

УТВЕРЖДЕНО

Коммерческий директор

ООО «Логика бизнеса»

_____ Г.Н. Подбуцкий

«__» _____ 2015 г.

**Система электронного документооборота
"Логика СЭД. СПО"**

Версия 4.5

Архитектура СЭД "Логика СЭД. СПО"

СОДЕРЖАНИЕ

I.	<u>ВВЕДЕНИЕ</u>	8
II.	<u>ОПИСАНИЕ ОСНОВНЫХ МОДУЛЕЙ СЭД «ЛОГИКА СЭД. СПО»</u>	10
II.1.	ПЕРЕЧЕНЬ И НАЗНАЧЕНИЕ ПРИКЛАДНЫХ МОДУЛЕЙ	10
II.1.1.	Модуль «ВХОДЯЩИЕ»	10
II.1.2.	Модуль «ИСХОДЯЩИЕ»	11
II.1.3.	Модуль «ВНУТРЕННИЕ»	12
II.1.4.	Модуль «ОБРАЩЕНИЯ ГРАЖДАН»	13
II.1.5.	Модуль «ОРД»	14
II.1.6.	Модуль «СПРАВОЧНИКИ»	15
II.1.7.	Модуль «ОТЧЕТЫ»	16
II.1.8.	Модуль «АРМ»	16
II.2.	ПЕРЕЧЕНЬ И НАЗНАЧЕНИЕ ФУНКЦИОНАЛЬНЫХ МОДУЛЕЙ	17
II.2.1.	Модуль ведения шаблонов	17
II.2.2.	Модуль создания и публикации информационных материалов	17
II.2.3.	Модуль поиска материалов	17
II.2.4.	Модуль администрирования	17
II.2.5.	Модуль управления доступом	17
II.2.6.	Модуль сканирования	18
II.2.7.	Электронная подпись	22
III.	<u>ВЗАИМОДЕЙСТВИЕ МОДУЛЕЙ СИСТЕМЫ</u>	28
IV.	<u>ОПИСАНИЕ ТРАНСПОРТНОЙ ПОДСИСТЕМЫ</u>	32
IV.1.	МЕЖВЕДОМСТВЕННЫЙ ЭЛЕКТРОННЫЙ ДОКУМЕНТООБОРОТ	32
IV.2.	ГОСТ	34
IV.3.	ТРАНСПОРТНЫЙ АГЕНТ	41
IV.4.	РЕПЛИКАЦИЯ	46
V.	<u>ИНТЕГРАЦИИ С ВНЕШНИМИ СИСТЕМАМИ</u>	60
V.1.	ИНТЕГРАЦИЯ С ВНЕШНИМИ СИСТЕМАМИ	60
V.2.	ИНТЕГРАЦИЯ НА ОСНОВЕ ПРЯМОГО ДОСТУПА К БАЗАМ ДАННЫХ	60
V.3.	ИНТЕГРАЦИЯ НА ОСНОВЕ УДАЛЕННОГО ВЫЗОВА ПРОЦЕДУР ВНЕШНИХ СИСТЕМ	60
V.4.	ИНТЕГРАЦИЯ НА ОСНОВЕ WSRP	61
V.5.	ОБЩАЯ СХЕМА ИНТЕГРАЦИИ СЭД «ЛОГИКА СЭД. СПО» С ВНЕШНИМИ СИСТЕМАМИ	63
VI.	<u>ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ</u>	64

VI.1.	АРХИТЕКТУРА СИСТЕМЫ	64
VI.2.	ОПИСАНИЕ АРХИТЕКТУРЫ	64
VI.3.	ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ	65
VI.4.	МЕТОДЫ ДОСТУПА К ИНФОРМАЦИИ	66
VI.5.	ПОРТЛЕТЫ	66
VI.6.	ЛОГИЧЕСКАЯ СТРУКТУРА ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ	67
<u>VII.</u>	<u>ПЕРЕЧЕНЬ ПРИКЛАДНЫХ ТАБЛИЦ СЭД «ЛОГИКА СЭД. СПО» С УКАЗАНИЕМ НАЗНАЧЕНИЯ И ОПИСАНИЕ СТРУКТУРЫ</u>	<u>69</u>
<hr/>		
VII.1.	СТРУКТУРА БАЗЫ ДАННЫХ	69
VII.2.	ОПИСАНИЕ ТАБЛИЦ	69
VII.3.	ТАБЛИЦЫ СТРУКТУРЫ	70
VII.4.	ТАБЛИЦЫ СВОЙСТВ АТТРИБУТОВ	74
VII.5.	ТАБЛИЦЫ ЖЦ КАРТОЧЕК	76
VII.6.	ТАБЛИЦЫ ПОЛЬЗОВАТЕЛЕЙ И РОЛЕЙ	78
VII.7.	ТАБЛИЦЫ ЗНАЧЕНИЙ И ИСТОРИИ	80
VII.8.	ТАБЛИЦЫ ПРАВ И СОСТОЯНИЙ	83
VII.9.	ПРОЧИЕ ТАБЛИЦЫ	85
<u>VIII.</u>	<u>СТРУКТУРА КОДА СЭД «ЛОГИКА СЭД. СПО»</u>	<u>86</u>
<hr/>		
VIII.1.	КОМПОНЕНТЫ	86
VIII.2.	ОСНОВНЫЕ ПОНЯТИЯ	86
VIII.3.	ОБЩАЯ СТРУКТУРА И ВЗАИМОСВЯЗИ	88
VIII.4.	ОБЪЕКТНАЯ МОДЕЛЬ	90
VIII.5.	ГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ (GUI)	92
VIII.6.	НАВИГАЦИЯ	97
<u>IX.</u>	<u>ПРИЛОЖЕНИЕ А. ОПИСАНИЕ ПРОТОКОЛОВ</u>	<u>100</u>

ТЕРМИНЫ И СОКРАЩЕНИЯ

В настоящем документе применены следующие термины с соответствующими определениями (Таблица 1).

Таблица 1. Соглашения по терминологии

Элемент	Описание
Атрибут (поле)	Основная единица хранения информации в карточке документа. Атрибут (поле) заполняется пользователем (вручную) или системой (автоматически)
Блок	Способ группировки данных. Блоки используются для создания шаблонов карточек, одни и те же блоки могут быть использованы в разных шаблонах. Блоки состоят из набора атрибутов
Вкладка	Блоки и поля карточки документа располагаются на отдельных вкладках. Чтобы получить доступ к полям определенной вкладки, достаточно щелкнуть мышью в области ее заголовка
Внутренние документы	Официальный документ, не выходящий за пределы подготовившей его организации
Входящий документ	Документ, поступивший в учреждение
Делопроизводство; документационное обеспечение управления	Отрасль деятельности, обеспечивающая документирование и организацию работы с официальными документами
Документ, документированная информация	Зафиксированная на материальном носителе информация с реквизитами, позволяющими ее идентифицировать
Исполнитель	Работник отдела-исполнителя, на которого возлагается осуществление поиска и поставки материальных ресурсов (требуемых услуг, работ) для организации, проведение анализа коммерческих предложений и (или) подготовка, оформление, согласование, визирование и выполнение иных функций, необходимых для заключения договора
Исходящий документ	Официальный документ, отправляемый из организации
Контролер	Сотрудник организации, осуществляющий контроль за исполнением поручений и решений, формализованных в РД, в соответствии с возложенными на него обязанностями
Куратор	Определенное распоряжением Инициатора должностное лицо, и/или руководитель структурного подразделения, ответственное за подготовку, заключение и исполнение конкретного договора, а также за соблюдение требований к делопроизводству при заключении договора и работы с договорной документацией
Линейный справочник	Содержит значения, которые используются для заполнения атрибутов (полей) карточек документов
Номенклатура дел	Систематизированный перечень наименований дел, заводимых в организации, с указанием сроков их хранения, оформленный в установленном порядке
Нумератор	В нумераторе задаются правила, по которым будет рассчитываться текущий номер для регистрируемого в системе документа. Таким образом, нумератор содержит сведения о «счетчике», но не позволяет задать формат регистрационного номера

Элемент	Описание
Организационно-распорядительный документ (ОРД)	Вид письменного документа, в котором фиксируют решения административных и организационных вопросов, а также вопросов управления, взаимодействия, обеспечения и регулирования деятельности органов власти, учреждений, организаций, предприятий их подразделений, должностных лиц
Ответственный исполнитель (исполнитель)	Должностное лицо организации, осуществляющее исполнение документа (разработку проекта документа и подготовку его к изданию)
Переход	Изменение статуса карточки документа (например, переход из статуса «Зарегистрирован» в статус «В дело»)
Подписант	Сотрудник организации, наделенный полномочиями подписывать документ
Пользовательское представление	Совокупность документов, сгруппированных по нескольким функциональным критериям, создающая удобную среду для решения задач пользователя системы
Поручение	Резолюция, часть резолюции, касающаяся выполнения определенного пункта резолюции определенным исполнителем
Раздел системы	Система организована в виде разделов. Каждый раздел системы объединяет информационные единицы определенного типа (Исходящие, Внутренние, ОРД и т. д.). В свою очередь, внутри каждого раздела разграничиваются пользовательские представления
Распорядительные документы	Документы, в которых фиксируются решения административных и организационных вопросов деятельности организации
Регистрационно-контрольная карточка (РКК) документа	Набор реквизитов документа, позволяющих зафиксировать информацию, содержащуюся в документе, и достаточных для его идентификации, представленный в виде единого учетного объекта в соответствии с правилами делопроизводства организации
Регистрационный индекс (номер) документа	Цифровое или буквенно-цифровое обозначение документа, присваиваемое документу при его регистрации
Реестр	Перечень (список) чего-либо
Резолюция документа	Реквизит, состоящий из надписи на документе, сделанной должностным лицом и содержащей принятое им решение. Указание по исполнению документа подчиненным лицам
Реквизиты документа	Обязательные элементы оформления документа организации
Система межведомственного электронного документооборота (МЭДО)	Федеральная информационная система, предназначенная для организации взаимодействия систем электронного документооборота (СЭД) участников межведомственного электронного документооборота
Согласование документа	Процедура, призванная подтвердить согласие с содержанием документа должностными лицами, не являющимися авторами документа, путем соответствующего визирования. Виза включает личную подпись и ЭП, ее расшифровку (ФИО), должность визирующего лица, дату визирования
Соисполнитель	Сотрудник организации, указанный в поручении в качестве одного из исполнителей поручения, если назначено несколько исполнителей поручения и данный сотрудник не указан первым
Списание документа в дело	Передача исполненного документа на временное оперативное хранение (срок определяется в номенклатуре дел)
Справочник	Место хранения нормативно-справочной информации
Статус документа	Текущая стадия выполнения документа. Статус меняется автоматически по мере прохождения жизненного цикла документа

Элемент	Описание
Формат нумератора	Формат номера может включать в себя, помимо счетчика, дополнительные параметры (текущую дату, разделители и т. д.), в зависимости от порядка формирования регистрационных номеров, принятого в Организации
Характеристика	Минимальная часть для описания материала учетной карточки. Одни и те же характеристики могут быть использованы в различных блоках
Шаблон	Принятая в системе форма для создания карточек. Шаблон состоит из блоков. Шаблон объединяет в себе описание процесса обработки, состав атрибутов, права пользователей
Электронная подпись	Реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной подписи и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе
Электронная почта	Один из компонентов системы автоматизации документооборота, средство доставки, отправки информации и ее передачи между пользователями как внутри организации, так и между организациями, имеющими соответствующие аппаратные и программные средства

Перечень используемых сокращений приведен в таблице (Таблица 2).

Таблица 2. Список используемых сокращений

Элемент	Описание
GUI	Graphic User Interface (Графический интерфейс пользователя)
JAI	Java Advanced Imaging (Инструмент Java для ввода, вывода и обработки графических изображений)
JVM	Виртуальная машина Java
OCSP	Online Certificate Status Protocol (Протокол получения статуса сертификата в реальном времени)
SOAP	Simple Object Access Protocol (Простой протокол доступа к объектам)
TWAIN	Technology Without Any Interesting Name (Стандартный протокол и интерфейс, определяющий взаимодействие между программами и устройствами захвата изображения, такими как сканеры и цифровые камеры)
UDDI	Universal Description Discovery & Integration (Инструмент для расположения описаний веб-сервисов для последующего их поиска другими организациями и интеграции в свои системы)
WSDL	Web Services Description Language (Язык описания веб-сервисов и доступа к ним)
WSRP	Web Services for Remote Portlets (Стандарт сетевого протокола для связи с удаленными портлетами)
APM	Автоматизированное рабочее место
БД	База данных
БН	Балансировщик нагрузки
ГОСТ	Государственный стандарт
ДОУ	Документационное обеспечение управления
ИС	Информационная система
ЛВС	Локальная вычислительная сеть
МАЭД	Модуль архивирования ЭД
МВ	Модуль взаимодействия в формате СЭД «Дело»
МВ ГОСТ	Модуль взаимодействия в формате ГОСТ Р 53898-2010

Элемент	Описание
МВ МЭДО	Модуль взаимодействия в формате МЭДО
МГО	Модуль генерации отчетов
МКД	Модуль конвертирования документов
МКП	Модуль крипто-провайдера
ММК	Модуль мобильных клиентов
МПЗ	Модуль периодических задач
МПР	Модуль почтовых рассылок
МПЭП	МПЭП модуль проверки ЭП
МР	Модуль репликации
МСД	Модуль поддержки сканирования документов
МУ	Модуль уведомлений
МУНД	Модуль управления неструктурированными данными
МЭДО	Система межведомственного электронного документооборота
МЭП	Модуль ЭП
НИИ	Научно-исследовательский институт
НИИИТ	Научно-исследовательский институт информационных технологий
НПА	Нормативный правовой акт
НПБ	Нормативно-правовая база
ОГ	Обращение гражданина
ООО	Общество с ограниченной ответственностью
ОРД	Организационно-распорядительный документ
ОС	Операционная система
ОЧП	Основная часть приложения
ПК	Персональный компьютер
ПМИ	Программа и методика испытаний
ПО	Программное обеспечение
РГ	Рабочая группа
РД	Распорядительный документ
РКК	Регистрационно-контрольная карточка
РФ	Российская Федерация
СА	Сертификационное агентство
Система	Федеральная государственная информационная система «Электронный документооборот уголовно-исполнительной системы»
СКЗИ	Средства криптографической защиты информации
СМ	Сервис мониторинга состояния сервера
СМК	Сервис мобильных клиентов
СП	Сервис приложения
СУБД	Система управления базами данных
СХПД	Сервис хранения и предоставления данных
СЭД	Система электронного документооборота
ТА	Транспортный агент
ТЗ	Техническое задание на создание информационной системы СЭД «Логика СЭД. СПО»
ТМ	Таблица маршрутизации
ТУ	Транспортный узел
УЦ	Удостоверяющие центры
ФЗ	Федеральный закон
ФИО	Фамилия, имя, отчество
ХНД	Хранилища неструктурированных данных
ХСД	Хранилища структурированных данных
ЭД	Электронный документ
ЭП	Электронная подпись

I. Введение

СЭД «Логика СЭД. СПО» (далее Система, СЭД) – автоматизированная многопользовательская система, сопровождающая процесс управления работой иерархической организации с целью обеспечения выполнения этой организацией своих функций. Процесс управления опирается на человеко-читаемые документы, содержащие инструкции для сотрудников организации, необходимые к исполнению.

Электронный документ (ЭД) – информация, зафиксированная на электронном материальном носителе. ЭД:

- создается, обрабатывается, хранится и передается с помощью электронных технических средств;
- может быть представлен в форме, пригодной для восприятия человеком, не обладающим специальными техническими навыками;
- при его составлении, хранении, передаче использован предусмотренный способ, позволяющий достоверно идентифицировать составителя ЭД.

Регистрационно-контрольная карта (РКК) – набор реквизитов документа, позволяющих зафиксировать информацию, содержащуюся в документе, и достаточных для его однозначной идентификации, представленный в виде единого учетного объекта в соответствии с правилами делопроизводства организации.

Пользователь СЭД (далее просто пользователь) – лицо или программно-аппаратный комплекс имеющий право участвовать в функционировании СЭД и использовать функционал СЭД или его часть задействуя заранее задекларированные способы взаимодействия.

Справочник – группа ЭД, объединенных по какому-либо логическому критерию, основная роль которых – предоставление нормативно-справочной информации для остальных ЭД.

Представление электронного документа в СЭД «Логика СЭД. СПО»

Электронный документ может быть представлен при помощи одной и более РКК, причем одна РКК, которая содержит основную регистрационно-контрольную информацию называется основной, а остальные – вспомогательными.

Данные о пользователе

Для каждого пользователя в СЭД обязательно содержится учетная и контрольная информация. Учетная информация о пользователе состоит из:

- числового идентификатора пользователя (ID);
- символьного идентификатора пользователя (логин);
- пароля пользователя;
- полного имени пользователя;

- электронного адреса пользователя (e-mail).

Первые два идентификатора являются обязательными, уникальными и однозначно соответствуют друг другу. Пароль также является обязательной частью учетной информации о пользователе.

Контрольная информация о пользователе может быть представлена в различных форматах, которые зависят от условий использования СЭД, но опирается на существование в СЭД РКК шаблона «Персона», которая однозначно связана с регистрационной информацией. Контрольная информация может содержать данные об:

- иерархическом положении пользователя в организации, которую он представляет;
- наименовании должности;
- местоположении пользователя (кабинет, этаж, здание);
- доступных средствах связи с ним (телефон, e-mail и т. д.);
- дополнительных ролях, возложенных на него должностной инструкцией.

Организации

Организация, в зависимости от размера может обслуживаться одним экземпляром СЭД. В этом случае каждая организация работает в отдельном виртуальном пространстве документов и справочников. Также существуют общие справочники, доступные всем или части организаций. Системообразующая информация (метаинформация), а именно: шаблоны документов, их статусы, жизненные циклы, линейные справочники являются общими для всех организаций.

II. Описание основных модулей СЭД «Логика СЭД. СПО»

II.1. Перечень и назначение прикладных модулей

II.1.1. Модуль «Входящие»

Модуль «Входящие» обеспечивает обработку и регистрацию всей входящей корреспонденции, поступившей из других организаций (Рисунок 1). Модуль позволяет создавать регистрационно-контрольную карточку (РКК) документа, куда заносится информация по документу: дата регистрации, номер исходящего, краткое содержание документа, адресат, подписант, организация-отправитель и т.д.

Предусмотрены возможности регистрации документа, постановки документа и поручений по документу на контроль, контроль результатов и сроков исполнения поручений.

Основными функциями модуля являются:

- создание и заполнение РКК документа (ввод данных, загрузка вложений);
- регистрация документа;
- отправка документа адресату;
- постановка документа на контроль;
- создание (внесение в РКК документа) резолюций;
- снятие документов с контроля;
- списание документов в дело.

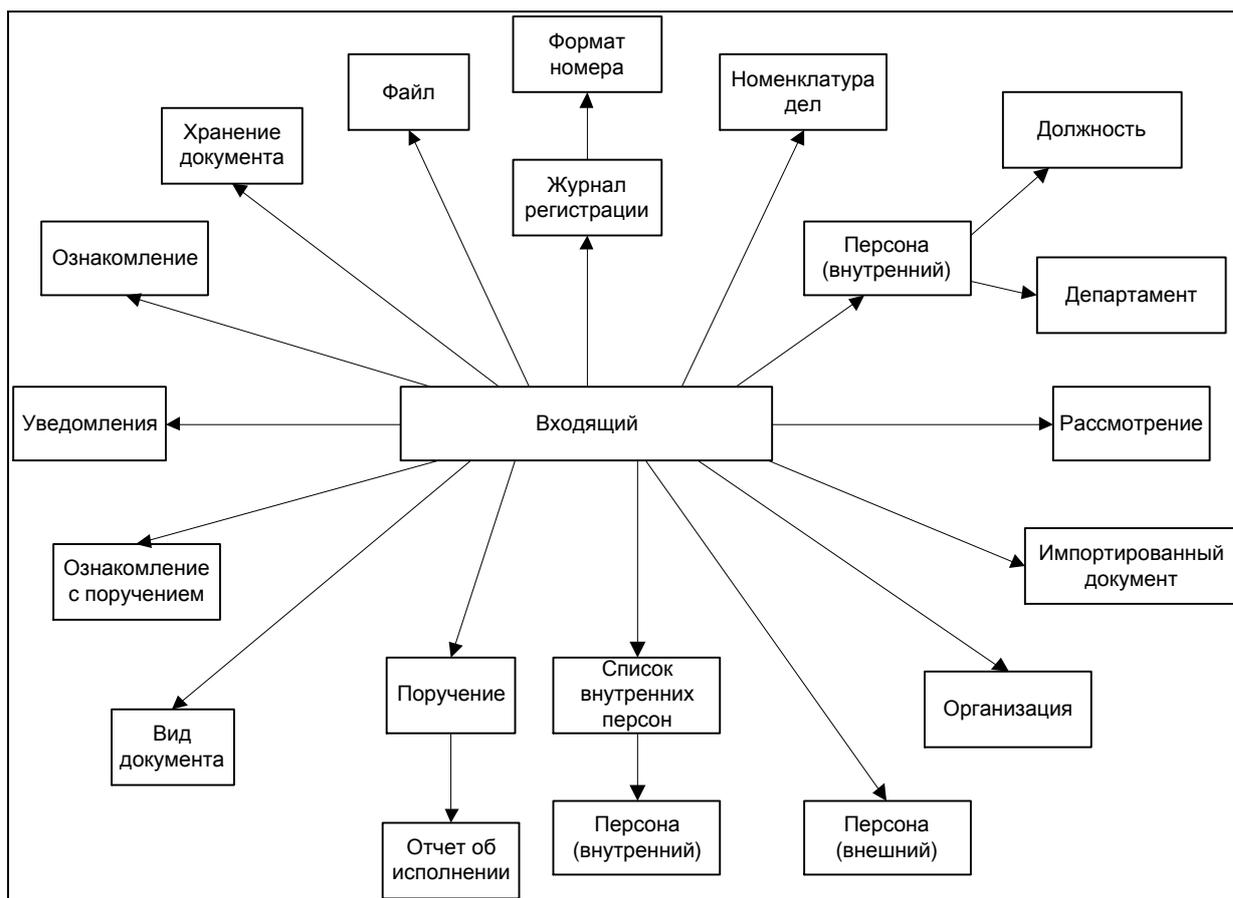


Рисунок 1 – Взаимосвязь шаблона «Входящий» с шаблонами Системы

II.1.2. Модуль «Исходящие»

Модуль «Исходящие» обеспечивает возможность обработки исходящих документов, отправляемых в другие организации (Рисунок 2). Модуль позволяет создавать регистрационно-контрольную карточку (РКК) документа, куда заносится информация по документу: автор проекта документа, подписант, согласующие, краткое содержание документа, организация-получатель документа и т.д.

Модуль позволяет создавать проект исходящего документа (с заполнением данных проекта документа), опрашивать проект по маршруту (на дальнейшее согласование и подписание), регистрировать подписанный проект документа.

Основными функциями модуля являются:

- создание и заполнение РКК проекта документа (ввод данных, загрузка вложений);
- отправка документа по маршруту (на дальнейшее согласование и подписание);
- указание связей по документу;
- регистрация документа;
- регистрация документа со статуса «Подготовка» (минуя согласование и подписание проекта);

- автоматическая отправка документа адресату;
- списание документов в дело.

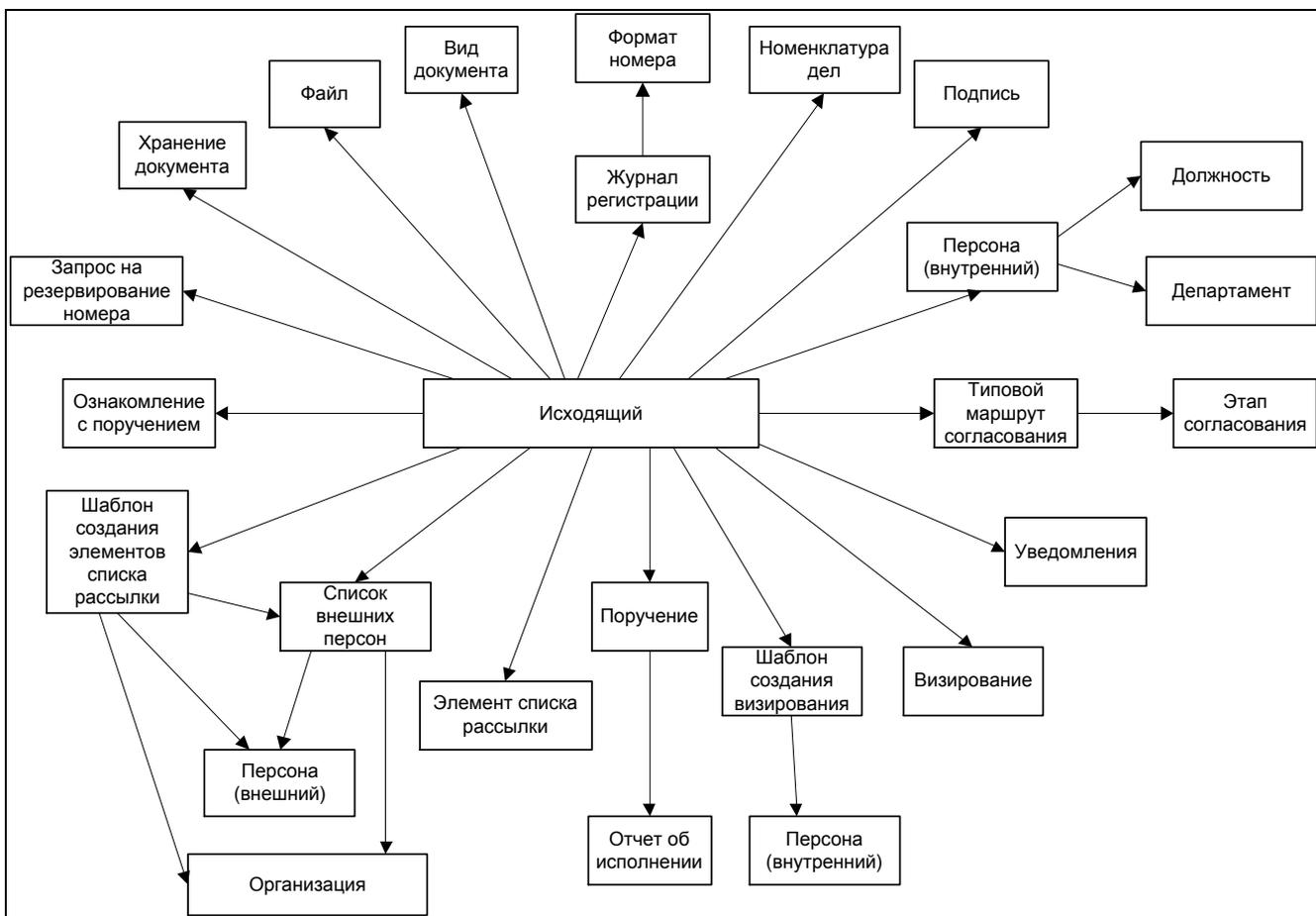


Рисунок 2 – Взаимосвязь шаблона «Исходящий» с шаблонами Системы

II.1.3. Модуль «Внутренние»

Модуль «Внутренние» обеспечивает возможность обработки внутренних документов, передаваемых между сотрудниками организации (Рисунок 3). Модуль позволяет создавать регистрационно-контрольную карточку (РКК) документа, куда заносится информация по документу: автор проекта документа, подписант, согласующие, краткое содержание документа, адресат документа и т.д.

Модуль позволяет создавать проект внутреннего документа (с заполнением данных проекта документа), опраывать проект по маршруту (на дальнейшее согласование и подписание), регистрировать подписанный проект документа, отправлять документ адресату.

Основными функциями модуля являются:

- создание и заполнение РКК проекта документа (ввод данных, загрузка вложений);
- отправка документа по маршруту (на дальнейшее согласование и подписание);

- указание связей по документу;
- регистрация документа;
- регистрация документа со статуса «Подготовка» (минуя согласование и подписание проекта);
- отправка документа адресату;
- постановка документа на контроль;
- создание (внесение в РКК документа) резолюций;
- снятие документов с контроля;
- списание документов в дело.

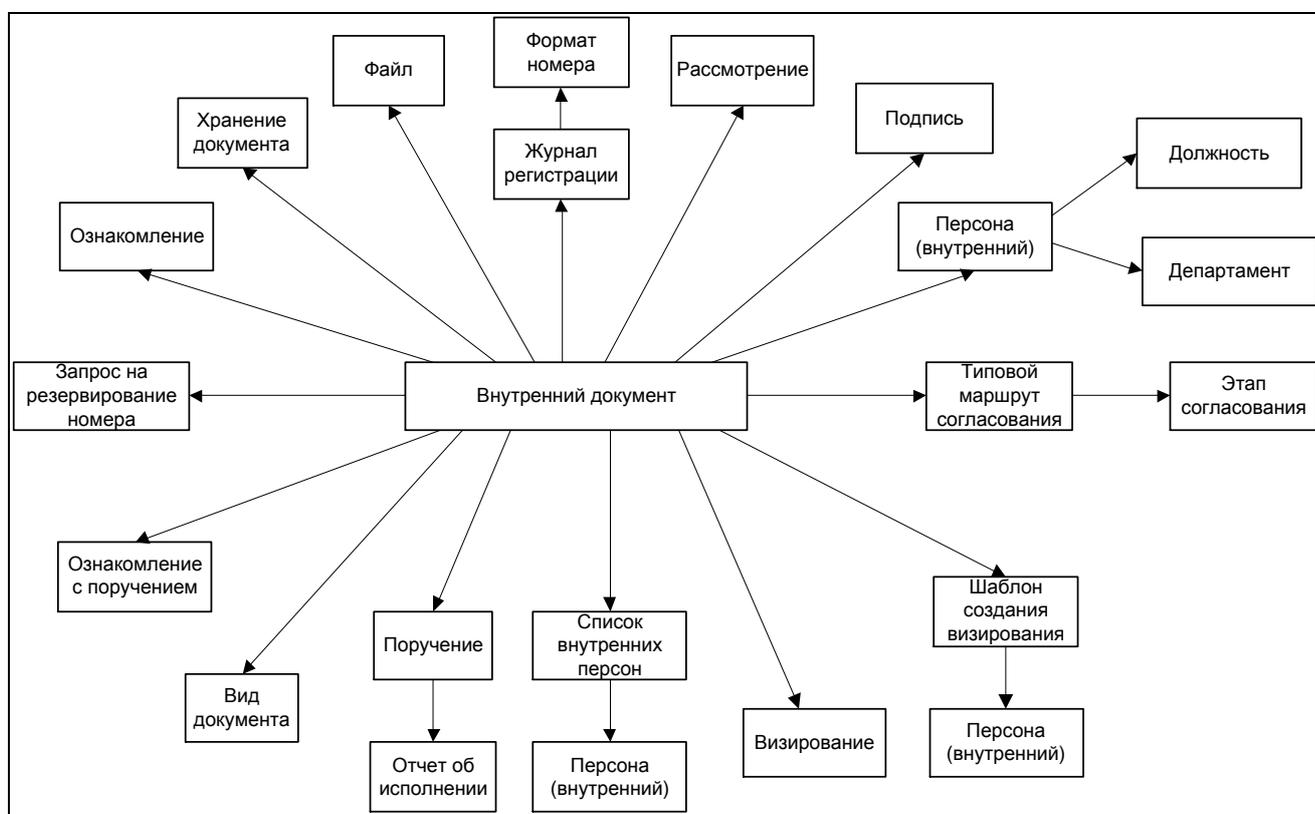


Рисунок 3 – Взаимосвязь шаблона «Внутренний» с шаблонами Системы

II.1.4. Модуль «Обращения граждан»

Модуль «Обращения граждан» обеспечивает возможность обработки поступающих в организацию обращений граждан (Рисунок 4). Модуль позволяет создавать регистрационно-контрольную карточку (РКК) документа, куда заносится информация по документу: краткое содержание документа, адресат, информация об авторе обращения, вопрос обращения и т.д.

Предусмотрены возможности регистрации документа, постановки документа и поручений по документу на контроль, контроль результатов и сроков исполнения поручений.

Основными функциями модуля являются:

- создание и заполнение РКК документа (ввод данных, загрузка вложений);
- регистрация документа;
- отправка документа адресату;
- постановка документа на контроль;
- создание (внесение в РКК документа) резолюций;
- снятие документов с контроля;
- списание документов в дело.

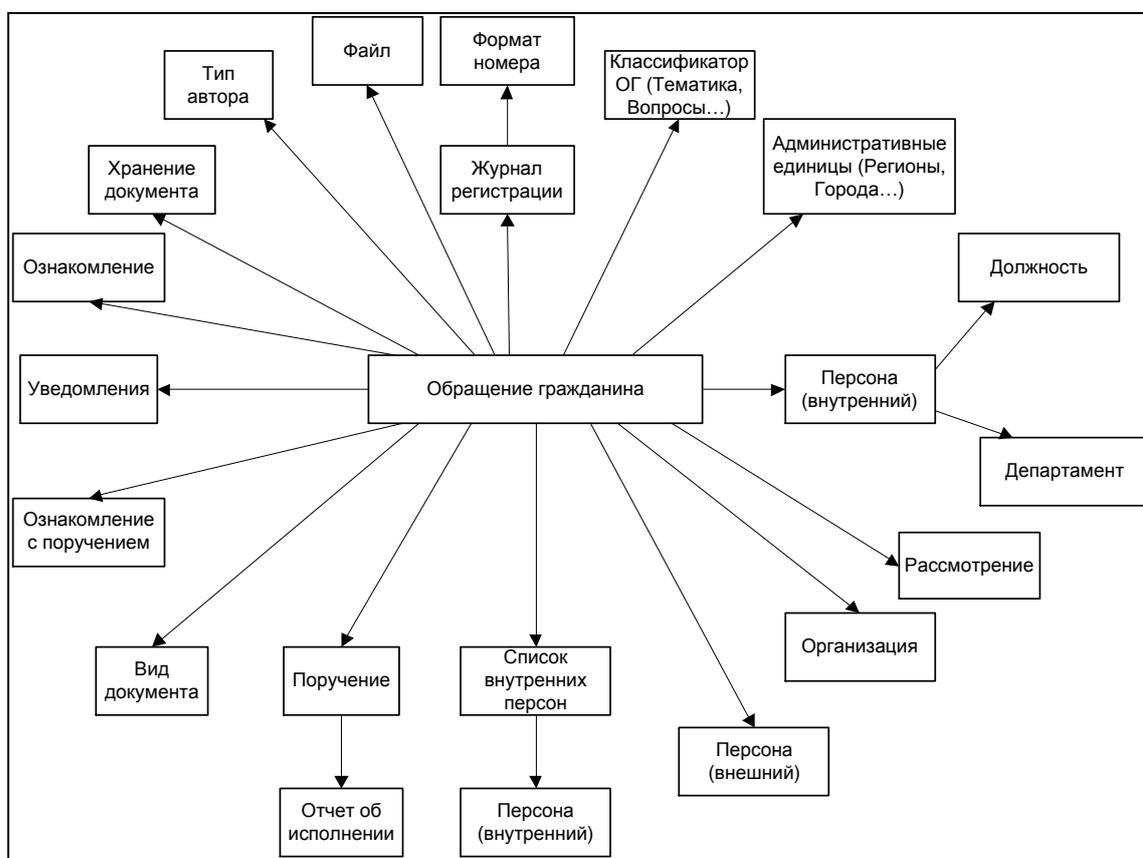


Рисунок 4 – Взаимосвязь шаблона «Обращения граждан» с шаблонами Системы

II.1.5. Модуль «ОРД»

Модуль «ОРД» обеспечивает возможность обработки организационно-распорядительных документов, разрабатываемых в организации (Рисунок 5). Модуль позволяет создавать регистрационно-контрольную карточку (РКК) документа, куда заносится информация по документу: автор проекта документа, подписант, согласующие, краткое содержание документа и т.д.

Модуль позволяет создавать проект организационно-распорядительного документа (с заполнением данных проекта ОРД), отправлять проект по маршруту (на дальнейшее согласование и подписание), регистрировать подписанный проект ОРД.

Основными функциями модуля являются:

- создание и заполнение РКК проекта ОРД (ввод данных, загрузка вложений);
- отправка ОРД по маршруту (на дальнейшее согласование и подписание);
- указание связей по документу;
- регистрация ОРД;
- регистрация ОРД со статуса «Подготовка» (минуя согласование и подписание проекта);
- списание документов в дело.

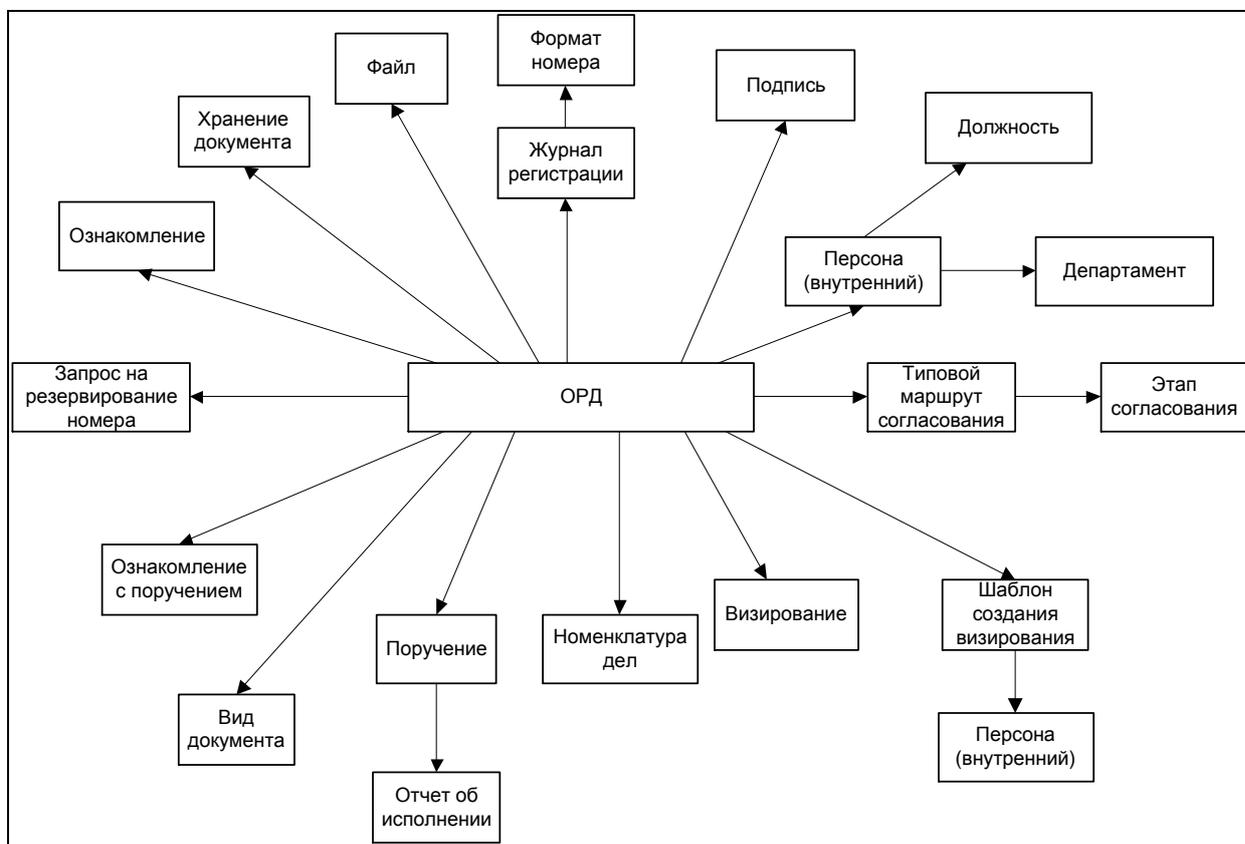


Рисунок 5 – Взаимосвязь шаблона «Обращения граждан» с шаблонами Системы

II.1.6. Модуль «Справочники»

Модуль «Справочники» обеспечивает возможность централизованного хранения справочной информации. Модуль позволяет создавать и использовать карточки справочников.

Основные справочники Системы:

- Персоны внутренние;

- Персоны внешние;
- Авторы обращений;
- Организации;
- Подразделения;
- Должности;
- Классификатор ОГ;
- Номенклатура дел;
- Нумератор;
- Формат нумератора;
- Журнал регистрации;
- Зоны ДОУ;
- Настройки для АРМ руководителя;
- Виды документов;
- Виды исходящих бланков.

II.1.7. Модуль «Отчеты»

Модуль «Отчеты» обеспечивает возможность формирования отчетов по выбранным параметрам.

Основные типы отчетов Системы:

- Отчеты по исполнительской дисциплине;
- Отчеты ОГ;
- Отчеты по ОРД;
- Реестры;
- Другие отчеты.

II.1.8. Модуль «АРМ»

Модуль «АРМ» разработан специально для руководителей и исполнителей и предназначен для удобства работы пользователей. Модуль «АРМ» разработан по принципу «одного окна» - когда вся значимая информация по документу отображается в одном окне, не требуя перехода в служебные карточки. Обеспечивает возможность быстрой обработки документов в минимальное количество нажатий.

II.2. Перечень и назначение функциональных модулей

II.2.1. Модуль ведения шаблонов

Модуль обеспечивает формирование и настройку шаблонов информационных материалов, включая определения составов блоков, входящих в них характеристик и допустимых значений к ним.

II.2.2. Модуль создания и публикации информационных материалов

Модуль создания и публикации информационных материалов обеспечивает:

- создание и редактирование карточек информационных материалов с их последующим согласованием, возможностью хранения версий карточек и вложенных файлов;
- просмотр отдельных карточек в соответствии с правами пользователя с возможностью удобной навигации по ее блокам (особенно актуально в случаях, когда карточка содержит несколько десятков характеристик);
- предоставление возможности пользователю организовать работу со списком «избранных» информационных ресурсов, формирование запросов редакторам, сохранение запросов на поиск – функционал «Личный кабинет».

II.2.3. Модуль поиска материалов

Модуль поиска материалов обеспечивает «простой» и «расширенный» поиск карточек информационных ресурсов, включая возможность использования простого языка запросов и контекстного поиска, как по значениям полей карточки, так и по документам, прикрепленным к карточкам.

II.2.4. Модуль администрирования

Модуль администрирования предоставляет функции аутентификации и авторизации пользователей в соответствии с принятой ролевой моделью и требованиями информационной безопасности при доступе ко всем разделам и ресурсам Системы.

II.2.5. Модуль управления доступом

Модуль управления доступом к данным Системы реализует следующие функции:

- поддержку основных функциональных ролей;
- создание новых функциональных ролей;
- разграничение доступа с учетом типа роли, подразделения, уровня доступа и применяемых шаблонов карточек;
- определение доступности функций (действий, предусмотренных процессом обработки) для пользователей (функциональных ролей пользователей, определенных как значение атрибута);

- определение доступности для пользователей (функциональных ролей пользователей, определенных как значение атрибута) атрибутов карточек, находящихся в определенном состоянии;
- журналирование действий пользователя - фиксирование и запись в журнал фиксируемых действий, производимых пользователями системы;
- получение сервисной информации - статистика работы с системой;
- сбор статистики по материалам – суммируемая информация по действиям пользователя в разрезе шаблона, материала, типу действия.

II.2.6. Модуль сканирования

Существует 2 типа сканирования:

- локальное, разделяется на 2 вида:
 - при помощи Java-апплета, библиотеки TWAIN (mmsc) и библиотеки JAI;
 - при помощи установленного в организации FineReader (v.6, ScriptEdition) и VBScript, встроенного в страницу.
- потоковое, когда сканирование происходит отдельным аппаратно-программным комплексом, результаты выкладываются в общую с порталом папку (имена файлов кодируются согласно штрихкоду на оригиналах и содержат номер РКК), портал при помощи периодической задачи вкладывает файлы в соответствующие РКК.

Апплет сканирования

Задача апплета - сканировать документы, и прикреплять их в качестве вложений к карточкам.

Сам апплет сканирования находится в проекте scanner-applet. Для работы необходим проект [mmsc](#), который представляет собой реализацию протокола TWAIN. Несмотря на то, что в проекте он находится в виде исходных файлов, он является сторонней opensource разработкой сообщества «[mmscompunting](#)». После сборки «scanner-applet.jar» он помещается в «DBMI-UserPortlets.war».

В проекте апплет сканирования помещен на «CardScan.jsp». Его объявление имеет следующий вид:

```
<object id="<portlet:namespace/>-applet"  
  classid="java:com/aplana/scanner/ScannerApplet.class"  
  type="application/x-java-applet" style="width: 900px; height: 650px;">  
  <param name="mayscript" value="true"/>  
  <param name="archive" value="<c:url value="/scanner-applet.jar"/>"/>  
  <param name="filename" value="<%= cardId.toString() %>"/>
```

```
<param name="targetUrl" value="<%= uploadUrl %>"/>
<param name="namespace" value="<portlet:namespace/>
"/>
```

Апплет запускается из архива «scanner-applet.jar», класс апплета «ScannerApplet.class». Два других основных параметра:

- «filename» - id карточки основания;
- «targetUrl» - обратный адрес, куда с помощью post-запроса будет отсылаться отсканированный документ ((адрес портала)/DBMI-UserPortlets/servlet/scanner-upload). На этом адресе установлен spring-контроллер, который производит последующий действия по прикреплению полученного вложения к карточке.

Сценарии работы:

1. Открыть карточку на редактирование.
2. В поле «Вложения» нажать кнопку «Сканировать». Открывается окно сканирования и загружается java-applet сканирования (при условии, что браузер поддерживает загрузку апплетов и в системе установлена java).
3. Изначально доступны только три кнопки и флажок:
 - *Новый документ* - создается новый документ;
 - *Сканировать* - открывается стандартный диалог сканирования и производится последующее сканирование;
 - *Выбрать* - выбирается устройство для сканирования;
 - *флажок «Показывать настройки»* - если не активен, то при нажатии кнопки «Сканировать» не отображается диалог настроек, а сразу происходит сканирование с настройками по умолчанию.
4. Нажать кнопку «Сканировать» - отображается диалог сканирования с выбором настроек. Установить нужные значения и произвести сканирование. После завершения процесса (можно производить сканирование несколько раз для создания многостраничного документа) в апплете появляется отсканированное изображение (или изображения в случае многостраничного документа), с которым возможны следующие действия:
 - *Сохранить* - документ сохраняется локально на диск. После сохранения документ удаляется в апплете и открывается новый документ;
 - *Отправить PDF (Tiff)* - документ отправляется в виде PDF(Tiff)-вложения в текущую карточку, после чего предлагается подписать документ, как и любое другое вложение. После чего открывается новый документ;
 - *Печать* - печать только выбранной страницы документа. Документ не закрывается;
 - *Поворот* - поворачивает страницу на 90 градусов по часовой стрелке;
 - *Удалить* - удаляет текущую страницу из документа.

5. Далее нажать кнопку «Новый документ» или кнопку «Назад», чтоб вернуться в карточку.

Механизмы работы

Устройство апплета:

- GUI апплета реализовано с помощью библиотеки «swing».
- Взаимодействие со сканером осуществляется через «com.aplana.scanner.ScannerController», который реализует twain-интерфейс ScannerListener. Этот интерфейс содержит метод «update», который срабатывает при возникновении события на сканере.
- В зависимости от ОС (Linux/Windows), используются разные классы, реализующие работу со сканером (унаследованы от абстрактного класса «Scanner»). Нужный класс определяется при загрузке апплета.
- Если при загрузке апплета в системе не найдены сканеры, то панель сканирования не отображается.
- После того, как сканер обработал документ, изображение берется из буфера сканера и добавляется в «com.aplana.scanner.ui.PageListModel pageListModel». Изображения в «pageListModel» хранятся в памяти с помощью «SoftReference» и одновременно в виде JPEG файла в TEMP-директории. Таким образом, осуществляется кэширование отсканированных изображений в рамках доступной памяти.
- Промежуточное (для хранения) преобразование отсканированной страницы в JPEG может исказить данные и из-за этого частично теряется смысл в дальнейшем сохранении в формате TIFF.
- «ScannerController» также реализует остальные функции апплета:
 - «printPage()» - печатать выбранную страницу;
 - «rotatePage()» - повернуть выбранную страницу;
 - «deletePage()» - удалить выбранную страницу;
 - «scan()» - сканировать;
 - «uploadPages(String)» - загрузить на портал. Входная строка «pdf» или «tiff»;
 - «savePages(File)» - сохранить.
- Все действия с отсканированными страницами выполняются с помощью «Task», унаследованных от «SwingWorker». Конкретные действия наследуются от «com.aplana.scanner.task.Task. (ImageTask, SaveTask, UploadTask)»:
 - «ImageTask» преобразует сканированную страницу в выходной PDF/Tiff;
 - «SaveTask» сохраняет документ локально на диске;
 - «UploadTask» загружает документ на портал через post-запрос. Процедура загрузки:

- все сканированные страницы, находящиеся в «pageListModel», с помощью iText конвертируются в tiff или pdf и сохраняются во временный файл;
- из этого файла создается объект типа «FilePart»;
- этот объект помещается в post-запрос на портал, где его обрабатывает «spring-controller com.aplana.dbmi.scanner.ScannerUploadController»;
- «ScannerUploadController» берет из запроса «Part», создает карточку шаблона «Файл», добавляет ее к документу основанию. В итоге возвращает на сервлет «id» созданной карточки (необходимо для процедуры подписи, на стадии тестирования);
- если «id» карточки получен, то пользователю выдается запрос на подпись вложения.

Внешнее потоковое сканирование

Внешнее сканирование осуществляется с помощью программно-аппаратного комплекса, установленного у заказчика (например, Fujitsu-Siemens fi-6670). СЭД к нему не привязывается. Результаты сканирования этого комплекса выкладываются в общую с порталом папку (имена файлов кодируются согласно штрихкоду на оригиналах и содержат номер РКК). В портале работает специальная задача «MaterialSyncModule», которая загружает файлы из этой папки, и на основании имени файла, прикрепляет их к соответствующим карточкам в портале.

Осуществление потокового сканирования

Для осуществления процесса потокового сканирования оператору необходимо:

- проверить наличие штрихкодов на документах;
- поместить пакет документов в потоковый сканер и запустить сканирование, нажав кнопку «Сканировать»;
- далее система сканирования произведет обработку пакета, присвоит ему название и поместит его в папку-хранилище (папка с окончанием «Inbox»), откуда его автоматически забирает СЭД:
 - название файлов формируется следующим образом: «cardID_ddmmyyhh24miss.pdf»;
 - если документ обрабатывается более 10 минут, система сканирования поместит его в папку для отложенной обработки (и сформирует задание для обработки этого документа в ночные часы).
- каждые 2 минуты СЭД забирает документы из папки-хранилища и перемещает их в подпапку «Bad»;
- далее СЭД производит поиск подходящей карточки для связи. Если карточка найдена, СЭД добавляет к ней файл и удаляет его из папки «Bad»;
- если карточка не нашлась, отсканированный файл остается в папке «Bad».

II.2.7. Электронная подпись

Принципы работы ЭП (без привязки к системе)

Работа ЭП основана на наиболее популярном алгоритме - асимметричной схемы ЭП. Схема подразумевает существование закрытого и открытого ключей. Закрытый ключ находится только у автора сообщения и никуда не пересылается. Открытый ключ генерируется вместе с закрытым, и может быть разослан кому угодно. С помощью закрытого ключа возможно только формирование ЭП, с помощью открытого - только проверка ЭП.

Автор сообщения, формирует с помощью закрытого ключа (находящегося только у него и никуда не пересылаемого) ЭП сообщения (шифрует сообщение с помощью закрытого ключа по одному из асимметричных алгоритмов). Отправляет исходный текст сообщения, ЭП сообщения (шифрованное сообщение), и открытый ключ (который находится в свободном доступе). Получатель, чтобы проверить подлинность сообщения, расшифровывает ЭП по открытому ключу, и сравнивает присланное и расшифрованное сообщения. Тем самым, подмена сообщения возможна только одновременным изменением открытого ключа и ЭП сообщения. Для обеспечения достоверности открытых ключей существуют сертификаты открытых ключей, которые выдаются удостоверяющими центрами. С помощью сертификатов, удостоверяющие центры (УЦ) гарантируют, что данный открытый ключ принадлежит именно тому человеку, который прислал сообщение. Чтобы не было подмены УЦ, сертификаты также имеют цифровую подпись и сертификат для ее проверки, выданный центральным УЦ.

ЭП в системе

В системе в качестве сообщений выступают карточки. Подписывать можно как вложения, так и набор полей карточки. В системе в качестве подписываемых данных выступают не поля карточки, а их HASH. При проверке подписи: подпись дешифруется и сравнивается с HASH-ем исходного текста. ЭП при этом хранится в отдельном атрибуте карточки, т.е. представляет собой «HtmlAttribute»:

```
Attribute_code="ADMIN_67129" Attr_name_rus="Подпись" Type="W"
```

Данный атрибут содержит следующие поля:

```
certHash - Хэш-сертификата  
signature - Сама электронная подпись  
pkcs7 - подпись в формате pkcs7.
```

Хэш-сертификата вычисляется по стандартному алгоритму, указанному в файле настроек «cryptoLayer.properties», например «HASH_ALGORITHM=GostR3411-94».

«PKCS#7» - стандарт, описывающий синтаксис криптографических сообщений.

Синтаксически анализируется атрибут «ADMIN_67129» - в «/UserPortlets/src/com/aplana/dbmi/crypto/SignatureData.java» метод «setSignatureData(String data, Card card)».

У персоны, в личной карточке персоны есть два атрибута, которые являются «CardLink» атрибутами, на карточке шаблона «Сертификат ЭП» (template_id=1228):

```
Attribute_code="ADMIN_6884958" Attr_name_rus="Сертификаты" Type="C"  
Attribute_code="ADMIN_6885058" Attr_name_rus="Текущий сертификат" Type="C",
```

В свою очередь, у карточки «Сертификат ЭП» есть BackLink атрибут на карточку персоны владельца сертификата:

```
Attribute_code="ADMIN_6884858" Attr_name_rus="Владелец сертификата" Type="B"
```

Карточка «Сертификат ЭП» также содержит:

```
Attribute_code="ADMIN_6884758" Attr_name_rus="Сертификат" Type="S"  
Attribute_code="ADMIN_6885158" Attr_name_rus="Хэш сертификата" Type="S"
```

Атрибут «Сертификат» (ADMIN_6884758) представляет собой сертификат открытого ключа в формате X.509, из которого обычными get-методами можно получить необходимую информацию (номер сертификата, владельца, дата начала действия, дата окончания действия, и т.д.).

Подпись карточек осуществляется в момент смены статусов. Для реализации данной задачи в таблице *workflow_move* предусмотрена *apply_ds numeric(1,0) NOT NULL DEFAULT 0*:

- 0 - не подписывать карточку;
- 1 - подписывать карточку;
- 2 - подписывать карточку и вложения (вложения берутся с помощью action «GetAttachments», который загружает вложения из документа основания).

Параметр из данной колонки записывается в «PortletSession» (в «CardPortlet») в виде атрибута «MI_APPLY_SIGNATURE» и потом на основе его значения в «CardButtonPane.jsp» принимается решение, присоединять ли подпись.

Два типа подписи

Атрибут «Подпись» содержит подпись в двух разных форматах:

- «просто подпись» (зашифрованный хэш-сообщения);
- подпись в формате «pkcs7».

«Просто подпись» может использоваться только внутри системы. Не может быть использована для взаимодействия с другими системами. Для этих целей необходима «подпись в формате pkcs7». Если, например, при отправке в другую систему, атрибут карточки «Подпись» содержит только «Просто подпись» (в атрибуте есть только «signature»), а «подпись в формате pkcs7» отсутствует (pkcs7=""), то фактически документ не подписан. В системе сначала проверяется «Подпись в формате pkcs7», а если она отсутствует, то «Просто подпись».

Ограничения на ЭП, при использовании различных СЭД

- формирование ЭП для карточки «Файл» должно производиться только по атрибуту «Материал» (вложение документа) и ни по каким иным атрибутам других карточек, т.к. документ может быть отправлен во внешнюю систему, в которой отсутствуют данные атрибуты, в связи с чем документ не пройдет проверку ЭП;
- экспорт по ГОСТ игнорирует все подписи, кроме pkcs7-подписей вложений (прямых).

Провайдеры ЭП

Для того чтобы иметь возможность работы с ЭП, необходимо иметь установленные в ОС необходимые провайдеры ЭП. По умолчанию в Windows и Java набирается около 8 предустановленных провайдеров, но они не используются в проекте СЭД «Логика СЭД. СПО». Поэтому необходимо установить их отдельно. На данный момент реализована работа с двумя провайдерами ЭП: КриптоПРО и ЛиссиКрипто. Данные криптопровайдеры реализуют отечественные стандарты:

- ГОСТ Р 34.10 94 «Информационная технология. Криптографическая защита информации. Система электронной цифровой подписи на базе асимметричного криптографического алгоритма»;
- ГОСТ Р 34.10 94 и ГОСТ Р 34.10-2001 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи»;
- Алгоритм зашифрования/расшифрования данных и вычисление имитовставки реализованы в соответствии с требованиями ГОСТ 28147 89 «Системы обработки информации. Защита криптографическая».

КриптоПро так же является удостоверяющим центром и имеет возможность выдавать сертификаты.

Пример файла настройки криптопровайдеров:
«/Portal/conf/dbmi/card/signature/cryptoLayer.properties.example».

Общая схема формирования ЭП

Создание ЭП производится в «CryptoApplet» (/CryptoModule/src/com/aplana/crypto/CryptoApplet.java). Главный метод - «getSignature(...)». Напрямую апплет на странице не отображается, а его вызов происходит через JS. JS-script размещен в файле «/UserPortlets/WebContent/js/crypto.js», и метод, вызывающийся для создания ЭП – «cryptoGetSignature(args)».

Для создания ЭП необходим закрытый ключ пользователя, который уже может быть установлен в реестре, или же может находиться у пользователя на специальном носителе ruToken (eToken), который представляет собой устройство в виде флеш-карты с пин-кодом для доступа и сертификатом пользователя с закрытым ключом. Для ее работы также необходимо установить драйверы, которые можно загрузить с сайта «http://rutoken.ru».

Таким образом, при формировании ЭП закрытый ключ пользователя никуда не отправляется.

Общая схема проверки ЭП

Для проверки ЭП в целях удобства использования и настройки была создана отдельная SOAP-служба. Ее реализация находится в проекте CryptoServer. Ее цель - проверка правильности цифровой подписи. При этом к ней имеют возможность подключаться несколько порталов. Сама служба запускается отдельно на сервере [TomCat](#). Также имеется возможность локальной (локально на портале) проверки ЭП, без подключения к внешней службе. Настройки производятся в файле «/Portal/conf/dbmi/card/signature/crypto.properties.example»:

```
verWebSer=true  
host=http://gossер:8080/CryptoServer/service/CryptoService?wsdl
```

Параметр «verWebSer» указывает, где проверять ЭП (true - на внешней службе, false - локально на портале) Параметр host - адрес сервиса. Веб-метод для проверки «boolean checkStringContentSignature(String document, String sign, String base64certificate)».

Сценарии работы

Подпись вложений:

- открыть карточку на редактирование;
- открыть форму добавления вложений;
- выбрать файлы;
- загрузить выбранные файлы;
- после загрузки файлов, произойдет подпись всех вложений. При этом:
 - если в карточке пользователя нет ни одного прикрепленного сертификата, то отобразится сообщение: «Невозможно подписать вложение в связи с отсутствием ключа ЭП. Обратитесь в отдел ЭП для получения нового сертификата»;
 - если не был обнаружен секретный ключ, то появляется ошибка получения данных. Дальнейший путь не отслежен, т.к. отсутствует секретный ключ.
- если файлы не были подписаны сразу, то последующая подпись невозможна;
- факт подписи вложения отображается в атрибуте «Вложения», столбец «Подпись».

Подпись карточек

Подпись карточек происходит автоматически. В любой момент времени подписать карточку невозможно. При изменении статуса карточек пользователю отображается запрос на прикрепление к карточке ЭП (флаг «apply_ds» в таблице «workflow_move»). Подписываются как карточки документов основания, так и карточки Согласований, Подписаний и т.д. Запрос на подпись и дальнейшая процедура аналогична процедуре подписания вложений.

Добавление/удаление сертификатов открытых ключей пользователей

- зайти под учетной записью администратора;
- перейти в карточку пользователя (например, через «Справочник»);
- произвести редактирование атрибута «Сертификаты»:
 - добавление: чтобы добавить сертификат, нужно нажать кнопку «Добавить Сертификат». После чего создается новая карточка «Сертификат». В блоке «Данные ЭП», нажав кнопку «Вставить из .seg файла», добавить сертификат. После чего сохранить карточку;
 - чтобы удалить сертификат, нужно перейти в карточку сертификата и перевести ее в статус «Дубликат».

Проверка ЭП

Подпись может проверяться в трех местах:

- атрибут «Подпись» карточки;
- колонка «Подпись» атрибута «Вложения» карточки;
- по кнопке «Проверить ЭП».

Настройка подписываемых атрибутов карточки

При формировании ЭП карточки, под подписание попадают не все атрибуты, содержащиеся в карточке. Необходимо самостоятельно указать, какие атрибуты карточки попадут под подписание (на основании шаблона). Эти настройки содержатся в файле «dbmi/card/signature/signAttributes.xml». Подписывать можно атрибуты не только текущей карточки, но и связанных с ней. Переход к ним осуществляется через тэг «<link id="...атрибут в текущей карточке..." type="B/C">». Там же указывается и тип атрибута (CardLink / BackLink). Допускается вложенность. Например, в шаблоне «Визирование» подписываются: вложения карточки основания и вложения текущей карточки. Чтоб подписать вложение недостаточно просто указать атрибут - Вложение (DOCLINKS), который является кардлинком на карточку «Файл», а нужно указать атрибут «MATERIAL» в этой карточке.

Важно: термин «подписываются» предполагает, что ЭП для карточки (в примере: для карточки визирования) будет формироваться на основании указанных атрибутов (в примере: на основании вложения карточки основания, вложения карточки визирования, а также атрибутов «ADMIN_14746670» и «JBR_VISA_D_A_CONSENT»). При этом подпись самих вложений не произойдет:

```
<!-- Шаблон "Визирование" -->
<template id="348">
  <link id="JBR_VISA_PARENT_DOC" type="B">
  <link id="DOCLINKS" type="C">
```

```
<attribute type="M" id="MATERIAL"/>
</link>
</link>
<attribute type="T" id="ADMIN_14746670"/>
<attribute type="D" id="JBR_VISA_D_A_CONSENT"/>
<link id="DOCLINKS" type="C">
  <attribute type="M" id="MATERIAL"/>
</link>
</template>
```

Проверка сертификатов

Перед наложением ЭП и при проверке ЭП происходит проверка статуса сертификата: отозван сертификат или нет. Происходить это может с помощью двух механизмов:

- проверка с помощью списка отозванных сертификатов CRL;
- онлайн проверка с помощью OCSP.

III. Взаимодействие модулей системы

Функционирование СЭД «Логика СЭД. СПО» обеспечивается как минимум двумя сервисами: сервис приложения (СП) и сервис хранения и предоставления данных (СХПД) (см. Рисунок 6). Взаимодействие СП и СХПД осуществляется через локальную вычислительную сеть (ЛВС).

СП состоит из одного или группы серверов приложений JBoss, которые обеспечивают выполнение приложения и дополнительных модулей.

Составными частями СП являются:

- виртуальная машина Java (JVM);
- сервер приложений JBoss (JBossAS);
- основная часть приложения (ОЧП);
- модуль конвертирования документов (МКД);
- модуль управления неструктурированными данными (МУНД);
- модуль почтовых рассылок (МПР);
- модуль уведомлений (МУ);
- модуль взаимодействия в формате МЭДО (МВ МЭДО);
- модуль взаимодействия в формате ГОСТ Р 53898-2010 (МВ ГОСТ);
- модуль взаимодействия в формате СЭД «Дело» (МВ Дело);
- модуль репликации (МР);
- модуль ЭП (МЭП);
- модуль крипто-провайдера (МКП);
- модуль проверки ЭП (МПЭП);
- модуль поддержки сканирования документов (МСД);
- модуль периодических задач (МПЗ);
- модуль мобильных клиентов (ММК);
- модуль генерации отчетов (МГО);
- модуль архивирования ЭД (МАЭД);
- транспортный агент (ТА).

В зависимости от дополнительно решаемых задач (отказоустойчивость, повышение производительности, обмен данными, мониторинг, комбинированные задачи и т. п.) СП может быть оборудован вспомогательными сервисами:

- балансировщик нагрузки (БН);
- сервис мониторинга состояния сервера (СМ);
- транспортные агенты (ТА);
- сервис мобильных клиентов (СМК).

Также для этих целей модули КД и МПЭП могут быть вынесены на отдельные серверы.

Для функционирования JBoss необходима виртуальная машина Java (JVM) и СУБД, которая может быть совмещена с СУБД Системы.

Основная часть Системы исполняется средствами JBossAS в виртуальном пространстве JVM.

Основная часть приложения Системы обеспечивает:

- идентификацию, аутентификацию и авторизацию пользователей приложения;
- управление правами пользователей на действия с ЭД;
- ввод-вывод сущностей приложения;
- версионирование ЭД;
- базовые функциональные блоки, используемые другими модулями;
- исполнение базовых бизнес-процессов;
- журналирование системных операций и событий;
- генерацию необходимых представлений графического пользовательского интерфейса.

Виртуальная машина Java – основная часть исполняющей системы Java. Виртуальная машина Java интерпретирует байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java. JVM обеспечивает выполнение программ, написанных на языке Java, а также других языках программирования.

JBossAS – Java EE сервер приложений с открытым исходным кодом, разработанный компанией JBoss Group.

Модуль конвертирования документов обеспечивает конвертирование файлов электронных вложений в форматы PDF при помощи внешних сервисов.

Модуль управления неструктурированными данными обеспечивает загрузку, версионирование, индексирование, поиск и выдачу файлов электронных вложений.

Модуль почтовых рассылок обеспечивает уведомление пользователей приложения о событиях, наступающих при прохождении ЭД определенных этапов бизнес-процессов посредством рассылки электронных почтовых сообщений.

Модуль уведомлений так же, как и МПР обеспечивает уведомление пользователей, но посредством визуальных всплывающих сообщений в графическом интерфейсе пользователя.

Модуль взаимодействия в формате МЭДО обеспечивает обмен ЭД с другими участниками МЭДО.

Модуль взаимодействия в формате ГОСТ 53898-2010 обеспечивает обмен ЭД с другими СЭД «Логика СЭД. СПО» и другими СЭД, поддерживающими обмен в формате ГОСТ 53898-2010.

Модуль взаимодействия в формате СЭД «Дело» обеспечивает обмен ЭД с другими СЭД «Логика СЭД. СПО» и другими СЭД, поддерживающими обмен в формате ГОСТ СЭД «Дело» производства компании «Электронные Офисные Системы».

Модуль репликации обеспечивает обмен ЭД с другими СЭД «Логика СЭД. СПО» методом полного отображения данных.

Модуль ЭП обеспечивает работу пользователей с ЭП согласно определенным бизнес-процессам во взаимодействии с модулями МКП и МПЭП.

Модуль крипто-провайдера обеспечивает алгоритмическое обеспечение формирования и проверки ЭП.

Модуль проверки ЭП обеспечивает формирование ЭП взаимодействуя с пользователем и специализированным аппаратно-программным комплексом (Рутокен).

Модуль поддержки сканирования документов обеспечивает считывание, преобразование и ввод в СЭД отсканированных документов взаимодействуя при этом со сканерами, установленными на рабочих местах (через драйверы TWAIN и SANE) и сканерами рабочих групп (через файловую систему).

Модуль периодических задач обеспечивает запуск и исполнение одноразовых и периодических сервисных задач, связанных с обеспечением бизнес-процессов и сервисным обслуживанием СЭД.

Модуль мобильных клиентов обеспечивает доступ к определенному функционалу СЭД с мобильных устройств.

Модуль генерации отчетов обеспечивает генерацию определенных параметризованных отчетов по запросам пользователей.

Модуль архивирования обеспечивает исполнение бизнес-процессов требующих архивирования ЭД и изъятия их из документооборота.

Транспортный агент обеспечивает прием и пересылку электронных сообщений, генерируемых модулями МВ ГОСТ, МВ Дело и МР по маршрутам, определенным в таблице маршрутизации.

Сервис хранения и предоставления данных

СХПД обеспечивает хранение, поиск и фильтрацию структурированных и неструктурированных данных. СХПД состоит из:

- хранилища структурированных данных (ХСД), которое основывается на одном или более СУБД PostgreSQL и обслуживаемого им (-и) экземпляра (-ов) БД;

- хранилища неструктурированных данных (ХНД), которое основано на файловой системе.

В зависимости от дополнительно решаемых задач (отказоустойчивость, повышение производительности, увеличение емкости, мониторинг, комбинированные задачи и т.п.) СХПД может быть оборудован вспомогательными сервисами:

- балансировщик нагрузки (к примеру, pgPool);
- агент мониторинга состояния сервиса.

Составные части СХПД могут быть расположены как на одном физическом сервере, так и на нескольких.

ХСД в свою очередь состоит из:

- СУБД PostgreSQL;
- БД СЭД версии соответствующей ОЧ Системы;
- БД JBoss.

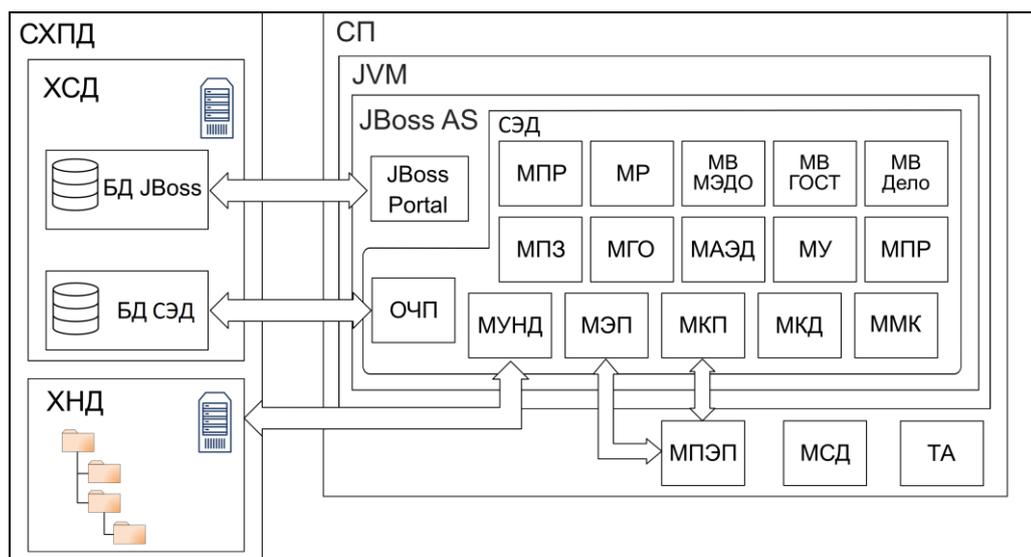


Рисунок 6 – Схема модулей системы

IV. Описание транспортной подсистемы

IV.1. Межведомственный электронный документооборот

Настройка МЭДО

МЭДО настроено с одной общей папкой в директории «/opt/jboss/medo.source». Настройки размещены в файле «\${JBOSS_HOME}/server/default/conf/dbmi/medo/options.properties»:

```
InFolder=/opt/jboss/medo.source/in_1server
InFolderProcessed=/opt/jboss/medo.source/in_1server/in1/br4j
InFolderDiscarded=/opt/jboss/medo.source/in_1server/nonex1/br4j
outFolderExport=/opt/jboss/medo.source/out_1server
ticketsPath=/opt/jboss/medo.source/ticket_1server
```

В указанных значениях указываются входящие каталоги для документов, а также для тех документов, которые зарегистрированы и не зарегистрированы. Также указывается исходящий каталог. Кроме того, нужно обязательно (если идет привязка к определенному сетевому интерфейсу) биндить ноду на 0.0.0.0.

После настройки конфигурации МЭДО, необходимо запустить задачу. Для этого необходимо войти в Систему под учетной записью администратора, перейти в «Администрирование системы», на вкладку «Задачи». Выбрать в «Task name» MEDO, в пункте «Start every» установить время, через которое будет включаться задача, и установить время запуска. Время необходимо устанавливать по времени сервера, на котором установлена Система. Выставляется время портала +1..4 минуты. Для облегчения мониторинга, можно вывести лог в отдельный файл: произвести редактирование файла \${JBOSS_HOME}/server/default/conf/jboss-log4j.xml и добавить следующий код:

```
<appender name="MEDOlog" class="org.jboss.logging.appender.DailyRollingFileAppender">
  <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
  <param name="File" value="${jboss.server.log.dir}/medo.log"/>
  <param name="Append" value="false"/>

  <!-- Rollover at midnight each day -->
  <param name="DatePattern" value=".'yyyy-MM-dd"/>

  <layout class="org.apache.log4j.PatternLayout">
    <!-- The default pattern: Date Priority [Category] Message\n -->
    <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
  </layout>
</appender>
```

```

</layout>
</appender>
<category name="com.aplana.medo" additivity="true">
  <appender-ref ref="MEDOlog" />
</category>

```

После перезагрузки портала создается файл «medo.log» в директории «\${JBOSS_HOME}/server/default/log/».

Настройка MEDOgate

Доставкой карточек, отправленных по МЭДО, а также созданием тикетов, занимается задача «MEDOgate». Ее настройки хранятся в файле «\${JBOSS_HOME}/server/default/conf/dbmi/medo-gate/route.properties».

```

InboundFolders=/opt/jboss/medo.source/out_1server
OutboundFolder=/opt/jboss/medo.source
TicketFolder=/opt/jboss/medo.source/out_1server

```

Настройка организаций в Системе

Для того чтобы организации могли производить обмен по МЭДО, нужно установить принадлежность к МЭДО, а также указать для каждой конкретной организации UUID и E-mail. Для этого необходимо зайти в систему под учетной записью администратора в раздел системы «Справочники» - «Организации». Выбрать организацию, которой планируется отправлять карточки по МЭДО (или создать новую организацию).

Обязательно нужно указать:

1. «UUID организации».

UUID организации указан в файле «export_template.xml» портала организации, куда планируется отправлять документы:

```

<xsl:attribute name="xdms:uid"> <!-- Обязательный -->
  <xsl:value-of select="'1D04CA3E-DF1A-0CB4-C325-XXXXXXXXXX002'" /> //сам uid
</xsl:attribute>
<xdms:organization> <!-- Обязательный -->
  <xsl:value-of select="'Организация 2'" /> //имя организации
</xdms:organization>

```

2. «Клиент МЭДО» выбрать значение «Да».

3. «Email». Необходимо указать название каталога входящих документов второго сервера, например, «in_2server».
4. «Наименование (справочное)».

На втором сервере (если необходимо), производятся аналогичные настройки, UUID организации необходимо указать первого сервера, указать входящий e-mail первого сервера и т.п.

Отправка и принятие документа по МЭДО

После регистрации исходящих документов отправка документа адресату (по МЭДО) осуществляется автоматически. Информация об отправке (статус оправленной карточки) отображается в документе на вкладке «Рассылка»: название организации, которой отправляется документ, дата отправки, способ отправки, статус обработки у адресата, и сам статус.

Входящие документы по МЭДО находятся в разделе «Входящие» - «На предварительной обработке МЭДО». Документы находятся в статусе «Поступивший из ВС». Далее необходимо обработать документ и принять решение о регистрации. В результате статус документа изменится на «Зарегистрирован» или «Отказано в регистрации» в случае регистрации или отказа в регистрации документа соответственно. При этом отправителю отправится уведомление о регистрации или об отказе в регистрации документа.

IV.2.ГОСТ

Настройка модуля ГОСТ

1. Настроить директории, в которые модуль ГОСТ будет выгружать и откуда скачивать пакеты. Для этого необходимо перейти в конфигурационном каталоге «JBoss» в файл «\$JBoss_HOME/server/default/conf/dbmi/gost/config.properties». Если данного файла еще не существует, то его необходимо создать путем переименования файла «config.example.properties». Далее нужно настроить следующие параметры:
 - «inFolder» указывается директория, откуда модуль ГОСТ забирает пакеты для загрузки. В дальнейшем будет ссылка на данный путь с помощью алиаса «\$GOST_IN_FOLDER»;
 - «inFolderProcessed» указывается директория, в которую модуль ГОСТ будет переносить успешно загруженные пакеты;
 - «inFolderDiscarded» указывается директория, в которую модуль ГОСТ будет переносить неразобранные пакеты. При любом исключении, возникающем в процессе загрузки, пакет отклоняется и помещается в данную директорию. Примечание: если по какой-либо причине файловой системы не удалось переместить данный пакет, то он останется в «\$GOST_IN_FOLER», но будет произведена попытка создания для этого пакета лок-файла с содержимым «Locked by distribution manager [%date%] by class com.aplana.distrmanager.LoadDoc». При этом в логе системы об этом событии будет свидетельствовать запись вида «Unable to move directory [processingDirectory] to [destFolder]. Trying to lock it.»;

- «outFolder» указывается директория, куда модуль ГОСТ будет выгружать пакеты для отправки. В дальнейшем будет ссылка на данный путь с помощью алиаса «\$GOST_OUT_FOLDER».
2. Настроить файл «\$JBOSS_HOME/server/default/conf/dbmi/dmsi/config.properties». В случае если данный файл не существует, необходимо его создать путем переименования файла «config.example.properties»:
 - «Standart». Вид стандарта, по которому создано данное сообщение (используется значение «Стандарт системы управления документами»);
 - «Version». Версия стандарта (используется значение 1.0);
 - «sys_id». Уникальный служебный идентификационный номер системы;
 - «System». Наименование системы управления документами;
 - «System_details». Дополнительные данные о системе управления документами. Все указанные выше значения используются при создании паспорта сообщения ГОСТ Р 53898-2010;
 - «Default_organization_id». Обязательное для заполнения значение. Для него нужно указать card_id карточки Организации, которая считается для данной СЭД организацией по умолчанию. Это означает, что если при выгрузке используется пользователь, не привязанный какой-либо конкретной организации, то считается, что он относится к данной организации.
 3. Запустить задачник «ГОСТ» через интерфейс администратора: Администрирование системы/Задачи.
 4. Настроить организации, участвующие во взаимодействии:
 - *Идентификационный номер организации* = %GOST_UUID%, где %GOST_UUID% - уникальный на уровне всех систем идентификатор данной организации;
 - *Внешняя организация* = Да;
 - *Способ отправки по умолчанию* = ГОСТ.

Выгрузка документа

1. Инициация выгрузки производится с помощью перевода карточки «Элемент списка рассылки» в статус «Готов к отправке». Это может производиться вручную или автоматически при регистрации документа (Исходящий, ОРД) Получателю исходящего. При успешной выгрузке пакета статус карточки изменится на «Отправлен», иначе «Не отправлен». При этом в случае успешного завершения произойдут следующие события (в случае неуспешного завершения часть из них может отсутствовать в зависимости от этапа, на котором произошла ошибка):

- в атрибут «Сообщение ГОСТ» привязывается карточка шаблона «Сообщение ГОСТ», которая содержит слепок документа на момент отправки: XML с описанием документа (паспорт сообщения), копии Файлов вложений. Таким образом, если этап успешно завершился, то дальнейшие попытки отправить документ согласно данному ЭСР не будут приводить к генерации нового сообщения;
 - на каждую попытку отправки (перевод карточки ЭСР в «Готов к отправке») будет производиться создание новой карточки «Статус отправки в МЭДО» (1224) с привязкой в атрибут «Информация об отправке».
2. Далее в пересылке выгруженного пакета участвует Транспортный агент.
 3. От принимающей стороны приходят уведомления и загружаются в атрибут «Информация о доставке»:
 - Об успешной загрузке - Статус карточки ЭСР переводится в «Доставлено»;
 - О некорректном пакете - Статус карточки ЭСР переводится в «Не отправлено»;
 - О регистрации - Статус карточки ЭСР переводится в «Зарегистрировано»;
 - Об отказе в регистрации - Статус карточки ЭСР переводится в «Отклонено».

Загрузка документа

1. Загрузка документа начинается с создания карточки «Доставка». Данная карточка представляет собой описание конверта сообщения.
2. Если пакет с таким идентификатором уже загружался, то он отклоняется. Таким образом, если Сообщение ГОСТ с данным идентификатором уже загружался, то повторно он не будет загружаться. Если есть такая необходимость, то нужно создавать новую ЭСР у отправителя.
3. К «Доставке» с помощью атрибута «Сообщение ГОСТ» привязывается карточка «Сообщение ГОСТ», содержащая в себе в качестве Вложений карточки файлы, содержащие пришедший пакет (файл паспорта и файлы вложений).
4. В дальнейшем при работе с документом уведомления привязываются к карточке «Доставка» с помощью Элементов списка рассылки. Таким образом, из документа можно найти информацию об уведомлениях перейдя в карточку «Доставка» с помощью атрибута «Исходный ресурс» и далее на вкладку со Списком рассылки. Здесь находятся ЭСР для уведомлений (само уведомление можно найти, перейдя по следующему пути: «Исходный ресурс» - «Лист рассылки» - «Сообщение ГОСТ» - «Уведомление ГОСТ»).

UUID

Ниже приведен список UUID, используемых в процессе обмена:

Отправитель

- «Элемент списка рассылки», атрибут «Идентификатор сообщения» (JBR_DIST_UUID) содержит в себе значение паспорта сообщения. Обозначение - %UUID_PASS%;
- Исходящий «Сообщение ГОСТ», атрибут «UUID карточки» (JBR_UUID) также содержит в себе значение паспорта сообщения %UUID_PASS%;
- «Статус отправки в МЭДО», атрибут «UUID карточки» (JBR_UUID) содержит в себе значение конверта сообщения %UUID_PACKET%.
- Входящее «Уведомление ГОСТ», атрибут «UUID карточки» (JBR_UUID) содержит в себе значение паспорта сообщения %UUID_PASS%.

Получатель

- Доставка:
 - атрибут «UUID пакета» JBR_DISTR_UUID содержит в себе значение конверта сообщения %UUID_PACKET%;
 - атрибут «UUID исходного сообщения» JBR_DISTR_SRC_UUID содержит в себе значение паспорта сообщения %UUID_PASS%.
- Входящий «Сообщение ГОСТ», атрибут «UUID карточки» (JBR_UUID) содержит в себе значение паспорта сообщения %UUID_PASS%.
- Исходящее «Уведомление ГОСТ», атрибут «UUID карточки» (JBR_UUID) содержит в себе значение паспорта сообщения %UUID_PASS%.

Схема работы

На рисунках представлены схемы работы UUID (Рисунок 7, Рисунок 8, Рисунок 9, Рисунок 10).

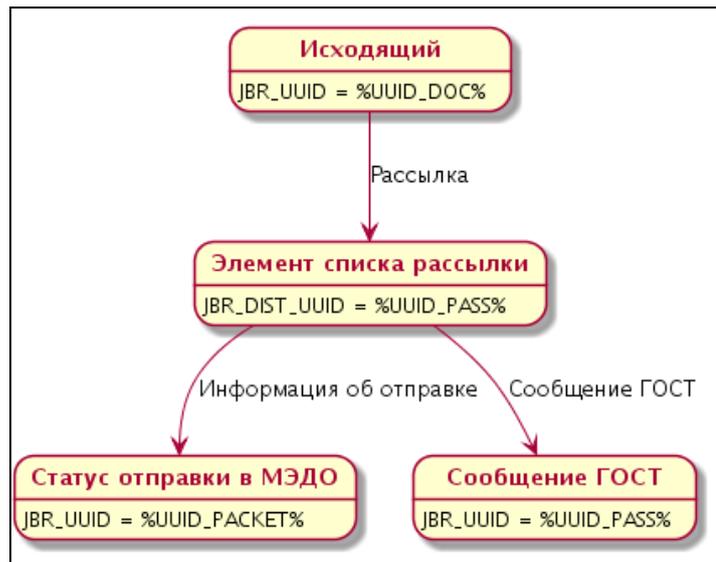


Рисунок 7 – Выгрузка документа

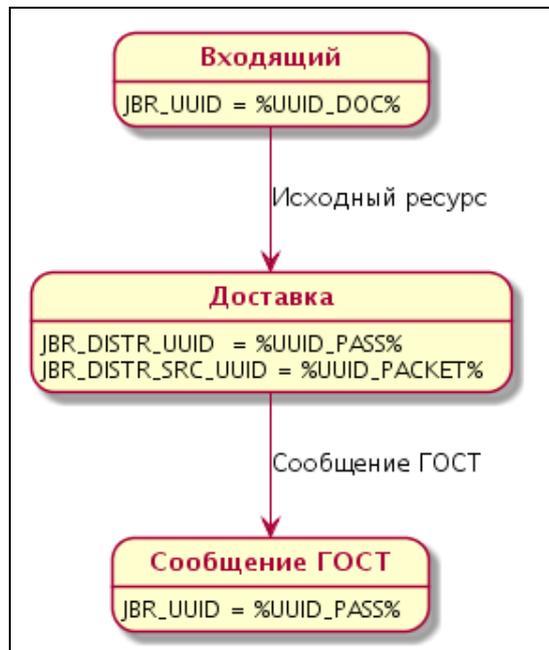


Рисунок 8 – Загрузка документа

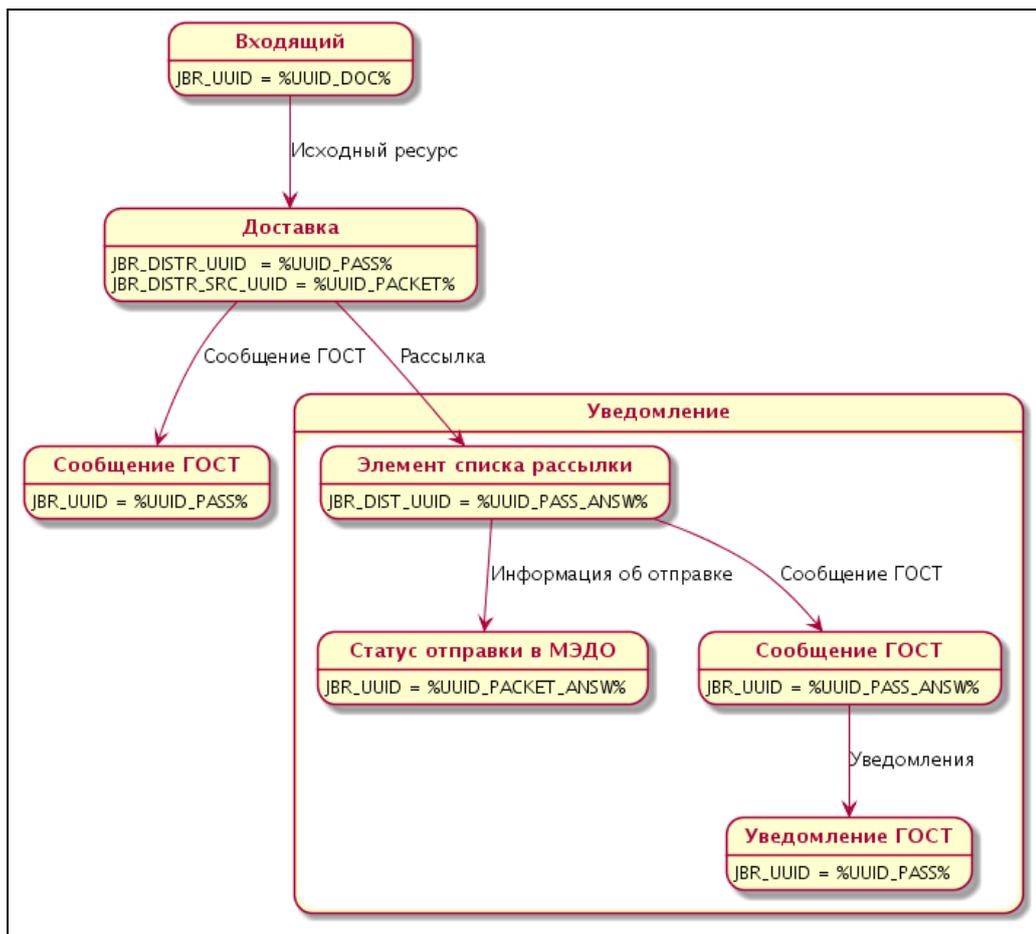


Рисунок 9 – Выгрузка уведомления

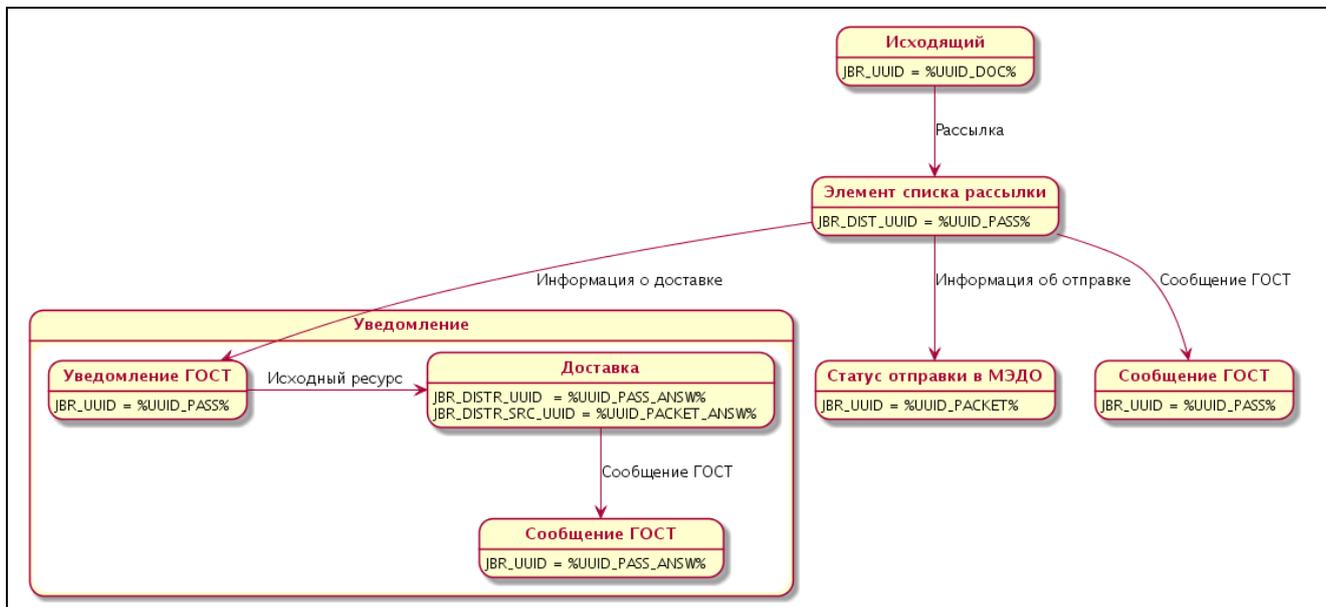


Рисунок 10 – Загрузка уведомления

Блокировки директорий

В процессе работы модулей для синхронизации используется лок-файл «folder.lock». Таким образом, если директория «заблокирована» с помощью данного файла, то никакой другой процесс не имеет право ее изменять или каким-либо образом с ней взаимодействовать.

Часто используемые случаи, где происходит блокировка:

1. В случае если модуль ГОСТ по результатам работы не смог переместить обрабатываемую директорию-пакет в каталог обработанных пакетов (как в случае успешной в «processed», так и в случае неуспешной в «discarded»), то файл блокировки будет содержать соответствующую отметку.
2. В процессе выгрузки пакета в файловую систему директория блокируется до момента полной выгрузки всех необходимых файлов, однако, в случае возникновения исключения в штатной ситуации, данная директория удаляется. Таким образом, файл блокировки для этого случая возникает, например, в случае отключения сервера во время процесса сохранения.
3. «ExportQueue» в процессе обработки пакета блокирует с сообщением:

locked by ExportQueue (Thu Jun 19 09:52:16 MSK 2014)

4. В процессе перемещения пакета в «InternalNode» с сообщением либо в «trash», если возникло исключение:

locked by FileReceiver write message (Thu Jun 19 09:52:16 MSK 2014)

5. В процессе считывания директории с пакетом с помощью «FileSystemReceiver» исходная директория помечается:

locked by FileReceiver (Thu Jun 19 09:52:16 MSK 2014)

6. В процессе отправки исходящего сообщения исходная директория помечается:

locked by Plugin Executor (Thu Jun 19 09:52:16 MSK 2014)

7. В процессе отправки исходящего сообщения целевая директория помечается:

locked by FileSender (Thu Jun 19 09:52:16 MSK 2014)

Также транспортным агентом используются дополнительные два вида блокировок. Для «folder.queued» это:

1. Обработанная кэширующим механизмом «ExportQueue», помечается:

placed in queue (Thu Jun 19 09:52:16 MSK 2014)

2. Директории, добавленные в механизм переправки «SendRetryStorage» помечаются:

placed in queue (Thu Jun 19 09:52:16 MSK 2014)

Для «folder.delme», если при работе транспортного агента по какой-либо причине не удалось удалить директорию, то она помечается:

Delete this folder (Thu Jun 19 09:52:16 MSK 2014)

Директории с необработанными сообщениями

- ГОСТ: «discarded». Директория, в которую модуль ГОСТ переносит неразобранные пакеты. При любом исключении, возникающем в процессе загрузки, пакет отклоняется и помещается в данную директорию. Если по какой-либо причине файловой системе не удалось переместить данный пакет, то он останется во Входящей папке, но будет произведена попытка создания для этого пакета лок-файла с содержимым «Locked by distribution manager [%date%] by class com.aplana.distrmanager.LoadDoc». При этом в логе системы об этом событии будет свидетельствовать запись вида «Unable to move directory [processingDirectory] to [destFolder]. Trying to lock it.».
- ТА: «trash». В данную папку попадают невалидные пакеты:
 - не указан uuid отправителя;
 - не указан uuid получателя;
 - описанные файлы вложений фактически не существуют;
 - ошибка при разборе XML паспорта или если файл паспорта не существует, не является файлом или по какой-либо иной причине не может быть открыт на чтение
 - при загрузке в случае ошибки разбора XML паспорта.
- ТА: «status fail». Директория, в которую переносятся пакеты, если ошибка произошла непосредственно при отправке (например, ошибка ввода-вывода). В этом случае дополнительно, если исключение помечено как «переотправляемое», данный пакет будет дополнительно добавлен в модуль переотправки. Для случая «FileSender» это производится, когда не удалось создать директорию назначения.

IV.3.Транспортный агент

Транспортный агент - отдельное WEB-приложение, не зависящее от исходных кодов системы. Поставляется в виде WAR-файла. Запускается в момент инициализации контекста (см. «TransportAgentContextListener#contextInitialized(ServletContextEvent arg)»).

Используемые термины:

- *Транспортный агент (ТА)* – программный модуль, обеспечивающий перемещение сообщений, а также генерацию и перемещение уведомлений о доставке между транспортными узлами в соответствии со своими способностями и своей таблицей маршрутизации;

- *Сообщение* – группа файлов, определенного формата, содержащая в себе кроме данных для получателя, информацию об отправителе и получателе сообщения, а также о необходимости уведомлений;
- *Уведомление* – особый вид сообщения, несущего информацию о событии, произошедшем в системе адресата, к примеру: получение сообщения, регистрация полученного документа и т.д.;
- *Уведомление о доставке (тикет)* – особый вид сообщения, несущего информацию о событии, произошедшем у транспортного агента, к примеру: доставка или недоставка сообщения;
- *Перемещение сообщения* – операция по изменению местоположения сообщения между транспортными узлами посредством использования различных транспортных механизмов: операции в файловой системе, электронная почта, Web-сервисы, а также более низкоуровневые протоколы передачи данных (HTTP(S), UDP);
- *Таблица маршрутизации (ТМ)* – файл, содержащий таблицу однозначных соответствий наименований (идентификаторов) адресатов и местоположений транспортных узлов (и транспортных узлов тикетов);
- *Транспортный узел (ТУ)* – место временного хранения сообщений перед и/или после перемещения их транспортными агентами.

Общие положения

Транспортные агенты могут быть выполнены в виде программных модулей как в составе приложения СЭД, так и автономно. Каждый ТА обладает такими идентифицирующими его свойствами:

- наименование ТА;
- UUID ТА.

Транспортные агенты предназначены для транспортировки сообщений между участниками электронного документооборота посредством перемещения сообщений от одного ТУ к другому. Транспортировка осуществляется при помощи различных транспортных средств, механизмов и протоколов. Каждый ТА обладает определенным набором «умений» транспортировать сообщения.

Каждый ТА может обслуживать несколько ТУ.

При удачном или неудачном перемещении сообщения из одного ТУ в другой ТА может генерировать уведомление о доставке (тикет) и перемещает его в ТУ тикетов. При перемещении тикета или уведомления соответствующий тикет не генерируется. Таким образом, реализуется функция прозрачной ретрансляции уведомлений и тикетов.

В каждый сгенерированный тикет в качестве отправителя вносятся данные, однозначно идентифицирующие данный ТА. Далее тикет может попасть непосредственно в СЭД участника или быть ретранслирован другими ТА далее по маршруту.

Доставка сообщений осуществляется на основе таблицы маршрутизации (ТМ), которой оборудован каждый ТА. Маршрутизация осуществляется на основе UUID адресатов. Кроме указания местоположения ТУ для каждого конкретного участника документооборота, существует указание ТУ по умолчанию, куда должны попадать сообщения, получатели которых не нашли соответствия в этой ТМ. Для каждого участника документооборота возможно указание местоположения ТУ тикетов.

ТА непосредственно не взаимодействуют друг с другом. ТА можно соединять в группы различной топологии (звезда, кольцо, ячеистая топология и т.д.). Благодаря возможности ретрансляции сообщения могут проходить сети ТА различных топологий.

Текущая реализация

Транспортируемые сообщения – только в формате ГОСТ 53898-2010.

Реализованные транспортные механизмы: файловые операции, операции с сообщениями электронной почты (SMTP и POP).

При использовании транспортных механизмов электронной почты все части сообщения помещаются как вложенные файлы внутрь сообщения электронной почты.

Тикеты формируются на основе информации в пересылаемых сообщениях, а именно: наименования и UUID отправителя и получателя, флага о необходимости генерации уведомлений. Тикеты помещаются в специализированную папку в файловой системе, доступной ТА.

Таблицы маршрутизации должны быть расположены в файле «conf/routeorg.xml» относительно текущей папки.

Настройка Транспортного агента

Основным конфигурационным файлом, который необходимо настраивать при развертывании ТА является «\$JBOSS_HOME/server/default/conf/dbmi/routeTable.xml». Дополнительно необходимо убедиться, что переименованы файлы «beans.xml», содержащие плагины, участвующие в обмене, и соответственно файлы с настройками «smtp.properties», «soz.properties».

Основные моменты, на которые стоит обратить внимание в контексте текущей решаемой задачи:

1. Элемент «Folders/outbound/@path» необходимо настроить на директорию «\$GOST_OUT_FOLDER».
2. Элемент «Folders/inbound/@path» необходимо настроить по следующему правилу: «\$GOST_IN_FOLDER = \$TA_INBOUND/\$GOST_MAIN_FOLDER». Где «\$TA_INBOUND» -

- это настраиваемый путь, «\$GOST_MAIN_FOLDER» - значение «mainFolder», сконфигурированное в «internalNode».
3. Далее необходимо построить конфигурацию «Internal». Все «internalNode», относящиеся к организациям-получателям ГОСТ (имеющие соответствующий UUID), должны быть настроены на один и тот же каталог «\$GOST_MAIN_FOLDER». При этом здесь можно использовать при конфигурации следующее: если есть множество организаций данного узла, относящихся к ГОСТ и, например, только один, относящийся к Репликации, то можно настроить «internalNode» по умолчанию: имеющий значение «uuid="default"». В любом случае «default» необходимо настроить для возможности корректной обработки сообщений не сконфигурированным получателям.
 4. Для «External» также необходимо обязательно настроить как минимум узел по умолчанию для случая, когда получатель не распознан в соответствии с текущими настройками. Для этого и для всех остальных настраиваемых «externalNode» настроить «send/address» и «receive/address», при этом элементы «send», «receive» могут включать в себя несколько «address».

Таким образом, агент будет собирать сообщения из всех указанных «ExternalNode» и дальше перенаправлять их на соответствующий «InternalNode» в соответствии с «internalNode/@uuid» получателя. Агент рассортировывает пакеты из «\$OUTBOUND» в соответствии с получателем в «ExternalNode» в соответствии с «externalNode/@uuid».

Таблица маршрутизации

Таблица маршрутизации вместе с настройками по транспорту размещается в конфигурационной директории приложения «\${JBOSS_HOME}/server/default/conf/dbmi/transportAgent/».

Таблица маршрутизации сохранена в файле «\${JBOSS_HOME}/server/default/conf/dbmi/transportAgent/routeTable.xml». Представляет из себя полное описание взаимодействия узлов ТА (Таблица 3).

Таблица 3. Краткое описание таблицы маршрутизации

Наименование элемента	Описание
TransportAgent	Определение агента, его наименование и UUID
Folders	Местоположение корневых папок
Tasks	Определение периодических задач - получателей и отправителей (без привязки к конкретным узлам)
TransportNode	Описание транспортных узлов (ТУ). Узлы могут быть внешними и внутренними по отношению к текущему ТА
TransportNode/Internal	Описание внутренних ТУ. Каждый internalNode определяется UUID, наименованием и корневой папкой данного внутреннего узла

Наименование элемента	Описание
TransportNode/External	Описание внешних ТУ. Каждый externalNode определяется UUID и наименованием. Кроме того, здесь прописан элемент address, в котором определены способы (по task) и адрес доставки и/или получения сообщений

Основные принципы работа ТА

Алгоритм отправки:

1. Задача «ExportQueue» считывает заголовок сообщения, определяет UUID получателя в зависимости от типа сообщения.
2. Производится поиск «externalNode» в «routeTable.xml». Если нет соответствия, то выбирается значение «default».
3. Для найденного «externalNode» в дочернем «send/address» определяет адрес (элемент «address»).
4. Заносится информация в кэш.
5. Другая задача на базе «Sender» активируется в заданное интервалами время.
6. Производится опрос «ExportQueue» на наличие «своих» сообщений по имени «task».
7. Происходит отправка по заданному адресу и обработка результатов отправки.
8. Переданные сообщения переносятся в папки отправленных документов.

Повторная отправка

В случае если при отправке сообщения возникает ошибка, есть возможность повторно отправить сообщение. Для этого в «routeTable.xml» заведен специальный task:

```
<task name="SendRetryStorage" bean="sendRetryStorage" period="300"/>
```

Количество повторных отправок определяется свойством «numberOfAttempts» бина «sendRetryStorage».

Плагин самостоятельно определяет необходимость повторной отправки. В случае ошибки отправки, плагин инициирует «SendException». Чтобы сообщение было отправлено повторно, нужно установить флаг «shouldRetry=true».

Алгоритм получения:

1. Задача на базе «Receiver» плагина активируется в заданное интервалами время.
2. Происходит поиск всех адресов для получения сообщений из «externalNode» в дочерних «receive/address».
3. Полученные сообщения сортируются в зависимости от UUID получателя по внутренним ящикам.

IV.4.Репликация

Условие начала репликации

Данную информацию можно получить, используя значение элемента «StartCondition», при этом следует учитывать, что если задан корень Root, то сработавшее условие распространяется на все дерево.

На данный момент реализованы следующие условия:

- Появилась карточка Рассмотрения:
 - в одном из статусов «Обработка помощником», «Отправлен на исполнение руководителем», «Рассмотрение», «Отправлен руководителем в дело»;
 - идентификатор (UUID узла репликации (org)) Организации рассматривающего не совпадает с идентификатором Организации регистратора документа.
- Появилась карточка Отчет об исполнении:
 - в одном из статусов «Обработка помощником», «Отправлен», «Принят», «Рассмотрение»;
 - идентификатор (UUID узла репликации (org)) Организации исполнителя не совпадает с идентификатором Организации регистратора документа.

Условие завершения репликации

Данную информацию можно получить, используя значение элемента «StopCondition», при этом следует учитывать, что если задан корень Root, то сработавшее условие распространяется на все дерево.

В системе определить остановлена репликация или активна можно с помощью атрибута «Реплицировать карточку» (REPLIC_IN_PROCESS). Значение 1 соответствует активной репликации, 0 - остановленной.

На данный момент реализовано условие остановки Репликации для документов Входящий и ОГ при переводе их в статус «Готов к списанию в дело», «В дело».

Примечание. Если для дерева уже была остановлена Репликации, то возвращение документа, например, обратно в Исполнение не приведет автоматически к запуску Репликации.

Получатели репликации

Данную информацию можно получить, используя значение элемента «Addressee», при этом следует учитывать, что если задан корень Root, то сработавшее условие распространяется на все дерево:

- в случае если карточка независимая, то рассылка идет на все узлы согласно конфигурации «ReplicationNodeConfig»;
- иначе адресат определяется с помощью «Addressee».

Текущие сконфигурированные правила определения получателя:

- организации Рассматривающих (только те, где сконфигурирован идентификатор (UUID узла репликации (орг));
- организации Исполнителей (только те, где сконфигурирован идентификатор (UUID узла репликации (орг)).

Состав реплицируемой информации

Действуют два правила для определения состава реплицируемой информации:

1. Если шаблон данной карточки не описан в настройках «ReplicationTemplateConfig», то карточка не реплицируется.
2. Если атрибут помечен как «Exclude» для некоего шаблона, то значения данного атрибута не попадают в пакет Репликации при реплицировании карточки данного шаблона.

Ключевые слова

Для поиска целевой карточки кроме поиска по UUID существует также дополнительный механизм, который в случае неудачи поиска по идентификатору пытается найти карточку с помощью сконфигурированных Keywords. Поиск производится по всем атрибутам:

- строковым, если значение непустое;
- числовым, если присутствует значение в пакете;
- дате, если присутствует значение в пакете;
- персоне, если присутствует значение в пакете.

Если карточка найдена, то происходит обновление UUID для нее на основе пришедшего пакета.

Общее описание работы

Основные положения

- обмен производится с помощью пакетов XML в виде файлов. Пакет представляет собой директорию, содержащую XML с описанием карточки и файлы материалов;
- выгрузка производится по событию синхронно: при сохранении или изменении статуса карточки производится выгрузка пакета с данными;
- загрузка производится по расписанию с помощью задачника «ReplicationTask»;

- выгружается слепок карточки, а не изменения, таким образом: никаких анализов изменений не производится, каждый раз выгружаются все значения (за исключением отфильтрованных настройками);
- обновление осуществляется напрямую с помощью SQL-запросов;
- для каждого адресата для каждой реплицируемой карточки создается карточка шаблона Репликация. Дальнейшая работа происходит с данной карточкой с помощью жизненного цикла;
- сверх реплицированной карточки существует дополнительно локальная копия.
- GUID карточки является уникальным идентификатором рабочей карточки и служит для связи с ней (GUID карточки = UUID репликации карточки). Примечание: для одной рабочей карточки может быть несколько карточек «Репликации» с одним и тем же «GUID карточки», например, в случае нескольких адресатов;
- Адресат (GUID узла) является уникальным идентификатором получателя данного пакета Репликации;
- Отправитель (GUID узла) является уникальным идентификатором отправителя данного пакета Репликации. Таким образом, данные три атрибута являются уникальным ключом для поиска единственной карточки Репликации;
- GUID итогового объекта является аналогом GUID карточки, но используется для связи локальной копии реплицированной карточки (также называется итоговым объектом);
- Дата отправки заполняется при обновлении карточки «Репликация» текущей датой;
- Дата доставки - использование данного атрибута на текущий момент не обнаружено;
- XML уведомления о доставке при поступлении пакета со статусом «обновляется XML-содержимым полученного уведомления», если карточка находится в данный момент в статусе Отправлен;
- История представляет из себя кардлинк на карточки шаблона «История репликации». При поступлении нового пакета для него в первую очередь создается карточка данного шаблона, в который записывается в атрибут «XML представление» ее значение. После этого созданная карточка прикрепляется в атрибут «История». После этого начинается обработка данного пакета;
- ID карточки содержит в себе «card_id» рабочей карточки, соответствующей данной карточки «Репликация»;

- Реплицировать карточку - флаг, указывающий на то, что данная Репликация активна. В случае если данный атрибут содержит значение 0, то это означает, что репликация по данной карточке уже завершена.

Пример XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ReplicationPackage xmlns:ns2="http://aplana.com/dbmi/ReplicationPackage/">
  <PackageType>card</PackageType>
  <Addressee>3d1bf250-5034-11e3-8f96-0800200c9a66</Addressee>
  <Sender>164a79d7-2aaf-42e7-a963-58a1fa636d10</Sender>
  <DateSent>2014-07-10T18:43:35.880+04:00</DateSent>
  <Card guid="7146c453-8451-4e42-9fc9-10069890ee74" card_id="12458115" status="556656"
  templateId="1044">
    <Attribute code="JBR_VERIFY_DS">
      <ListValue>1450</ListValue>
    </Attribute>
    .....
    <Attribute code="JBR_TCON_TERM">
      <DateValue>2014-08-07T23:59:00.000+04:00</DateValue>
    </Attribute>
  </Card>
</ns2:ReplicationPackage>
```

Блок Общие характеристики

Следующие атрибуты относятся к Репликации и добавлены в блок «Общие характеристики», т.е. содержатся во всех шаблонах (Рисунок 11):

- UUID репликации карточки - уникальный идентификатор данной карточки. Служит для связи с карточкой Репликации и в целом для идентификации данной карточки в рамках нескольких систем (на узле, куда реплицируется данная карточка, есть своя карточка с таким же идентификатором, являющаяся копией данной);
- Владелец репликации карточки - идентификатор узла, инициировавшего репликацию;

- Версия репликации - атрибут заполняется при выгрузке пакета;
- Флаг версии репликации - атрибут является сигнализатором того, что идет загрузка и данную карточку не нужно выгружать (он выставляется только при загрузке карточек типа «Файл»).

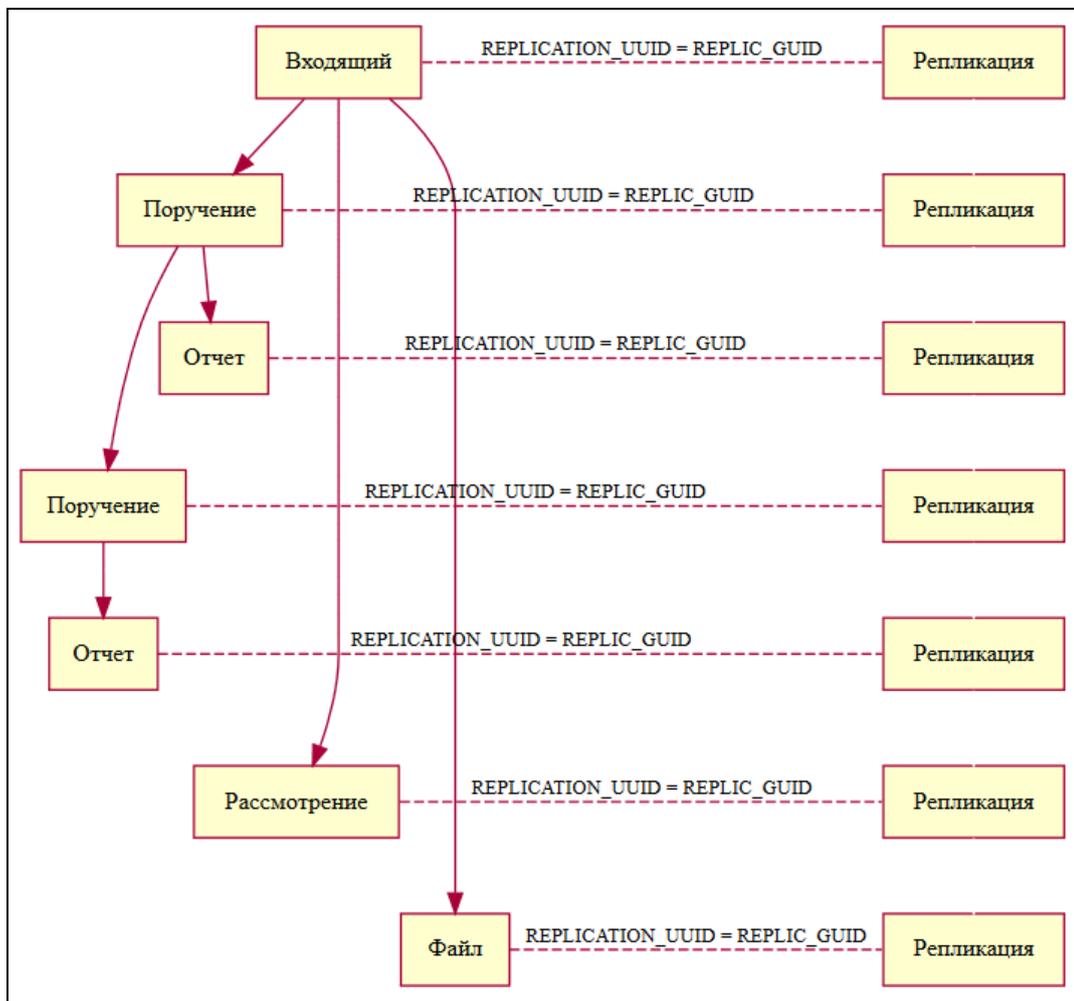


Рисунок 11 – Взаимосвязи карточек

Отправка данных

1. При первой выгрузке создается карточка в статусе «Черновик» и после выгрузки переводится в статус «Отправлен» (Рисунок 12).
2. В дальнейшем используется существующая карточка и после выгрузки также переводится в статус «Отправлен» (из статуса «Принято», если для данной карточки уже приходило уведомление об обновлении карточки на принимающей стороне).



Рисунок 12 – Жизненный цикл карточек шаблона «Репликация» для выгрузки

Загрузка данных

1. Новая карточка создается в статусе «Черновик» (Рисунок 13). При переходе в статус «Необработанный» (из любого статуса) происходит старт обработки пакета. При этом используется информация из самой карточки, т.к. все поступающие XML-данные предварительно загружаются в нее в виде кардлинки в атрибут «История». На данном этапе проверяются связи, если для какого-либо GUID не найдена соответствующая карточка, то карточка переводится в статус «Необходим дозапрос», иначе в «Принято».
2. При переводе в статус «Принято» запускается загрузка целевой карточки. Сначала выполняется проверка, не заблокирована ли карточка, которую необходимо обновить:
 - если заблокировано, то «Репликация» переходит в статус «В очередь»;
 - если карточки еще не существует, то она создается в искусственном статусе «Незавершенная репликация»;
 - если дата изменения целевой карточки позднее даты поступившей, то «Репликация» переводится в статус «Коллизия»;
 - если ранее, то обновляется.
3. При переводе в статус «Необходим дозапрос» создается целевая карточка в статусе «Незавершенная репликация».
4. При переводе в статус «Коллизия» происходит разрешение коллизии:
 - если данный сервер является Владельцем репликации, то изменения не производятся, а отправляется пакет с текущим состоянием карточки;
 - если данный сервер не является Владельцем репликации, то карточка обновляется.

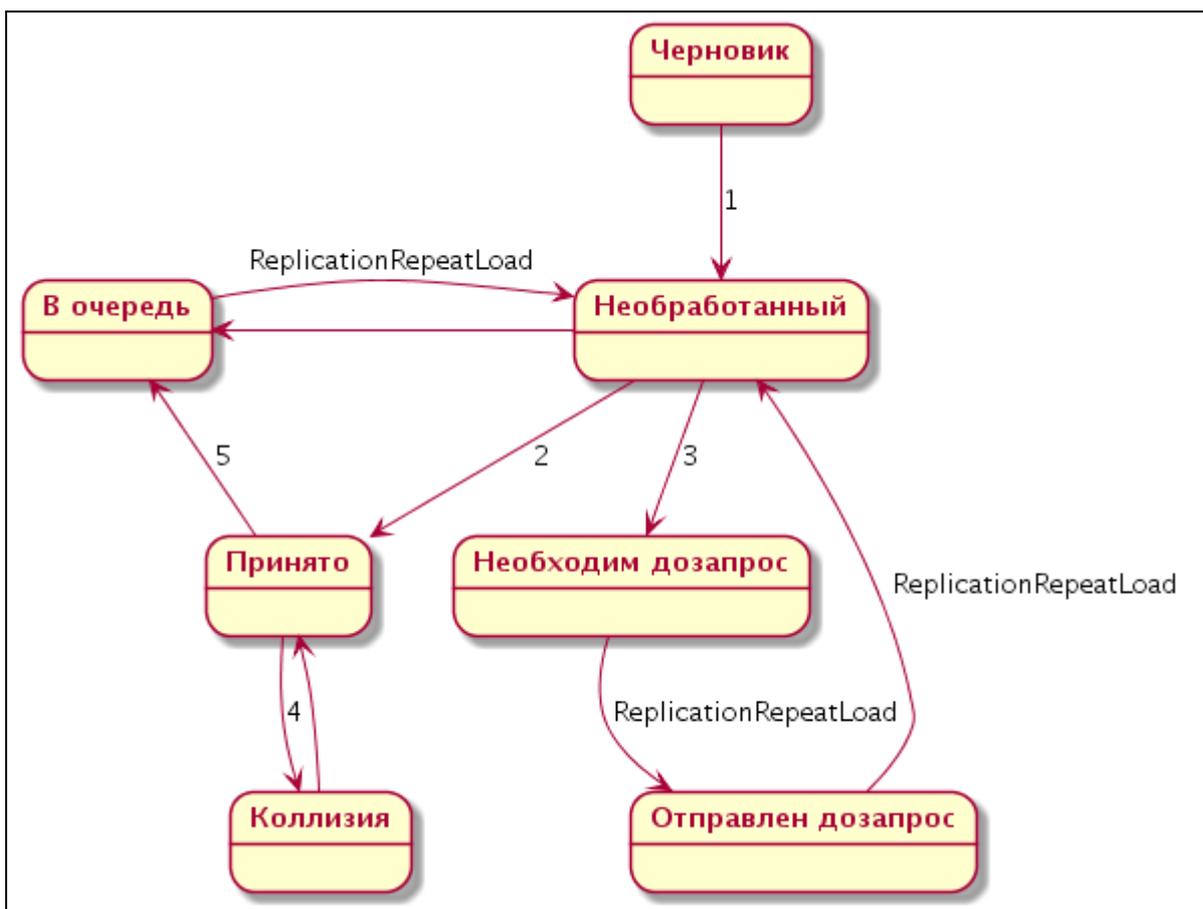


Рисунок 13 – Жизненный цикл карточек шаблона «Репликация» для механизма загрузки данных

Задачник «ReplicationRepeatLoad»

Дополнительно к загрузке пакетов на каждой итерации задачника запускается «ReplicationRepeatLoad». Функции задачника:

1. Для карточек в статусе «Необходим дозапрос» производит отправку пакета с дозапросом несуществующих карточек, на которые ссылается данная карточка. После этого карточка переводится в статус «Отправлен дозапрос».
2. Для карточек в статусе «Отправлен дозапрос» производится проверка, все ли зависимости разрешены. Если все, то карточка переводится в статус «Необработанный» (в связи с чем запускается процесс, описанный выше).
3. Для карточек в статусе «В очередь» переводит в статус «Необработанный».
4. Для карточек в статусах «Ошибка», «Черновик» осуществляется дозапрос.

Промежуточный и итоговый документ

При поступлении пакета получателю кроме непосредственно реплицированной карточки создается дополнительная карточка такого же шаблона, в которую копируются изменения из

реплицированной карточки по заданным правилам. Таким образом, промежуточная карточка является копией карточки на отправителе, а итоговая является объектом, с которым работает пользователь. При этом итоговая иерархия карточек может содержать внутренние данные, необходимые для работы только в данной конкретной системе.

В качестве примера можно привести регистрацию документа в системе получателя под новым номером: как следствие, отчеты и поручения будут иметь названия, включающие новые регистрационные данные.

Можно выделить два основных блока, обеспечивающих работу данного механизма (Рисунок 14):

- копирование изменений из промежуточной карточки в итоговую;
- копирование изменений из итоговой карточки в промежуточную с последующим сохранением последней, чтобы сработали процессоры и произошла репликация.

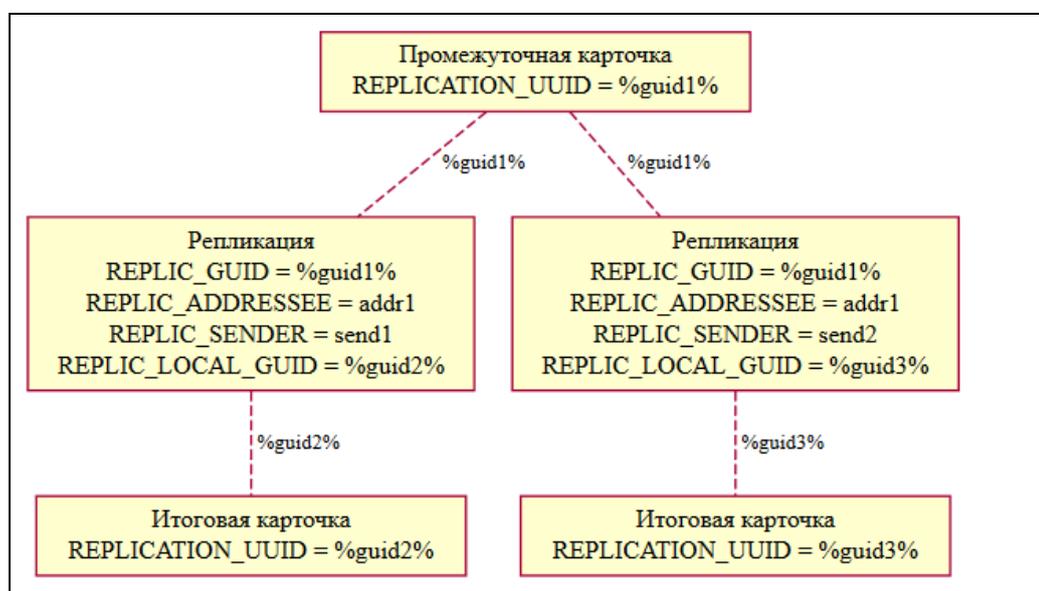


Рисунок 14 – Схема связи промежуточной и итоговой карточек

Копирование изменений из Промежуточной карточки в Итоговую

Сигналом для начала загрузки изменений служит успешное обновление промежуточной карточки. Признаком этого служит перевод карточки Репликация в статус «Принято» после отработки процессора загрузки изменений.

Далее создается/изменяется на основе того же пакета итоговая карточка, но с новым локальным UUID, при этом предусмотрена возможность произвести предобработку пакета и постобработку перед сохранением полученной карточки специфичным в зависимости от шаблона способом.

Список всех шаблонов, для которых производится копирование:

- Входящий;
- ОГ;

- Поручение;
- Рассмотрение;
- Отчет об исполнении;
- Отчет об исполнении внешним исполнителем;
- Ознакомление с поручением;
- Файл;
- Перенаправление ОГ;
- Результаты рассмотрения.

Базовая функциональность таких обработчиков включает функциональность фильтрации атрибутов в зависимости от настроек.

Для шаблонов Входящий и ОГ произведены настройки по фильтрации следующих атрибутов:

- промежуточный документ (репликация) (REPLICATED_DOC_LNK);
- итоговый документ (репликация) (REPLIC_LOCAL_DOC_LNK);
- тип документа репликации (REPLICATED_DOC_TYPE);
- регистратор (JBR_REGD_REGISTRAR);
- журнал регистрации (JBR_REGD_REGJOURN);
- регистрационный номер (JBR_REGD_REGNUM);
- введенный вручную номер (JBR_MANUALLY_NUMBER);
- предыдущие регистрации (JBR_REG_LAST_REG);
- регистрационный номер (цифровая часть) (JBR_REGD_REGNUM_D);
- дата регистрации (JBR_REGD_DATEREG);
- индекс номенклатуры (JBR_REGD_INDEX).

Дополнительно для шаблонов Входящий и ОГ реализована следующая логика для случая, если создается итоговый документ:

- добавляется связь между документами с помощью атрибута «Промежуточный документ (репликация)» (REPLICATED_DOC_LNK);
- атрибуту «Тип документа репликации» (REPLICATED_DOC_TYPE) выставляется значение «Итоговый документ»;

- создается карточка «Информация о регистрации», в которую заносятся данные о Регистрационном номере, Регистрационной дате, Регистраторе и Организации, к которой он относится. Данная карточка записывается в атрибут «Предыдущие регистрации» документа.

Копирование изменений из Итоговой карточки в Промежуточную

Копирование изменений из итоговой карточки в промежуточную происходит при тех же событиях, что и Репликация карточки: сохранение и изменение статуса, после отработки процессора Репликации.

В данном случае вся основная работа производится обработчиком для соответствующего шаблона, который поддерживает обработку двух типов событий: создание карточки и изменение карточки. Отличием является принцип фильтрации: в случае, если карточка новая, то копируются все атрибуты; если изменяется существующая, то только сконфигурированный список атрибутов. Сам процессор срабатывает только на изменения карточек типа «итоговая», поэтому событие «создание карточки» происходит в процессе работы данного копирующего механизма.

Создание новых карточек производится для случая, если данная карточка является итоговой и к ней привязана некоторая другая карточка, которая входит в множество «поддерживаемых» репликацией, но которая не является итоговой согласно формальным признакам (нет карточки Репликации, ссылающейся на нее с помощью local UUID). Это свидетельствует о том, что данная карточка принадлежит к множеству итоговых, но появилась только локально (например, создали дочернее поручение). В результате есть необходимость создать промежуточную версию данной карточки, чтобы она реплицировалась отправителю. Для справочного шаблона, не нужно создавать промежуточные карточки, а нужно использовать ту же самую карточку в процессе копирования.

Текущие настройки обработчиков

Справочные шаблоны:

- Персона (Внутренний);
- Персона (Внешний);
- Организация;
- Департамент;
- Должность.

Шаблоны, участвующие в копировании. Список копируемых атрибутов:

- Входящий:
 - Резолюции;
 - Рассмотрение.
- ОГ:

-
- Перенаправленные обращения;
 - Результаты рассмотрения;
 - Резолюции;
 - Рассмотрение.
- Поручение:
- Резолюция;
 - Связанные поручения;
 - Исполнитель;
 - Соисполнители;
 - К сведению;
 - Должность фамилия и инициалы лица, подписавшего резолюцию;
 - Дата подписания;
 - Отправил на исполнение;
 - На контроле;
 - Срок;
 - Контролер;
 - Предварительный срок;
 - Вложения;
 - Отчет об исполнении;
 - Отчеты внешних исполнителей;
 - Категория срочности;
 - Подпись.
- Рассмотрение:
- Предыдущий рассматривающий;
 - Рассматривающий;
 - Ответственный рассматривающий;
 - Рассмотреть до;
 - Комментарий;
 - Дата исполнения;
 - Срочность;
 - Категория срочности;
 - Подпись.
-

-
- Отчет об исполнении:
 - Исполнитель;
 - Дата утверждения;
 - Утверждающий скрытый;
 - Текущий отчет;
 - Комментарий;
 - Дата исполнения;
 - Отчет;
 - Вложения;
 - Подготовленные документы;
 - Рассмотрен в АРМ скрытый;
 - Фрагмент отчета скрытый;
 - Контролер;
 - Категория срочности;
 - Причина отправки на доработку;
 - Подпись.
 - Отчет об исполнении внешним исполнителем:
 - Текущий отчет;
 - Отчет;
 - Вложения;
 - Подготовленные документы;
 - Рассмотрен в АРМ скрытый;
 - Фрагмент отчета;
 - Комментарий;
 - Дата исполнения;
 - Исполнитель;
 - Контролер.
 - Ознакомление с поручением:
 - Рассмотрен в арм скрытый;
 - Кому ознакомиться;
 - Ознакомиться до;
-

- Дата ознакомления.
- Файл;
- Перенаправление ОГ;
- Результаты рассмотрения.

Для Входящего и ОГ дополнительно к базовой реализована следующая функциональность: поиск карточек «Информация о регистрации» с указанным номером, датой и регистратором. Если карточки найдены, и они привязаны к промежуточной карточке, то никакие действия не производятся, если не привязаны, то создается новая карточка «Информация о регистрации» с номером регистрации, датой регистрации, регистратором и организацией данного регистратора и прикрепляется с помощью атрибута «Предыдущие регистрации» к промежуточному документу.

Настройка модуля Репликация

1. Настроить директорию, в которую модуль Репликация будет выгружать, и собирать пакеты. Для этого необходимо открыть в конфигурационном каталоге «JBoss» файл «`$(JBoss_HOME)/server/default/conf/dbmi/replication/ReplicationNodeConfig.xml`». Если данного файла еще не существует, то его необходимо создать путем переименования файла «`ReplicationNodeConfig.xml.example`». Далее нужно настроить следующие параметры:
 - `IncomingFolder` - указывается директория, из которой модуль Репликация забирает пакеты для загрузки. В дальнейшем следует ссылаться на данный путь с помощью алиаса `$REPLIC_IN_FOLDER`;
 - `OutgoingFolder` - указывается директория, в которую модуль Репликация будет выгружать пакеты для отправки. В дальнейшем следует ссылаться на данный путь с помощью алиаса `$REPLIC_OUT_FOLDER`.
2. Настроить транспортную составляющую узла Репликации. Для этого необходимо открыть в конфигурационном каталоге JBoss в файл «`$(JBoss_HOME)/server/default/conf/dbmi/replication/ReplicationNodeConfig.xml`»:
 - `ServerGUID` - указывается уникальный идентификатор данной СЭД (узла Репликации). На данный UUID опирается логика модуля по разрешению Коллизий и вычисление отправителя при инициации Репликации. Если данный узел (СЭД) поддерживает работу только одной организации, то необходимо выставить значение, равное идентификатору данной организации. `Organizations` в данном случае настраивать не обязательно;
 - `Organizations GUID` - указываются уникальные идентификаторы организаций, базирующихся на данной СЭД. На данный UUID опирается лишь логика, проверяющая относится ли пришедший пакет к данной СЭД или помещен ошибочно в директорию Входящих. Если СЭД поддерживает несколько организаций, то нужно перечислить в этом элементе идентификаторы всех этих

организаций. Для ServerGUID следует указать некий абстрактный UUID, который будет служить именно идентификатором СЭД;

- ReplicationMember - указываются уникальные идентификаторы всех остальных узлов, участвующих в процессе Репликации. Таким образом, если есть, например, три СЭД (узла), участвующих в Репликации, то в настройки данной попадет ServerGUID двух оставшихся. Настройки заключается в том, что, когда изменяется карточка, не относящаяся к конкретному адресату, то необходимо выполнить рассылку пакета с Репликацией всем остальным узлам (например, это Организация, Департамент, Персона и т.п.).

3. Проверить, существует ли файл «\$JBOSS_HOME/server/default/conf/dbmi/replication/ReplicationTemplateConfig.xml». Если нет, то необходимо переименовать соответствующий файл «ReplicationTemplateConfig.xml.example».
4. Запустить задачник «ReplicationTask» через интерфейс администратора: Администрирование системы - Задачи.
5. Настроить организации, участвующие во взаимообмене. Для Репликации иницирующим требованием является синхронизированность справочников: все СЭД должны иметь один и тот же набор Персон (внутренних), участвующих во взаимообмене, привязанных к Организациям. Организации, участвующие во взаимообмене, также должны быть синхронизированы. На данный момент обязательным требованием является синхронизированность Персон (не карточек) на всех системах, участвующих в обмене. Таким образом, для всех организаций, участвующих в Репликации необходимо настроить атрибут UUID узла репликации (org) (ORG_REPL_UUID).

V. Интеграции с внешними системами

V.1. Интеграция с внешними системами

Данное решение предоставляет широкий спектр вариантов и методов интеграции приложений. Разделяя системы по архитектурным особенностям реализации, предлагается следующий набор вариантов:

- интеграция систем имеющие Web-интерфейс;
- интеграция на основе прямого доступа к базам данных;
- интеграция на основе удаленного вызова процедур внешних систем;
- интеграция на основе WSRP.

V.2. Интеграция на основе прямого доступа к базам данных

Интеграция систем в портале на основе прямого доступа к базам данных сводится к набору операций:

- осуществление соединения с базой данных;
- подготовка соответствующего SQL оператора;
- выполнение SQL оператора и получение набора результатов;
- отображение набора результатов в портлете.

Большинство современных средств разработки имеют соответствующие инструменты, позволяющие построить простейший SQL Портлет не прибегая к написанию кода.

V.3. Интеграция на основе удаленного вызова процедур внешних систем

В случае если система не имеет web-интерфейса, а интеграция на основе прямого доступа к базам данных не целесообразна или невозможна, используется метод удаленного вызова процедур. Данный метод подразумевает обращение к процедурам (методам) одной системы непосредственно из другой. В случае интеграции такой системы в портал портлет непосредственно обращается к внешним процедурам, используя одну из принятых технологий.

В настоящее время имеется множество реализаций удаленного вызова процедур (RPC): CORBA, COM, .NET Remoting, Java RMI и др. Различия заключаются в простоте использования и возможности поддержки систем. Часто функциональные возможности расширяются за счет добавления дополнительных свойств, таких как поддержка транзакционности.

Удаленный вызов процедур использует принцип инкапсуляции интегрируемых приложений. Если приложению требуется информация, которая содержится в другом приложении, оно

выполняет запрос к этому приложению напрямую. При необходимости изменения данных другого приложения, приложение осуществляет изменение посредством вызова приложения – владельца данных. Каждое приложение поддерживает целостность своих данных и может изменять их, не воздействуя на другие приложения.

V.4. Интеграция на основе WSRP

Фактически, любой из описанных выше подходов к интеграции может быть сведен к следующей схеме (Рисунок 15).

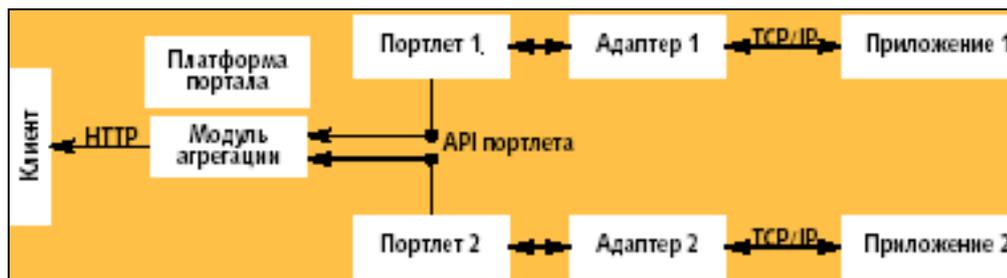


Рисунок 15 - Обобщенная схема «традиционной» интеграции

В такой интеграции участвуют четыре компонента: приложение, адаптер приложения на стороне портала, портлет и сам портал в качестве модуля агрегации данных. Приложение выступает поставщиком информации, и в общем случае нет никаких формальных ограничений относительно того, как эти данные должны быть представлены и переданы от источника к потребителю. Поэтому приложение должно иметь на стороне портала адаптер приложения. В качестве базового протокола используется TCP/IP, а протокол верхнего уровня может быть любым: DCOM, RMI или собственный протокол приложения. Непосредственно с адаптером взаимодействует портлет, который через интерфейсы адаптера получает данные, обрабатывает их на уровне логики задачи, формирует «презентационные» данные и отправляет результат порталу. Портал объединяет результаты, полученные от разных портлетов, и отправляет их клиенту. В рассмотренном варианте портлет отвечает как за логику задачи, так и за способ представления результатов.

Целью стандартизации WSRP (Web Services for Remote Portals) является не только устранение «узких» мест ранее описанных вариантов, но и расширение функциональности, например, за счет возможностей автоматического подключения удаленных портлетов, отвечающих стандарту. Достигается это посредством формализации способа взаимодействия между порталом и приложением, а также унификацией технологии представления результатов, введя для всех WSRP-сервисов одинаковые «презентационные» интерфейсы, позволяющие любому WSRP-совместимому portalу использовать их.

Первая задача решалась соглашением о том, что WSRP-сервисы являются Web-сервисами, следовательно, в основе работы с ними лежит стек протоколов UDDI, WSDL и SOAP. Вторая задача решалась разработкой и согласованием «презентационных» WSRP-интерфейсов. Интеграция по стандарту WSRP показана на рисунке (Рисунок 16).

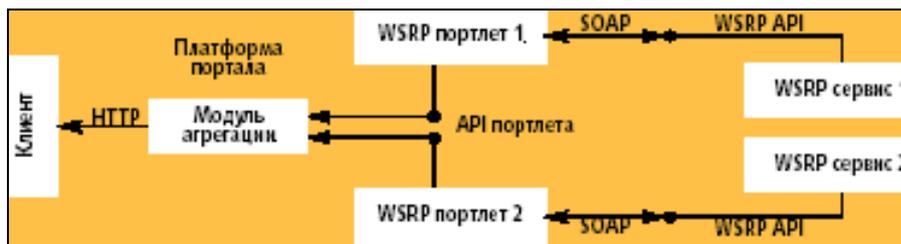


Рисунок 16 - WSRP интеграция

Технология COM позволила создавать внешние интерфейсы для приложений, содержащих объектно-ориентированные структуры выходных данных произвольной сложности. Похожим образом новый стандарт определяет структуры данных и интерфейсы, позволяющие удаленным приложениям предоставлять свои данные универсальным способом: через WSRP-сервисы. Например, для получения данных от сервиса необходимо использовать интерфейс `getMarkup()`. Один из параметров этого интерфейса позволяет передать информацию о том, в каком виде пользователь (в нашем случае портал) желает видеть возвращаемые сервисом данные. По спецификации сервис может вернуть данные в любом из имеющихся типов данных MIME. Можно даже определить массив, который в приоритетном порядке будет содержать желательные для возврата типы данных.

Если сервис не поддерживает графический интерфейс, то данные могут быть возвращены в виде HTML-таблицы или в формате XML. Первая версия стандарта закладывает основы его дальнейшего развития, рассматривая более сложные ситуации: реакция WSRP-сервисов на действия пользователей, сохранение промежуточных состояний сервиса, обработка ошибок, передача данных в двоичном виде, шифрование и др. В частности, интерактивная работа подразумевает сохранение текущих пользовательских установок; следовательно, необходимо контролировать состояние текущего сеанса для данного WSRP-сервиса. WSRP-сервисы могут иметь и конфигурационные установки. Стандарт указывает способ сохранения и получения таких установок через представителя WSRP-сервиса на портале, т.е. WSRP-портлета.

Привлекательность WSRP-интеграции заключается еще и в том, что WSRP-совместимый портал имеет встроенный модуль, который берет на себя функции поиска и встраивания WSRP-сервисов. Результатом его работы является автоматически сгенерированный представитель WSRP-сервиса на портале — WSRP-портлет (Portlet proxy). Таким образом, интеграция приложений через WSRP-сервисы осуществляется административными средствами портала, никаких дополнительных действий по разработке промежуточных модулей не требуется.

V.5. Общая схема интеграции СЭД «Логика СЭД. СПО» с внешними системами

Общая схема взаимодействия СЭД «Логика СЭД. СПО» с внешними системами приведена на рисунке (Рисунок 17).

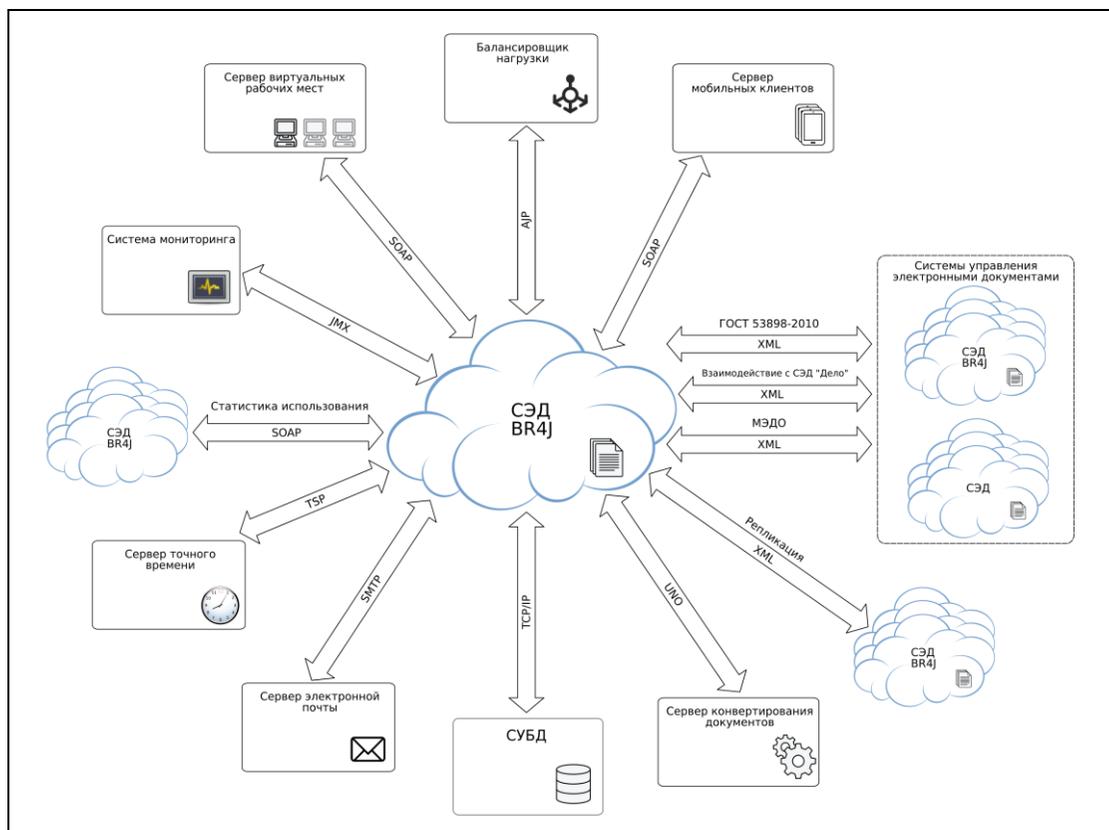


Рисунок 17 - Общая схема взаимодействия СЭД «Логика СЭД. СПО» с внешними системами

VI. Перечень используемых технологий

VI.1. Архитектура Системы

Техническое решение представляет собой единую информационную среду, объединяющую всю необходимую информацию для пользователей и позволяющую осуществлять унифицированную обработку информационного содержания (контента).

Решение реализовано на базе следующего набора продуктов:

- Сервер приложений - JBoss Enterprise Application Platform version 4.3;
- J2EE Portal Портал - JBoss Enterprise Portal version 2.6;
- Модуль хранения и обработки Информационного содержания портала (портальное приложение).

Для организации хранения данных предназначена СУБД PostgreSQL 9.3.

VI.2. Описание Архитектуры

Архитектура системы использует классическую для Web-приложений 3-х уровневую модель (Рисунок 18):

- уровень представления (Presentation Layer) – отвечает за представление информации, состоит из Web-страниц, построенных с использованием технологий HTML, JavaScript, JSP, Servlets.
- уровень бизнес-логики (Business Logic Layer) – отвечает за переработку информации, представлен Java классами, реализующими бизнес-логику. А также Java классами, отвечающими за доступ к данным (Data Access) - JDBC для доступа к базе данных приложения.
- уровень базы данных (Database Layer) – отвечает за хранение информации, представлен базой данных СУБД PostgreSQL 9.

Система рассчитана на интенсивный режим работы, подразумевающий круглосуточную эксплуатацию с перерывами на техническое обслуживание.

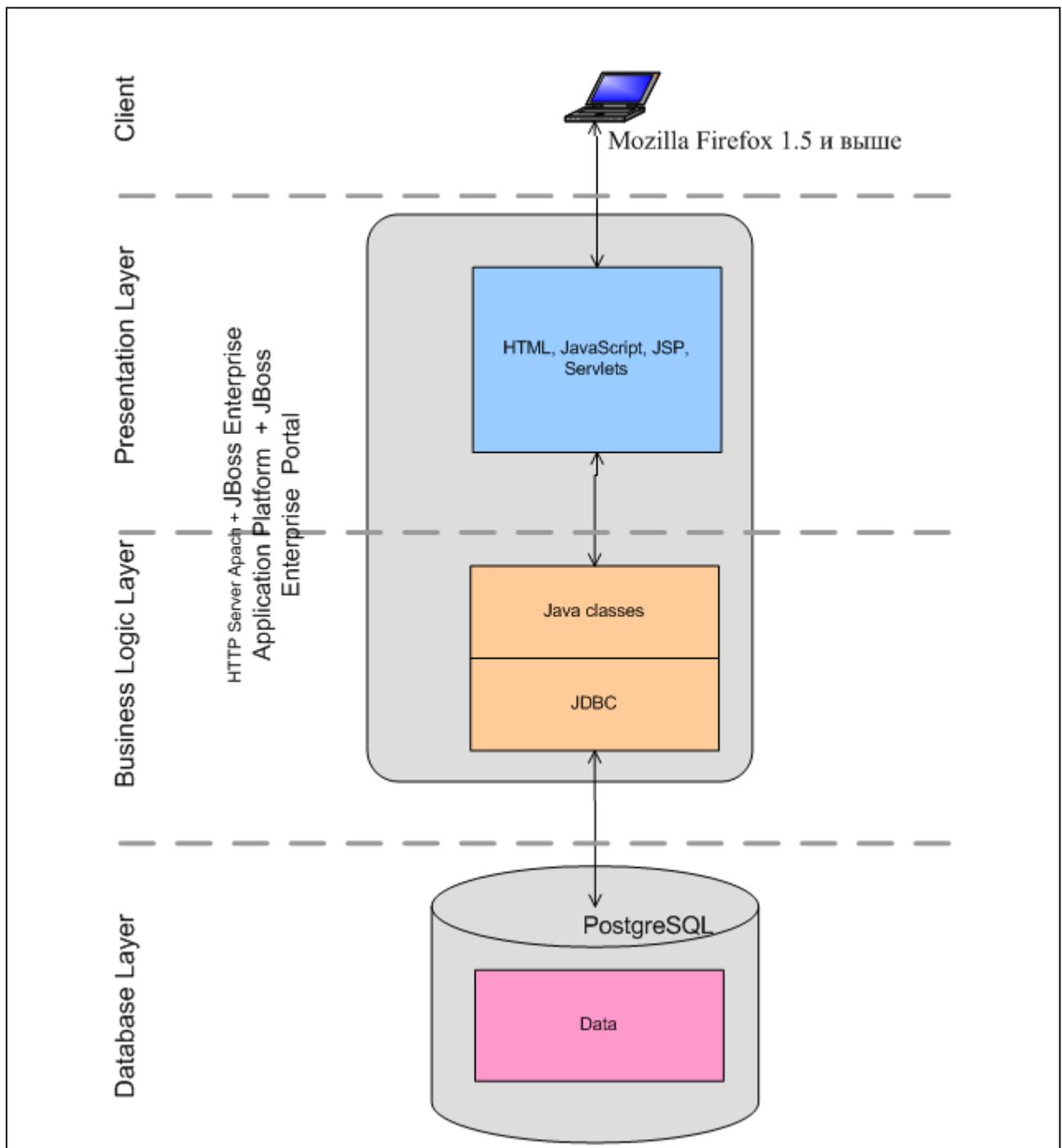


Рисунок 18 - Трехуровневая модель архитектуры системы

VI.3.Используемые технологии

Перечень используемых технологий:

- Portlet Specification and API 1.0 (JSR-168);
- Web Services for Remote Portlets 1.0 (WSRP);
- Java Server Faces 1.2 (JSR-252);

- Java Management Extension (JMX) 1.2;
- Full J2EE 1.4 compliance.

VI.4. Методы доступа к информации

Система - это унифицированное представление приложений в виде портлетов, позволяющее пользователям указывать способы отображения наборов информационных блоков и ресурсов в одном контексте, на одной странице. Процедуры отображения таких наборов могут изменяться в зависимости от требований, предъявляемых запрашивающим устройством. При каждом запросе, поступающем от устройства, выполняются следующие действия по объединению:

- сбор сведений о пользователе, устройстве и выбранном языке;
- выбор активных портлетов из набора приложений, доступ к которым разрешен пользователю;
- объединение вывода активных портлетов в единую страницу, пригодную для отображения на экране устройства;
- система также имеет возможность создания пользовательской модели навигации, включающей следующие функции:
 - многоуровневая навигация;
 - настраиваемые темы и оболочки;
 - настраиваемое размещение на странице портлетов (и, следовательно, информации).

VI.5. Портлеты

Система строится с использованием механизмов портлетов (Рисунок 19).

Портлеты - это небольшие приложения системы, разработка, внедрение и просмотр которых осуществляется независимо друг от друга, так же, как и управление портлетами.

Администраторы могут изменять настройки системы, выбирая портлеты и располагая их в желаемом порядке, в результате чего получают измененные Web-страницы отображения информационных ресурсов системы.

Портлеты, являясь полноценными приложениями, разработанными в соответствии с архитектурой модель-представление-управление, предусматривают несколько состояний и режимов представления, а также функции работы с событиями и сообщениями. По аналогии с сервлетами, работающими в рамках сервера приложений, портлеты работают в рамках контейнера портлетов, входящего в состав компонента J2EE Portal, который обеспечивает среду их выполнения.

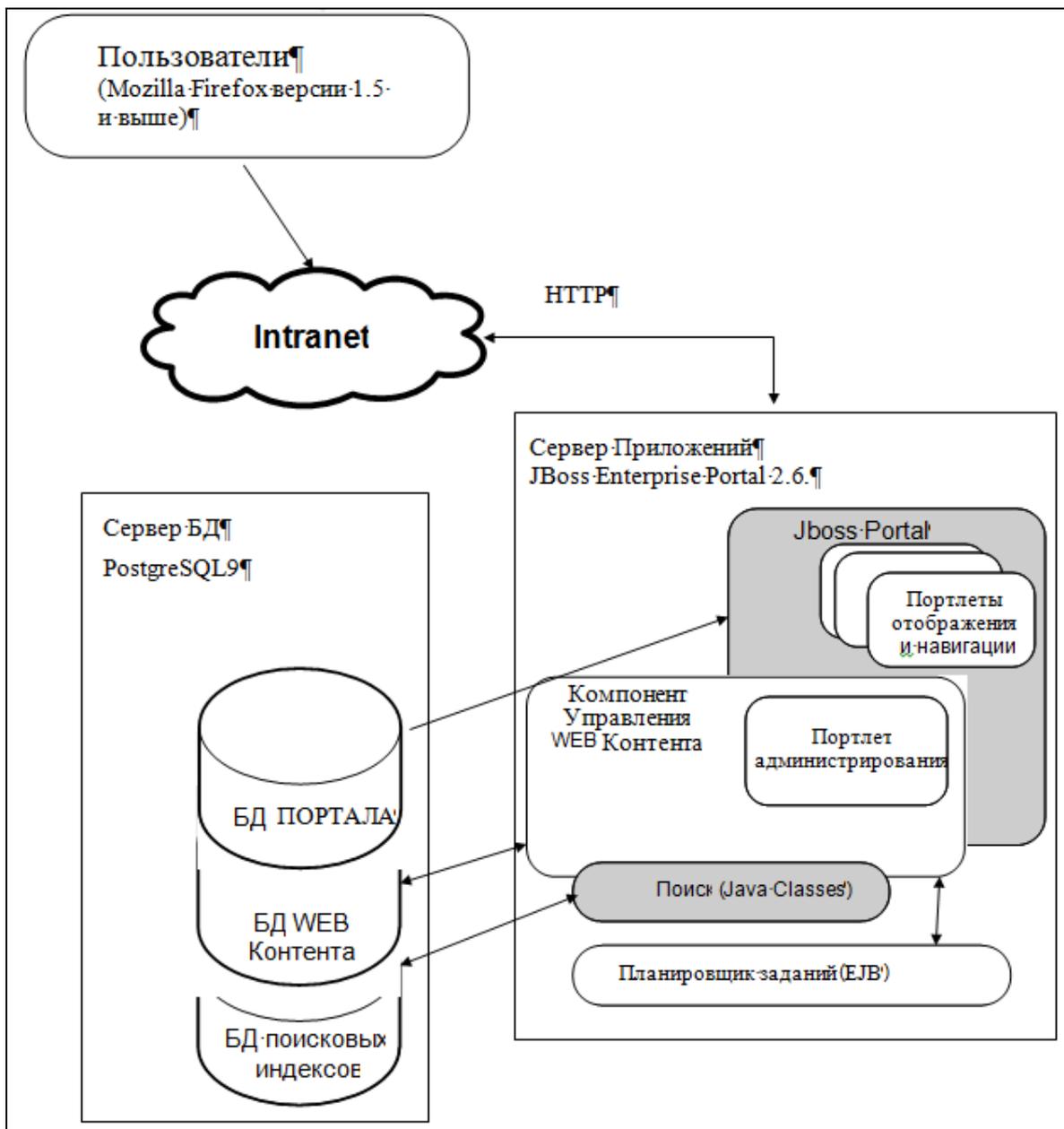


Рисунок 19 - Схема взаимодействия компонентов системы

VI.6. Логическая структура представления информации

Любые информационные материалы (информация, добавляемая пользователем) в Системе хранятся в одной из следующих форм:

- учетная карточка – это описание информационного ресурса, хранящееся в системе и используемое при поиске и обработке. Учетная карточка создается по определенному для каждого типа информационных ресурсов, с заданным набором свойств – шаблону. К каждой карточке может быть прикреплен файл;

-
- шаблон – это принятая в решении форма для создания учетной карточки материала, шаблон состоит из блоков. Для шаблона дополнительно определяется расположение блоков, права доступа и пр.;
 - блок характеристик (атрибутов) – это способ группировки данных, используемых для создания шаблонов учетных карточек информационных ресурсов. Одни и те же блоки могут быть использованы в разных шаблонах. Блоки состоят из описания (названия на русском и английском языках) и набора классификационных характеристик. Дополнительно для блока определяются способы расположения характеристик, обязательность их заполнения, возможность отображения, а также порядок и ширина поля в результатах поиска;
 - характеристика (атрибут) – это минимальная часть для описания информационного ресурса учетной карточкой. Одни и те же характеристики могут быть использованы в различных блоках. Могут быть датой, числом, текстом, справочником (линейный и иерархический список), ссылкой на другие карточки, полями специальных форм, встраиваемыми текстовыми редакторами.

VII. Перечень прикладных таблиц СЭД «Логика СЭД. СПО» с указанием назначения и описание структуры

VII.1. Структура базы данных

База данных состоит из таблиц:

- таблицы структуры;
- таблицы свойств атрибутов;
- таблицы жизненного цикла (ЖЦ) карточек;
- таблицы пользователей и ролей;
- таблицы значений и истории;
- таблицы прав и состояний;
- прочие таблицы.

VII.2. Описание таблиц

Схема Dbmi_trunk:

- **attribute** – содержит параметры атрибутов;
- **attribute_value** – содержит значения атрибутов;
- **attribute_value_hist** – история значений атрибутов;
- **attribute_view_param** – настройки отображения атрибутов;
- **attr_block** – содержит параметры блоков;
- **default_attribute_value** – значение атрибутов по умолчанию;
- **block_view_param** – параметры отображения блоков;
- **card** – содержит параметры карточек;
- **card_access** – права доступа к карточкам;
- **card_status** – статусы карточек;
- **card_version** – версии карточек;
- **person** – информация о пользователях; синхронизируется с таблицей jbr_users схемы public;

- **system_role** – роли в системе;
- **person_role** – присвоение определенной роли определенному пользователю;
- **person_ungrouped_role** – список ролей, назначенных пользователям без привязки к группе ролей;
- **person_role_group** – группы ролей пользователей;
- **group_role** – состав групп ролей системы;
- **system_group** - список групп ролей системы;
- **values_list** – список значений линейных справочников;
- **reference_list** – список линейных справочников;
- **workflow** – список переходов;
- **workflow_move** – параметры перехода;
- **workflow_move_required_field** – обязательность заполнения атрибутов при определенном переходе;
- **xml_data** – содержит разметку для определенных объектов.

Схема public:

- **jbp_roles** – порталные роли;
- **jbp_users** – порталные пользователи, синхронизируемые с пользователями портала.

VII.3. Таблицы структуры

- **template** – список шаблонов карточек;
- **attribute** – список атрибутов;
- **attr_block** – список блоков;
- **tab** – список закладок;
- **tab_template** – привязка закладок к шаблонам;
- **tab_block** – привязка блоков к закладкам;
- **template_attribute** – привязка атрибутов к шаблонам;
- **template_block** – привязка блоков к шаблонам.

Схема взаимодействия таблиц структуры представлена на рисунке (Рисунок 20).

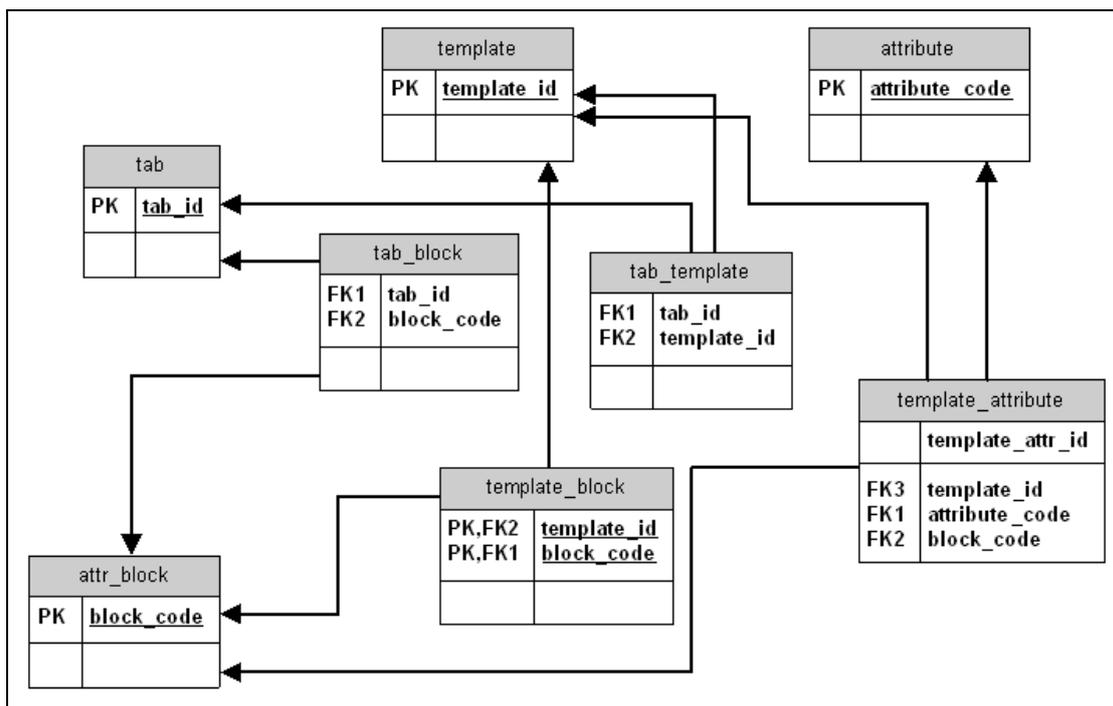


Рисунок 20 – Взаимодействие таблиц структуры

Описание полей таблицы **template** приведено в таблице (Таблица 4).

Таблица 4. Описание полей таблицы **template**

Название поля	Тип	Описание
template_id	numeric(9,0)	Уникальный номер шаблона
template_name_rus	character varying(128)	Русское название шаблона
template_name_eng	character varying(128)	Английское название шаблона
is_active	numeric(1,0)	1 – шаблон активен, 0 – нет
locked_by	numeric(9,0)	Id пользователя, заблокировавшего шаблон
lock_time	timestamp(6) without time zone	Время блокировки шаблона
is_system	numeric(1,0)	1 – шаблон системный
workflow_id	numeric(9,0)	Id ЖЦ шаблона
show_in_createcard	numeric(1,0)	1 – показывать шаблон на странице создания новых карточек
show_in_search	numeric(1,0)	1 – показывать карточки данного шаблона в поиске

Описание полей таблицы **attribute** приведено в таблице (Таблица 5).

Таблица 5. Описание полей таблицы **attribute**

Название поля	Тип	Описание
attribute_code	character varying(20)	Код атрибута
attr_name_rus	character varying(128)	Русское название атрибута
attr_name_eng	character varying(128)	Английское название атрибута
data_type	character(1)	Тип атрибута (A,B,C,D,E,H,I,L,M,S,T,U,W,Y)

Название поля	Тип	Описание
block_code	character varying(20)	Код блока
order_in_block	numeric(4,0)	Порядок атрибута в блоке
column_width	Numeric	Ширина колонки в списке карточек
display_length	numeric(4,0)	Длина атрибута в карточке
rows_number	numeric(4,0)	Количество строчек для атрибута
is_mandatory	numeric(1,0)	1 – атрибут обязательный
is_active	numeric(1,0)	1 – атрибут активный
is_system	numeric(1,0)	1 – создан системный
ref_code	character varying(20)	Код справочника для атрибута
locked_by	numeric(9,0)	Id пользователя, заблокировавшего атрибута
lock_time	timestamp(6) without time zone	Время блокировки атрибута
is_readonly	numeric(1,0)	1 – атрибут только для чтения
is_hidden	numeric(1,0)	1 – атрибут скрытый

Таблица Attribute. Типы атрибутов

- А - атрибут безопасности (SecurityAttribute);
- В - бэклинк-атрибут (BackLinkAttribute);
- С - кардлинк-атрибут (CardLinkAttribute);
- D - атрибут дата/время (DateAttribute);
- Е - типизированный кардлинк (TypedCardLinkAttribute);
- Н - иерархичный атрибут (TreeAttribute);
- I - числовой атрибут (IntegerAttribute);
- L - справочный атрибут (ListAttribute);
- М - материал-атрибут (MaterialAttribute);
- Р - атрибут-список шаблонов (TemplateListAttribute);
- R - атрибут-период дат (DatePeriodAttribute);
- S - строковый атрибут (StringAttribute);
- Т - многострочный текстовый атрибут (TextAttribute);
- U - атрибут-ссылка на пользователя (PersonAttribute);
- W - html-атрибут (HtmlAttribute);
- Y - атрибут-история (CardHistoryAttribute).

Описание полей таблицы **attr_block** приведено в таблице (Таблица 6).

Таблица 6. Описание полей таблицы attr_block

Название поля	Тип	Описание
block_code	character varying(20)	Код блока
block_name_rus	character varying(128)	Русское название блока
block_name_eng	character varying(128)	Английское название блока
is_active	numeric(1,0)	1 – блок активен, 0 – нет
is_system	numeric(1,0)	1 – блок системный
locked_by	numeric(9,0)	Id пользователя, заблокировавшего блок
lock_time	timestamp(6) without time zone	Время блокировки блока

Описание полей таблицы **tab** приведено в таблице (Таблица 7).

Таблица 7. Описание полей таблицы Tab

Название поля	Тип	Описание
tab_id	numeric(9,0)	Id шаблона
name_eng	character varying(100)	Русское название закладки
name_rus	character varying(100)	Английское название закладки

Описание полей таблицы **tab_template** приведено в таблице (Таблица 8).

Таблица 8. Описание полей таблицы tab_template

Название поля	Тип	Описание
template_id	numeric(9,0)	Id шаблона
tab_id	numeric(9,0)	Id закладки
order_lr	numeric(3,0)	Порядок отображения закладки в шаблоне

Описание полей таблицы **tab_block** приведено в таблице (Таблица 9).

Таблица 9. Описание полей таблицы tab_block

Название поля	Тип	Описание
block_code	character varying(20)	Код блока
tab_id	numeric(9,0)	Id закладки
Layout	Numeric	Порядок отображения блока в закладке (100-199 – отображение закладки в левой области, 200-299 – отображение закладки в правой области, 300-399 – отображение закладки во всю ширину карточки)

Описание полей таблицы **template_attribute** приведено в таблице (Таблица 10).

Таблица 10. Описание полей таблицы template_attribute

Название поля	Тип	Описание
template_attr_id	numeric(9,0)	Id привязанного к шаблону атрибуту
template_id	numeric(9,0)	Id шаблона
block_code	character varying(20)	Код блока
attribute_code	character varying(20)	Код атрибута

Название поля	Тип	Описание
is_mandatory	numeric(1,0)	1 – атрибут обязательный
order_in_list	numeric(4,0)	Порядок отображения атрибута при отображении в списке карточек
column_width	Numeric	Ширина колонки в списке карточек
is_hidden	numeric(1,0)	1 – атрибут скрытый
is_readonly	numeric(1,0)	1 – атрибут только для чтения

Описание полей таблицы **template_block** приведено в таблице (Таблица 11).

Таблица 11. Описание полей таблицы template_block

Название поля	Тип	Описание
template_id	numeric(9,0)	Id шаблона
block_code	character varying(20)	Код блока
Layout	Numeric	Порядок отображения блока в шаблоне (100-199 – отображение закладки в левой области, 200-299 – отображение закладки в правой области, 300-399 – отображение закладки во всю ширину карточки)
is_active	numeric(1,0)	1 – блок активен

VII.4. Таблицы свойств атрибутов

- **attribute_option** – свойства атрибутов;
- **xml_data** – xml-свойства атрибутов;
- **default_attribute_value** – значения атрибутов по умолчанию;
- **reference_list** – список справочников;
- **values_list** – список значений справочников.

Схема взаимодействия таблиц свойств атрибутов представлена на рисунке (Рисунок 21).

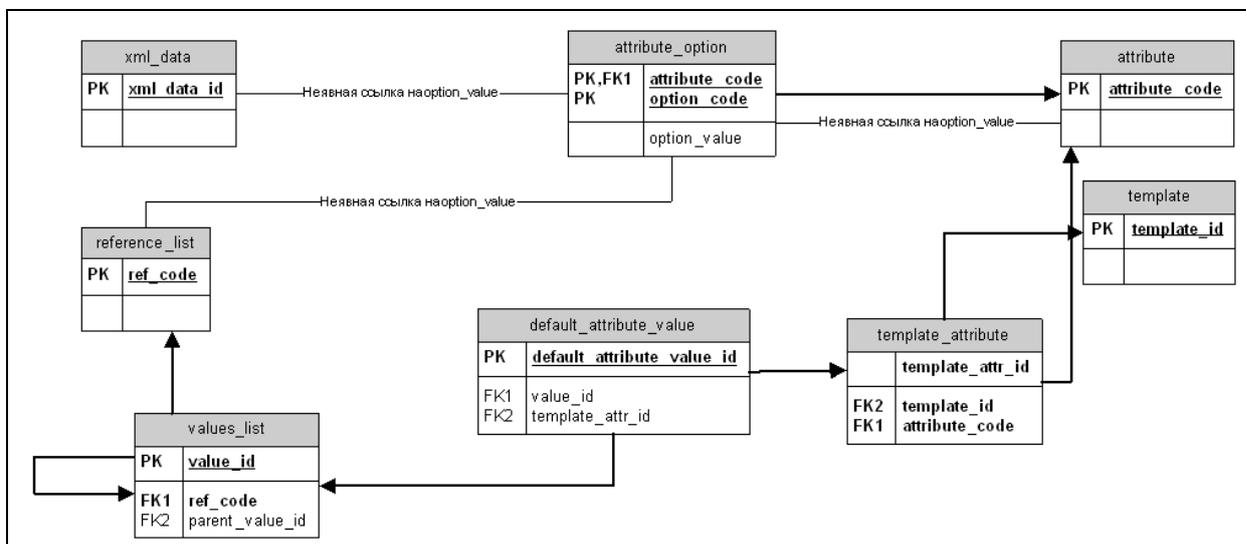


Рисунок 21 – Взаимодействие таблиц свойств атрибутов

Описание полей таблицы **attribute_option** приведено в таблице (Таблица 12).

Таблица 12. Описание полей таблицы attribute_option

Название поля	Тип	Описание
attribute_code	character varying(20)	Код атрибута
option_code	character varying(20)	Код свойства (LINK, REFERENCE, SEARCH, FILTER, UPLINK, RESTRICTED, SINGLEVALUED, ROWS, TIMEPATTERN, SHOWTIME)
option_value	character varying(100)	Значение свойства

Таблица attribute_option. Коды свойств

- LINK – задание кардлинка в родительской карточке, которое соответствует беклинку;
- REFERENCE – задание справочника для атрибута;
- SEARCH – задание поисковой XML для простого или типизированного кардлинка;
- FILTER – задание фильтра для кардлинка;
- UPLINK – задание кардлинка в родительской карточке для случаев, когда имеется дерево связанных карточек и требуется найти родительскую карточку для карточек нижнего уровня;
- SINGLEVALUED – атрибут имеет единственное значение;
- ROWS – количество строк при отображении многострочного атрибута;
- TIMEPATTERN – формат даты/времени для соответствующих атрибутов;
- SHOWTIME – показывать время для атрибутов дата/время.

Описание полей таблицы **xml_data** приведено в таблице (Таблица 13).

Таблица 13. Описание полей таблицы Xml_data

Название поля	Тип	Описание
xml_data_id	numeric(9,0)	Id xml-свойства
xml_type	character varying(20)	Тип свойства (SEARCH)
xml_data	Bytea	Значение свойства
is_system	numeric(1,0)	1 – свойство системное
Description	character varying(100)	Описание свойства

Описание полей таблицы **default_attribute_value** приведено в таблице (Таблица 14).

Таблица 14. Описание полей таблицы default_attribute_value

Название поля	Тип	Описание
default_attribute_value_id	numeric(9,0)	Id значения по умолчанию
template_attr_id	numeric(9,0)	Id атрибута в шаблоне
number_value	Numeric	Числовое значение по умолчанию
string_value	character varying(4000)	Строковое значение по умолчанию
date_value	timestamp without time zone	Значение даты по умолчанию
value_id	numeric(9,0)	Справочное значение по умолчанию
long_binary_value	Bytea	Бинарное значение по умолчанию

Описание полей таблицы reference_list приведено в таблице (Таблица 15).

Таблица 15. Описание полей таблицы reference_list

Название поля	Тип	Описание
ref_code	character varying(20)	Код справочника
Description	character varying(512)	Описание справочника

Описание полей таблицы values_list приведено в таблице (Таблица 16).

Таблица 16. Описание полей таблицы values_list

Название поля	Тип	Описание
value_id	numeric(9,0)	Id значения справочника
ref_code	character varying(20)	Код справочника, для которого описано значение
value_rus	character varying(128)	Русское название значения
value_eng	character varying(128)	Английское название значения
order_in_level	numeric(4,0)	Порядок отображения значения справочника
is_active	numeric(1,0)	1 – значение активное
parent_value_id	numeric(9,0)	Ссылка на value_id родительского справочника

VII.5. Таблицы ЖЦ карточек

- **action** – список СОБЫТИЙ СИСТЕМЫ;
- **card_status** – статусы шаблонов;
- **workflow** – список ЖЦ;
- **workflow_move** – список переходов ЖЦ;
- **workflow_move_required_field** – обязательность заполнения атрибутов при смене статусов карточек.

Схема взаимодействия таблиц ЖЦ представлена на рисунке (Рисунок 22).

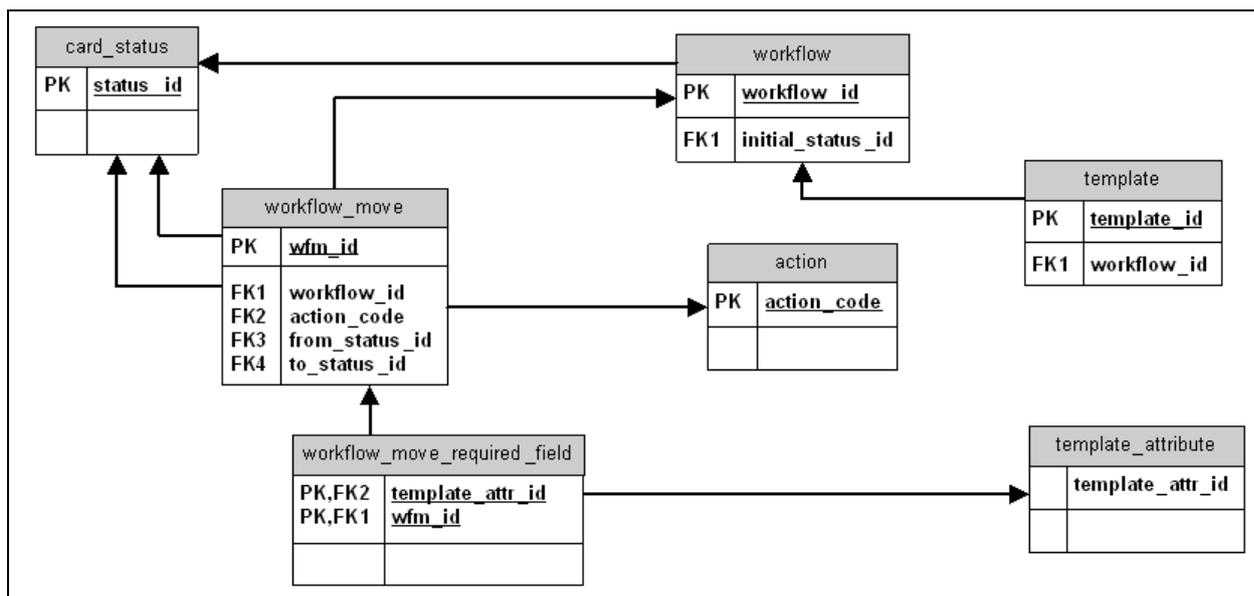


Рисунок 22 – Взаимодействие таблиц ЖЦ

Описание полей таблицы **action** приведено в таблице (Таблица 17).

Таблица 17. Описание полей таблицы **action**

Название поля	Тип	Описание
action_code	character varying(20)	Уникальный код события
action_name_rus	character varying(100)	Русское название события
action_name_eng	character varying(100)	Английское название события
locked_by	numeric(9,0)	Id пользователя, заблокировавшего статуса
lock_time	timestamp(6) without time zone	Время блокировки статуса

Описание полей таблицы **card_status** приведено в таблице (Таблица 18).

Таблица 18. Описание полей таблицы **card_status**

Название поля	Тип	Описание
name_rus	character varying(100)	Русское название статуса
name_eng	character varying(100)	Английское название статуса
default_move_name_rus	character varying(100)	Русское название перехода в этот статус по умолчанию
default_move_name_eng	character varying(100)	Английское название перехода в этот статус по умолчанию
status_id	numeric(9,0)	Id статуса
locked_by	numeric(9,0)	Id пользователя, заблокировавшего статуса
lock_time	timestamp(6) without time zone	Время блокировки статуса

Описание полей таблицы **workflow** приведено в таблице (Таблица 19).

Таблица 19. Описание полей таблицы workflow

Название поля	Тип	Описание
workflow_id	numeric(9,0)	Id ЖЦ
initial_status_id	numeric(9,0)	Начальный статус ЖЦ
name_rus	character varying(100)	Русское название ЖЦ
name_eng	character varying(100)	Английское название ЖЦ
is_active	numeric(1,0)	1 – ЖЦ активный
locked_by	numeric(9,0)	Id пользователя, заблокировавшего ЖЦ
lock_time	timestamp(6) without time zone	Время блокировки ЖЦ

Описание полей таблицы **workflow_move** приведено в таблице (Таблица 20).

Таблица 20. Описание полей таблицы workflow_move

Название поля	Тип	Описание
wfm_id	numeric(9,0)	Id перехода
workflow_id	numeric(9,0)	Id ЖЦ
name_rus	character varying(100)	Русское название перехода
name_eng	character varying(100)	Английское название перехода
from_status_id	numeric(9,0)	Начальный статус перехода
to_status_id	numeric(9,0)	Конечный статус перехода
need_confirmation	numeric(1,0)	1 – выдавать предупреждение перехода
action_code	character varying(20)	Код события
confirmation_rus	character varying(200)	Русский текст подтверждения
confirmation_eng	character varying(200)	Английский текст подтверждения
close_card	numeric(1,0)	1 – закрыть карточку после перехода

Описание полей таблицы **workflow_move_required_field** приведено в таблице (Таблица 21).

Таблица 21. Описание полей таблицы workflow_move_required_field

Название поля	Тип	Описание
wfm_id	numeric(9,0)	Id перехода
template_attr_id	numeric(9,0)	Id атрибута в шаблоне
must_be_set	numeric(1,0)	Признак обязательности атрибута (1,0,2)

VII.6. Таблицы пользователей и ролей

- **person** – список пользователей;
- **system_role** – список ролей системы;
- **person_role** – список всех ролей, назначенных пользователям;
- **person_ungrouped_role** – список ролей, назначенных пользователям без привязки к группе ролей;
- **person_role_group** – группы ролей пользователей;
- **group_role** – состав групп ролей системы;
- **system_group** - список групп ролей системы.

Схема взаимодействия таблиц пользователей и ролей представлена на рисунке (Рисунок 23).

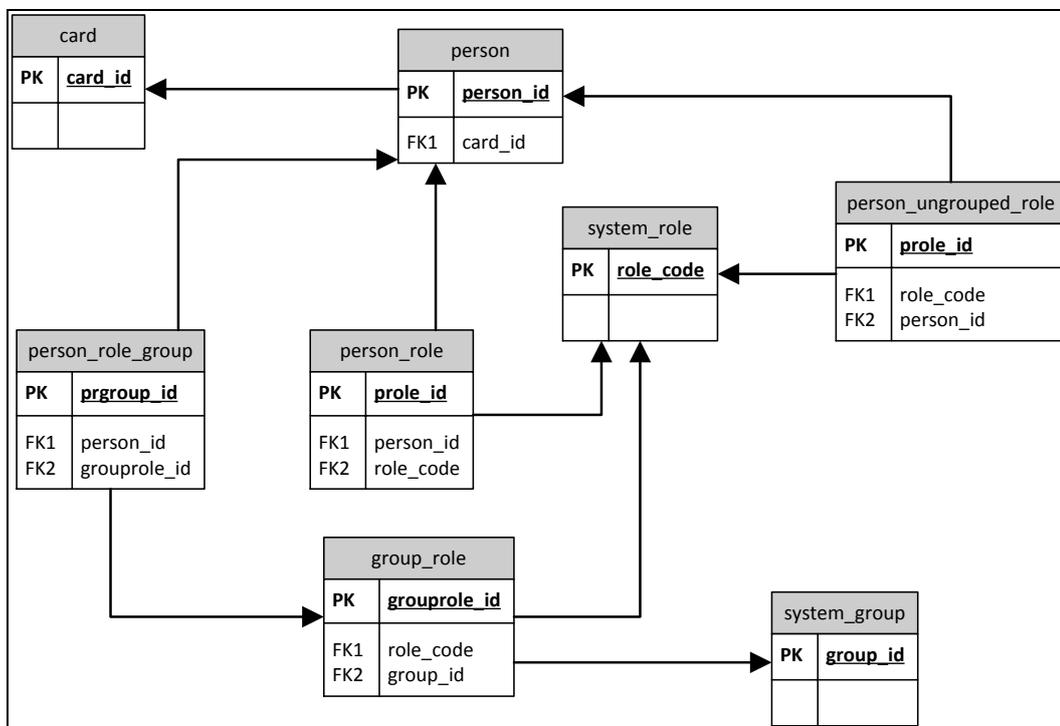


Рисунок 23 – Взаимодействие таблиц пользователей и ролей

Описание полей таблицы **person** приведено в таблице (Таблица 22).

Таблица 22. Описание полей таблицы **person**

Название поля	Тип	Описание
person_id	numeric(9,0)	Id пользователя
person_login	character varying(40)	Логин пользователя
full_name	character varying(256)	Полное имя пользователя
email	character varying(128)	Почтовый ящик
sync_date	timestamp without time zone	Время последней синхронизации
is_active	numeric(1,0)	1 – пользователь активен
locked_by	numeric(9,0)	Id пользователя, заблокировавшего ЖЦ
lock_time	timestamp(6) without time zone	Время блокировки ЖЦ
card_id	numeric(9,0)	Id карточки пользователя
replication_uuid	character varying(40)	UUID репликации

Описание полей таблицы **system_role** приведено в таблице (Таблица 23).

Таблица 23. Описание полей таблицы **system_role**

Название поля	Тип	Описание
role_code	character varying(20)	Код роли
role_name_rus	character varying(100)	Русское название роли
role_name_eng	character varying(100)	Английское название роли

Описание полей таблицы **person_role** приведено в таблице (Таблица 24).

Таблица 24. Описание полей таблицы person_role

Название поля	Тип	Описание
prole_id	numeric(9,0)	Id роли для пользователя
person_id	numeric(9,0)	Id пользователя
role_code	character varying(20)	Код роли

Описание полей таблицы **person_ungrouped_role** приведено в таблице (Таблица 25).

Таблица 25. Описание полей таблицы person_ungrouped_role

Название поля	Тип	Описание
prole_id	numeric(9,0)	ID роли для пользователя
person_id	numeric(9,0)	ID пользователя
role_code	character varying(20)	Код роли

Описание полей таблицы **person_role_group** приведено в таблице (Таблица 26).

Таблица 26. Описание полей таблицы person_role_group

Название поля	Тип	Описание
prgroup_id	numeric(9,0)	ID группы ролей для пользователя
person_id	numeric(9,0)	ID пользователя
group_code	character varying(20)	Код группы ролей

Описание полей таблицы **group_role** приведено в таблице (Таблица 27).

Таблица 27. Описание полей таблицы group_role

Название поля	Тип	Описание
grouprole_id	numeric(9,0)	ID группы ролей
group_code	character varying(20)	Код группы ролей
role_code	character varying(20)	Код роли

Описание полей таблицы **system_group** приведено в таблице (Таблица 28).

Таблица 28. Описание полей таблицы system_group

Название поля	Тип	Описание
group_code	character varying(20)	Код группы ролей
group_name_rus	character varying(100)	Русское название группы ролей
group_name_eng	character varying(100)	Английское название группы ролей

VII.7. Таблицы значений и истории

- **card** – список карточек;

- **attribute_value** – список значений атрибутов в карточках;
- **card_version** – история версий карточек;
- **attribute_value_hist** – история изменения атрибутов;
- **action_log** – таблица действий с карточками.

Схема взаимодействия таблиц значений и истории представлена на рисунке (Рисунок 24).

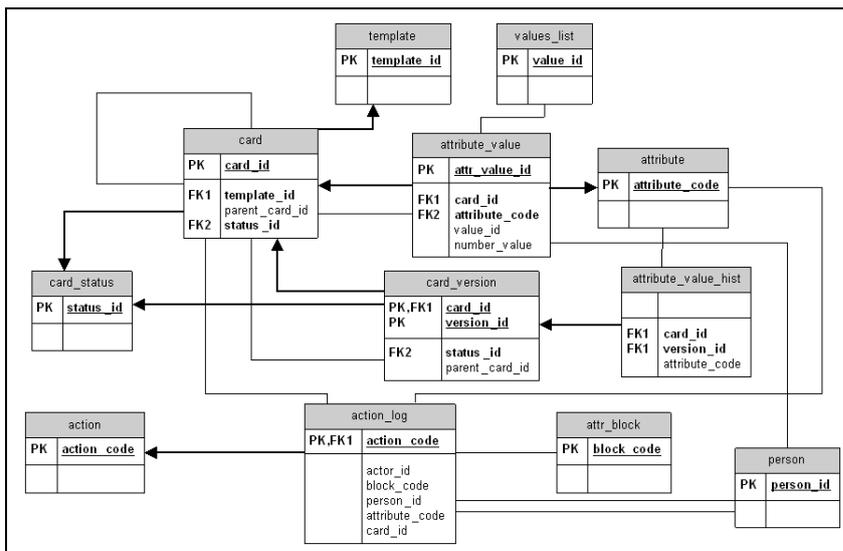


Рисунок 24 – Взаимодействие таблиц значений и истории

Описание полей таблицы **card** приведено в таблице (Таблица 29).

Таблица 29. Описание полей таблицы **card**

Название поля	Тип	Описание
card_id	numeric(9,0)	Id карточки
template_id	numeric(9,0)	Id шаблона
is_active	numeric(1,0)	1 – карточка активна
parent_card_id	numeric(9,0)	Id родительской карточки
file_storage1	Bytea	Файловое хранилище
external_path	character varying(512)	Внешняя папка для хранения
locked_by	numeric(9,0)	Id пользователя, заблокировавшего ЖЦ
lock_time	timestamp(6) without time zone	Время блокировки ЖЦ
file_name	character varying(256)	Имя файла
status_id	numeric(9,0)	Id статуса карточки
file_store_url	character varying(512)	Адрес файлового хранилища

Описание полей таблицы **attribute_value** приведено в таблице (Таблица 30).

Таблица 30. Описание полей таблицы **attribute_value**

Название поля	Тип	Описание
attr_value_id	numeric(12,0)	Id значения атрибута
card_id	numeric(9,0)	Id карточки
attribute_code	character varying(20)	Код атрибута

Название поля	Тип	Описание
number_value	Numeric	Числовое значение атрибута + ссылка на карточку или персону
string_value	character varying(4000)	Строковое значение атрибута
date_value	timestamp without time zone	Дата/время
value_id	numeric(9,0)	Ссылка на справочник
another_value	character varying(256)	Прочее значение
long_binary_value	Bytea	Бинарное значение

Описание полей таблицы **card_version** приведено в таблице (Таблица 31).

Таблица 31. Описание полей таблицы card_version

Название поля	Тип	Описание
card_id	numeric(9,0)	Id карточки
version_id	numeric(9,0)	Id версии
version_date	timestamp without time zone	Дата версии
parent_card_id	numeric(9,0)	Id родительской карточки
status_id	numeric(9,0)	Id статуса карточки
file_storage	Bytea	Файловое хранилище
external_path	character varying(512)	Внешнее хранилище
file_name	character varying(256)	Название файла
file_store_url	character varying(512)	Адрес файлового хранилища

Описание полей таблицы **attribute_value_hist** приведено в таблице (Таблица 32).

Таблица 32. Описание полей таблицы attribute_value_hist

Название поля	Тип	Описание
card_id	numeric(9,0)	Id карточки
version_id	numeric(9,0)	Id версии
attribute_code	character varying(20)	Код атрибута
number_value	Numeric	Числовое значение атрибута
string_value	character varying(4000)	Строковое значение атрибута
date_value	timestamp without time zone	Дата/время
value_id	numeric(9,0)	Ссылка на справочное значение
another_value	character varying(256)	Прочее значение атрибута
long_binary_value	Bytea	Бинарное значение атрибута

Описание полей таблицы **action_log** приведено в таблице (Таблица 33).

Таблица 33. Описание полей таблицы action_log

Название поля	Тип	Описание
action_code	character varying(20)	Код действия
log_date	timestamp without time zone	Дата действия
actor_id	numeric(9,0)	Id пользователя, совершившего действие
ip_address	character varying(15)	Ip-адрес машины
card_id	numeric(9,0)	Id карточки
template_id	numeric(9,0)	Id шаблона
block_code	character varying(20)	Код блока атрибута
attribute_code	character varying(20)	Код атрибута
person_id	numeric(9,0)	Id пользователя, совершившего

Название поля	Тип	Описание
		действие

VII.8. Таблицы прав и состояний

- **card_access** – права на доступ к карточкам;
- **attribute_view_param** – права на видимость, редактируемость и обязательность атрибутов;
- **block_view_param** – состояния блоков в шаблонах по статусам;
- **tab_template_state** – состояние закладок в шаблонах.

Схема взаимодействия таблиц прав и состояний представлена на рисунке (Рисунок 25).

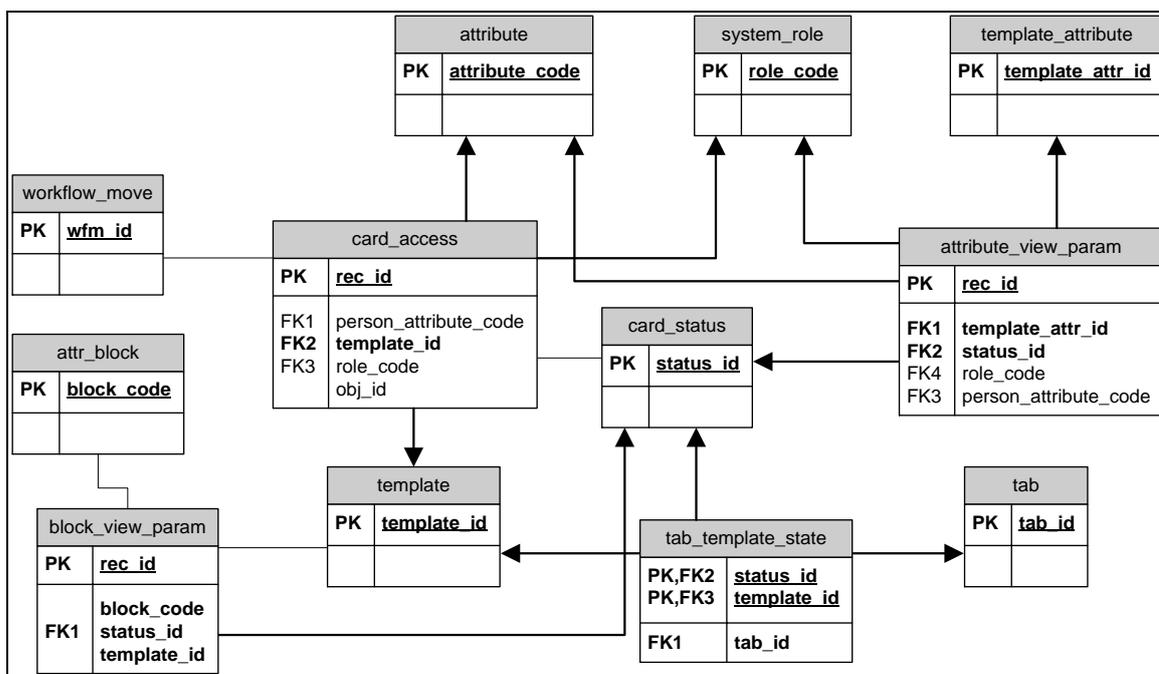


Рисунок 25 – Взаимодействие таблиц прав и состояний

Описание полей таблицы **card_access** приведено в таблице (Таблица 34).

Таблица 34. Описание полей таблицы **card_access**

Название поля	Тип	Описание
rec_id	numeric(9,0)	Идентификатор права доступа
permission_type	numeric(9,0)	Уровень доступа
object_id	numeric(9,0)	Ссылка на id статуса или id перехода
role_code	character varying (20)	Код роли
person_attribute_code	character varying (20)	Код персон-атрибута
template_id	numeric(9,0)	Идентификатор шаблона

Возможные уровни доступа:

- 1 – смена статуса;
- 2 – чтение карточки;
- 3 – редактирование карточки;
- 4 – создание карточки.

Описание полей таблицы **attribute_view_param** приведено в таблице (Таблица 35).

Таблица 35. Описание полей таблицы attribute_view_param

Название поля	Тип	Описание
rec_id	numeric(9,0)	Идентификатор правила
template_attr_id	numeric(9,0)	Идентификатор атрибута в шаблоне
status_id	numeric(9,0)	Идентификатор статуса
role_code	character varying (20)	Код роли
is_mandatory	numeric(1,0)	1 – атрибут обязательный
is_hidden	numeric(1,0)	1 – атрибут скрытый
is_readonly	numeric(1,0)	1 – атрибут только для чтения

Описание полей таблицы **access_template_rule** приведено в таблице (Таблица 36).

Таблица 36. Описание полей таблицы access_template_rule

Название поля	Тип	Описание
rule_id	numeric(9,0)	Идентификатор правила
operation_code	char (1)	Код операции (С - создание карточки)

Описание полей таблицы **block_view_param** приведено в таблице (Таблица 37).

Таблица 37. Описание полей таблицы block_view_param

Название поля	Тип	Описание
rec_id	numeric(9,0)	Идентификатор состояния
template_id	numeric(9,0)	Идентификатор шаблона
block_code	character varying (20)	Код блока
status_id	numeric(9,0)	Идентификатор статуса
state_block	Numeric	0 – блок свернут, 1 – свернут, если все атрибуты пустые, 2 – блок развернут

Описание полей таблицы **tab_tempalte_state** приведено в таблице (Таблица 38).

Таблица 38. Описание полей таблицы access_card_rule

Название поля	Тип	Описание
tab_id	numeric(9,0)	Идентификатор закладки
template_id	numeric(9,0)	Идентификатор шаблона
status_id	numeric(9,0)	Идентификатор статуса

VII.9. Прочие таблицы

Прочие используемые таблицы:

- **Installed_script** – история установленных скриптом (script_name, script_date);
- **Event_log** – история событий работы с карточками (action_log_id, action_code, log_date, actor_id, ip_address, card_id, template_id, block_code, attribute_code, person_id, is_success);
- **Event_log_detail** – подробности истории событий работы с карточками (event_detail_id, action_log_id, message text, description, code_message, type_message);
- **Delegation** – список делегирований (delegation_id, src_person_id, delegate_person_id, start_at, end_at);
- **Delegation_role** – список полномочий (delegation_role_id, name_rus, name_eng);
- **Scheduler_task** – список задач, запускаемых при старте сервера автоматически (task_id, task_module, interval, unit, date, info_ru, info_en, is_active, args_xml_data_id);
- **Person_department_access** – права пользователей на документы пользователей различных подразделений (только НПО) (person_id, dep_card_id, permission_type, template_id, person_attribute_code).

VIII. Структура кода СЭД «Логика СЭД. СПО»

VIII.1. Компоненты

- **База данных** – EnterpriseDB (PostgreSQL + Oracle syntax). Таблицы, sequences, триггеры для заполнения primary keys;
- **Business layer** – JBoss Application Server, EJB;
- **GUI** – JBoss Portal, portlets (JSR-168), AJAX.

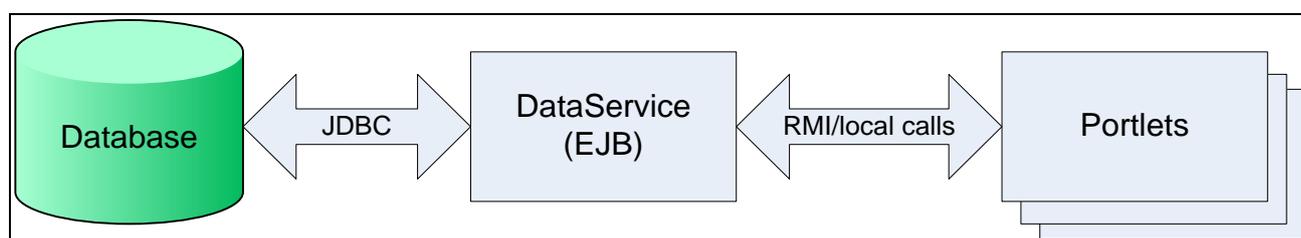


Рисунок 26 – Схема взаимодействия компонентов системы

VIII.2. Основные понятия

В основе системы лежит понятие карточки. Карточка – это набор некоторых характеристик (атрибутов), каждая из которых имеет (в данной карточке) одно или несколько значений. Конкретный набор характеристик, входящих в карточку, определяется шаблоном, по которому эта карточка создана. Связь карточки с шаблоном – постоянная. При изменении шаблона новые характеристики появляются (с пустым значением) во всех карточках, созданных по этому шаблону. При этом характеристики, удаленные из шаблона, остаются в карточке до ее первого сохранения.

Характеристики в текущей версии системы всегда существуют в рамках блоков характеристик. Шаблон карточки составляется из блоков, а не из отдельных характеристик. В каждом шаблоне должен обязательно присутствовать блок «Общие характеристики», который содержит ряд характеристик, необходимых для правильного функционирования системы. Характеристики могут перемещаться между блоками.

Для каждого шаблона определен свой жизненный цикл (workflow). Жизненный цикл – это граф, в котором узлами являются статусы карточек, а ребрами – переходы. Также в жизненном цикле определяется некий начальный статус. Этот статус получают все карточки, созданные по данному шаблону. Статус является еще одним свойством карточки (не характеристикой). Изменение статуса – отдельная операция в системе. Статус карточки может изменяться только в соответствии с жизненным циклом ее шаблона.

Важным принципом системы является «вечное» хранение (почти) всех объектов. Поэтому для них отсутствует операция «Удаление». Вместо этого системные объекты имеют флаг

«active», снятие которого указывает на то, что объект больше не должен использоваться для операций в системе (шаблон – для создания карточек, пользователь – для входа в систему и т.д.). Тем не менее, ссылки на этот объект могут оставаться в других объектах, сохраняя целостность исторических данных. Карточки же в конечном итоге попадают в такой статус, в котором они не включаются в обычные списки, но могут быть найдены отдельным запросом.

Система позволяет управлять правами пользователей. Для этого в ней определяется понятие роли. Для каждой роли определяется набор прав на выполнение операций над объектами (просмотр, изменение, создание) или действий. Специальная роль «Администратор» позволяет обладающему ей пользователю выполнять операции по изменению системных объектов (шаблон, характеристика, статус и т.д.). Остальные роли позволяют назначить права пользователей на операции с карточками (включая смену статуса) по отдельным шаблонам и статусам, а также права на работу с отдельными характеристиками этих карточек. Кроме статических ролей, в системе присутствуют «динамические» роли (этот термин отсутствует в коде). Динамическая роль пользователя определяется только по отношению к конкретной карточке. Пользователь имеет динамическую роль в карточке, если он упомянут в характеристике этой карточки.

VIII.3. Общая структура и взаимосвязи

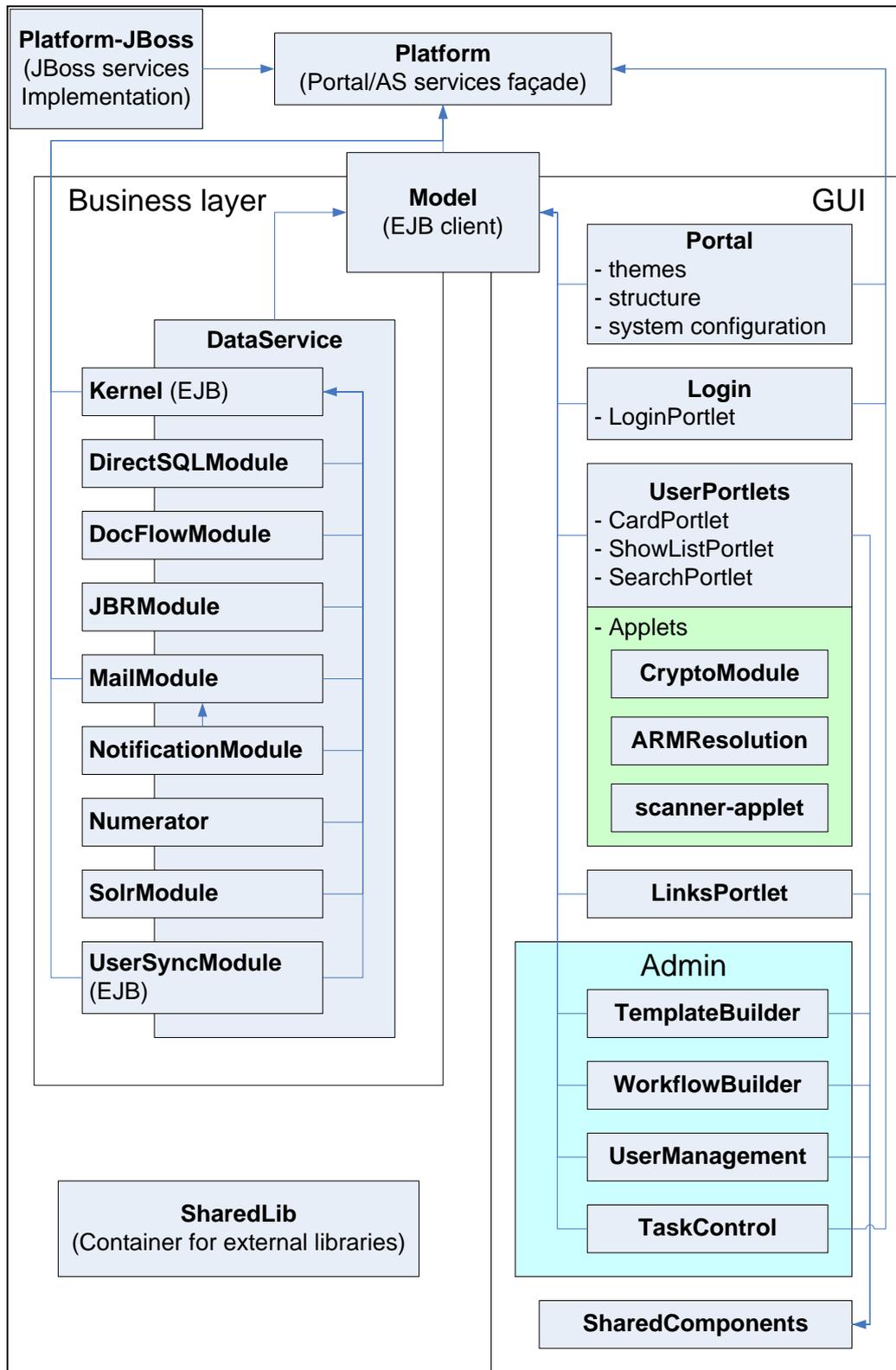


Рисунок 27 – Схема взаимосвязи проектов

Перечень проектов:

- *Platform* – объявление сервисов, реализация которых зависит от конкретной платформы (сервер приложений, портал).
- *Platform-JBoss* – реализация сервисов из проекта Platform на платформе JBoss Portal.
- *Model* – модель объектов системы, интерфейс EJB бизнес-слоя.
- *DataService* – контейнерный проект для enterprise приложения.
- *Kernel* – ядро бизнес-слоя. Содержит механизм диспетчеризации запросов со стороны клиентских приложений (GUI), реализацию основных операций с объектами (DAO), механизм подключения дополнительных модулей.
- *DirectSQLModule* – реализация функции прямых SQL-запросов со стороны клиентского кода.
- *DocFlowModule* – реализация универсальных процессов системы документооборота: подписание, согласование, рассмотрение.
- *JBRModule* – реализация специфических требований заказчика.
- *MailModule* – взаимодействие системы с почтовым сервером.
- *NotificationModule* – формирование и рассылка уведомлений об изменении карточек в системе, как периодических, так и событийных.
- *Numerator* – универсальный механизм присвоения регистрационных номеров документам.
- *SolrModule* – реализация полнотекстового поиска по документам в системе с использованием Apache Solr.
- *UserSyncModule* – синхронизация с каталогом пользователей JBoss Portal.
- *Portal* – описание структуры разделов и страниц системы, темы и стили оформления, конфигурационные файлы системы.
- *Login* – портлет входа в систему.
- *UserPortlets* – набор портлетов, составляющих основной интерфейс системы.
- *CryptoModule* – апплет, осуществляющий подписание файлов на стороне клиента при их добавлении в систему.
- *ARMResolution* – апплет, позволяющий добавлять графические и аудиорезолюции в документы.
- *Scanner-applet* – апплет потокового сканирования документов.

- *LinksPortlet* – простой портлет для отображения ссылок на различные компоненты системы.
- *TemplateBuilder* – набор портлетов для управления шаблонами карточек и атрибутами.
- *WorkflowBuilder* – портлет для управления жизненными циклами документов.
- *UserManagement* – портлет для управления правами доступа пользователей в системе.
- *TaskControl* – портлет для управления периодическими задачами в системе.
- *SharedComponents* – набор GUI-компонентов, используемых портлетами из различных проектов.
- *SharedLib* – контейнер для хранения сторонних библиотек, используемых в системе.

VIII.4. Объектная модель

Каждый тип объекта, хранящийся в системе, описан отдельным классом в package `com.aplana.dbmi.model` (проект Model). Эти классы используются для передачи данных между GUI и business layer. Все эти классы должны быть порождены от `com.aplana.dbmi.model.DataObject`.

Каждый объект, хранящийся в системе (БД), имеет собственный уникальный идентификатор. В Java-коде он представляется объектом `com.aplana.dbmi.model.ObjectId`, который состоит из ссылки на класс объекта и собственно идентификатора (хранящегося в БД). Такая форма идентификатора позволяет распознать тип объекта по его идентификатору, затрудняет смешивание идентификаторов различных объектов, а также скрывает конкретный тип ключа в БД (в настоящее время в различных таблицах используются как числовые, так и строковые ключевые поля). Зачастую в коде возникает необходимость ссылаться на конкретные объекты в БД. Чтобы не включать конкретные значения идентификаторов в код, класс `ObjectId` имеет статический метод `predefined()`, который позволяет использовать синонимы для реальных идентификаторов. Синонимы задаются в файле `Portal/conf/dbmi/objectids.properties`. Например, если в этом файле содержится строка:

```
stringattribute.jbr.uzdo.signature=ADMIN_67129
```

то в коде можно создать `ObjectId` для этого объекта следующим образом:

```
ObjectId id = ObjectId.predefined(StringAttribute.class, "jbr.uzdo.signature");
```

Важно заметить, что синоним объекта, передаваемый функции, не включает его тип.

Обработка различных настроечных xml-файлов в системе также построена с использованием этого метода, поэтому практически везде, где может быть задан идентификатор объекта, может также использоваться и его синоним.

В проекте Model также определяются действия, реализованные в системе (package `com.aplana.dbmi.action`). Каждое действие должно реализовывать интерфейс `com.aplana.dbmi.action.Action`. Наиболее важные действия описаны ниже.

Search. Ключевое действие системы – поиск карточек по заданным критериям. Поиск может работать в двух режимах: по содержимому характеристик (`setByAttributes(true)`) и по идентификаторам карточек (`setByCode(true)`). В первом режиме необходимо задать списки шаблонов (`setTemplates()`) и статусов (`setStatuses()`), по которым фильтруются карточки. Списки могут быть пустыми (но не `null`), в таком случае фильтрация по данному критерию не производится. Необходимо задать строку для поиска в строковых характеристиках (в т.ч. пустую). Можно ограничить набор характеристик, в которых будет осуществляться поиск заданной строки (`addStringAttribute()`). Можно задать ограничения на значения других характеристик карточки (`addXxxAttribute()`). Также в этом режиме поддерживается полнотекстовый поиск по файлам, вложенным в карточки (`setByMaterial(true)`).

В режиме поиска по идентификаторам в метод `setWords()` передается список чисел-идентификаторов карточек, разделенных запятыми. Также в этом режиме можно дополнительно фильтровать карточки по атрибутам, установив `setByAttributes(true)`, и задав фильтры аналогично первому режиму (кроме строковых характеристик).

Во всех режимах можно также указать желаемый набор возвращаемых характеристик, требуемую сортировку карточек и фильтр для постраничной выборки (`setFilter()`).

Объекты поиска могут создаваться не только из кода, но и из специальных xml-файлов. Пример такого файла:

```
<search byAttr="true">
  <name lang="ru">На подпись</name>

  <!-- Критерии поиска -->   <!-- Везде используются синонимы id -->
  <template>jbr.sign</template> <!-- может быть несколько тегов template -->
  <status>sign</status>     <!-- может быть несколько тегов status -->
  <attribute id="jbr.sign.person" type="user"> <!-- м.б. несколько -->
    <value var="current"/> <!-- спец. значение – текущий пользователь -->
  </attribute>

  <!-- Настройки формата вывода -->
  <column id="jbr.sign.comment" width="50" link="true"/>
  <column id="jbr.sign.parent" width="150" />
</search>
```

Результат выполнения действия `Search` выполняется в объекте `com.aplana.dbmi.action.SearchResult`. Карточки, возвращаемые в `SearchResult`, содержат не все характеристики, а только те, которые соответствуют столбцам, содержащимся в коллекции `getColumns()`. Поэтому перед выполнением операций по изменению этой карточки необходимо получить ее полную версию. Возвращаемый набор столбцов определяется следующим образом:

- если столбцы перечислены в объекте поиска (setColumns()), то столбцы возвращаются (если соответствующие характеристики существуют);
- если все возвращаемые карточки созданы по одному шаблону, то возвращаются характеристики, у которых указано «Отображать в результатах поиска» для данного шаблона (настраивается через административный интерфейс в свойствах шаблона);
- возвращается predetermined набор столбцов (hard-coded).

LockObject/UnlockObject. Блокировка/разблокировка объекта. Эти действия предназначены для предотвращения одновременного изменения объектов разными пользователями. Объект, заблокированный одним пользователем, не может быть изменен другим. Пользователь не может изменить не заблокированный им предварительно объект. Выполнение этих действий требует от пользователя прав на изменение данного объекта.

ChangeState. Осуществляет перевод карточки из одного состояния в другое. Должны быть указаны карточка (setCard()) и требуемый переход (setWorkflowMove()). Для выполнения этого действия карточка также должна быть заблокирована данным пользователем. По этому действию в системе выполняется большинство кода, выполняющего специфические функции документооборота: перевод в другие статусы связанных карточек, рассылка уведомлений и т.п.

CreateCard. Создает новую карточку по заданному шаблону (setTemplate()). Это действие не производит никаких изменений в базе данных, а конструирует объект карточки. Возвращенный объект будет иметь все характеристики, соответствующие запрошенному шаблону, заполненные значениями по умолчанию (или пустыми, если значения не заданы), правильный начальный статус. Идентификатор новой карточки равен null. Созданный объект пригоден для сохранения в системе через соответствующую операцию. Сразу после сохранения новая карточка будет заблокирована создавшим ее пользователем.

ListProject. Осуществляет поиск карточек, связанных с данной карточкой через некоторую характеристику. Возвращаемый данным действием результат такой же, как у действия Search.

VIII.5. Графический интерфейс пользователя (GUI)

Система реализована в виде трех «порталов» (в терминологии JBoss Portal):

- Основной – dbmi (<http://localhost:8080/portal/auth/portal/dbmi>);
- Рабочее место руководителя – boss (<http://localhost:8080/portal/auth/portal/boss>);
- Административный – dbmi-admin (<http://localhost:8080/portal/auth/portal/dbmi-admin>).

Ниже приведены скриншоты двух основных страниц системы: список карточек (с поиском) (Рисунок 28) и просмотр/редактирование карточки (Рисунок 29).

На этих скриншотах показаны три портлета, составляющих основу пользовательского интерфейса системы. Все эти портлеты расположены в проекте UserPortlets и взаимодействуют друг с другом.

СИСТЕМА ДОКУМЕНТООБОРОТА
Portal navigation Админов Админ Выход

Входящие
Исходящие
Внутренние
ОРДОГ
Ответы на ОГ ИЗ
Ответы на ИЗ
Внешнее исполнение
Личный кабинет
Кабинет помощника
Отчеты
Справочники

По номеру
По номеру исходящего
По типу контроля
По отправителю
По адресату
По статусу
Черновики

SearchPortlet

По номеру
 По характеристикам
 Полнотекстовый
 Расширенный поиск

ShowListPortlet

Входящие по регистрационному номеру
[Обновить список](#)

		Регистрационный номер	Дата регистрации	Номер исходящего	Дата документа	Краткое содержание (наименование текста)	Отправитель	Статус	
		654	2010-08-25	исх-010	2010-08-23	Отчет в печатной форме	ЗАО Телекоммуникационный центр Останкино Москва(ЗА...	Готов к списанию в дело	
		699	2010-09-01	1423-АО	2010-09-01	Проверка шага 2 в АРМе	ЗАО "АйТи"(ЗАО "АйТи")	Рассмотрение	
		711	2010-09-13	524586	2010-09-13	проверка переходов с ошибкой "Изменение статуса карточки ..."		Рассмотрение	
		722	2010-09-17	72547	2010-09-17	будем тестировать поручения второго уровня	"Bentley", Москва("Bentley", Москва)	Исполнение	
		723	2010-09-21	37465	2010-09-21		ЗАО АйТи Информационные технологии, Москва(ЗАО АЙТ...	Исполнение	
		724	2010-09-22	465865	2010-09-22		"Bentley", Москва("Bentley", Москва)	Исполнение	

Рисунок 28 – Список карточек (с поиском)



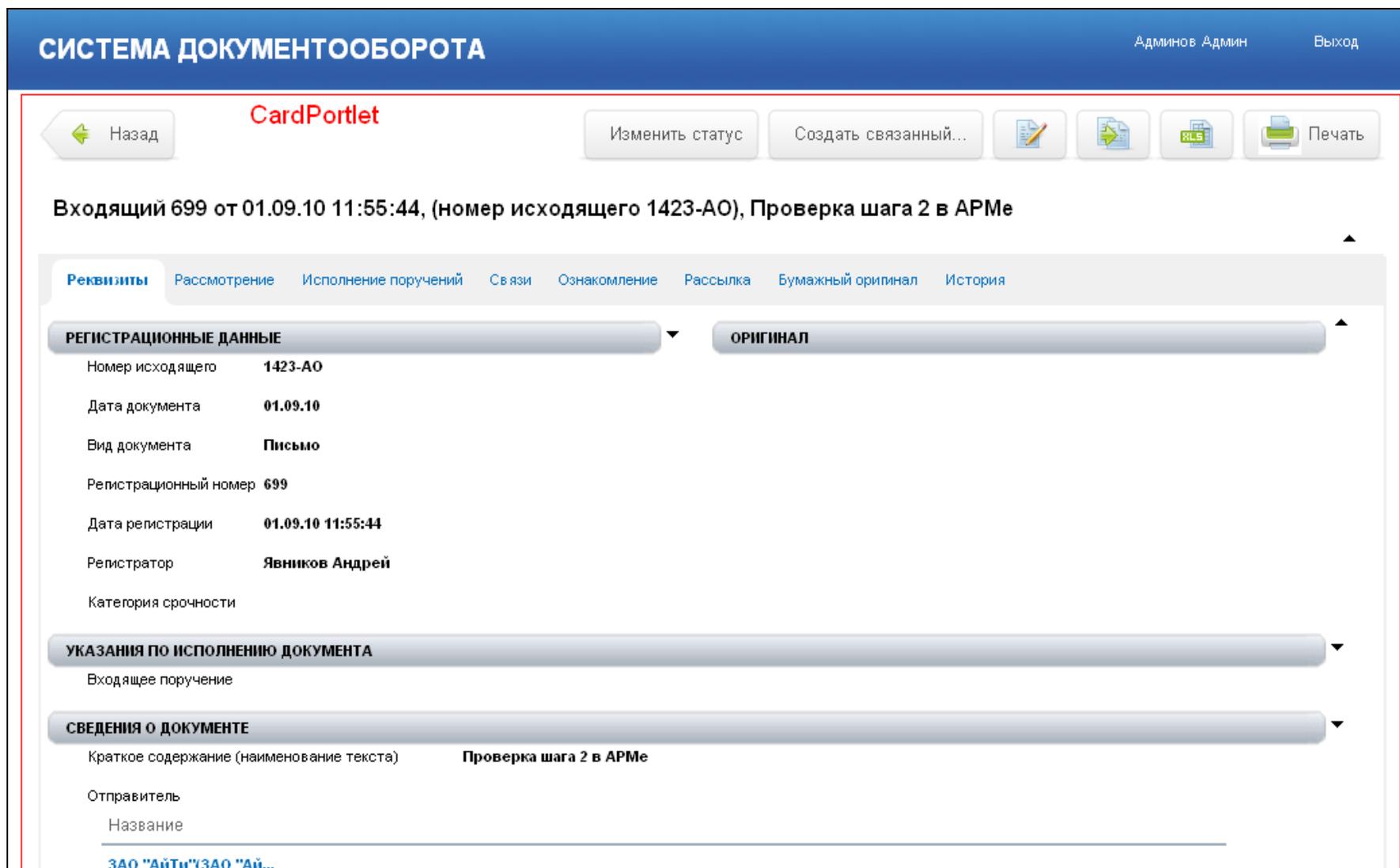


Рисунок 29 – Просмотр карточки

SearchPortlet. Формирует запросы на поиск карточек в системе и передает их ShowListPortlet, расположенному на той же странице (через application scope переменную). Имеет два режима: простой и расширенный. Первый из них показан на скриншоте выше (Рисунок 28). Во втором позволяет выбрать шаблоны карточек, по которым ведется поиск, и наложить ограничения (фильтры) на различные характеристики.

ShowListPortlet. Обращается к business layer системы для выполнения поиска карточек и отображает результаты поиска в виде таблицы. Может получать поисковые запросы:

- от SearchPortlet, расположенного на той же странице - через application scope переменную;
- из настроек портлета;
- из параметров страницы портала.

Портлет поддерживает сортировку таблицы по различным столбцам, разбиение на страницы, экспорт в csv-файл и печать таблицы. Для каждой строки таблицы формируется ссылка, позволяющая перейти на отдельную страницу портала для просмотра или редактирования данной карточки. Кроме того, имеется кнопка (ссылка), позволяющая создать новую карточку.

CardPortlet. Портлет реализует функции просмотра, создания и редактирования отдельных карточек. Располагается на отдельной странице портала вне видимой пользователю иерархии. Требуется как минимум один параметр через URL: номер карточки для просмотра/редактирования, или указание на режим создания карточки.

В режиме создания карточки портлет предлагает выбрать пользователю один из шаблонов. Этот этап может быть пропущен, если портлету через URL будет передан специальный параметр, указывающий идентификатор шаблона. Затем портлет запрашивает у business layer формирование новой карточки по заданному шаблону и переходит в режим редактирования карточки.

В режимах просмотра и редактирования карточки портлет запрашивает карточку через business layer, а в режиме редактирования производит ее блокировку для данного пользователя. После этого формируется страница, содержащая все характеристики карточки, каждая из которых отображается через соответствующий класс AttributeEditor. Настройка редакторов для характеристик производится в файле Portal/conf/dbmi/card/editors.xml. Каждый AttributeEditor работает с одним определенным типом характеристики. Для одного и того же типа характеристики могут существовать несколько классов AttributeEditor'ов. Пример фрагмента файла editors.xml:

```
<attributes package="com.aplana.dbmi.card">
  <select property="type">
    <case value="S">
```

```
<editor class="StringAttributeEditor"/>
<select property="id">
  <case value="jbr.uzdo.signature"> <!-- синоним id, см. описание выше -->
    <editor class="SignatureAttributeEditor"/>
    <viewer class="JspAttributeViewer">
      <parameter name="jsp">/WEB-INF/jsp/html/attr/SignatureView.jsp</parameter>
    </viewer>
  </case>
</select>
</case>
<!-- ... -->
</select>
</attributes>
```

Отображается дерево выбора контроля для конкретной характеристики. Выбор основывается на свойствах (bean properties) объекта этой характеристики. Самый внешний select осуществляет выбор по типу характеристики (attribute.getType()), внутри для выбора чаще используется ее идентификатор (attribute.getId()), но могут использоваться и другие properties характеристики (см. проект Model, класс com.aplana.dbmi.model.Attribute и его потомков). Внутри тегов case могут использоваться теги editor и viewer. Editor используется в режиме редактирования карточки, viewer – в режиме просмотра, а также в режиме редактирования для read-only характеристик. Портлет использует для характеристики наиболее подходящий editor/viewer, спускаясь по дереву разбора:

- Если ни один case из некоторого select не подходит для данной характеристики, используется контроль, объявленный на одном уровне с самим select (или выше, если его нет).
- Если не найден ни один подходящий editor для характеристики, то используется подходящий viewer; если же не найден viewer, то используется предопределенный viewer, отображающий строковое представление значения характеристики (attribute.getStringValue()).

Классы, реализующие контролы, должны реализовать интерфейс com.aplana.dbmi.card.AttributeEditor. Если же класс реализует также интерфейс com.aplana.dbmi.card.Parametrized, то через файл editors.xml для него могут быть заданы параметры (см. JspAttributeViewer в примере выше). Количество параметров, их имена, смысл, обязательность определяется классом контроля.

VIII.6. Навигация

В системе используются возможности JBoss Portal по организации структуры портала и навигации. В файлах Portal/WebContent/WEB-INF/*-portal-object.xml определяются разделы и страницы, из которых состоит система. Для каждой страницы определяется набор портлетов, а также свойств (properties), позволяющие организовывать различные представления данных. Главное из них – defaultSearch. Оно задает путь к xml-файлу поиска (относительно Portal/conf/dbmi). Из этого файла конструируется объект поиска, который вызывается при обращении пользователя к данной странице.

Для отображения навигации в общей стилистике системы используется собственная jsp-страница, подменяющая стандартную страницу навигации JBoss Portal. Она находится здесь: Portal/WebContent/WEB-INF/header/tabs.jsp. Для ее использования необходимо при установке системы модифицировать настройки JBoss Portal.

Business layer

Business layer реализован как единое J2EE приложение, основным внешним интерфейсом которого является stateless session Enterprise Java Bean (EJB). Интерфейс EJB описан в классе com.aplana.dbmi.service.DataService (проект Model) (предпочтителен для использования класс com.aplana.dbmi.service.DataServiceBean). EJB содержит минималистический набор операций, позволяющий считывать (метод getByld) и сохранять (метод saveObject) объекты в системе, запрашивать перечни объектов определенного типа (методы listAll, listChildren и filter), выполнять иные действия, не вписывающиеся в данную парадигму (метод doAction), а также определять доступность тех или иных операций конкретному пользователю (методы canCreate, canChange и canDo). Классы объектов, которые могут передаваться для выполнения операций, также описаны в проекте Model.

Реализация EJB находится в проекте Kernel. Класс com.aplana.dbmi.service.impl.DataServiceBean является диспетчером запросов, он с помощью com.aplana.dbmi.service.impl.QueryFactory создает обработчик конкретного запроса и вызывает его, передавая всю необходимую информацию. QueryFactory, конструирует обработчики конкретных запросов, опираясь на информацию из файлов queries.xml. Такой файл может присутствовать в каждом модуле приложения, от также размещен в проекте Kernel: conf/queries.xml. Пример фрагмента файла:

```
<queries>
  <object type="Card">
    <store><!-- Описание обработки сохранения объекта типа Card -->
      <access>CardModify</access>    <!-- extends AccessCheckerBase -->
      <pre-process>ValidateCard</query> <!-- extends ProcessorBase -->
      <query>SaveCard</query>        <!-- extends SaveQueryBase -->
      <specific property="template.id" value="33">
        <!-- По-особому обрабатываем карточки, у которых
```

```

        getTemplate().getId() == 33 -->
        <access>AdminAction</access> <!-- заменяет CardModify -->
        <pre-process>CalculateSomeField</pre-process>
        <!-- вызывается и ValidateCard, и CalculateSomeField -->
        <post-process>NotifyAdmins</post-process>
    </specific>
</store>
</object>
<!-- ... -->
</queries>

```

Имена классов в этом файле указываются без package. При этом используются package по умолчанию (разные для различных тегов, см. QueryFactory). Можно задать конкретный package для конкретного класса:

```
<access package="com.aplana.dbmi.mymodule">MyCardModify</access>
```

либо изменить значение package по умолчанию в корневом теге:

```
<queries access-package="com.aplana.dbmi.mymodule">
```

Обработчики, как и редакторы атрибутов, могут поддерживать интерфейс `com.aplana.dbmi.service.impl.Parametrized`, и получать параметры, заданные в `queries.xml`. В таком случае имя класса обработчика следует писать не внутри тега, а в атрибуте `class`:

```

<post-process class="CardNotification">
    <parameter name="beanName" value="notifyApproval"/>
</post-process>

```

Обработка запроса происходит следующим образом:

- Если для данного запроса прописан `access`, он вызывается для проверки прав пользователя на выполнение данного запроса. Если прав недостаточно, выбрасывается исключение.
- Вызываются все препроцессоры (`pre-processor`), найденные для данного запроса. Они предназначены для подготовки объекта к операции или изменения действия. При этом может выполняться валидация объекта, заполнение вычисляемых полей и т.д.
- Вызывается основной обработчик действия – `query`. Он производит необходимые действия с БД.
- Вызываются все постпроцессоры (`post-processor`), найденные для данного запроса. Они могут рассылать уведомления о данной операции, вызывать изменение других объектов

и т.п. Кроме того, постпроцессоры могут тем или иным образом модифицировать результат операции, возвращаемый пользователю.

Вся эта обработка происходит в единой транзакции. При возникновении исключения на любом этапе операция откатывается, клиенту возвращается ошибка.

Приложение DataService может включать в себя дополнительные модули. Для этого достаточно включить jar-файл в приложении и прописать его в application.xml. Если в этом модуле будет файл queries.xml, то он будет автоматически «подхвачен» QueryFactory и использован для построения обработчиков запросов. Существенная часть функциональности системы реализована именно в отдельных подключаемых модулях.

IX. Приложение А. Описание Протоколов

SOAP (Simple Object Access Protocol - простой протокол доступа к объектам) - протокол обмена структурированными сообщениями в распределенной вычислительной среде. Протокол используется для обмена произвольными сообщениями в формате XML и для удаленного вызова процедур. SOAP является расширением протокола XML-RPC.

SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, HTTPS и др. Чаще всего SOAP используется поверх HTTP. SOAP является одним из стандартов, на которых базируются технологии веб-служб.

AJP (The Apache JServ Protocol) - это бинарный протокол, который может проводить входящие запросы с веб-сервера до сервера приложений, который находится позади веб-сервера. Также он поддерживает некоторое наблюдение за состоянием сервера, в том числе позволяет проводить ping сервера приложений. AJP обычно используется в системах со сбалансированной нагрузкой, где один или более front-end рассылают запросы в один или более серверов приложений. Сессии направляются к нужному серверу приложений, используя механизм роутинга, где каждый сервер приложений получает свое имя.

JMX (The Java Management Extensions) - стандартизированный программный интерфейс (API), предназначенный для мониторинга и управления ресурсами, устройствами, сервисами и виртуальными машинами Java. Технология JMX разработана согласно спецификации JSR-003 и входит в J2SE начиная с версии 1.5. JMX позволяет поставщику реализовывать такие функции, как вывод конфигурационных параметров, а также дает возможность пользователям редактировать параметры. Также в JMX входит уровень уведомлений, который может использоваться приложениями, осуществляющими управление, для слежения за событиями, такими как запуск сервера приложений.

TSP (Time Stamp Protocol, Протокол Штампа Времени) - это криптографический протокол, позволяющий создавать доказательство факта существования электронного документа на определенный момент времени. Штамп времени (time-stamp) - это подписанный ЭП документ, центр штампов времени удостоверяет, что в указанный момент времени ему было предоставлено значение хэш-функции того документа, факт существования которого необходимо подтвердить. Само значение хэш-функции также указывается в штампе.

Значение хэш-функции от документа, на который получен штамп времени, служит для связи штампа с документом. При включении в штамп времени не самого документа, а значения хэш-функции, сохраняется конфиденциальность документа перед центром штампов времени.

UNO (Universal Network Objects) - компонентная модель, основанная на интерфейсах, применяемая в OpenOffice.Org. UNO обеспечивает взаимодействие объектов, созданных с применением различных технологий (OLE/COM, CLI, Web) и языков программирования (Python, C,

C++, Java, Basic), функционирующих на одном или нескольких компьютерах Internet- или Intranet-сети.

SMTP (Simple Mail Transfer Protocol - простой протокол передачи почты) - сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP. Описан в документах RFC 821 (1982 год); последнее обновление в RFC 5321 (2008).

XML (Extensible Markup Language - расширяемый язык разметки) - рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком.

XML является подмножеством SGML.

TCP/IP (Transmission Control Protocol/Internet Protocol - протокол управления передачей/Протокол Internet) - набор сетевых протоколов передачи данных, используемых в сетях, включая сеть Интернет. Протоколы работают друг с другом в стеке - это означает, что протокол, располагающийся на уровне выше, работает «поверх» нижнего, используя механизмы инкапсуляции. Например, протокол TCP работает поверх протокола IP.