МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики

Кафедра программного обеспечения   
и администрирования информационных систем

**Система обработки документов: хранение, распознавание и поиск**

Курсовая работа

Направление 09.03.03 Прикладная информатика   
Профиль «Прикладная информатика в юриспруденции»



|  |  |  |
| --- | --- | --- |
| Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_ | д. ф.-м. н., проф. | Артёмов М. А. |
| Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_ | 3 курс, 10 группа | Дегтярёва А. В. |
| Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_ | к.т.н., преп. | Старикова А. А. |

Воронеж 2018

**Аннотация**

Данная работа посвящена проектированию и практической реализации системы обработки документов. В ходе работы рассмотрены основные этапы разработки информационной системы: проведен анализ функциональности существующих систем электронного документооборота и сформированы требования к проектируемой системе, описаны реляционная и документоориентированная модели данных, с применением основных средств языка UML описана структура и основные функции системы. На основе проведенного анализа выполнена практическая реализация спроектированных модулей и разработан прототип системы.

Содержание

[Введение 4](#_Toc8298240)

[1. Постановка задачи 6](#_Toc8298241)

[2. Анализ 7](#_Toc8298242)

[2.1. Анализ существующих приложений 7](#_Toc8298243)

[2.2. Анализ функциональности приложения 9](#_Toc8298244)

[2.3. Анализ структуры приложения 12](#_Toc8298245)

[2.4. Анализ моделей данных 15](#_Toc8298246)

[2.4.1. Анализ реляционной модели данных 15](#_Toc8298247)

[2.4.2. Анализ документоориентированной модели данных 16](#_Toc8298248)

[2.5. Анализ пользовательского интерфейса 16](#_Toc8298249)

[3. Средства реализации 21](#_Toc8298250)

[4. Требования к программному и аппаратному обеспечению 22](#_Toc8298251)

[5. Реализация 23](#_Toc8298252)

[5.1. Структура системы 23](#_Toc8298253)

[5.2. Модели данных 23](#_Toc8298254)

[5.2.1. Реляционная модель данных 23](#_Toc8298255)

[5.2.2. Документоориентированная модель данных 23](#_Toc8298256)

[5.3. Диаграмма классов 26](#_Toc8298257)

[Заключение 27](#_Toc8298258)

[Список литературы 28](#_Toc8298259)

[Приложение 1. SQL-скрипт для создания базы данных 30](#_Toc8298260)

[Приложение 2. Описание сущностей базы данных 36](#_Toc8298261)

[Приложение 3. Листинг классов данных 41](#_Toc8298262)

Введение

Сотрудники любой организации тратят большое количество времени на поиск и обработку необходимых документов и информации, объем которой с каждым днем увеличивается. Внедрение системы электронного документооборота (СЭД) дает ряд преимуществ по сравнению с ручной обработкой: сокращение временных затрат на все рутинные операции с документами, обеспечение конфиденциальности информации, сокращение дублирования и оптимизация хранения документов, повышение качества данных, а следовательно, качества работы сотрудников и обслуживания клиентов.

В настоящее время существует множество СЭД, обладающих богатыми функциональными возможностями и широкой областью применения. Однако для большинства предприятий малого и среднего бизнеса функциональность этих систем избыточна, кроме того, их внедрение влечет высокие начальные затраты и скрупулезную работу по обучению пользователей.

Одной из важнейших функций таких систем является хранение и поиск документов. Любой документ, хранящийся в системе электронного документооборота, состоит из двух частей: его содержимого и набора реквизитов, так называемой карточки электронного документа, на основе которых осуществляется поиск, классификация и группировка. Если документ подлежит регистрации, то на него заводится еще и регистрационная карточка (РК). Возможен полнотекстовый поиск, поиск по реквизитам и регистрационным данным. Полнотекстовый поиск осуществляется по данным в самом документе, в том числе по словоформам через встроенные средства СУБД. Атрибутивный поиск осуществляется через специальную форму по нескольким значениям из полей карточек. Эта функциональность может использоваться отдельно от систем документооборота, а также интегрироваться с любой другой системой.

Таким образом, целесообразно создать программу, которая обеспечит легкий поиск и сохранность данных, быстрый доступ к документам и будет обладать понятным интерфейсом, легкостью внедрения и возможностью интеграции с другими системами с целью повышения эффективности работы организации.

1. Постановка задачи

Спроектировать систему обработки документов, предоставляющую следующую функциональность:

* загрузка и регистрация документов;
* представление документов в виде набора значений атрибутов;
* хранение документов;
* хранение атрибутов документов;
* атрибутивный поиск документов;
* полнотекстовый поиск документов, объем содержимого которых не более 40000 символов;
* просмотр документов;
* просмотр карточек документов;
* удаление документов;
* редактирование карточек документов;
* разграничение прав доступа;
* создание шаблонов для каждого вида документов;
* создание каталогов.

Реализовать спроектированные модули и разработать прототип клиентской части системы, провести модульное и интеграционное тестирование.

1. Анализ
   1. Анализ существующих приложений

В настоящее время существует большое число систем электронного документооборота. Наиболее известными являются: Documentum, Е1 Евфрат, Directum, Дело, 1С: Документооборот 8, Docsvision, ELMA. Сравнительный анализ их функциональности приведен в табл. 2.1.

Таблица 2.1. Сравнительный анализ СЭД

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Характеристики** | **Documentum** | **Е1 Евфрат** | **Directum** | **Дело** | **1С: Документооборот 8** | **Docsvision** | **ELMA** |
| **Загрузка и регистрация документов** | | | | | | | |
| Сканирование | + | + | + | + | + | + | + |
| Распознавание | + | + | + | + | + | + | + |
| Возможность добавления в РК дополнительных атрибутов при регистрации | + | – | – | + | + | – | – |
| Конструктор карточек | – | + | + | – | – | + | + |
| Зависимость набора атрибутов РК от выбранного типа документа | + | + | + | + | + | + | + |
| Наличие обязательных атрибутов регистрационной карточки | + | + | + | + | + | + | + |
| Заполнение атрибутов значениями из системных справочников | + | + | + | + | + | + | + |
| Автоматическое формирование регистрационного номера | + | + | + | + | + | + | + |
| Поточное сканирование документов | + | + | + | + | + | + | + |
| Автоматическое извлечение атрибутов | + | + | + | + | – | + | + |
| **Хранение документов** | | | | | | | |
| Хранение атрибутов | + | + | + | + | + | + | + |
| СУБД | + | + | + | + | + | + | + |
| **Поиск** | | | | | | | |
| Создание каталогов | + | + | + | + | + | + | + |
| Полнотекстовый поиск | + | + | + | + | + | + | + |
| Атрибутивный поиск | + | + | + | + | + | + | + |
| Поиск по шаблонам | + | – | – | – | + | + | + |
| Создание шаблонов поиска | + | – | – | – | + | + | + |
| Сохранение поисковых запросов | + | + | + | + | + | + | – |
| **Разграничение прав доступа** | | | | | | | |
| Разграничение прав доступа на уровне каталога | – | + | + | + | + | + | + |
| Разграничение прав доступа на уровне вида документов | + | – | – | + | + | – | + |
| Разграничение прав доступа на уровне конкретного документа | + | + | + | + | + | + | + |
| **Интеграция со сторонними системами** | | | | | | | |
| Встроенная интеграция с системами организации | + | + | + | + | + | + | + |
| Наличие открытого API | + | + | + | + | + | + | + |

* 1. Анализ функциональности приложения

На основе проведенного анализа существующих решений выделена необходимая функциональность приложения:

* загрузка и регистрация документов;
* хранение документов;
* поиск документов;
* просмотр документов;
* удаление документов;
* редактирование карточек документов;
* управление доступом к данным;
* создание шаблонов для каждого вида документов.

Загрузка и регистрация должна включать следующую функциональность:

* загрузка документа;
* автоматическое присвоение документу уникального регистрационного номера;
* представление документа в виде набора значений атрибутов (регистрационная карточка документа).

Хранение должно включать следующую функциональность:

* настройка иерархической структуры хранения (каталогов);
* хранение файлов документов;
* хранение значений атрибутов документа в базе данных.

Поиск документов должен включать следующую функциональность:

* просмотр содержимого каталогов;
* атрибутивный поиск документов по основным атрибутам (вид документа, регистрационный номер, дата загрузки, ФИО регистратора);
* расширенный атрибутивный поиск документов по шаблонам поиска;
* полнотекстовый поиск документов, объем содержимого которых не более 40000 символов.

Просмотр документов должен включать следующую функциональность:

* просмотр текста документа;
* просмотр содержимого карточки документа.

Управление доступом должно осуществляться с помощью механизма создания ролей, которые назначаются пользователю и определяют набор его прав. Разграничение прав доступа должно включать следующую функциональность:

* разграничение прав доступа на уровне иерархии хранения документов, которое подразумевает под собой наличие или отсутствие прав на просмотр содержимого каталога;
* разграничение прав доступа на уровне видов документов.

Для каждого вида документа должна быть предусмотрена возможность предоставления следующих прав на документы данного вида:

* право на просмотр текста документа;
* право на просмотр карточки документа;
* право на редактирование карточки;
* право на удаление документа.

Создание шаблонов подразумевает под собой формирование наборов атрибутов с указанием их обязательности. Такая функциональность необходима, чтобы обеспечить независимость системы от предметной области, так как в организациях могут быть различные виды и атрибуты документов. Шаблоны используются для регистрации и поиска документов.

В ходе анализа и формирования требований к проектируемой системе выделены три функциональных блока. Функциональность, связанная с администрированием системы доступна пользователю с ролью «Администратор». Возможность загрузки и регистрации документов доступна пользователям, которые обладают ролью «Администратор». Остальные возможности системы не требуют наличия особой роли и доступны всем пользователям в соответствии с политикой разграничения доступа к данным конкретной организации. Диаграмма вариантов использования системы обработки документов приведена на рис. 2.1.

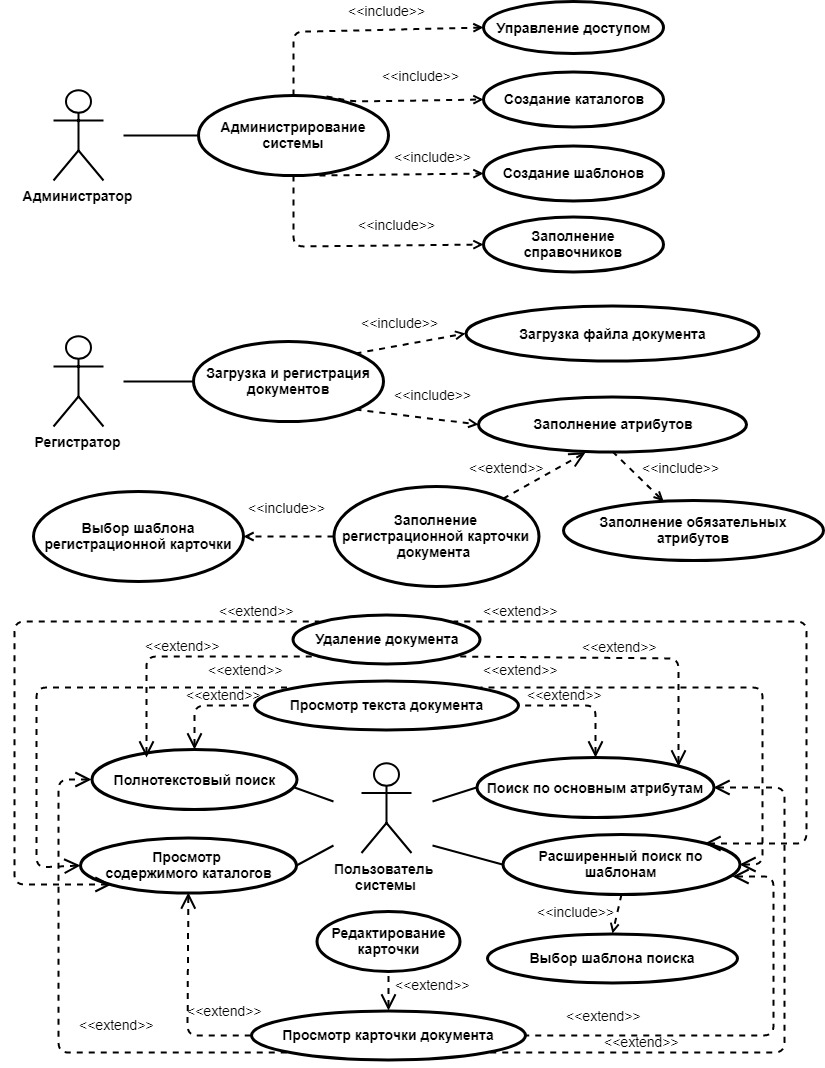


Рис. 2.1. Диаграмма вариантов использования системы обработки документов

* 1. Анализ структуры приложения

Исходя из наличия нескольких независимых функциональных блоков, для разрабатываемой системы выбрана сервис-ориентированная архитектура, при которой на стороне сервера имеется несколько независимо обновляемых и заменяемых компонентов - микросервисов.

Данная архитектура имеет следующие преимущества:

* возможность независимого развертывания микросервисов;
* возможность независимого масштабирования микросервисов;
* возможность независимой разработки каждого компонента.

**Независимое развертывание** позволяет изменять функциональность одного сервиса, не затрагивая другие и даже не приостанавливая их работу, а также заменить один микросервис другим или использовать каждый отдельно. К тому же, у микросервисов выше доступность: даже если один из них вышел из строя, это не обязательно приведет к сбою всего приложения.

**Независимая масштабируемость** позволяет отмасштабировать один из сервисов, во время повышенной нагрузки на него, без необходимости в масштабировании каждого компонента, что делает масштабирование гибким и снижает расходы.

**Независимая разработка** дает возможность использования наиболее подходящих технологий при разработке каждого микросервиса, в том числе использование нескольких БД с различными типами моделей данных.

Перечисленные преимущества позволят в дальнейшем изменять конфигурацию системы обработки документов в зависимости от нужд конкретной организации.

Структурная схема приложения приведена на рис. 2.2.

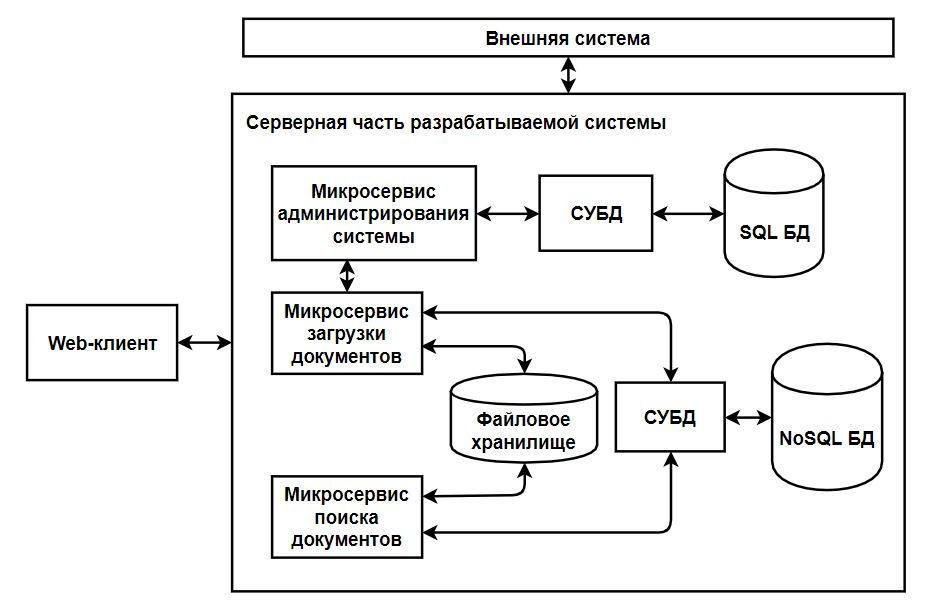


Рис. 2.2. Структурная схема приложения

Микросервис администрирования необходим для настройки системы и управления доступом. Данный компонент предоставит возможности заполнения справочников, создания каталогов и шаблонов, а также предоставления пользователям прав доступа. Для хранения информации, связанной с работой рассматриваемого сервиса, выбрана реляционная база данных.

Микросервис загрузки документов предоставит пользователям с ролью «Регистратор» возможность загрузки и регистрации документов. Загружаемые документы будут храниться в виде файлов на сервере, образуя файловое хранилище. Значения атрибутов загружаемых документов будут храниться в нереляционной (NoSQL) базе данных.

Причины выбора NoSQL базы данных для хранения метаданных документов:

* быстродействие, обеспечивающее эффективный поиск;
* возможность хранения больших объемов информации;
* легкость масштабирования;
* отсутствие ограничений на типы хранимых данных;
* отсутствие схемы и определенной структуры, что позволит хранить данные, состоящие из произвольного набора атрибутов и обеспечит независимость системы от предметной области.

Микросервис поиска документов предоставит пользователям системы функциональность, связанную с поиском документов и дальнейшей их обработкой. Для этого он взаимодействует с нереляционной базой данных, описанной выше.

Клиентская часть реализует пользовательский интерфейс, формирует запросы к сервисам и обрабатывает ответы от них. Отделение пользовательского интерфейса (UI) от микросервисов позволит поддерживать UI целиком, легко сохраняя его связность. В случае изменений границ в UI не придётся одновременно обновлять несколько компонентов, нарушая изолированность и независимость, обеспечиваемые самой архитектурой.

Клиентом системы может выступать любое приложение, однако использование web-клиента на рабочих местах дает определенные преимущества:

* отсутствие необходимости устанавливать приложение по клиентским местам и платить за дополнительные лицензии;
* техническая простота, так как от пользователя требуется просто запустить браузер и правильно набрать адрес;
* поддержка любых браузеров и, соответственно, любых операционных систем и всех гаджетов;
* доступ к системе из любой точки мира, где есть доступ во всемирную сеть.

Клиентские приложения и внешние системы будут взаимодействовать с компонентами разрабатываемой системы по сети. Все микросервисы должны предоставлять API для доступа к их функциональности.

* 1. Анализ моделей данных
     1. Анализ реляционной модели данных

Реляционная модель данных включает:

* атрибут;
* шаблон;
* вид документа;
* каталог;
* роль;
* права на вид документа;
* права на каталог.

Для атрибута необходимо хранить:

* уникальный идентификатор;
* наименование;

Для шаблона необходимо хранить:

* уникальный идентификатор;
* наименование;
* набор атрибутов шаблона с указанием их обязательности.

Для вида документа необходимо хранить:

* уникальный идентификатор;
* наименование;
* набор шаблонов для данного вида.

Для каталога необходимо хранить:

* уникальный идентификатор;
* наименование;
* идентификатор родительского каталога;

Для роли необходимо хранить:

* уникальный идентификатор;
* наименование.

Для прав на вид документа необходимо хранить:

* уникальный идентификатор вида документа;
* уникальный идентификатор роли, которой принадлежат права;
* наличие права доступа к тексту документов данного вида;
* наличие права доступа к карточке документов данного вида;
* наличие права удаления документов данного вида;
* наличие права изменения карточки документов данного вида.

Для прав на каталог необходимо хранить:

* уникальный идентификатор каталога;
* уникальный идентификатор роли, которой принадлежит право доступа к каталогу.
  + 1. Анализ нереляционной модели данных

Нереляционная модель данных включает метаданные документа. Для документа необходимо хранить:

* регистрационный номер;
* вид документа;
* каталог;
* дату регистрации;
* путь к файлу документа;
* сведения о пользователе, загрузившем документ;
* текст, если объем документа менее 40000 символов;
* набор значений атрибутов регистрационной карточки;
* набор прав на документ.
  1. Анализ пользовательского интерфейса

Интерфейс приложения должен включать:

* главное окно программы;
* окно администрирования;
* окно загрузки и регистрации документов;
* окно поиска документов;
* окно просмотра документа;
* окно редактирования карточки документа.

На рис. 2.4 представлена схема главного окна программы, включающего:

* главное меню;
* панель инструментов;
* область навигации;
* область просмотра содержимого каталогов.

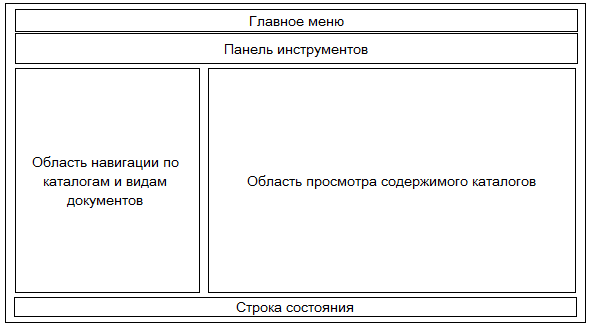


Рис. 2.4. Схема главного окна программы

Схема окна администрирования представлена на рис. 2.5. Оно содержит главное меню, панель инструментов и следующие вкладки:

* вкладка управление доступом, используемая для выдачи привилегий;
* вкладка каталоги, используемая для создания, редактирования и структурирования каталогов;
* вкладка справочники, используемая для заполнения справочников;
* вкладка шаблоны, используемая создания шаблонов поиска и регистрации документов.

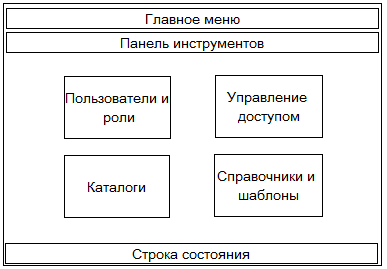


Рис. 2.5. Схема окна администрирования

Схема окна загрузки и регистрации документов представлена на рис. 2.6.

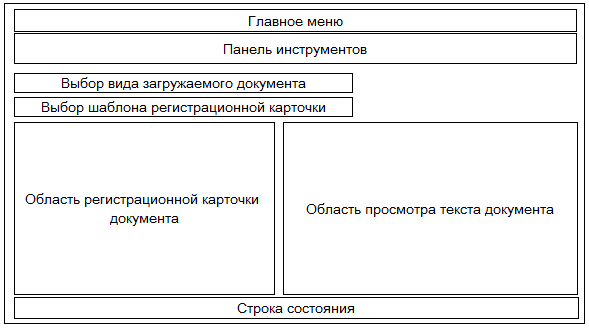


Рис. 2.6. Схема окна загрузки и регистрации документов

Оно содержит следующие области:

* главное меню;
* панель инструментов;
* область, содержащая обязательные для заполнения поля;
* область выбора шаблона регистрационной карточки;
* область регистрационной карточки документа, поля которой зависят от выбранного шаблона;
* область просмотра текста загружаемого документа.

Схема окна атрибутивного поиска представлена на рис. 2.7. Оно содержит следующие области:

* главное меню;
* панель инструментов;
* область выбора шаблона поиска;
* область атрибутов поиска, которая зависит от выбранного шаблона;
* область результатов поиска;
* строка состояния.

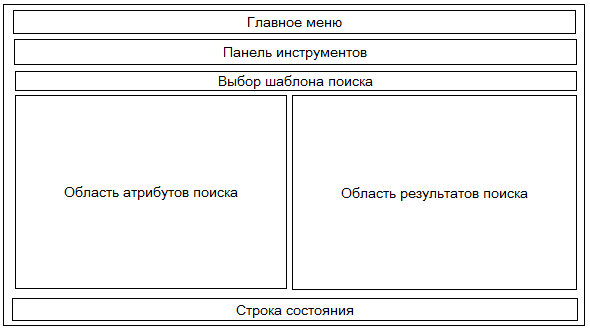


Рис. 2.7. Схема окна атрибутивного поиска

Схема окна просмотра документа представлена на рис. 2.8, оно содержит:

* главное меню;
* панель инструментов;
* область просмотра карточки документа;
* область просмотра текста документа;
* строку состояния.



Рис. 2.8. Схема окна просмотра документа

Схема окна редактирования карточки документа представлена на рис. 2.9, оно содержит область полей карточки документа, кнопки сохранения изменений и выхода.

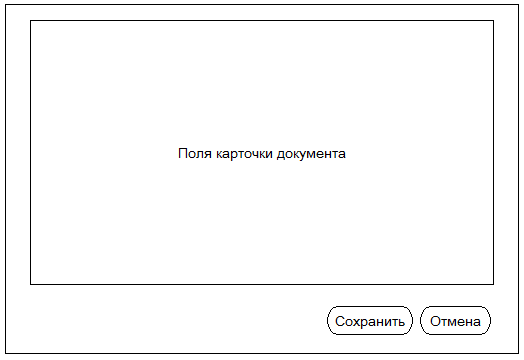


Рис. 2.9. Схема окна редактирования карточки документа

1. Средства реализации

Для разработки приложения выбраны следующие программные средства:

* среда разработки Microsoft Visual Studio Community 2017;
* язык программирования С#;
* язык программирования SQL;
* язык моделирования UML;
* СУБД MS SQL Server 2017;
* программный инструмент моделирования StarUML 2.8.1;
* инструментарий для проектирования баз данных и программных систем Sybase PowerDesigner 16.5.

1. Требования к программному и аппаратному обеспечению

Требования к программному обеспечению клиента: Microsoft .NET Framework 4.7.

Требования к аппаратному обеспечению клиента:

* операционная система Windows 8.1 и операционные системы более поздней версии;
* дисковое пространство 150Мб;
* оперативная память не менее 4ГБ;
* рекомендуемая частота процессора 2,0 ГГц или выше.

Требования к программному обеспечению на сервере:

* Microsoft .NET Framework 4.7;
* MS SQL Server 2017.

Требования к аппаратному обеспечению на сервере:

* операционная система Windows 8.1 и операционные системы более поздней версии;
* дисковое пространство 500Мб для установки программы и дисковое пространство для хранения базы данных, которое зависит от объема данных;
* оперативная память не менее 4 ГБ;
* рекомендуемая частота процессора 2,0 ГГц или выше;
* процессор x64: AMD Opteron, AMD Athlon 64, Intel Xeon с поддержкой Intel EM64T, Intel Pentium IV с поддержкой EM64T.

1. Реализация
   1. Структура системы

Предполагается, что аутентификация пользователей будет происходить на стороне внешней системы, в ходе которой для пользователя генерируется токен, который в дальнейшем используется как средство авторизации запросов к сервисам разрабатываемой системы.

* 1. Модели данных
     1. Реляционная модель данных

На основе анализа реляционной модели данных спроектирована база данных. Логическая модель базы данных в нотации Баркера представлена на рис. 5.1.

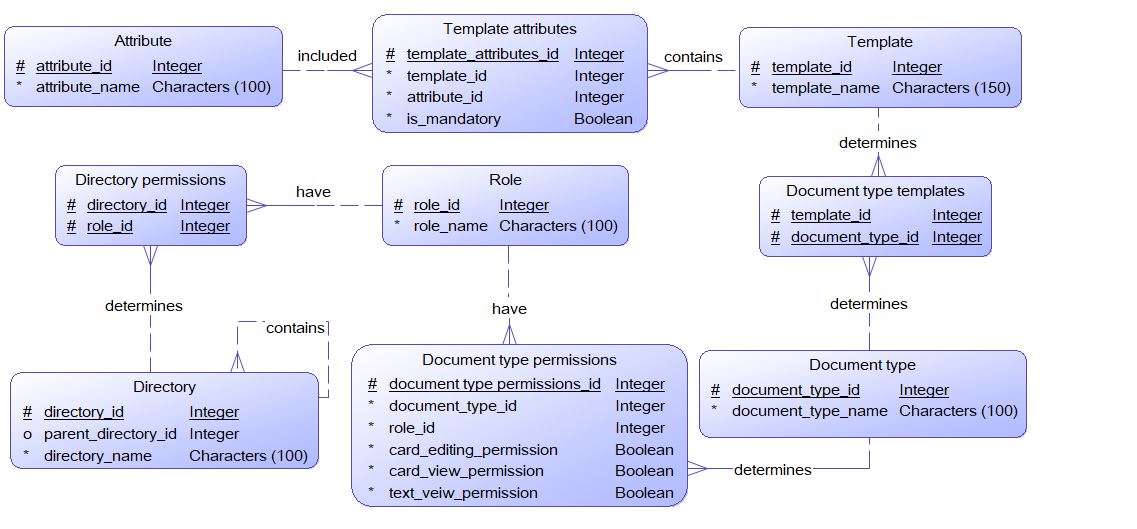


Рис. 5.1. Логическая модель базы данных в нотации Баркера

Листинг SQL-скрипта для создания базы данных приведен в приложении 1. Описание сущностей базы данных содержится в приложении 2.

* + 1. Документоориентированная модель данных

Документоориентированная база данных (далее - ДБД) хранит данные в коллекциях, состоящих из объектов (документов) в формате JSON, которые не имеют заранее определенного формата и схемы. На основе анализа модели данных спроектирована база данных, состоящая из одной коллекции с названием documents, в которой хранятся документы следующего вида:

{

\_id: ‘…’,

upload\_date: ‘…’,

directory\_name: ‘…’,

document\_type: ‘…’,

user: ‘…’,

file\_path: ‘…’,

text: ‘…’,

attributes:

{

attribute\_name: ‘attribute\_value’,

attribute\_name: ‘attribute\_value’,

…

}

permissions:

{

card\_editing: [‘role\_name’, ‘role\_name’, …],

card\_view: [‘role\_name’, ‘role\_name’, …]

text\_view: [‘role\_name’, ‘role\_name’, …]

delete: [‘role\_name’, ‘role\_name’, …]

}

}

Описание полей документов содержится в табл. 5.1.

Таблица 5.1. Описание полей объектов, хранящихся в ДБД

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Описание** |
| \_id | Object ID | Уникальный идентификатор документа,  который генерируется автоматически. Используется в качестве регистрационного номера документа. |
| upload\_date | Date | Дата и время загрузки документа |
| directory\_name | String | Каталог, в котором хранится документ |
| document\_type | String | Вид документа |
| user | String | ФИО пользователя, загрузившего документ |
| file\_path | String | Путь к файлу документа в файловом хранилище |
| text | String | Содержимое документа для полнотекстового поиска (храним только для документов, объем которых менее 40000 символов) |
| attributes | Object | Набор атрибутов и их значений |
| permissions | Object | Набор возможных прав на документ |
| card\_editing | Array | Набор имен ролей, которые имеют право редактирования карточки данного документа |
| card\_view | Array | Набор имен ролей, которые имеют право просмотра карточки данного документа |
| text\_view | Array | Набор имен ролей, которые имеют право просмотра текста данного документа |
| delete | Array | Набор имен ролей, которые имеют право удаления данного документа |

* 1. Диаграмма классов

В соответствии с моделями данных для каждого микросервиса реализованы классы данных, структура и взаимосвязи между ними представлены на диаграммах классов (рис. 5.2 – 5.4).

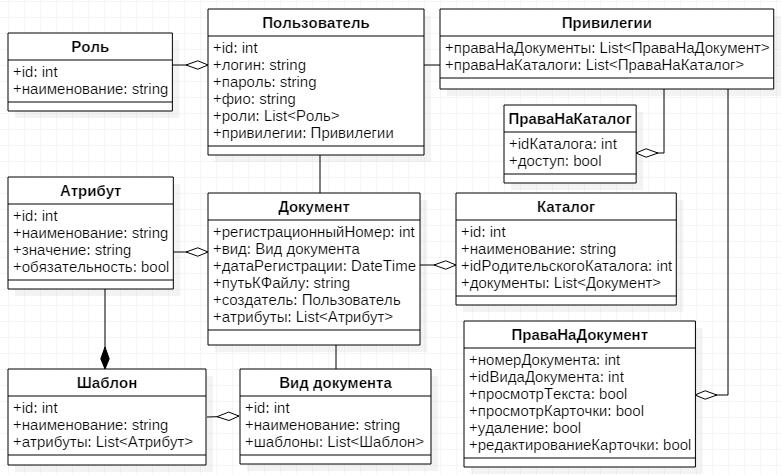


Рис. 5.2. Диаграмма классов

Листинг классов данных приведен в приложении 3.

Заключение

В ходе данной работы спроектирована система обработки документов, предоставляющая следующую функциональность:

* загрузка и регистрация документов;
* представление документа в виде набора значений атрибутов;
* распознавание текстов документов;
* хранение документов;
* хранение атрибутов документов;
* атрибутивный поиск документов;
* просмотр документов;
* удаление документов;
* редактирование карточек документов;
* разграничение прав доступа;
* создание шаблонов для каждого вида документов;
* создание каталогов.

Выполнена практическая реализация спроектированной системы и разработка прототипа клиентского приложения.

Список литературы

1. ЕВФРАТ-Документооборот. Руководство пользователя. – URL: http://www.etax.ru/files/14/documents/doc\_130\_EVFRAT%20Rukovodstvo%20polzovatelya.pdf (дата обращения: 15.04.2018).
2. ЕВФРАТ-Документооборот. Учебное пособие для пользователей. – URL: http://asu.ugatu.ac.ru/library/17/uchebnoe\_posobie\_dlja\_pol\_zovatelei.pdf (дата обращения: 15.04.2018).
3. Инструкция по работе в сэд «Дело» (толстый клиент). – URL: http://dmee.ru/docs/100/index-32726.html (дата обращения: 20.04.2018).
4. Леоненков А.В. Самоучитель UML / А.В. Леоненков // StudFiles – URL: https://studfiles.net/preview/3568463/ (дата обращения: 10.06.2018).
5. Организационный документооборот и архив. Типовое решение. – URL: http://parentalnotes.com/wp-content/uploads/2016/12/Document\_management\_and\_archive.pdf (дата обращения: 18.05.2018).
6. Платформа ELMA BPM. Руководство пользователя. – URL: https://www.elma-bpm.ru/kb/help/Platform/content/User\_Guide\_User\_index.html (дата обращения: 20.04.2018).
7. Система электронного документооборота (СЭД): что такое, особенности и рекомендации. – URL: http://fb.ru/article/279568/sistema-elektronnogo-dokumentooborota-sed-chto-takoe-osobennosti-i-rekomendatsii (дата обращения: 03.04.2018).
8. Системы ECM/СЭД. – URL: http://www.ecm-portal.ru/practice/software/ (дата обращения: 14.06.2018).
9. СОВРЕМЕННЫЕ СИСТЕМЫ ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА. КРАТКИЙ ОБЗОР. – URL: http://docplace.ru/wp-content/uploads/2017/07/sed.pdf (дата обращения: 03.04.2018).
10. СЭД «DIRECTUM». ОБЩАЯ инструкция пользователя по работе в системе (подробная). – URL: http://document.vstu.edu.ru/document-uit/directum/directum-all/158-instruktsiya-po-rabote-s-sistemoj-directum/file (дата обращения: 15.04.2018).
11. СЭД «DIRECTUM». ОБЩАЯ инструкция пользователя по работе в системе (подробная). – URL: http://document.vstu.edu.ru/document-uit/directum/directum-all/158-instruktsiya-po-rabote-s-sistemoj-directum/file (дата обращения: 15.04.2018).
12. Умалатова Л.М. ОБЗОР РОССИЙСКОГО РЫНКА СИСТЕМ ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА / Л.М. Умалатова // Международный научно-технический журнал «ТЕОРИЯ. ПРАКТИКА. ИННОВАЦИИ»: электрон. журн. – 2018. – №4. – URL: http://www.tpinauka.ru/2018/04/Umalatova.pdf (дата обращения: 05.06.2018).
13. 1С: Документооборот 8. – URL: http://v8.1c.ru/doc8/ (дата обращения: 20.04.2018).
14. EMC Documentum Webtop. Версия 6.7. Руководство пользователя. – URL: http://asud.mobilegtes.ru:7777/asud\_mgtes/help/ru\_RU/userguide.pdf (дата обращения: 15.04.2018).
15. Docsvision 5. Система управления документами и бизнес-процессами. Приложение "Управление документами". Руководство пользователя. – URL: http://www.mm-27.ru/docsvision/DV\_DocumentManagement\_UserGuide\_ru.pdf (дата обращения: 15.04.2018).
16. Documentum 5. Архитектура платформы. – URL: http://t-asu.ru/images/page41/documentum\_5.pdf (дата обращения: 15.04.2018).

Приложение 1. SQL-скрипт для создания базы данных

CREATE SCHEMA `documents`;

CREATE TABLE `documents`.`attribute` (

`attribute\_id` int(11) NOT NULL AUTO\_INCREMENT,

`attribute\_name` varchar(100) NOT NULL,

PRIMARY KEY (`attribute\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`template` (

`template\_id` int(11) NOT NULL AUTO\_INCREMENT,

`template\_name` varchar(150) NOT NULL,

PRIMARY KEY (`template\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`template\_attributes` (

`template\_id` int(11) NOT NULL,

`attribute\_id` int(11) NOT NULL,

`is\_mandatory` tinyint(4) NOT NULL,

PRIMARY KEY (`template\_id`,`attribute\_id`),

KEY `attr\_fk\_idx` (`attribute\_id`),

CONSTRAINT `attr\_fk` FOREIGN KEY (`attribute\_id`) REFERENCES `attribute` (`attribute\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `tem\_fk` FOREIGN KEY (`template\_id`) REFERENCES `template` (`template\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`document\_type` (

`document\_type\_id` int(11) NOT NULL AUTO\_INCREMENT,

`document\_type\_name` varchar(100) NOT NULL,

PRIMARY KEY (`document\_type\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`document\_type\_templates` (

`template\_id` int(11) NOT NULL,

`document\_type\_id` int(11) NOT NULL,

PRIMARY KEY (`template\_id`,`document\_type\_id`),

KEY `doc\_type\_fk\_idx` (`document\_type\_id`),

CONSTRAINT `doc\_type\_fk` FOREIGN KEY (`document\_type\_id`) REFERENCES `document\_type` (`document\_type\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `template\_fk` FOREIGN KEY (`template\_id`) REFERENCES `template` (`template\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`role` (

`role\_id` int(11) NOT NULL AUTO\_INCREMENT,

`role\_name` varchar(100) NOT NULL,

PRIMARY KEY (`role\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`document\_type\_permissions` (

`document\_type\_id` int(11) NOT NULL,

`role\_id` int(11) NOT NULL,

`card\_editing\_permission` tinyint(4) NOT NULL,

`card\_view\_permission` tinyint(4) NOT NULL,

`text\_view\_permission` tinyint(4) NOT NULL,

`delete\_document\_permission` tinyint(4) NOT NULL,

PRIMARY KEY (`document\_type\_id`,`role\_id`),

KEY `role\_perm\_fk\_idx` (`role\_id`),

CONSTRAINT `doc\_type\_perm\_fk` FOREIGN KEY (`document\_type\_id`) REFERENCES `document\_type` (`document\_type\_id`),

CONSTRAINT `role\_perm\_fk` FOREIGN KEY (`role\_id`) REFERENCES `role` (`role\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`directory` (

`directory\_id` int(11) NOT NULL AUTO\_INCREMENT,

`parent\_directory\_id` int(11) DEFAULT NULL,

`directory\_name` varchar(100) NOT NULL,

PRIMARY KEY (`directory\_id`),

KEY `parent\_dir\_fk\_idx` (`parent\_directory\_id`),

CONSTRAINT `parent\_dir\_fk` FOREIGN KEY (`parent\_directory\_id`) REFERENCES `directory` (`directory\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `documents`.`directory\_permissions` (

`directory\_id` int(11) NOT NULL,

`role\_id` int(11) NOT NULL,

`access` tinyint(4) NOT NULL,

PRIMARY KEY (`directory\_id`,`role\_id`),

KEY `role\_dir\_perm\_fk\_idx` (`role\_id`),

CONSTRAINT `dir\_perm\_fk` FOREIGN KEY (`directory\_id`) REFERENCES `directory` (`directory\_id`),

CONSTRAINT `role\_dir\_perm\_fk` FOREIGN KEY (`role\_id`) REFERENCES `role` (`role\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Приложение 2. Описание сущностей базы данных

Таблица 1. Attribute

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| attribute\_id | Уникальный идентификатор атрибута |
| attribute\_name | Наименование атрибута |

Таблица 2. Template

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| template\_id | Уникальный идентификатор шаблона |
| template\_name | Наименование шаблона |

Таблица 3. Document type

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| document\_type\_id | Уникальный идентификатор вида документа |
| document\_type\_name | Наименование вида документа |

Таблица 4. Document type templates

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| template\_id | Уникальный идентификатор шаблона |
| document\_type\_id | Уникальный идентификатор вида документа |

Таблица 5. Role

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| role\_id | Уникальный идентификатор роли |
| role\_name | Наименование роли |

Таблица 6. Template attributes

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| template\_id | Уникальный идентификатор шаблона |
| attribute\_id | Уникальный идентификатор атрибута |
| is\_mandatory | Обязательность атрибута для данного шаблона |

Таблица 7. Document type permissions

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| document\_type\_id | Уникальный идентификатор вида документа |
| role\_id | Уникальный идентификатор роли |
| card\_editing\_permission | Наличие у роли права редактирования карточки документов данного вида |
| card\_view\_permission | Наличие у роли права просмотра карточки документов данного вида |
| text\_view\_permission | Наличие у роли права просмотра текста документов данного вида |
| delete\_document\_permission | Наличие у роли права удаления документов данного вида |

Таблица 8. Directory

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| directory\_id | Уникальный идентификатор каталога |
| parent\_directory\_id | Уникальный идентификатор родительского каталога |
| directory\_name | Наименование каталога |

Таблица 9. Directory permissions

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| directory\_id | Уникальный идентификатор каталога |
| role\_id | Уникальный идентификатор роли |
| access | Наличие у роли права доступа к каталогу |

Приложение 3. Листинг классов данных