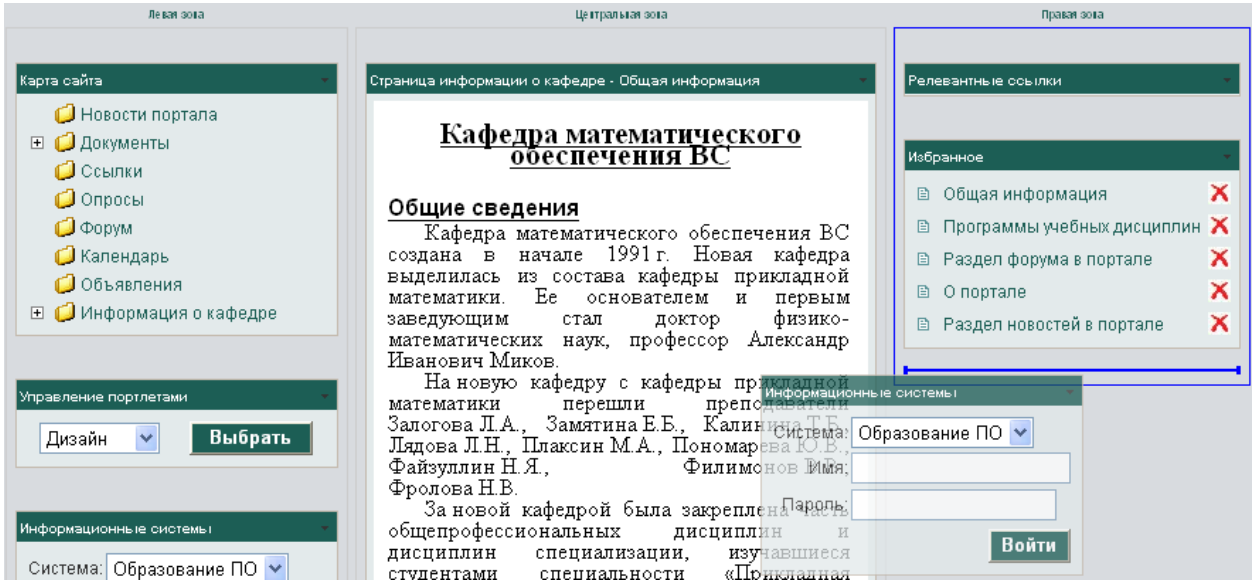

Рис. 2.26. Изменение расположения Web-частей в режиме «Дизайн»

Режим «Каталог» имеет те же возможности, что и режим «Дизайн». Однако в этом режиме отображается еще одна Web-часть – каталог Web-частей. При помощи этой Web-части можно выбирать, в какие зоны помещать скрытые в данный момент Web-части (рис. 2.27).

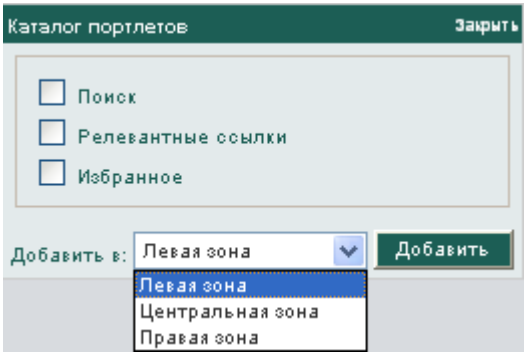

Рис. 2.27. Каталог Web-частей (портлетов)

Имеется ограничение на работу с Web-частями: невозможно закрывать Web-части главного окна и управления Web-частями.

2.4. Применяемые программные средства

Приложение WebMETAS является составной частью CASE-системы METAS. Программное ядро CASE-системы METAS (MDK METAS) позволяет настраиваться на различные программные платформы, работать под управлением различных операционных систем Microsoft Windows, использовать для создания информационных систем различные реляционные СУБД и источники данных, для которых существуют драйверы ODBC.

Связь с моделями и программными компонентами программного ядра MDK METAS показана на рис. 2.28.

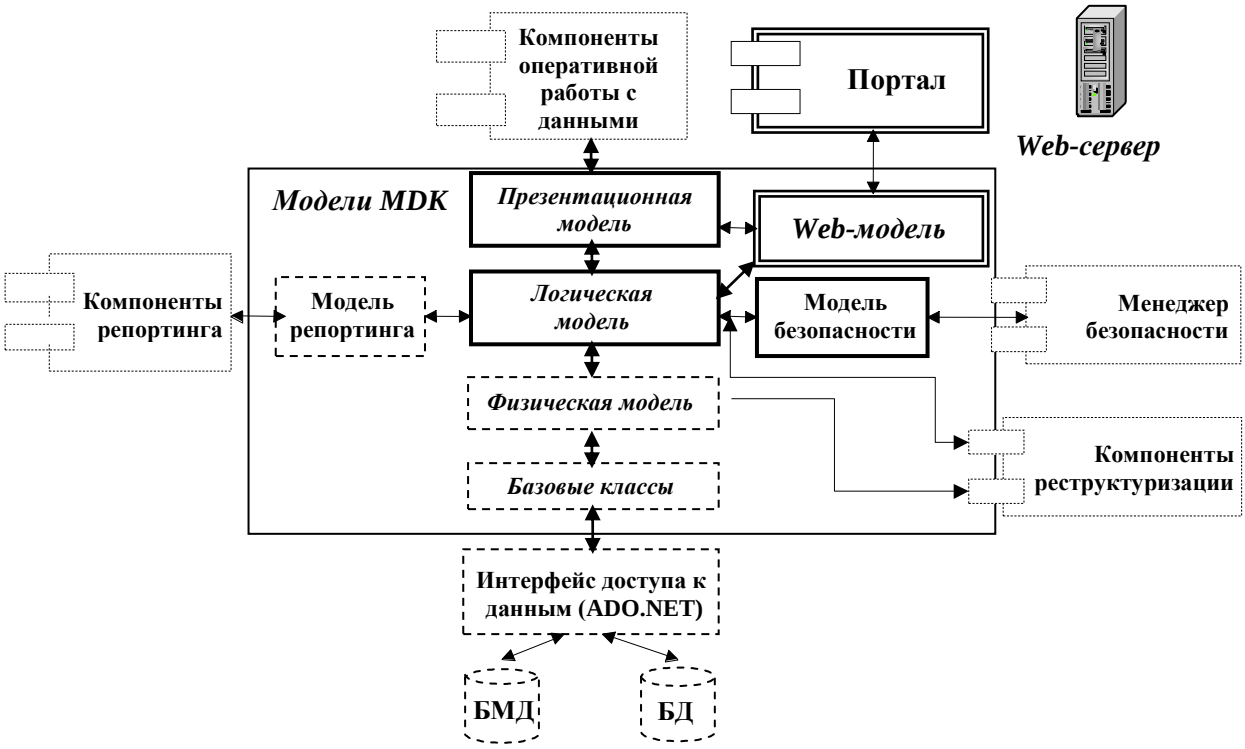

Рис. 2.28. Схема многоуровневой организации метаданных METAS

Для разработки приложений использована инструментальная среда Microsoft Visual Studio .NET.

Для функционирования runtime-компонентов необходимо установить .NET Framework (распространяется бесплатно), драйверы ODBC (при установке операционной системы) и СУБД (можно использовать, в частности, Microsoft SQL Server Express, которая распространяется бесплатно), сервер IIS. В основе портала лежит технология Microsoft ASP.NET 2.0. В данной разработке использовались следующие компоненты ASP.NET: встроенная модель безопасности, аутентификации и авторизации пользователей (Security Model), привязка к данным (Data Binding), Web-части (Web Parts), пользовательские элементы управления (User Controls).

2.5. Используемые технические средства и требования к аппаратуре

Требования к аппаратуре определяются требованиями, соблюдение которых необходимо для установки используемого программного обеспечения.

Минимальные требования: процессор Intel Celeron, 128 Мб ОЗУ.

3. Специальные условия применения и требования организационного, технического и технологического характера

Для работы с программами *нет необходимости создания специальных условий применения и выполнения особых требований организационного, технического и технологического характера*. Условия применения и требования определяются требованиями к применяемому программному и аппаратному обеспечению, перечисленными выше, а также выполнением лицензионных соглашений при использовании необходимого для работы программ программного обеспечения.

Требования к квалификации пользователей определяются выполняемыми ими обязанностями. Минимальные требования:

- навыки работы в среде Microsoft Windows.

Разработчики и администраторы, создающие портал и его компоненты, выполняющие его настройку, *должны удовлетворять следующим квалификационным требованиям*:

- знания и навыки в области проектирования реляционных баз данных;
- навыки администрирования реляционных СУБД;
- навыки Web-программирования.

4. Условия передачи программной документации или ее продажи

Продажа и передача программ и программной документации возможна на основе договора с АНО науки и образования «Институт компьютинга». Условия передачи и/или продажи полностью определяются договором, заключаемым заказчиками/покупателями с АНО «Институт компьютинга».

Представителем АНО «Институт компьютинга», имеющим право заключать договоры на передачу и/или продажу программного обеспечения, на разработки программ с использованием представленных в данном документе средств, является заместитель директора Лядова Л.Н. (e-mail: lnlyadova@mail.ru; тел.: +7 (342) 222-37-95).

Библиографический список

1. *Елманова Е.* Web-порталы: классификация и назначение / Е. Елманова [Электронный ресурс] (http://www.compress.ru/article.aspx?id=10932&iid=439).
2. *Елманова Н.* Web-порталы: назначение, преимущества, особенности и средства / Н. Елманова [Электронный ресурс] (http://www.interface.ru/misc/mnogoe_1.htm).
3. *Файфер Д.* Корпоративная портализация электронного бизнеса / Д. Файфер, Т. Кемпф, М. Краммер // Сетевой журнал №3.2001 [Электронный ресурс] (http://www.setevoi.ru/cgi-bin/text.pl/magazines/2001/3/78).
4. *Троелсен Э.* C# и платформа .NET. / Э. Троелсен // СПб.: Питер, 2002.
5. *Шапошников И.В.* ASP.NET / И.В. Шапошников // СПб.: БХВ-Петербург, 2002.
6. *Акопянтц А.* Что такое портал? / А. Акопянтц [Электронный ресурс] (http://akop.ru/personal/4847).
7. Что делает портал порталом? [Электронный ресурс] (http://www.perm-dom.ru/modules.php?op=modload&name=Reviews&file=index&req=showcontent&id=1).
8. *Галкин Г.* Такие разные корпоративные порталы / Г. Галкин // Сетевой журнал №5.2002 [Электронный ресурс] (http://www.setevoi.ru/cgi-bin/text.pl/magazines/2002/5/54).
9. *Галкин Г.* Изменить лицо IT / Г. Галкин // Сетевой журнал №1.2001 [Электронный ресурс] (http://www.setevoi.ru/cgi-bin/text.pl /magazines/2001/1/40).
10. *Phifer G.* Portals Provide a Fast Track to SOA / G. Phifer [Электронный ресурс] (http://www.bijonline.com/index.cfm?section=article&aid=82).
11. Сервис-ориентированная архитектура [Электронный ресурс] (http://www.citforum.ru/internet/webservice/soa/).
12. *Лехманн М.* Соединяя Web-сервисы (Weaving Web Services Together) / М. Лехманн [Электронный ресурс] (http://citforum.ru/internet/webservice/weav_web/).
13. WebSphere Portal. Обзор [Электронный ресурс] (http://www-306.ibm.com/software/info/ecatalog/ru_RU/products/R106329H88430D48.html?&S_TACT=none&S_CMP=none).
14. *Эдди С. Э.* XML Справочник / С. Э. Эдди // СПб.: Питер, 2000.
15. [Электронный ресурс] (http://www.citforum.ru/internet/xslt/xslt.shtml#section-Introduction).
16. [Электронный ресурс] (http://knowhow.virtech.ru/asp/default.asp).
17. [Электронный ресурс] (http://www.dotsite.ru/Tutorials/ASP.NET).
18. *Рейли Д. Дж.* Создание приложений Microsoft ASP.NET / Д. Дж. Рейли // М.: Русская редакция, 2002.

19. *Жарикова Е.* Выбор системы управления контентом интернет-ресурс / Е. Жарикова [Электронный ресурс] (http://www.ibusiness.ru/marcet/tele/20905/).
20. Сайт разработчика системы ABO.CMS [Электронный ресурс] (http://www.abocms.ru/).
21. Сайт разработчика системы UMI.CMS [Электронный ресурс] (http://www.umi-cms.ru/).
22. Аналитический портал коммерческих систем управления сайтом [Электронный ресурс] (http://www.cmsmagazine.ru/).
23. *Hongjing W.* A Reference Architecture for Adaptive Hypermedia Systems / W. Hongjing [Электронный ресурс] (http://wwwis.win.tue.nl/ah2001/papers/wu.pdf).
24. *Брусиловский П. Л.* Технологии и методы адаптивной гипермедиа / П.Л. Брусиловский // User Modeling and User Adapted Interaction, 1996, v 6, n 2-3, стр. 87-129.
25. *Brusilovsky P.* Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies / P. Brusilovsky // Proceedings of Workshop "WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems (ITS'98), San Antonio, TX, August 16-19, 1998.
26. *Mathé N.* User-centered indexing for adaptive information access / N. Mathé, J. Chen // User Models and User Adapted Interaction 6, 1996.
27. *Kaplan C.* Adaptive hypertext navigation based on user goals and context / C. Kaplan, J. Fenwick, J. Chen // User Models and User Adapted Interaction 3 (3), 1993. Pp. 193-220.
28. *Hohl H.* Hypadapter: An adaptive hypertext system for exploratory learning and programming / H. Hohl, H.-D. Böcker, R. Gunzenhäuser // User Models and User Adapted Interaction 6, 1996.
29. *Grunst G.* Adaptive hypermedia for support systems / G. Grunst // Adaptive user interfaces: Principles and practice. Amsterdam: North-Holland, 1993. Pp. 269-283.
30. *Vassileva J.* A task-centered approach for user modeling in a hypermedia office documentation system / J. Vassileva // User Models and User Adapted Interaction 6, 1996.
31. *Boy G. A.* On-line user model acquisition in hypertext documentation / G. A. Boy // Agent Modeling for Intelligent Interaction, Sydney, Australia, 1991. Pp. 34-42.
32. *Höök K.* A glass box approach to adaptive hypermedia / K. Höök, J. Karlgren, A. Warn, N. Dahlbäck, C. G. Jansson, K. Karlgren,. and B. Lemaire // User Models and User AdaptedInteraction 6, 1996.
33. *Brusilovsky P.* ISIS-Tutor: An adaptive hypertext learning environment / P. Brusilovsky, L. Pesin // JCKBSE'94, Japanese-CIS Symposium on knowledge-based software engineering, Pereslavl-Zalesski, Russia, 1994. Pp. 83-87
34. *Thomas C. G.* Using agents to improve the usability and usefulness of the World-Wide Web / C.G. Thomas, G. Fischer // Fifth International Conference on User Modeling, UM-96, Kailua-Kona, Hawaii, 1996. Pp. 5-12.

35. *Lei Y.* An Ontology-based Approach to Web Site Design and Development / Y. Lei, E. Motta, J. Domingue [Электронный ресурс] (http://kmi.open.ac.uk/people/yuangui/resources/thesis/).

36. WebML User Guide version 3.0 [Электронный ресурс] (http://www.webml.org).

37. *Лядова Л. Н.* CASE-технология METAS / Л. Н. Лядова, С. А. Рыжков // Межвузовский сб. науч. трудов / Математика программных систем. Пермь: Перм. ун-т. 2003. С. 4-19.

38. *Борисова Д. А.* Компонент реструктуризации CASE-системы METAS / Д. А. Борисова // Межвузовский сб. науч. трудов / Математика программных систем. Пермь: Перм. ун-т. 2003. С. 34-43.

39. *Савельева Н.* Системы управления контентом / Н. Савельева [Электронный ресурс] (http://cmslist.ru/articles/cms-review).

40. *Борисова Д.А.* Иерархическая модель данных как основа реализации информационной системы, управляемой метаданными / Д. А. Борисова, Л. Н. Лядова // Межвузовский сб. науч. трудов / Математика программных систем. Пермь: Перм. ун-т. 2006. С. 4-13.

41. *Куделько Е. Ю.* Модель объектов пользовательского интерфейса METAS / Е. Ю. Куделько, Л. Н. Лядова // Межвузовский сб. науч. трудов / Математика программных систем. Пермь: Перм. ун-т. 2006. С. 40-55.

42. *Хлызов А. В.* Создание динамически настраиваемых web-ориентированных информационных систем / А. В. Хлызов, М. В. Чичагова // Научно-технический рецензируемый журнал «Инженерный вестник 1(21)/1 ‘2006» / Белорусский государственный университет информатики и радиоэлектроники – Минск, 2006. С. 189-192.

43. *Chichagova M.* Access Rights Inheritance in Information Systems Controlled by Metadata / M. Chichagova, L. Lyadova // Information Journal “Information Theories and Applications” Vol. 14, Number 2, 2007. Pp. 189-192.

44. Web дизайн. Design как стиль жизни. История, теория, практика дизайна. [Электронный ресурс] (http://www.rosdesign.com/design/fdesign_2.htm).

45. Методология проектирования программных продуктов. [Электронный ресурс] (www.stu.ru/inform/glaves2/glava18/gl_18_1.html).

46. *Дам Э.* Пользовательские интерфейсы нового поколения / Э. Дам // Открытые системы. №6, 1997. С.34-37.

47. *Куделько Е. Ю.* Генерация и настройка экранных форм на основе метаданных / Е. Ю. Куделько // Математика программных систем: Межвуз. сб. науч. тр. / Перм. ун-т. – Пермь, 2003. С.51-59.

48. *Рыжкова Е. А.* Реализация удаленного доступа к ресурсам информационной системы «Образование Пермской области» / Е. А. Рыжкова, А. В. Хлызов // Информатика в школе: Тезисы докладов X юбилейной областной научно-методической конференции 9-10 января 2006 г. «Рождественские чтения» / Перм. ун-т. – Пермь, 2006. С.86-88.

49. *Хлызов А. В.* Разработка средств создания порталов с использованием многоуровневых метаданных / А. В. Хлызов // Технологии Microsoft в теории и практике программирования. Материалы конференции / Изд-во Нижегородского госуниверситета – Нижний Новгород, 2006. С.310-313.

50. *Рыжкова Е. А.* Генерация Web-интерфейса для динамически настраиваемых информационных систем / Е. А. Рыжкова, А. В. Хлызов // Технологии Microsoft в теории и практике программирования. Материалы конференции / Новосибирский университет – Новосибирск, Академгородок, 2006. С.32-34.

51. *Лядова Л. Н.* Реализация порталов дистанционного обучения на основе CASE-технологии METAS / Л. Н. Лядова, А. В. Урезалов, А. В. Хлызов // Proceedings of the Second International Conference “Modern (e-) Learning” / Sofia, Institute of Information Theories and Applications FOI ITHEA, 2007. С. 153-161.

52. *Lyadova L.* Implementation of distant learning portals based on CASE-Technology METAS / L. Lyadova, A. Urezalov, A. Khlyzov // International Journal “Information Technologies and Knowledge” Vol. 2, 2008. Pp. 489-495.

Приложение А. Руководство пользователя портала

При первом обращении к порталу открывается стартовая страница портала (рис. А.1).

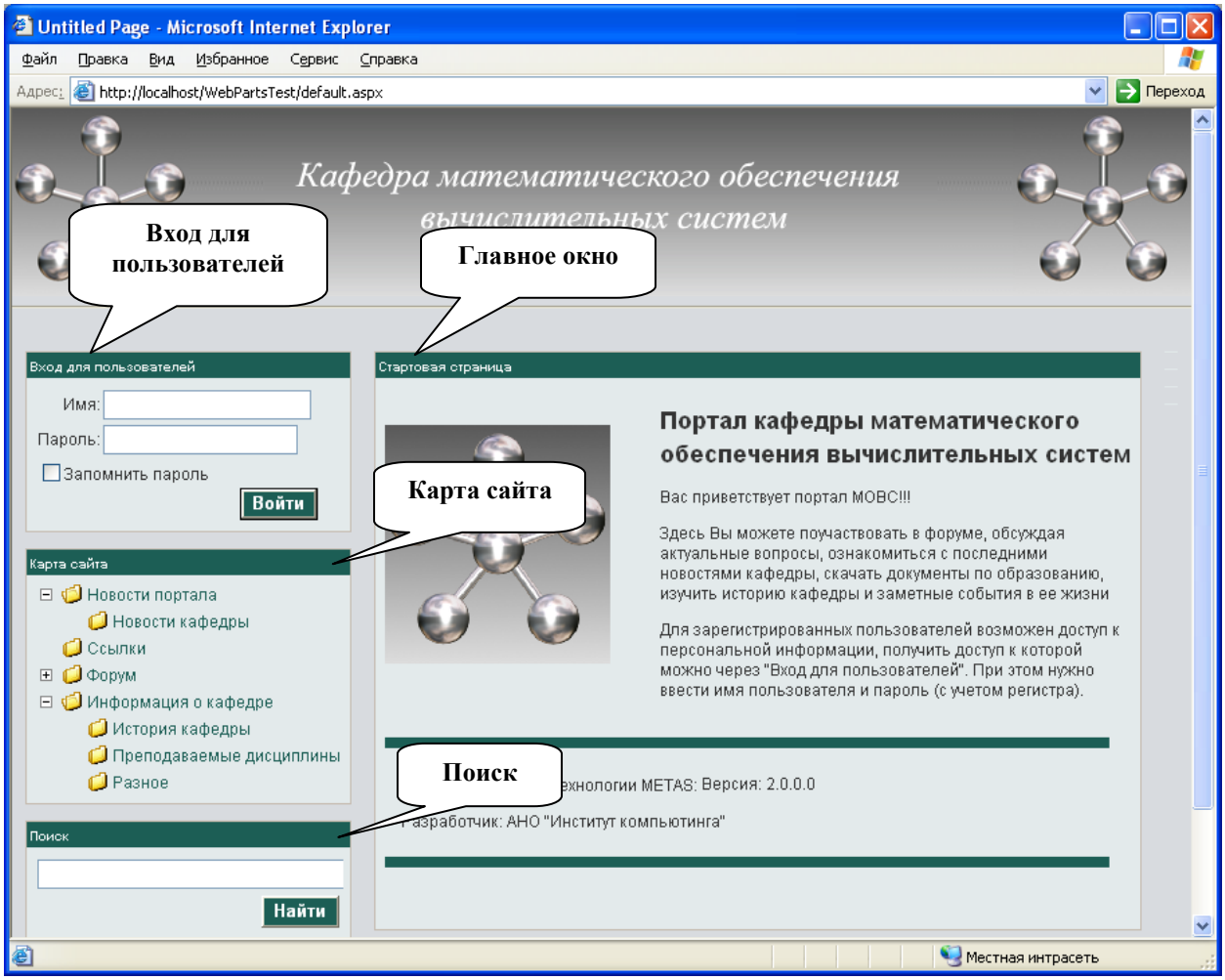

Рис. А.1. Стартовая страница портала

Неаутентифицированный пользователь может просматривать общедоступные ресурсы портала, такие как новости, форум и т.д. и осуществлять поиск среди них.

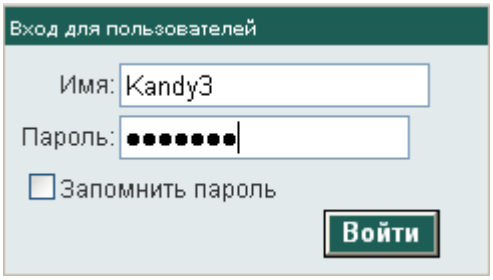

Рис. А.2. Веб-часть аутентификации пользователей

Чтобы получить доступ к другим ресурсам и функциям портала, зарегистрированному в портале пользователю необходимо ввести имя и пароль в веб-части с заголовком «Вход для пользователей» (рис. А.2) и нажать на кнопку «Войти».

В случае успешной проверки данных, предоставленных пользователем, перезагружается карта сайта, отображая доступные пользователю для просмотра ресурсы портала (рис. А.3).

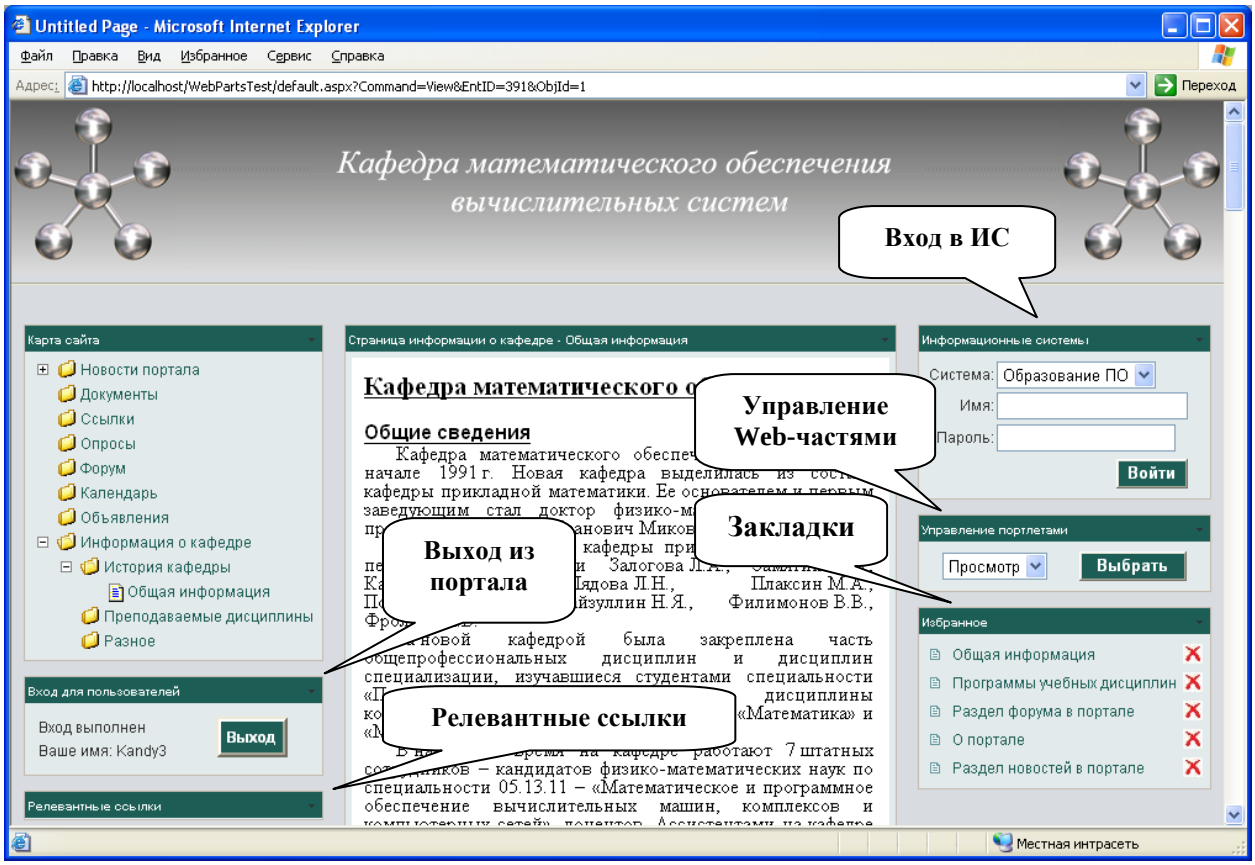

Рис. А.3. Вид портала для аутентифицированного пользователя

Веб-часть аутентификации при этом изменяет свое представление. В новом представлении отображается имя пользователя и кнопка «Выход», щелкнув мышью по которой можно выйти из портала.

Аутентифицированному пользователю доступны также веб-части для входа в информационные системы, зарегистрированные в портале, настройки рабочего места, закладок и релевантных ссылок.

Карта сайта предназначена для навигации по иерархии объектов портала. Для того чтобы отобразить информацию, соответствующую вершине карты сайта, необходимо щелкнуть по ней указателем мыши. После этого в главном окне будет выдана информация по этой вершине.

Если выбранная вершина – объект, то будет выдана страница данного объекта (рис. А.3). Если у объекта имеются дочерние объекты, то они также

будут отображены в карте сайта.

Если выбранная вершина группирует однотипные объекты, то в главном окне будет выдан список страниц объектов (рис. А.4). В карте они будут отображены как дочерние вершины.

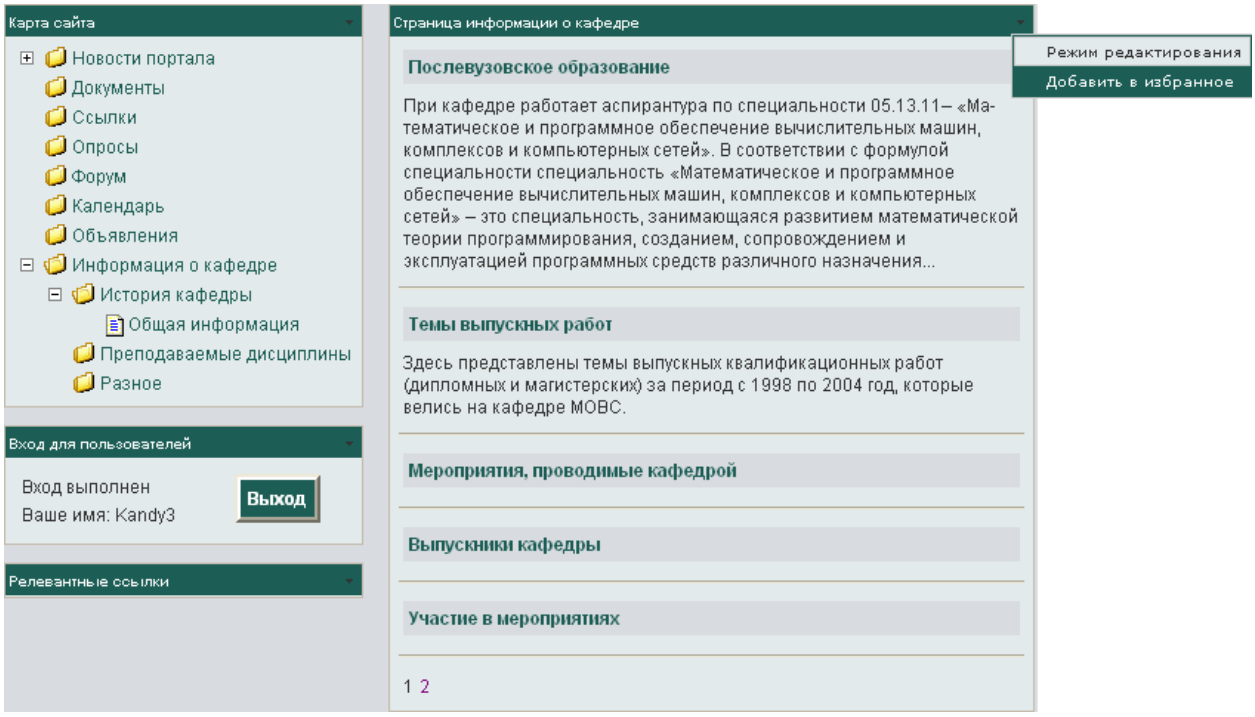

Рис. А.4. Список страниц объектов

Пользователь может добавить страницу со списком в закладки (веб-часть «Избранное»), нажав на кнопку «Добавить в избранное».

Если пользователь обладает правом доступа к объектам типа объектов, отображаемых в списке, то он может перейти к таблице отображаемых объектов при помощи кнопки «Режим редактирования» (рис. А.5). При желании пользователь может переключиться обратно в режим списка, нажав на кнопку «Режим чтения».

Таблица отображает только те объекты, правом на просмотр которых обладает пользователь. Если в отношении некоторых объектов таблицы пользователь имеет право редактирования или удаления, то для каждого объекта (строки в таблице) будут доступны кнопки соответственно редактирования или удаления.

В таблице первые три колонки предназначены для совершения операций над объектами. Доступны операции просмотра информации об объекте таблицы (колонка П), редактирования информации об объекте (колонка Р) и удаления объекта (колонка У). Кроме того, можно сортировать объекты по заголовку колонки как по убыванию, так и по возрастанию.

Просмотр информации об объекте можно осуществить и при помощи щелчка на соответствующей вершине дерева объектов.

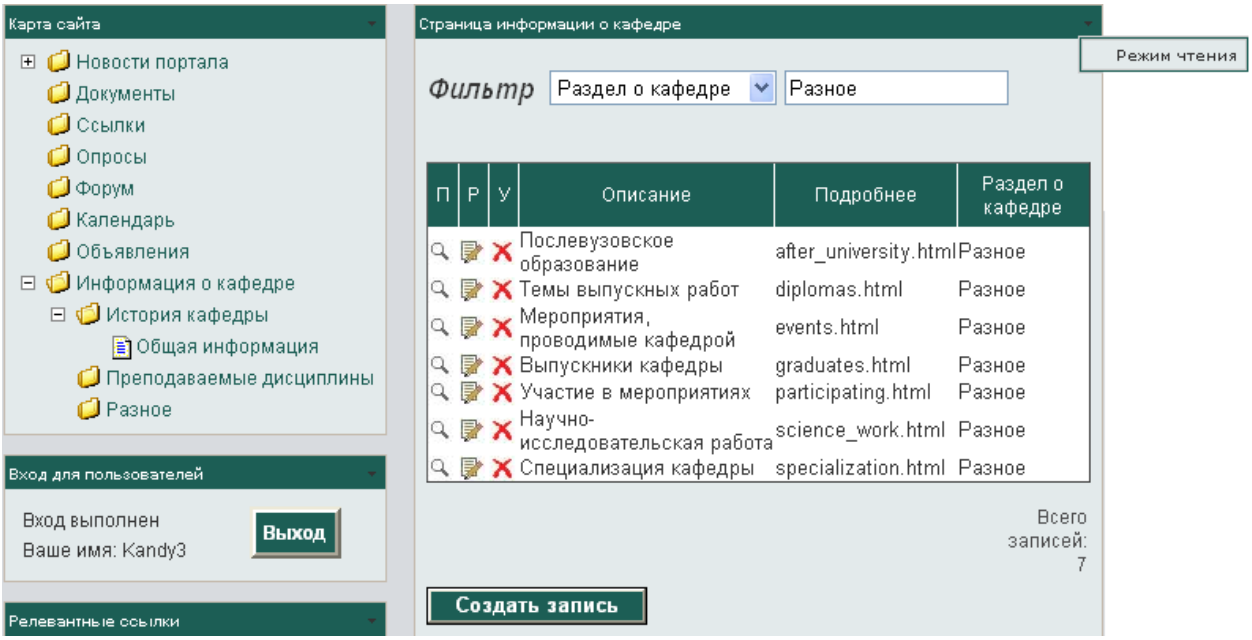

Рис. А.5. Таблица объектов

В таблице отображаются не более 10 записей. Для просмотра других записей можно воспользоваться кнопками «<<» и «>>».

Создать экземпляр сущности можно, нажав на соответствующую кнопку под таблицей.

Общее количество объектов данной сущности можно посмотреть справа под таблицей.

Также есть возможность фильтрования отображаемых объектов по всем возможным атрибутам. Для этого в поле «Фильтр» нужно выбрать параметр, по которому производится фильтрация, а затем в соседнем поле ввести значение, равняться которому будут должны значения атрибутов отображаемых объектов.

При доступе к объекту в режиме редактирования отображается форма, представленная на рис. А.6.

На форме отображаются атрибуты редактируемого объекта с соответствующими значениями. Атрибуты распределены по закладкам. В случае, если атрибут – это краткая информация об объекте, связанном с данным, то более подробную информацию о нем можно будет просмотреть при помощи кнопки «Перейти к родительскому объекту», которая находится справа от атрибута. Также справа находится кнопка «Показать список объектов», при помощи которой можно просмотреть все объекты данного типа.

Для ввода значений атрибутов объекта необходимо либо ввести информацию с клавиатуры, либо выбрать необходимое значение из списка.

Кроме того, на отдельной закладке «Дочерние объекты» приводятся ссылки для навигации к дочерним объектам, представленным в виде таблицы соответствующего типа объектов.

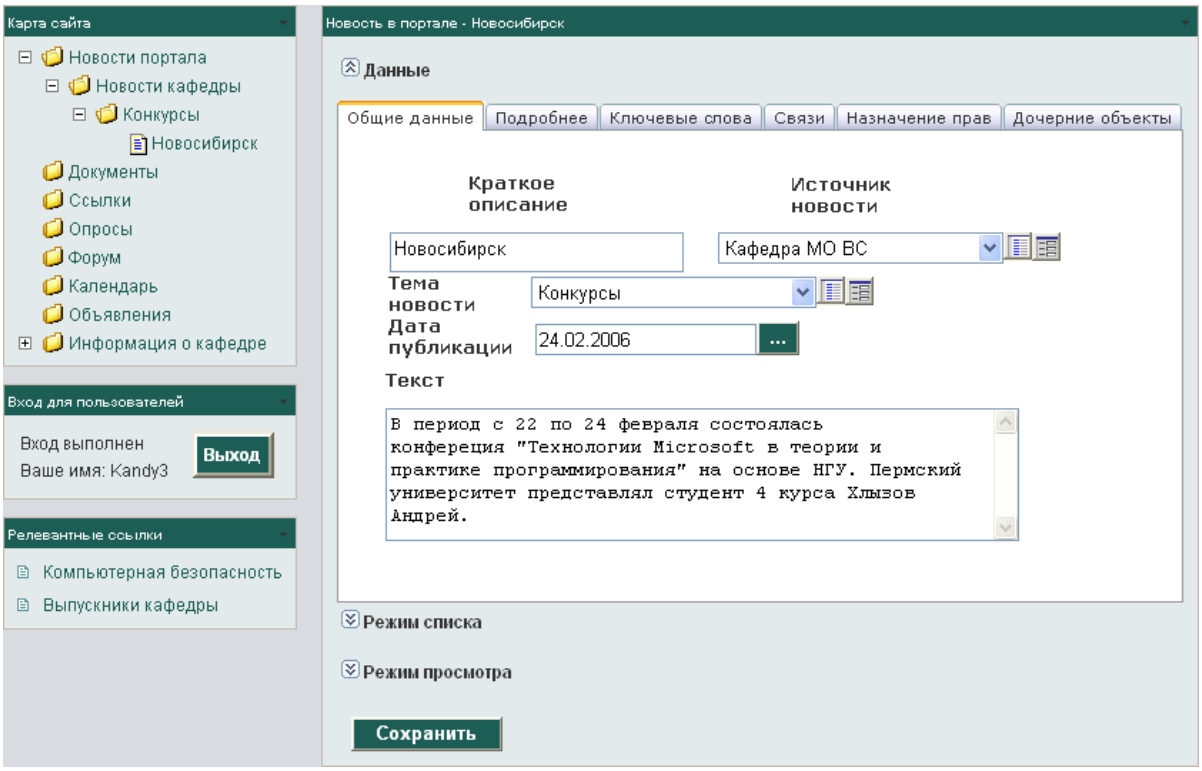

Рис. А.6. Форма редактирования объекта

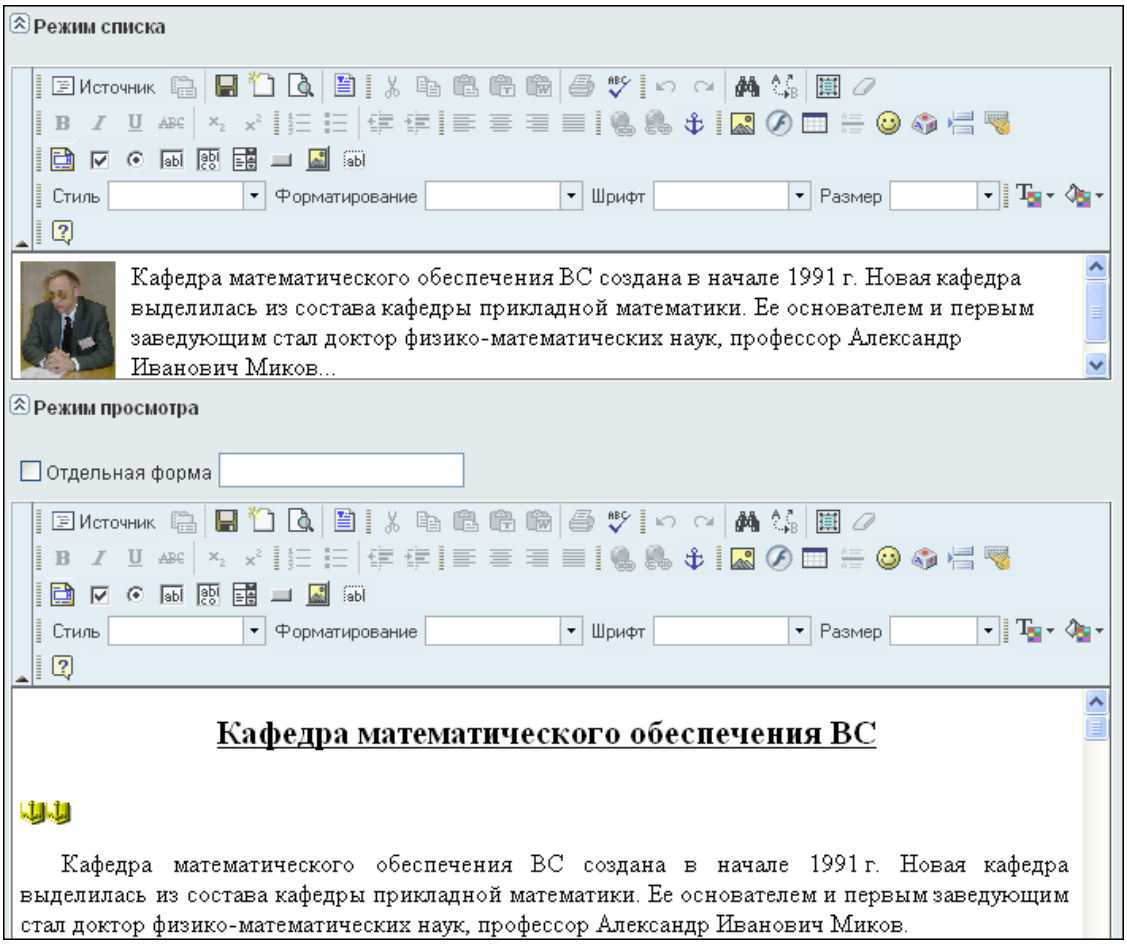

Рис. А.7. Редактирование представлений режимов списка и просмотра

На форме пользователь также может задать списковое представление объекта и представление объекта в режиме просмотра. Для режима просмотра нужно либо указать текст при помощи редактора, расположенного на странице, либо указать компонент с расширением .ASCX, который будет загружаться вместо простого текста.

Помимо обычных элементов управления для ввода и редактирования атрибутов объекта, возможно отображение нестандартных элементов. Как правило, такие элементы используются для работы с атрибутами таких типов, как «Файл», «Классификатор», «ОКАТО» и т.д.

Атрибут «Файл» представляет собой информацию о прикрепленном к объекту физическом файле, хранящемся в базе данных системы. Для его представления на форме редактирования используется элемент, представленный на рис. А.8.

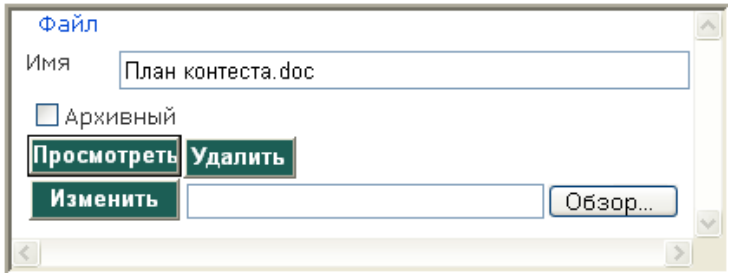

Рис. А.8. Элемент управления для работы с атрибутом типа «Файл»

Файл может отсутствовать. Если файл отсутствует, то в поле «Имя» будет отображено сообщение «Нет документа».

Файл может быть архивным. Если файл архивный, то из операций доступен лишь его просмотр.

Для просмотра файла надо нажать на кнопку «Просмотр». После этого откроется окно, предлагающее либо сохранить файл на компьютере пользователя, либо открыть файл.

Для того чтобы удалить прикрепленный файл, надо нажать на кнопку «Удалить». При этом не будет отображена форма с подтверждением удаления файла.

Для того чтобы изменить файл, надо сначала выбрать файл. Для этого необходимо нажать на кнопку «Обзор». Откроется диалоговое окно для выбора файла. Затем надо нажать на кнопку «Изменить» для фиксации изменений в базе данных.

Для того чтобы сделать файл архивным, надо нужно поставить соответствующую галочку «Архивный». После этого доступна будет только операция просмотра файла.

Также на форме есть несколько дополнительных закладок.

Закладка «Назначение прав» служит для задания прав пользователей и групп на чтение, изменение и удаление данного объекта (рис. А.9). Закладка доступна только для тех пользователей, которые могут передавать права на объект.

Группа	Чтение	Изменение	Удаление	
Administrators	☐	☐	☐	Редактировать
Guests	☑	☐	☐	Изменить Отменить
Пользователи	☐	☐	☐	Редактировать

Рис. А.9. Закладка «Назначение прав» формы редактирования

Закладка «Ключевые слова» предназначена для добавления и удаления слов, по которым можно будет найти объект при помощи веб-части «Поиск» (рис. А.10).

Слово	
Конкурс	Удалить
Хлызов	Удалить
Андрей	Удалить
	Добавить

Рис. А.10. Закладка «Ключевые слова»

Закладка «Связи» служит для задания релевантных объектов, ссылки на которые будут отображаться в веб-части «Релевантные ссылки» при просмотре или редактировании данного объекта (рис. А.11). Чтобы добавить новую ссылку, нужно внизу первой колонки ввести название типа этого объекта (при этом будут выдаваться подсказки предикативного ввода) и название самого объекта внизу второй колонки (для второй колонки также работает предикативный ввод).

Тип	Объект	
Страница информации о кафедре	Компьютерная безопасность	Удалить
Страница информации о кафедре	Выпускники кафедры	Удалить
		Добавить

Рис.** Ошибка! Текст указанного стиля в документе отсутствует.**.1. Закладка «Релевантные ссылки»

После внесения всех изменений нужно нажать на кнопку «Сохранить» для их фиксирования в базе данных. После подтверждения изменений будет отображена страница с таблицей сущности, содержащей отредактированный объект.

Аутентифицированный пользователь также может использовать модуль доступа к информационным системам (рис. А.12). Данный модуль позволяет

пользователю обращаться к зарегистрированным в портале ИС, описанным при помощи технологии METAS. Для получения доступа к необходимой ИС, нужно выбрать ИС из списка, представленного в веб-части «Информационные системы», и нажать на кнопку «Войти».

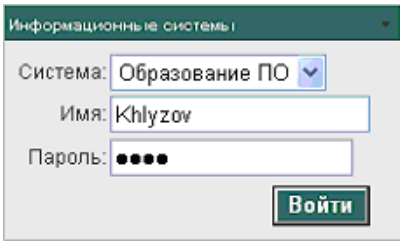

Рис. А.12. Web-часть «Информационные системы»

После этого будет перенастроена карта сайта и скрыты все веб-части, не имеющие отношения к ИС («Релевантные ссылки», «Избранное» и т.д.) (рис. А.13).

Рис. А.13. Вид страницы после входа в ИС «Образование Пермской области»

Приложение В. Руководство администратора портала

Помимо возможности оперирования с информационным наполнением портала через Web-интерфейс, администратор WebMETAS может работать с метаданными портала при помощи Win-приложения. Оно обеспечивает администратора возможностью реструктуризации портала, создания, изменения и удаления типов и экземпляров объектов и связей между ними, настройки пользовательского интерфейса и прав зарегистрированных пользователей, а также подключения бизнес-операций.

Реструктуризация портала

При помощи средства реструктуризации пользователь может настраивать типы объектов и связи между ними.

Для удобства представления информации пользователь может создавать различные схемы, в которых он выбирает типы объектов для отображения. Выбрать схему можно из списка, отображаемого в верхней части окна (рис. В.1).

В каждой схеме пользователь может создавать типы объектов. Для создания типа нужно щелкнуть правой кнопкой мыши по пустой области схемы и выбрать пункт «Создать сущность…» из появившегося меню. Пользователь также может добавить к схеме уже существующие типы объектов. Для этого нужно выбрать пункт «Сущности подсхемы» из того же меню и поставить галочку у типов, которые необходимо добавить.

Для каждой типа объекта пользователь может задавать атрибуты. Для задания атрибутов достаточно щелкнуть правой кнопкой мыши по типу объекта, отображаемого на схеме, и выбрать пункт «Конструктор». После этого отобразится форма с таблицей, представляющей атрибуты (рис. В.2). Для каждого атрибута задается его название, тип, описание, обязательность и права на просмотр и изменение. Также можно назначать значение по умолчанию и тип задания значения (вычислимый или задаваемый пользователем).

Между типами объектов пользователь может задавать связи. При создании связи указываются соединяемые типы объектов, название связи и ее тип (1:М, М:М и т.д.). Для того чтобы создать связь, нужно щелкнуть правой кнопкой мыши по пустой области схемы и выбрать пункт «Добавить связь» (рис. В.3).

Для редактирования связи нужно щелкнуть левой кнопкой мыши по линии, соединяющей отображаемые типы объектов, а затем по той же линии щелкнуть правой кнопкой и выбрать пункт «Свойства связи» (рис. В.4).

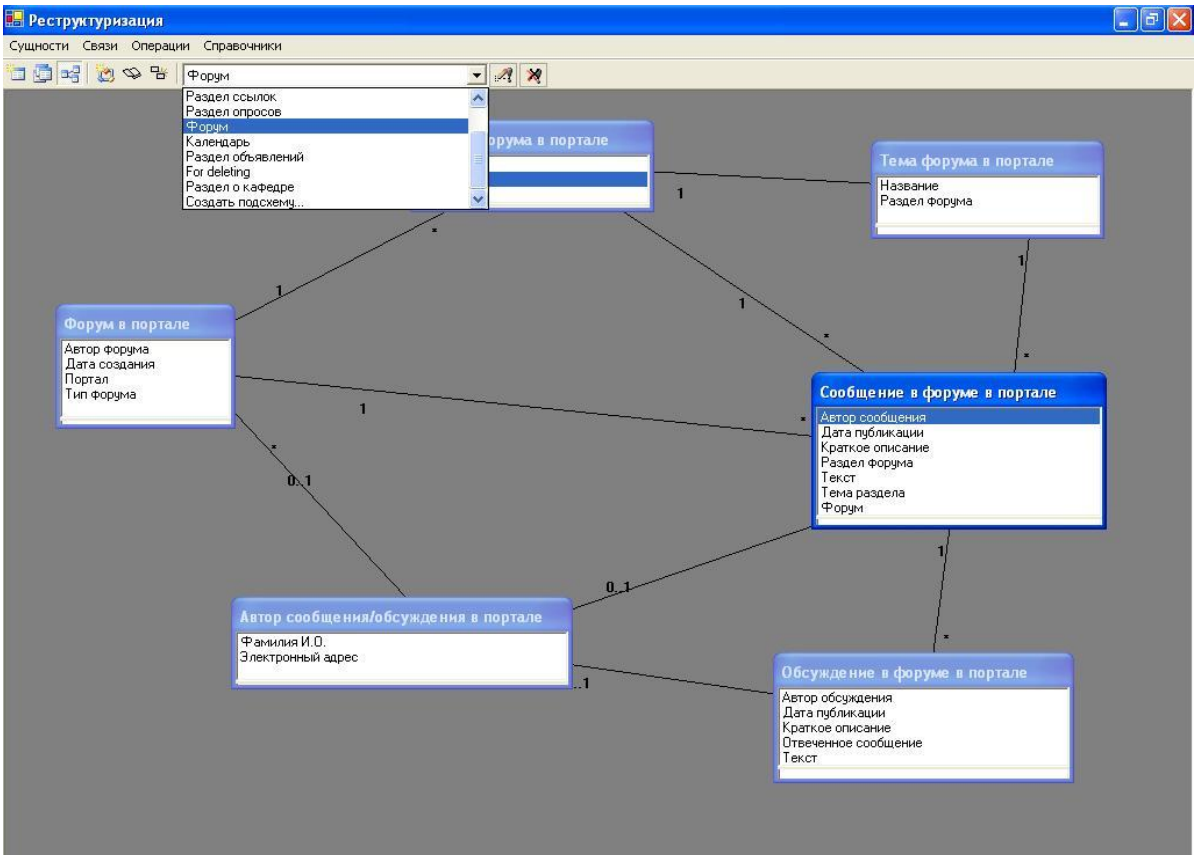

Рис. В.1. Выбор схемы

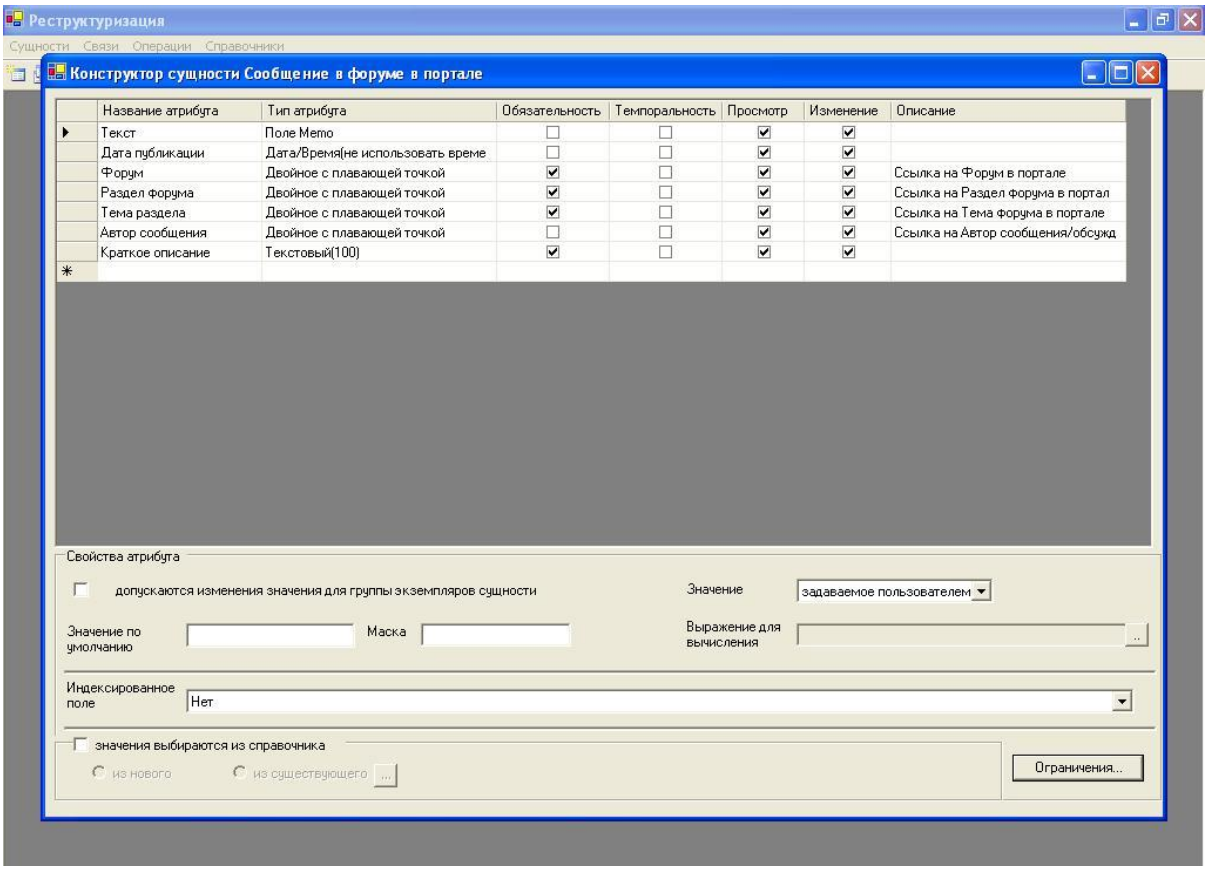

Рис. В.2. Изменение атрибутов типа объекта

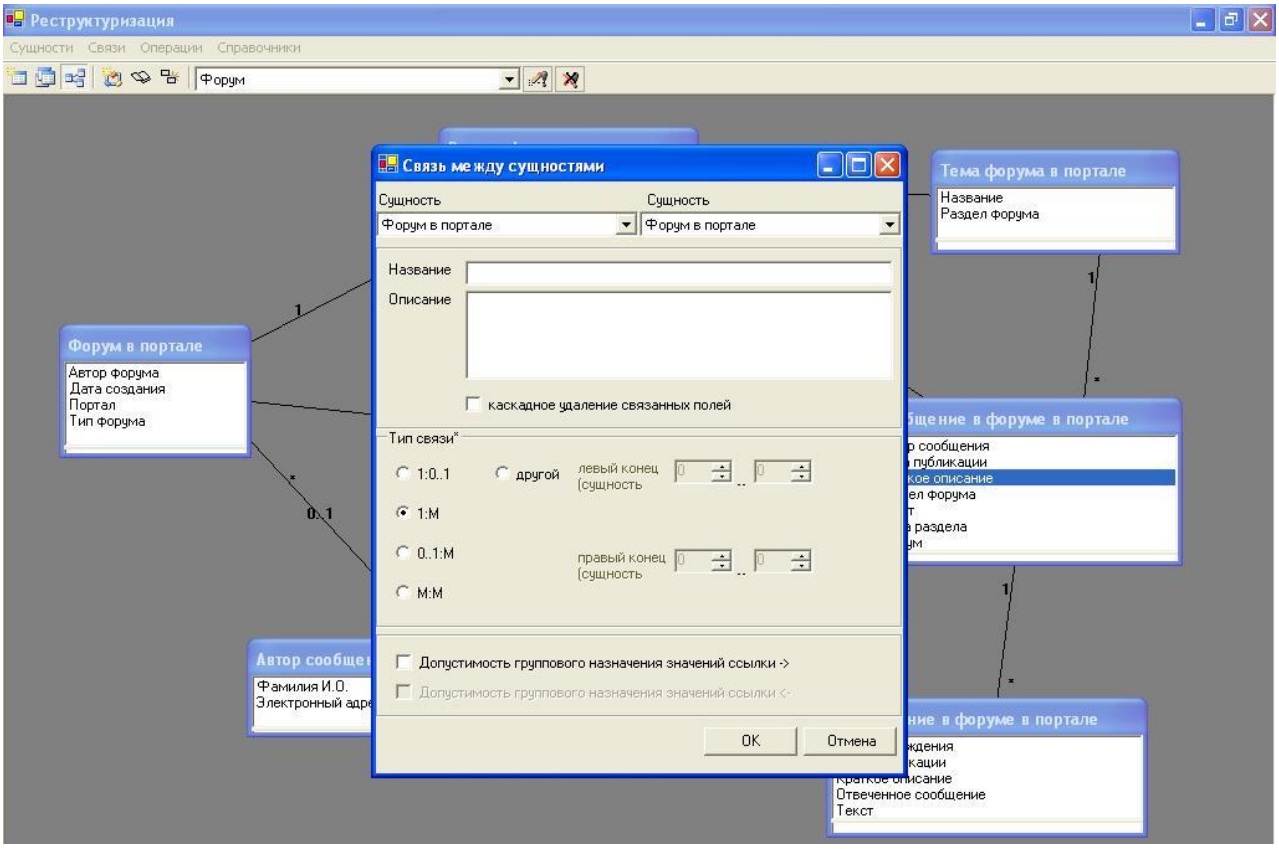

Рис. В.3. Добавление связи

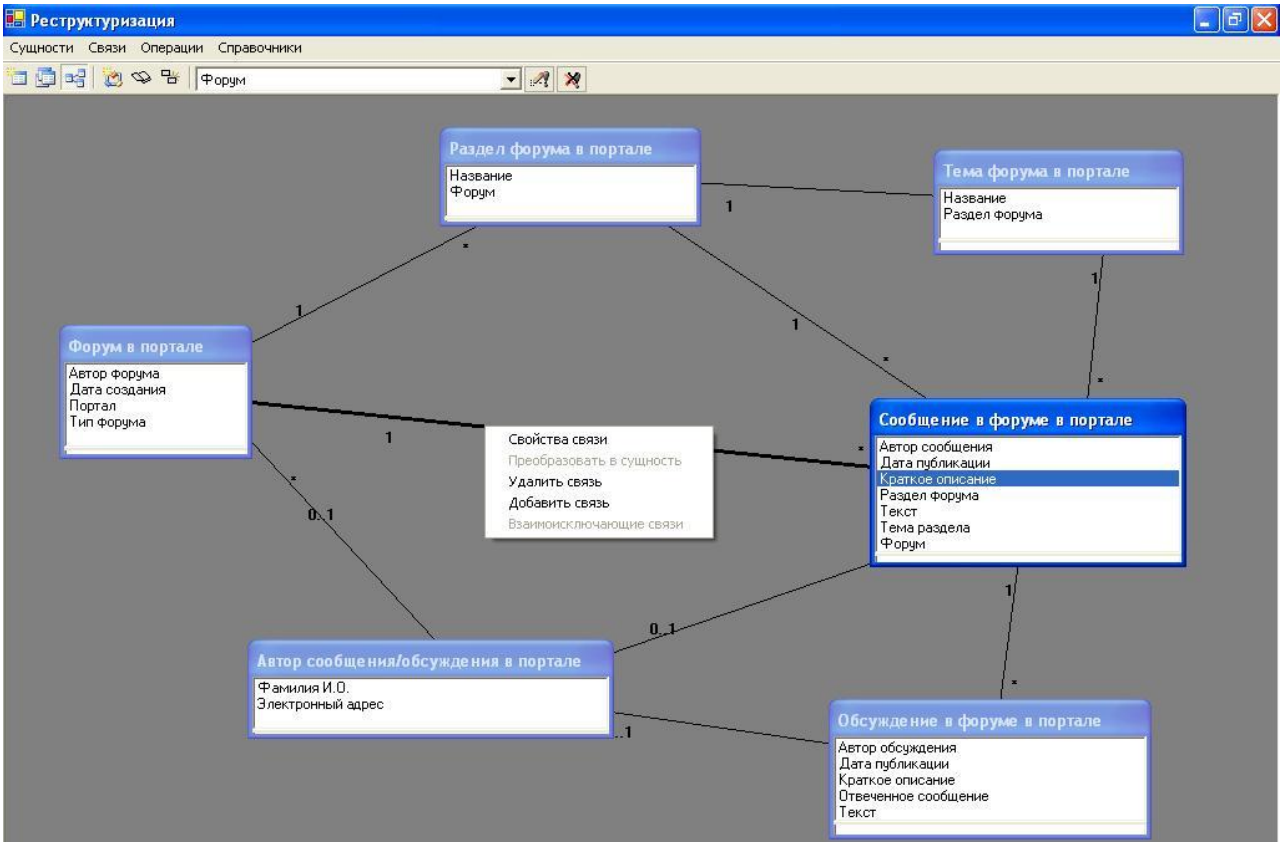

Рис. В.4. Свойства связи

Настройка пользовательского интерфейса

При помощи средства настройки Web-интерфейса пользователь может изменять иерархию объектов, представленную в WebMETAS деревом объектов и меню, а также задавать способ обработки выбора вершины в дереве объектов.

Левая часть окна работает в режимах просмотра и настройки. Для переключения между режимами нужно щелкнуть правой кнопкой мыши по дереву объектов и выбрать соответствующий пункт меню.

Режим настройки

Для добавления типа вершины в дерево объектов необходимо щелкнуть правой кнопкой мыши по вершине, которая будет родительской по отношению к новой, и выбрать пункт «Добавить». Аналогично можно удалять и изменять вершины дерева объектов. После выбора пункта меню будет отображена форма для задания параметров вершины (рис. В.5). В качества параметров задаются название вершины, текст и иконка, которые будут отображаться в дереве объектов и др. После задания значений необходимых параметров нужно нажать кнопку «Сохранить» в правом нижнем углу окна.

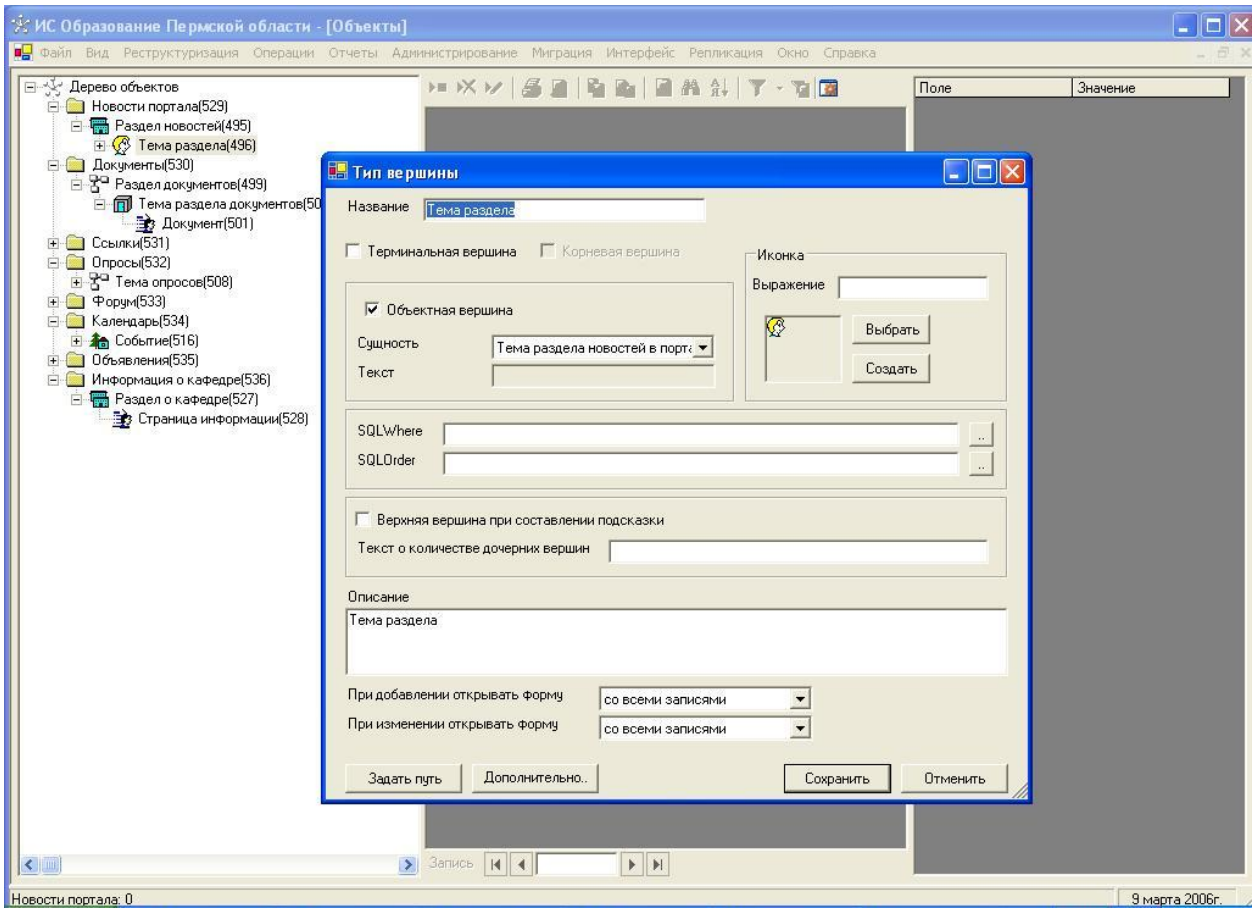

Рис. В.5. Создание типа вершины

Средняя часть окна служит для отображения информации о дочерних объектах в виде таблицы. Для того чтобы настроить таблицу дочерних объектов, нужно вначале выбрать вершину в дереве объектов, которая представляет родительский объект, а затем щелкнуть правой кнопкой мыши по средней части окна и выбрать пункт «Редактировать поля». После этого откроется форма для задания отображаемых полей дочерних объектов (рис. В.6). Выбрать отображаемые атрибуты можно путем перетаскивания соответствующих полей объектов, отображаемых в левой части формы, в список «Выбранные поля». Для каждого отображаемого поля можно задать ширину, выравнивания и др.

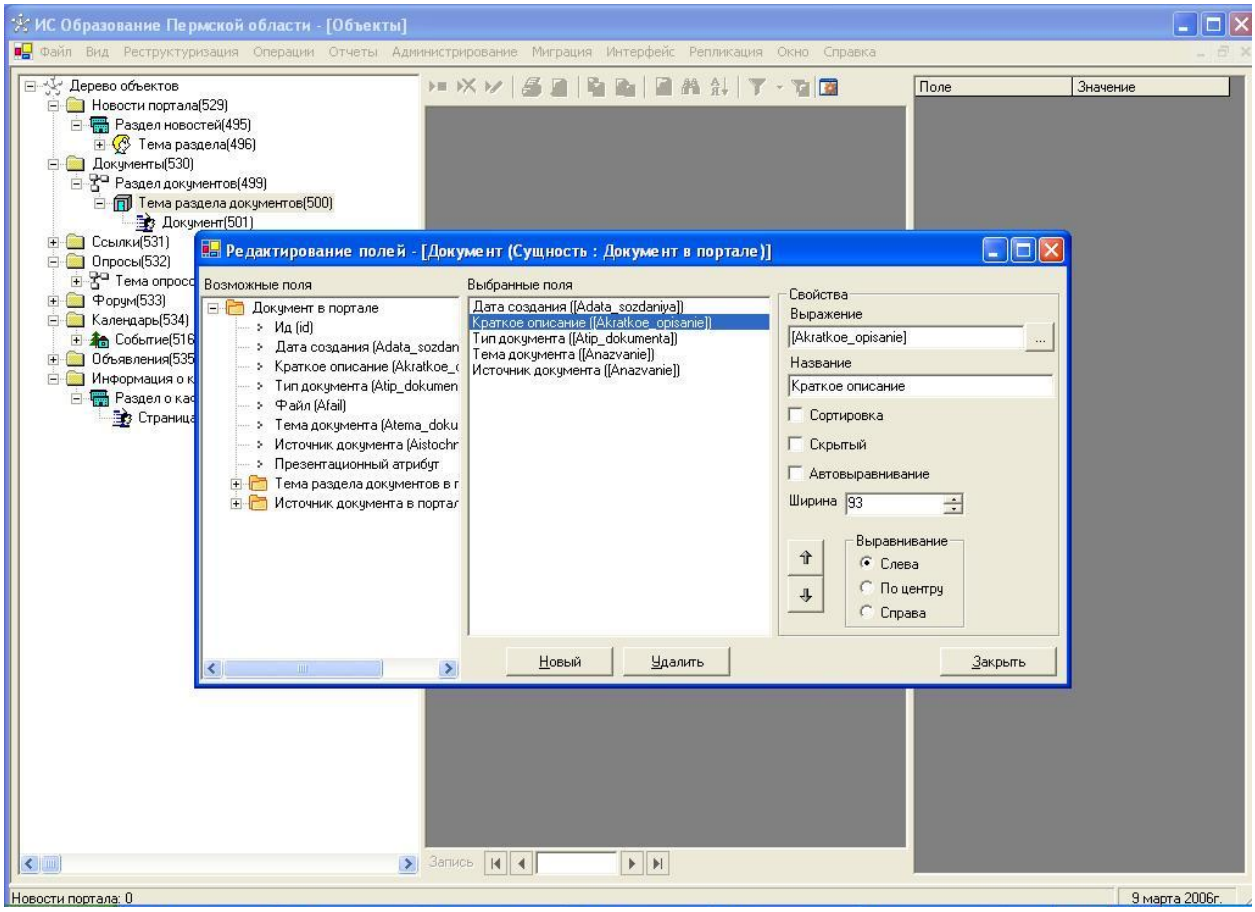

Рис. В.6. Настройка таблицы дочерних объектов

Правая часть окна служит для отображения информации об объекте, выбранном в таблице дочерних объектов. Для настройки отображаемых атрибутов объекта нужно вначале выбрать вершину в дереве объектов, которая представляет родительский объект, а затем щелкнуть правой кнопкой мыши по средней части окна и выбрать пункт «Редактировать поля». Настройка отображаемых атрибутов выполнятся аналогично.

Помимо вышеперечисленного администратор имеет возможность настройки внешнего вида форм, отображаемых как в Win-приложении, так и в Web-интерфейсе.

Для того чтобы настроить форму, нужно выбрать соответствующую вершину в дереве объектов и щелкнуть по ней правой кнопкой мыши. После этого нужно выбрать в контекстном меню пункт «Настроить форму». После этого откроется форма, представленная на рис. В.7.

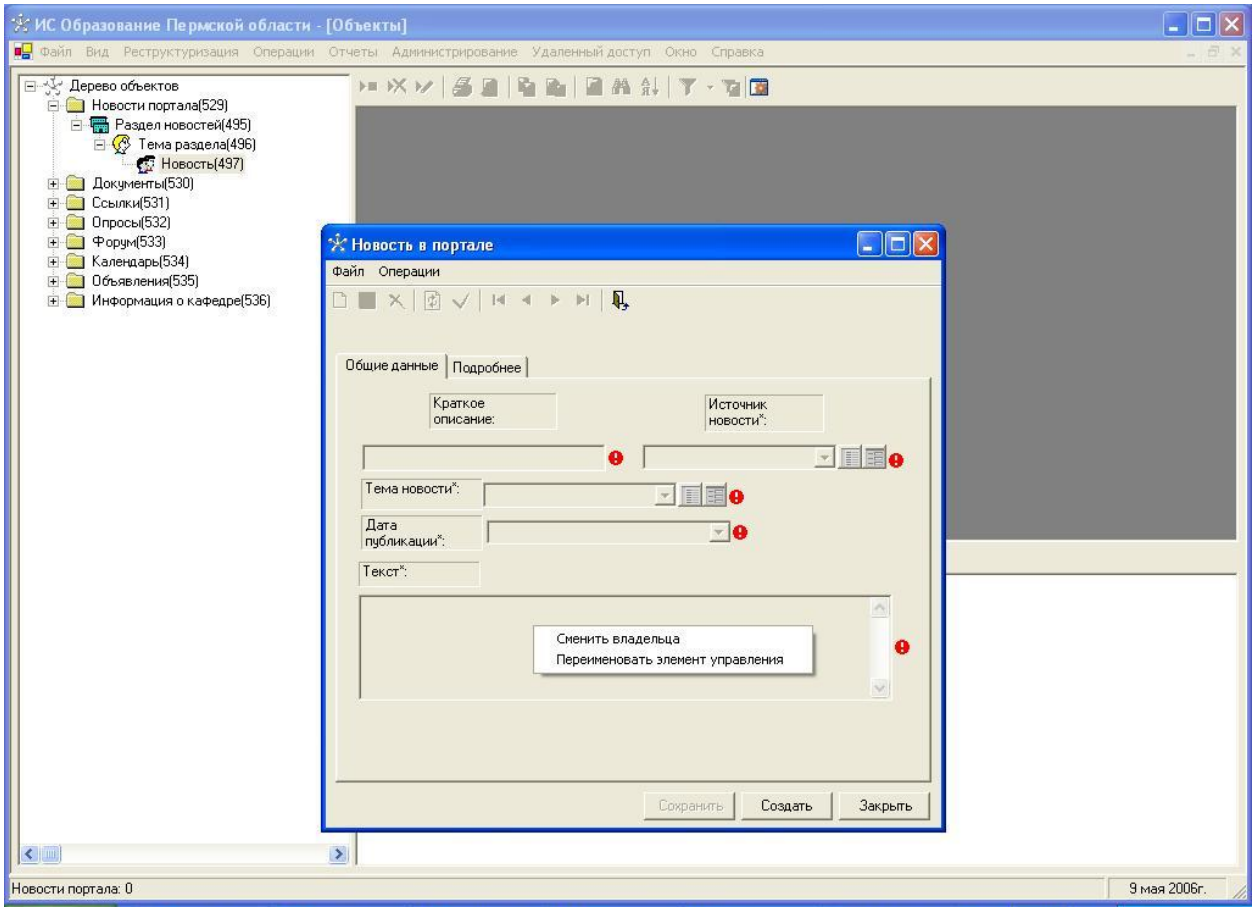

Рис. В.7. Настройка формы редактирования объекта

Для изменения расположения какого-либо атрибута формы нужно перетащить его на требуемое место, удерживая нажатой левую кнопку мыши.

Для изменения ширины или высоты атрибута нужно подвести указатель мыши к краю атрибута до того момента, когда появится двунаправленная стрелка. Затем, удерживая нажатой левую кнопку мыши, можно изменить размер атрибута.

Для того чтобы изменить название атрибута, отображаемое рядом с ним, нужно щелкнуть правой кнопкой мыши по соответствующему атрибуту и выбрать пункт меню «Переименовать элемент управления». После этого необходимо ввести новое название атрибута.

Для того чтобы переместить атрибут на другую закладку, нужно щелкнуть правой кнопкой мыши по соответствующему атрибуту и выбрать пункт меню «Сменить владельца». После этого нужно выбрать новую закладку и нажать на кнопку «Сохранить».

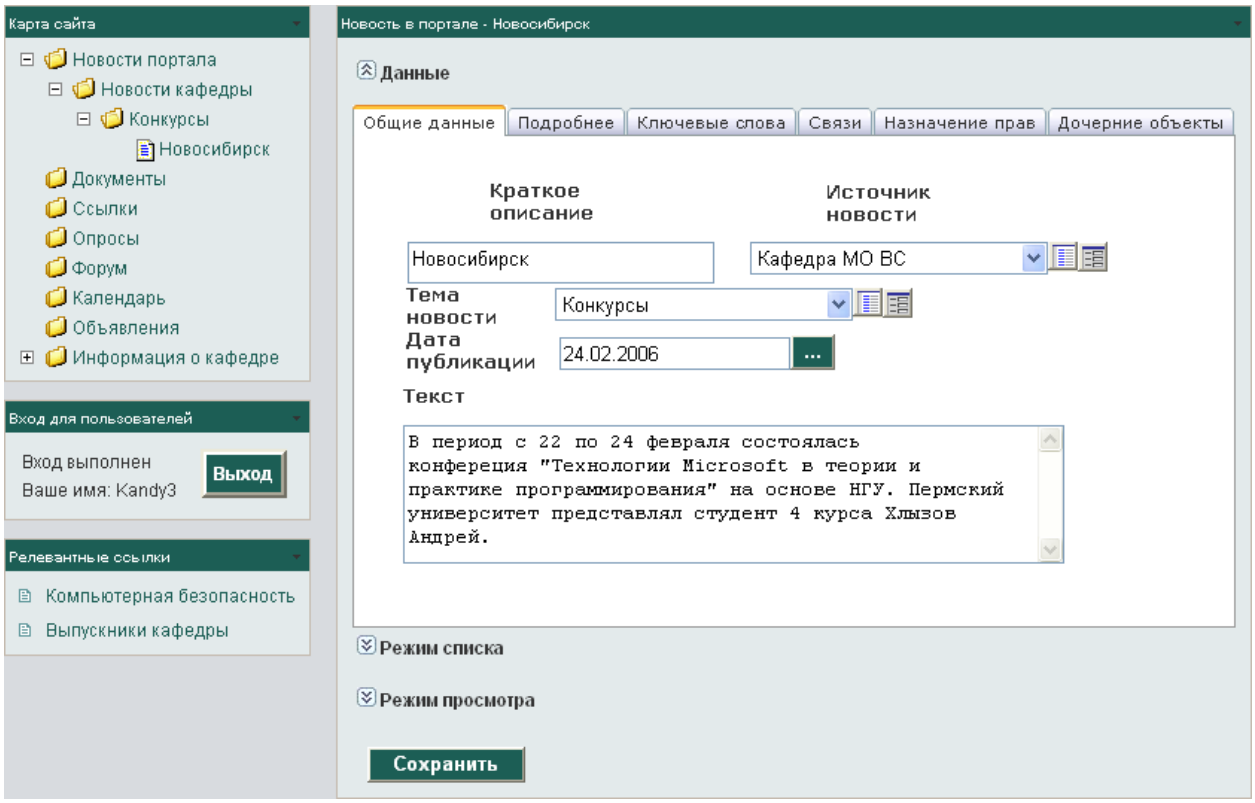

Рис. В.8. Вид настроенной формы в Web-браузере

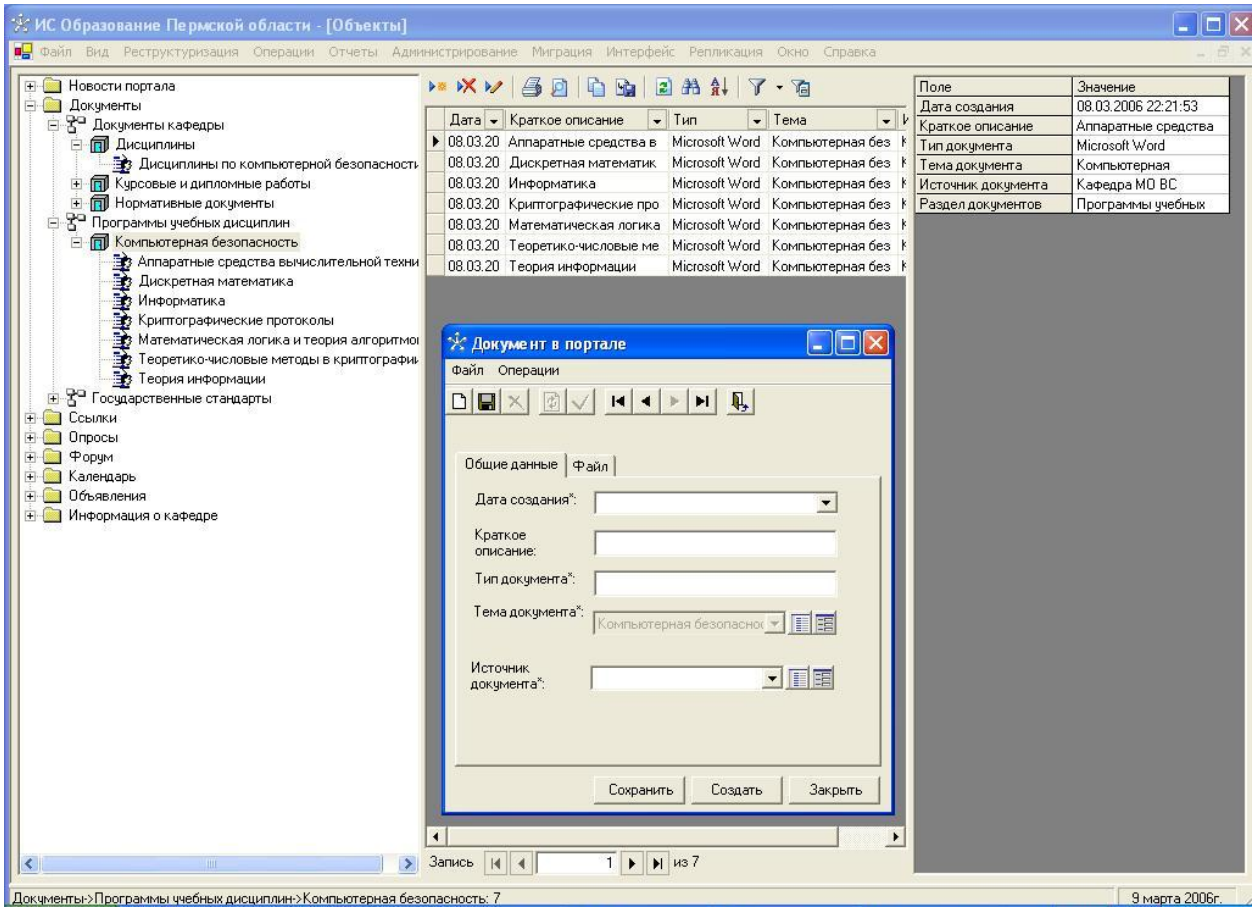

Рис. В.9. Добавление объекта портала

Вид настроенной формы редактирования в Web-браузере представлен на рис. В.8.

В режиме просмотра пользователь может изменять информационное наполнение портала.

Для изменения, создания и удаления объектов дерева нужно щелкнуть правой кнопкой мыши в соответствующем месте дерева объектов и выбрать соответствующий пункт меню. При создании и изменении объектов откроется форма для редактирования информации об объекте (рис. В.9). После внесения изменений нужно нажать на кнопку «Сохранить». Оперировать с объектами можно также в таблице дочерних объектов.

Назначение прав удаленным пользователям

Для того чтобы назначить права удаленным пользователям, администратору нужно выбрать пункт меню Удаленный доступ->Удаленные пользователи и группы.

Форма удаленных пользователей и групп имеет 2 закладки: «Группы» и «Пользователи» (рис. В.10).

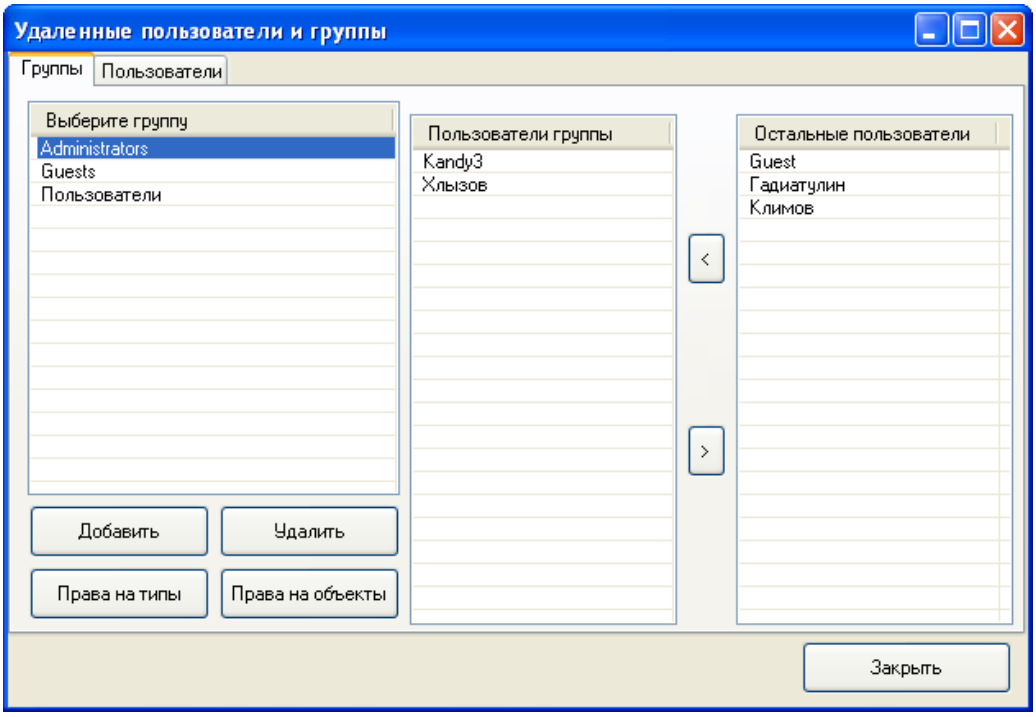

Рис. В.10. Удаленные пользователи и группы

Опишем интерфейс для работы с группами пользователей.

Имеется возможность добавления и удаления групп. Для того чтобы добавить группу, необходимо щелкнуть по кнопке «Добавить». После этого появится новая запись для группы в левом списке. Для того чтобы удалить группу, необходимо щелкнуть по кнопке «Удалить».

В правой части окна находятся 2 списка пользователей. Первый список содержит пользователей, которые входят в данную группу. Второй список содержит остальных пользователей портала. Для того чтобы добавить пользователя в группу, нужно выбрать соответствующую запись в списке остальных пользователей и щелкнуть по кнопке «<». Для того чтобы удалить пользователя из группы, нужно выбрать соответствующую запись в списке пользователей группы и щелкнуть по кнопке «>».

Для назначения прав группы на типы объектов нужно выбрать соответствующую запись в списке групп слева и щелкнуть по кнопке «Права на типы». После этого будет отображена форма с правами (рис. В.11).

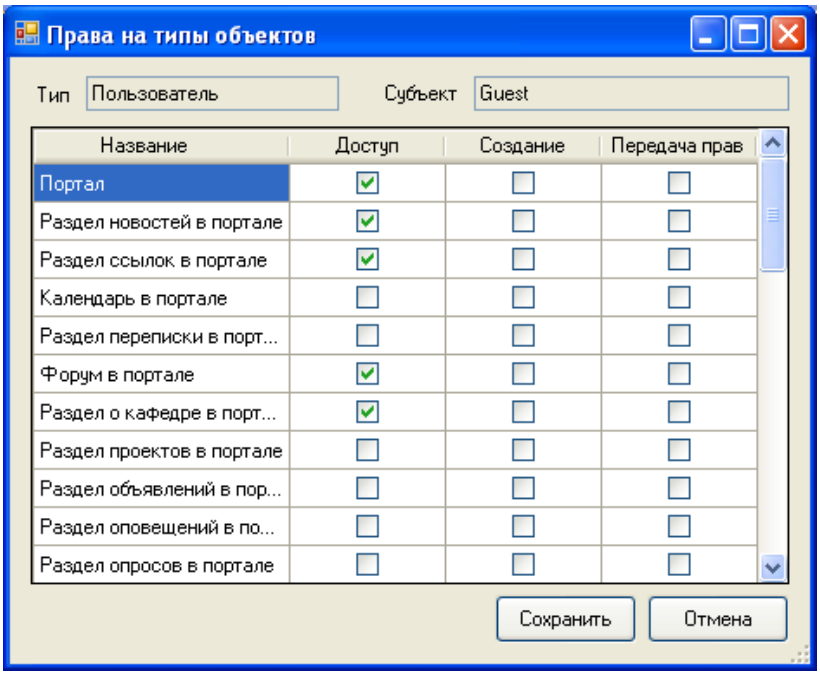

Рис. В.11. Назначение прав на типы объектов группе или пользователю

Можно назначать права на доступ к объектам выбранного типа, создание объектов и передачу прав на объекты другим пользователям. Для того чтобы назначить определенное право на определенный тип объекта группе, нужно поставить галочку в соответствующей строке типа и соответствующем столбце права. По завершении работы с правами нужно нажать на кнопку «Сохранить». Для отмены изменений нужно нажать на кнопку «Отмена».

Для назначения прав группы на объекты портала нужно выбрать соответствующую запись в списке групп слева и щелкнуть по кнопке «Права на объекты». После этого будет отображена форма с правами (рис. В.12).

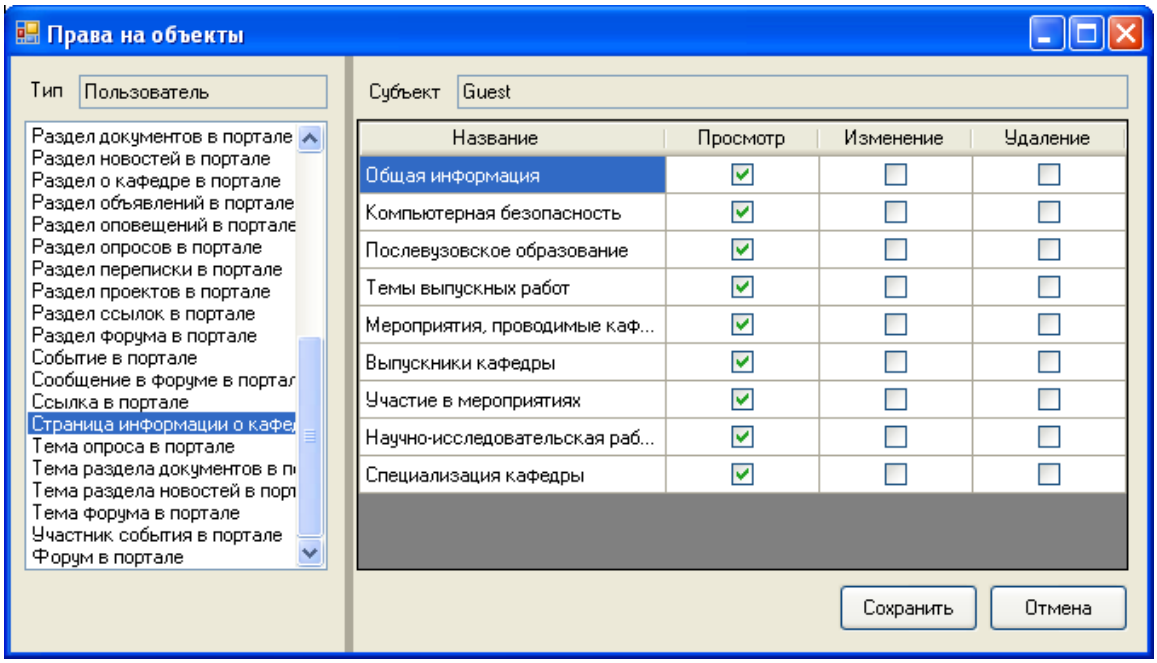

Рис. В.12. Назначение прав на объекты группе или пользователю

Слева на форме находится список всех типов объектов. При выборе типа в списке в правой части формы отображается таблица прав на объекты этого типа для данной группы. Можно назначать права на просмотр, изменение и удаление объектов. Для того чтобы назначить определенное право на определенный объект группе, нужно поставить галочку в соответствующей строке типа и соответствующем столбце права. По завершении работы с правами нужно нажать на кнопку «Сохранить». Для отмены изменений нужно нажать на кнопку «Отмена».

Работа с пользователями аналогична работе с группами за тем лишь исключением, что вместо списков пользователей данной группы и остальных пользователей, имеются списки групп, в которые входит пользователь, и остальных групп. Работа с ними аналогична работе со списками закладки «Группы».

СОДЕРЖАНИЕ