

Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Иркутский государственный университет»  
(ФГБОУ ВО «ИГУ»)

Институт математики и информационных технологий  
Кафедра вычислительной математики и оптимизации

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
БАКАЛАВРА**

по направлению 01.03.02 Прикладная математика и информатика  
профиль «Математические методы и информационные технологии»

**РАЗРАБОТКА И РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ СИСТЕМЫ  
ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА «ТЕЗИС» НА ПЛАТФОРМЕ  
CUVA ДЛЯ АДМИНИСТРАЦИИ Г. ЕКАТЕРИНБУРГА**

Студента 4 курса очного отделения  
группы 02421-ДБ  
Подрядчикова Владимира  
Валерьевича

Руководитель:  
канд. физ.-мат. наук, доц.  
\_\_\_\_\_ Захарченко В. С.

Консультант:  
инженер-аналитик ООО «Сибирский  
центр информационных технологий»  
\_\_\_\_\_ Левченко К. О.

Допущена к защите  
Зав. кафедрой,  
д-р физ.-мат. наук, проф.  
\_\_\_\_\_ Аргучинцев А. В.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
ГЛАВА 1. ОБЗОР ПЛАТФОРМЫ .....	7
1.1 Cuba – платформа для создания информационных систем .....	7
1.2 Ключевые особенности и технологии .....	10
1.4 Уровни и модули приложения .....	12
1.5 Основные понятия. Плагин Cuba Studio .....	14
1.6 Базовые процессы в СЭД ТЕЗИС .....	18
ГЛАВА 2. ТРЕБОВАНИЯ К МОДИФИКАЦИЯМ И ИХ ОПИСАНИЕ .....	22
2.1 Создание и настройка документа муниципальных услуг .....	22
2.2 Настройка поиска штрихкода при сканировании документов с помощью плагина ТЕЗИС Помощник .....	27
2.3 Группа периодических поручений .....	29
ГЛАВА 3. ПРОГРАММНЫЙ КОД ДОРАБОТОК .....	38
3.1 Документ муниципальных услуг .....	38
3.1.1 Создание сущностей .....	38
3.1.2 Добавление в меню «Документ» на главном экране вкладок «Список МУ», «Создать МУ» .....	41
3.1.3 Автозаполнение поля «Исполнитель» в экране редактирования документа .....	42
3.1.4 Настройка локализации .....	43
3.1.5 Работа с просроченным документом .....	45
3.2 Считывание штрихкода .....	49
3.2.1 Имеющийся функционал считывания изображения .....	49
3.2.2 Поиск подизображения со штрих-кодом .....	50
3.3 Группа периодических поручений .....	51
3.3.1 Кратко о сущностях .....	51
3.3.2 Отмена периодических поручений в группе .....	51
ЗАКЛЮЧЕНИЕ .....	60
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	61
ПРИЛОЖЕНИЯ .....	63
Приложение 1 .....	63
Приложение 2 .....	65

## **ВВЕДЕНИЕ**

В настоящее время существует огромное количество предприятий самого разного предназначения. Однако все их часто объединяет работа с большим объёмом бумаг. Заявления, договоры, декларации, накладные, ордера, лицензии, сертификаты, - всё это виды документации, что каждый день передаются из рук в руки в процессе жизненного цикла предприятия. Организация системы, в которой передача информации, работа с документами будет проводиться максимально быстро и точно, - одна из основных проблем многих учреждений, так как от выполнения этой цели напрямую зависит эффективность предприятия. Каждый неразумно потраченный час лишает потенциальной прибыли, уменьшает качество и доступность оказываемых услуг.

До сих пор существуют предприятия, процесс ведения документов на которых не является автоматизированным, то есть происходит непосредственно с физическими носителями: бумажными документами, книгами, справками и тому подобным. Согласно данным исследования, проведенного компанией ELMA в августе-сентябре 2023 года, менее 3% крупных российских компаний ведут полностью в бумажном варианте. Оставшиеся 96% компаний уже находятся на пути к полной цифровизации документов и процессов, однако далеко не все процессы на предприятии подвержены автоматизации. Несмотря на то, что около 55% опрошенных компаний так или иначе перевели в цифровой вид делопроизводство и архивное хранение, количество тех, кто смог собрать в рамках одного архива бумажные и электронные документы, не так велико — всего 18%. [5].

Работа с бумажными носителями неудобна из-за траты времени на перемещение документов непосредственно сотрудниками, выделения помещений под архивы для хранения бумаг, траты средств на бумагу и обеспечение условий хранения. Работа ощутимо замедляется, так как сотрудникам самим приходится искать нужны бумаги, ждать почту или факс из другого отделения, искать подходящего работника для постановки печати, заверения или утверждения документов. Также такой подход порождает

уязвимости: важный документ может быть потерян или испорчен сотрудником, есть риск угрозы неавторизованного доступа постороннего к конфиденциальным данным и подделка документов. К тому же процесс работы и случаи ошибок сложно отследить.

Тяжело эффективно организовать и согласовать бизнес-процессы на предприятии, где документооборот осуществляется без специально сконструированной системы, в которой бы каждый сотрудник понимал структуру процесса и мог следить за продвижением работы, а также движением документов по предприятию. Более того, вышеупомянутые риски нельзя решить простым переносом данных на компьютер. Организация хранения документов должна контролироваться специально сконструированной для этого системой, отвечающей стандартам работы с документами внутри организации

Для решения этой проблемы существуют системы электронного документа оборота (СЭД). СЭД – это автоматизированная многопользовательская система, сопровождающая процесс управления работой организации с целью обеспечения выполнения этой организацией своих функций. Её основные возможности включают в себя [1]:

1. Хранение всех документов в базе данных
2. Разграничение доступа сотрудников в зависимости от их должности и выполняемой задачи
3. Создание удобной коммуникации между сотрудниками
4. Поиск нужных данных в большом объеме информации
5. Отслеживание пути проведения документов

Система электронного документооборота строго регулирует размещение документов, устанавливая четкие нормы их хранения. Документы находят свое место в выделенном хранилище, где проводится тщательный учет и каталогизация. Значительным аспектом функционала СЭД является система управления доступом к документам, настраиваемая администратором с учетом индивидуальных правил в системе [1].

Сохранение истории изменения документов обеспечивает прозрачность, тем самым снижает риск подделки документов. Администратор всегда может выяснить, какой сотрудник внёс изменения, когда и откуда документ подвергся редактированию. К тому же СЭД располагает специальными методами для безопасного необратимого процесса удаления.

За многие годы работы предприятия система позволяет формировать архив. Современные вычислительные системы могут хранить большой объем данных с минимальными затратами. Работа с архивом во многом производится автоматически, самой системой. По истечении срока хранения данные переносятся в архив, где все документы расположены в надлежащих каталогах [1].

Одной из ключевой задач системы является управления процессами проведения документов. Обычно в СЭД присутствуют такие стандартные операции как постановка задачи для сотрудника, назначение ответственных за задачу, передачу данных между сотрудниками, контроль выполнения и установка сроков, создание условий для параллельной и последовательной работы.

Из дополнительных функций можно отметить напоминания о сроках сдачи заданий, просроченных поручений, уведомления о задержке документов и срочных задачах. Такие уведомления могут отсылаться по электронной почте, приходить в виде SMS-сообщений. Имея полную картину, сотрудник сможет оптимизировать свой процесс работы, расставив приоритеты по заданиям, а также минимизировать ошибки из-за собственной невнимательности или непунктуальности. А руководство сможет отследить прогресс сотрудника в случае больших задержек и отставаний от графика.

Поиск по электронному архиву имеет ряд значительных преимуществ в сравнении с бумажным аналогом. Обычно в бумажном архиве документы содержат два-три признака: номер и название документа, дата поступления. Современные СЭД могут обеспечивать широкий спектр возможностей поиска документов. У каждого документа может любое количество атрибутов, по

которым можно сортировать большие объемы данных. А полнотекстовый поиск добавляет возможность искать документы по своему содержанию, что существенно повышает эффективность поиска по сравнению с обыкновенным поиском по бумажному архиву [1].

Актуальность данной темы объясняется тем, что на многих российских предприятиях до сих пор отсутствует автоматизированный документооборот, либо же всё еще находится на стадии внедрения и некоторые рабочие процессы осуществляются с помощью бумажных носителей. В 2019 году в рамках программы «Цифровая экономика Российской Федерации» стратегия перехода к цифровой среде в различных сферах, включая бизнес и государственные службы, становится одной из основных [7] [3]. Были созданы проекты, направленные на регулирование цифровой среды, создание информационной инфраструктуры.

Объектом исследования будут принципы разработки систем электронного документооборота на платформе, предназначенной для создания корпоративных приложений, основные элементы такой системы, а также задачи, с которыми сталкиваются программисты в ходе её создания.

Предметом исследования является процесс разработки СЭД ТЕЗИС на платформе CUBA для администрации города Екатеринбург в рамках деятельности Сибирского центра информационных технологий.

Целью работы является получение практического опыта в создании основных элементов системы и разработке решений для оптимизации и повышения эффективности бизнес-процессов.

Для достижения цели будут выполнены следующие задачи:

- Изучение архитектуры и технологий платформы
- Описание доработок, необходимых для функционирования системы по стандарту организации
- Программная реализация

## **ГЛАВА 1. ОБЗОР ПЛАТФОРМЫ**

### **1.1 Cuba – платформа для создания информационных систем**

Нужда в цифровой трансформации существует уже давно. За последние 15 лет было создано не мало конфигураций, сред программирования, фреймворков, которые нацелены на работу с созданием проектов, отвечающим всевозможным стандартам ведения производства и оказания услуг.

Прежде чем начать работу над проектом, нужно определить технологическую базу, то есть платформу для создания нашего проекта. Сроки, стоимость автоматизации, запланированные задачи платформы, условия труда, - всё это может напрямую повлиять на успех всего процесса, поэтому необходим правильный выбор инструментов и технологий. Выбирая инструмент, нужно обратить внимание на такие критерии как скорость реализации проекта, его продуктивность, стоимость разработки и поддержания работоспособности, мобильность и гибкость применения.

В Сибирском центре информационных технологий работа с СЭД происходит на Cuba Platform. Платформа написана на языке Java и располагает огромным инструментарием для создания корпоративных проектов. Одна из идей платформы состоит в комбинировании java-технологий с инструментами ускоренной разработки. Использование доступных средств не ограничено лицензией, то есть весь код приложения находится под контролем владельца проекта, а затраты на обновление и поддержку ПО снижаются.

Cuba Platform является продуктом, созданным компанией Haulmont Technology. Haulmont Technology — это ведущий разработчик программных продуктов в сфере корпоративных информационных систем. Платформа была представлена в 2011 году, и с тех пор получила признание в мире разработки бизнес-приложений благодаря своей гибкости, масштабируемости и сильному фокусу на потребностях предприятий. Сегодня Haulmont Technology является ключевым игроком в области корпоративных информационных технологий.

Платформа подходит для создания таких приложений как:

- Бизнес-приложения
- Управленческие и информационные системы
- Системы планирования ресурсов предприятия (ERP) — ПО, созданное для оптимизации и автоматизации бизнес-процессов на предприятии. Все бизнес-процессы координируются друг с другом, создавая единый достоверный источник данных
- Разработка программно-аппаратной составляющей web и мобильных приложений

Но не подходит для:

- Веб-сайты
- Проекты, не имеющие ограничения по количеству одновременно работающих пользователей.
- Проекты, не требующие стандартной функциональности такой, как ролевая модель, разграничение доступа и т. п.

Набор инструментов платформы Cuba открывает возможности для эффективного создания проектов, избегая трудоемкого старта с нуля. Это позволяет быстро разрабатывать прототипы, обладающие базовой функциональностью. Процессы быстрого конфигурирования, редактирования моделей данных, генерации скриптов и кода, а также наличие справочных материалов сопровождают программиста на каждом этапе работы.

Важно отметить, что платформа не ограничивается своим инструментарием. В среде разработки предусмотрена библиотека аддонов (add-ons), которые легко интегрируются в проект, расширяя его функциональность. Библиотека, известная как Cuba Marketplace, предоставляет разнообразные решения, широко применяемые на предприятиях различных отраслей.

Сфера применения платформы Cuba охватывает области такие, как работа с финансами, бухгалтерия, управление персоналом. Она демонстрирует свою



эффективность и адаптируемость в разнообразных бизнес-сценариях, что делает ее востребованной на рынке корпоративных решений.

Гибкость платформы проявляется также в ее способности интегрироваться с различными сторонними системами. Благодаря интеграционной готовности, разработчики могут легко внедрять решения, адаптированные к конкретным потребностям заказчика, и обеспечивать непрерывный обмен данными с другими корпоративными системами. К тому же элементом платформы является ее открытость для кастомизации. Разработчики имеют возможность создавать собственные модули и аддоны, настраивать существующие компоненты под уникальные требования проекта.

Open-source решение платформы позволяет заказчику не зависеть компании, создававшей приложение. Платформа распространяется под Apache 2.0 лицензией, исходный код доступен на GitHub. Лицензионных ограничений на проект не накладывается. У Cuda существует бесплатная и коммерческая версии. Бесплатная Cuda пригодна для использования, но имеет ограниченные размеры моделей данных. В неё также отсутствуют некоторые инструменты для удобства программирования: генераторы отчетов, кода, диаграмм, поиск по тексту. Стоимость лицензии на данный момент на одного человека около 20 тысяч рублей за год пользования. Любой пользователь может начать ознакомление с платформой и в дальнейшем, если понадобится, перейти на лицензию. Для заказчиков проекта тоже есть плюсы. Если после внедрения системы возникнут какие-либо вопросы или требования, с ними можно обратиться в другую компанию, изначально не создававшую приложение. Открытый код приложения и подробная документация открывают возможности для обучения персонала, к тому же заказчик не имеет проблем с наймом специалистов.

## 1.2 Ключевые особенности и технологии

Подробно рассмотрим ключевые особенности и модули платформы. Словарь использованных сокращений и терминов будет приведен в приложении 1.

**Поддержка Java 8, 11.** Платформа работает не на самой последней версии java, но всё еще имеет доступ к последним языковым возможностям API, а также имеет совместимость с современными библиотеками и фреймворками. Версии 8 и 11 широко распространены и стабильны, автоматически получают обновления для укрепления защиты от потенциальных угроз и уязвимости.

**Поддержка распространенных браузеров: Google Chrome, Opera, Mozilla Firefox, Safari, Microsoft Edge.** Poly В проект на Cuba Platform полностью интегрирована система сборки библиотеки Polymer, что открывает возможности создания front-end порталов с веб-интерфейсом.

**Работа на серверах с Java EE Web Profile, Apache Tomcat.** Java EE - спецификация для разработки многопоточных приложений. Представляет набор API и сервисов для создания корпоративных приложений. Web profile является её профилем с более узконаправленным интерфейсом, направленным на создание web-приложений. Этот профиль обычно предназначен для легких и средних по сложности приложений. Включает в себя базовые сервлеты, JSP, JSF, JPA и другие технологии. Apache Tomcat - сервер приложений с открытым исходным кодом, который реализует спецификацию Java Servlet, JSP и JEL. Предназначен для развертывания приложения.

**Базы данных: Oracle, MS SQL, PostgreSQL.** Выбор между популярными базами данных позволяет подобрать подходящую БД под конкретный проект. К тому же есть возможность использования разных баз данных в одном проекте.

**Хранилище Amazon S3 File Storage.** Обеспечение интеграции с Amazon Simple Storage Service позволяет хранить файлы в облачном хранилище Amazon Web Services. Amazon S3 является высокомасштабируемым, надежным облачным хранилищем, обеспечивающим высокую доступность данных и защиту от потери информации. Поддерживаются функции разграничения

доступа, резервного копирования и восстановления данных, а система тарификации позволяет оптимизировать затраты на хранение данных в зависимости от их объема и типа.

**Поддержание работы в случае перебоев и отказов. Кластерная система серверов.** Приложения могут продолжать стабильно работать даже при отказе некоторых серверов. Кластеризация позволяет объединить несколько серверов в единую группу, работающую как единое целое, то есть несколько экземпляров приложения развертывается на различных серверах. При отказе одного или нескольких узлов в кластере другие узлы продолжают обслуживать запросы пользователей без прекращения работы приложения. Для равномерного распределения запросов применяется механизм балансировки нагрузки и репликация данных. Режим кластера также включает в себя инструменты для мониторинга узлов, обнаружение сбоев и автоматического восстановления системы. Платформа предоставляет административные панели для управления кластером, отслеживания производительности и просмотра журнала событий.

Платформа располагает широким спектром технологий:

1. **Vaadin** – основной фреймворк для создания универсального пользовательского интерфейса. Предоставляет набор компонентов для построения веб-приложений: кнопки, таблицы, формы и т.д. Клиентский уровень создан на его основе по следующим принципам [7]:
  - в виртуальной машине java формируется дерево визуальных компонентов;
  - информация о состоянии пользовательского интерфейса передается в браузер в формате JSON;
  - на основе JSON создается DOM, обеспечивающий представление документа в виде структуры со своими свойствами и методами, который связывает веб-страницы со скриптами;
  - визуальные компоненты берутся из библиотеки Google Web Toolkit;

- любые действия пользователя (заполнение полей, нажатие кнопок) передаются на сервер в формате JSON.

Такой подход позволяет избежать прямого использования HTML, CSS и JavaScript. Слой Generic UI позволяет работать с визуальным представлением экранов в формате XML, инициализировать и обрабатывать данные в контроллерах экранов, являющимися Java-классами.

2. Фреймворк **Spring** обеспечивает взаимодействие между контроллерами, веб-сервисами в среднем и клиентском слоях. Основные возможности сервисов [2]:
  - обращение к сущностям и их атрибутам;
  - написание запросов на языке JPQL;
  - вызов методов среднего слоя;
  - передача данных в форматах XML или JSON;
  - контроль доступа в зависимости от прав пользователя.
3. Фреймворк **EclipseLink** реализует технологию ORM, позволяя моделировать java-объекты для удобного хранения в базе данных, выполнять к ней запросы по всем стандартам Java Persistence API.
4. **Elasticsearch** – стандарт для реализации полнотекстового поиска, упрощающего нахождение необходимых документов в канцелярских архивах.
5. **Gradle** – система сборки проекта.
6. **jBPM** – модуль Workflow, предоставляющий ресурсы для создания, проектирования и управления бизнес-процессами.

## 1.4 Уровни и модули приложения

Все приложения строятся на трех основных уровнях: клиентский, средний слой и слой базы данных [2]. На первых двух уровнях существуют блоки приложения, являющиеся обособленной программой, чаще всего

выполняющейся на JVM. На каждом уровне может существовать несколько таких блоков, обеспечивая тем самым высокую масштабируемость системы [2]. Архитектура системы представлена на рисунке 1, где;

- Веб-клиент – реализация универсального пользовательского интерфейса, с которым пользователь взаимодействует в браузере.
- Веб-портал – дополнительный блок, обеспечивающий интерфейс для взаимодействия со сторонними приложениями и мобильными устройствами.
- Средний слой – ядро системы, содержащее основную бизнес-логику, модели сущностей, экранов. Осуществляет взаимодействие с базой данных.
- Слой базы данных – база данных на Oracle, MS SQL или PostgreSQL.
- Frontend UI – аналог веб-клиента для внешних пользователей.

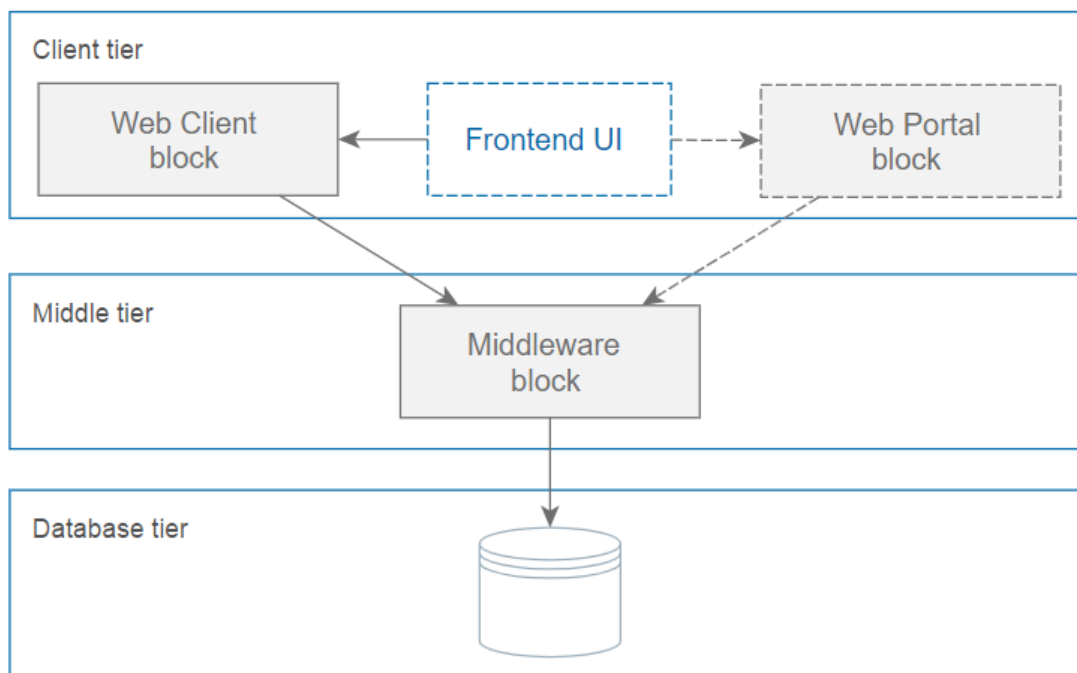


Рисунок 1 – Уровни и блоки Cubix-приложения

Источник: [2]

Архитектура приложения делится на модули (см. рис. 2) []:

- Global – содержит модели всех сущностей, интерфейсы сервисов и другие классы с общедоступной функциональностью. Классы сущностей содержат аннотации: @Entity @Listeners и т.п. Все сущности зарегистрированы в файле persistence.xml.
- Core – содержит сервисы, реализованные на основе интерфейсов Global-модуля, и другие компоненты. Сервисы аннотируются как @Service. Из них можно обращаться к данным сущностей в Global и вызывать другие сервисы. Классы с логикой сервисов доступны из других компонентов только в том случае, если объявлены их интерфейсы в глобальном модуле, что сделано эффективного масштабирования приложения.
- Gui – содержит компоненты пользовательского интерфейса веб-клиента.
- Web – реализация пользовательского интерфейса веб-клиента на Vaadin.
- Portal – веб-портал на Spring MVC.
- Front – реализация Frontend UI на JavaScript

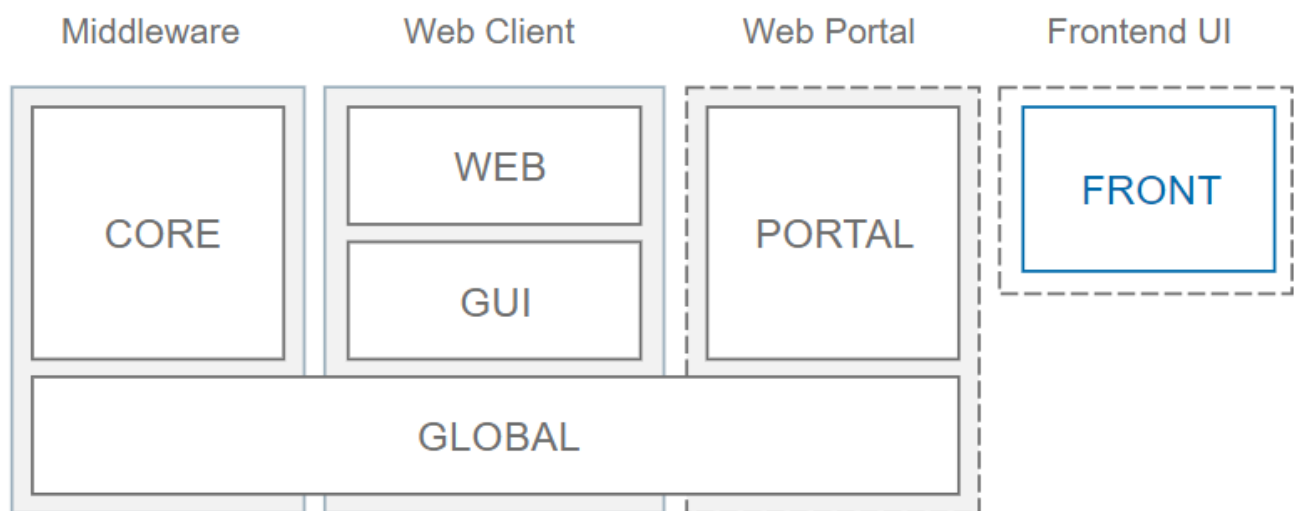


Рисунок 2 – Модули Cuba-приложения

Источник: [2]

## 1.5 Основные понятия. Плагин Cuba Studio

В платформе используются следующие элементы [2] [7]:

1. Сущность – объект со своими данными, характеристиками и функциями.  
В СЭД самые распространенные сущности – это документы, справочники, карточки.
2. Метаданные – фреймворк для работы с сущностями. Предоставляет интерфейс для получения информации о сущностях, их атрибутах, отношениях между сущностями, регламентирует допустимые типы данных и позволяет создавать различные механизмы работы [7].  
Интерфейсы представлены на рисунке 3.

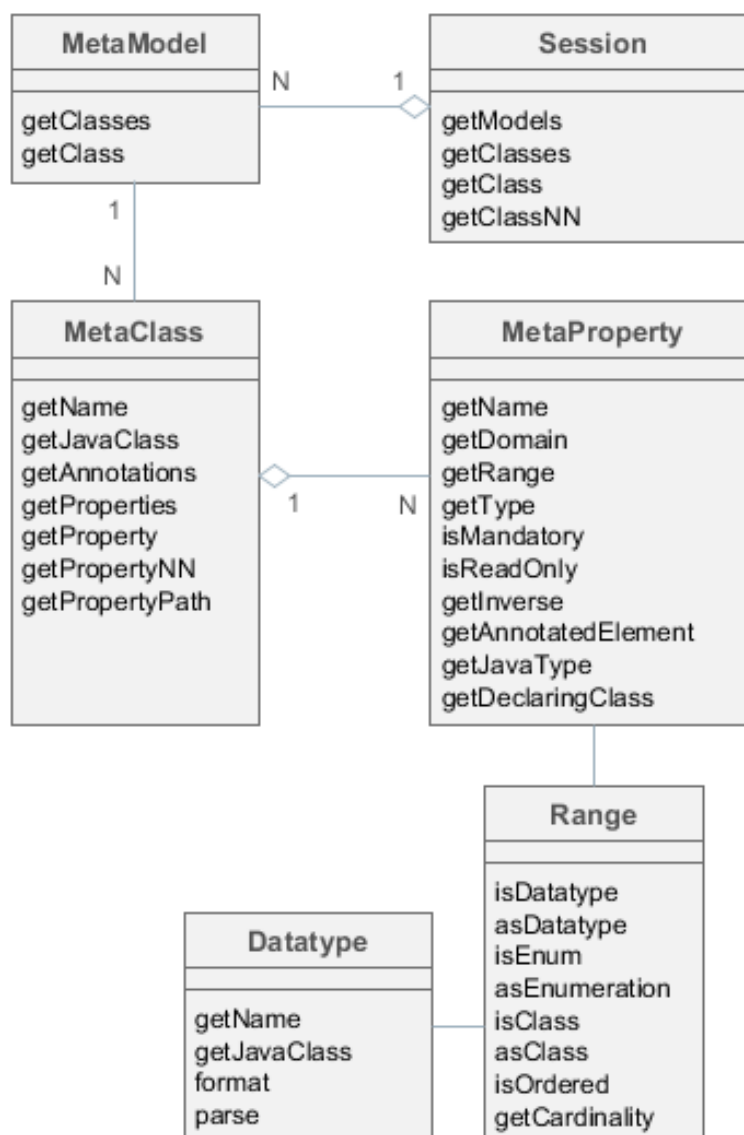


Рисунок 3 – Интерфейс метаданных

Источник: [2]

- Session - точка входа в фреймворк метаданных. Позволяет получать экземпляры MetaClass по имени или классу.
- MetaModel - редко используемый интерфейс для группировки метаклассов.
- MetaClass - интерфейс метаданных класса сущности.
- MetaProperty - интерфейс метаданных атрибута сущности.
- Range – интерфейс, описывающий тип атрибута сущности.

3. Сервис – определяют множество операций среднего слоя для клиентского уровня. В сервисе можно получать данные сущностей, сессии, сохранять и обрабатывать эти данные [2].

- Интерфейс сервиса расположен в модуле Global и доступен на среднем и клиентском слоях.
- Бин с логикой сервиса содержится в модуле Core и доступен только из среднего слоя.
- ServiceInterceptor – профилактика исключений.



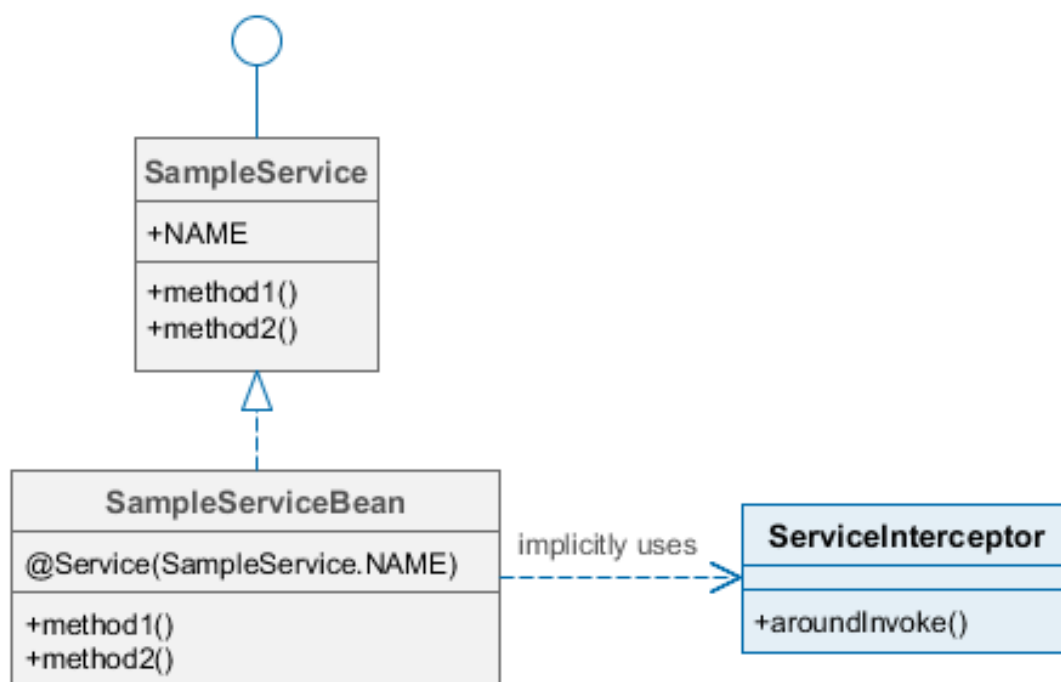


Рисунок 4 – Основные компоненты сервиса

Источник: [2]

4. Представление (View) – механизм загрузки данных сущностей. Решает проблему жадной загрузки, при которой приходится загружать все атрибуты сущности базы данных даже в том случае, если из них понадобится всего часть [7].
5. Дескриптор экрана – это XML-файл, описывающий визуальные компоненты, компоненты данных и параметры экрана.
6. Контроллер экрана – класс экрана, содержащий логику инициализации экрана и обработки событий таких как, например, вызов сервиса.
7. Сессия пользователя - объект класс `UserSession`, содержащий информацию о текущем пользователе, его правах доступа к данным.

**Cuba Studio** – плагин IntelliJIDEA, предоставляющий среду разработки со своим инструментарием для проектов на Cuba. Также существует как отдельное приложение.

Краткий обзор некоторых инструментов [2]:

- Cuba CLI – инструмент командной строки, позволяющий настраивать проект и создавать объекты. Инструмент позволяет разрабатывать Cuba-проекты в любой интегрированной Java-среде.
- Designer – инструмент, добавляющий удобный интерфейс для создания сущностей, заполнения атрибутов и ограничений.
- Визуальный редактор экранов – настройка внешнего вида экрана без прямого изменения xml-дескрипторов.
- Генераторы кода – код сущностей и экранов генерируется автоматически при использовании инструментов их создания.
- Visual Components Library – библиотека визуальных компонентов для экрана.
- Создание схем бизнес-процессов.
- Генератор маршрутов URL - предоставляет API для генерации ссылок на экран.

## **1.6 Базовые процессы в СЭД ТЕЗИС.**

Система электронного документооборота ТЕЗИС разработана компанией Haulmont, разработчиками платформы Cuba. Это готовая программа для автоматизации движения документов на предприятии и цифровизации основных процессов [5]. Рассмотрим несколько базовых процессов, реализованных в системе.

На главном экране системы ТЕЗИС есть три основных поля: папки действий, верхнее меню и основной экран работы. Папки действий (левая часть экрана) содержат задачи, договоры, документы и прочие сущности, разделенные по группам и имеющие отношения к авторизованному на данный момент пользователю. Верхнее меню содержит перечень основных объектов работы. С его помощью пользователь может создавать документы, задачи, просматривать список всех документов и поручений в системе, обращаться к архиву, настройкам системы, справочникам. С помощью настроек

системы пользователь может изменить внешний вид основного экрана работы, размещать виджеты по своему усмотрению. На рисунке 5 на главном экране расположен виджет с историей созданных пользователем задач.

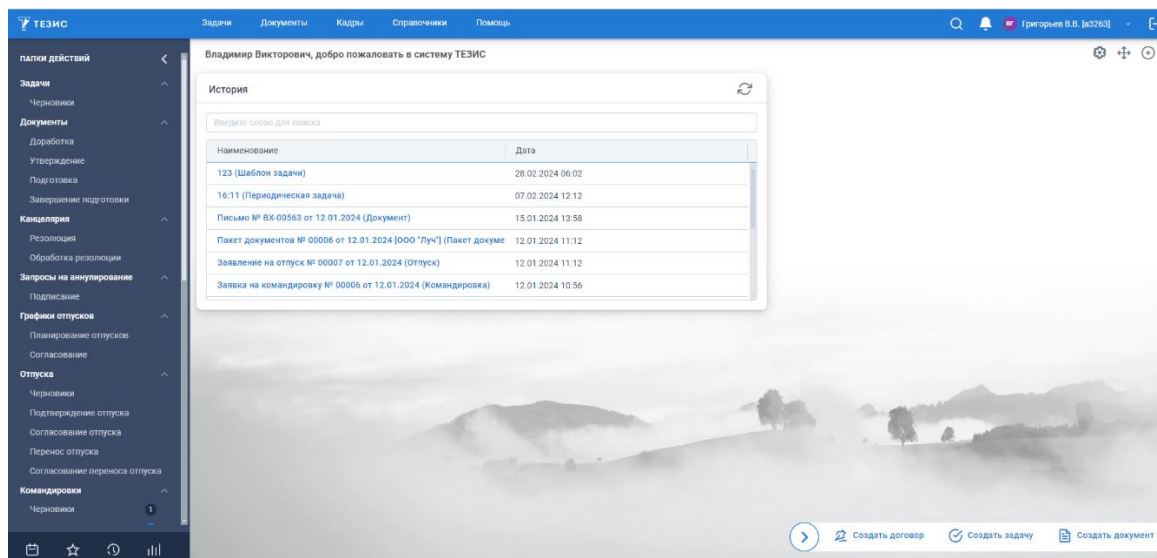


Рисунок 5 – Главный экран системы ТЕЗИС

Задачи во многом определяют занятость сотрудников, поэтому заниматься их созданием приходится практически постоянно. Во вкладке «Задачи» при нажатии кнопки «Создать задачу» открывается окно редактирования задачи (см. рис. 6). Заполнив поля, указав исполнителей, добавив необходимые вложения и т.п., можно пустить вход рабочий процесс.

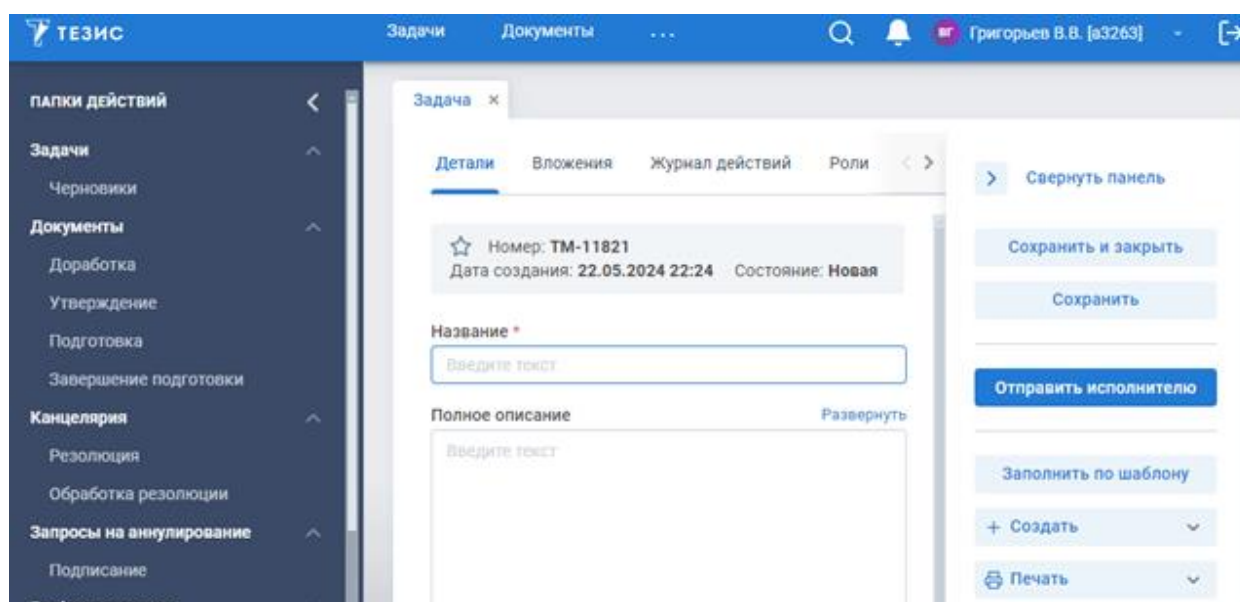


Рисунок 6 – Создание задачи

А система иерархии позволит проследить за его выполнением на протяжении всего жизненного цикла задачи (см. рис. 7).

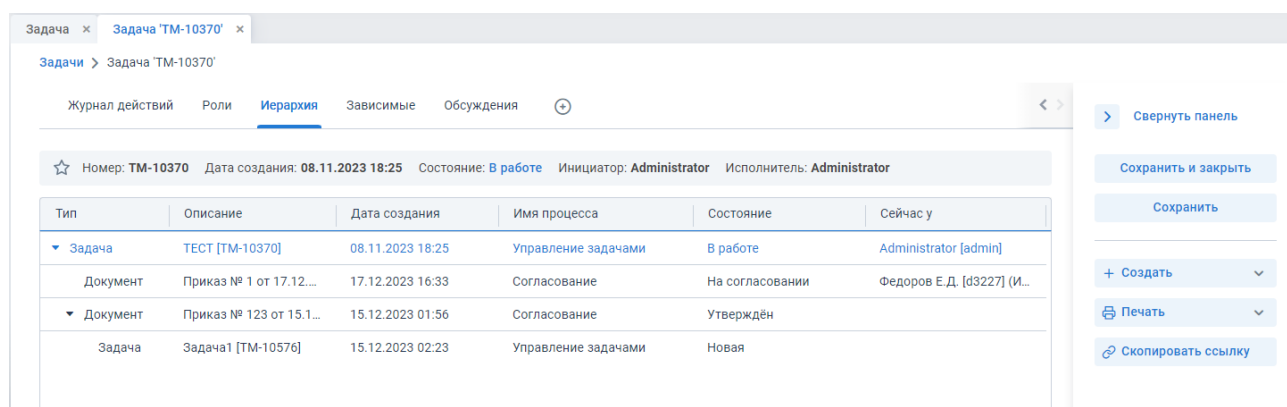


Рисунок 7 – Контроль исполнения

Чаще всего основанием для создания задачи являются документы. Во вкладке «Документы» при нажатии кнопки «Создать документ» открывается окно редактирования документа (см. рис. 8). Аналогично созданию задачи, заполняем необходимые поля. Из возможных операций с документами в системе реализованы ознакомление, регистрация, резолюция и согласование. При выборе одной из них откроется модальное окно для указания участников процесса (см. рис. 9). После подтверждения документ будет отправлен в работу.

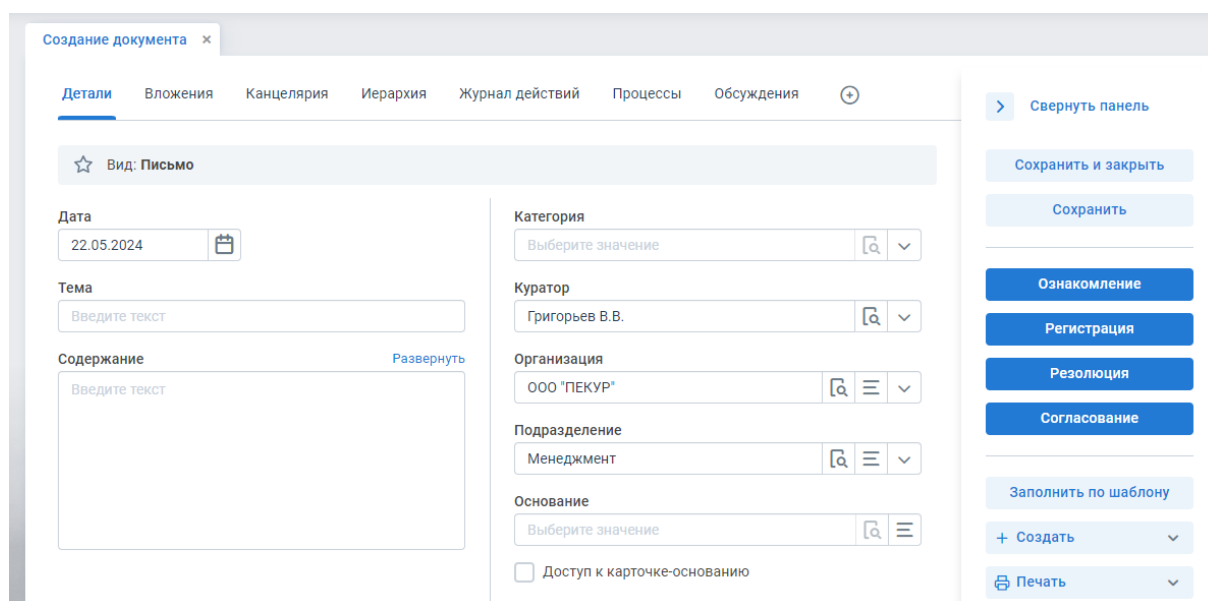


Рисунок 8 – Создание документа

Подтверждение

Детали Вложения

+ Добавить

	Роль	Пользователь	Очередность	E-mail	Трей
Инициатор	Григорьев В.В. [a3263] (Гене)			✓	✓
Согласующий	Бочарова И.В. [e3263] ( )	1	✓	✓	
Утверждающий	Григорьев В.В. [a3263] (Гене)		✓	✓	
Секретарь	Бочарова И.В. [e3263] (Секр)		✓	✓	

Комментарий Развернуть

Введите текст

Завершить к  ☐ Отправить в ЭДО

Отправить Отменить

Рисунок 9 – Согласование документа

Рассмотрев имеющийся функционал штатной версии системы, можно сделать вывод, что уже на старте мы получаем готовый продукт, содержащий в себе реализацию работы с объектами предприятия. Дальнейшая работа заключается в расширении этой системы до соответствия стандартам организации. Все возможности СЭД ТЕЗИС представлены на рисунке 10.



Рисунок 10 – Все возможности СЭД Тезис

Источник: [7]

## ГЛАВА 2. ТРЕБОВАНИЯ К МОДИФИКАЦИЯМ И ИХ ОПИСАНИЕ

### 2.1 Создание и настройка документа муниципальных услуг.

В данной главе будет приведено описание требуемых доработок: требования к функционалу и описание бизнес-процессов. Бизнес-процессы проиллюстрированы с помощью BPMN-нотации [7] для наглядного представления их структуры. Значение символов нотации можно посмотреть в приложении 2.

Документ муниципальных услуг – пример одного из видов документов, с которыми постоянно производится работа в системе. С создания подобных объектов обычно и начинается работа с СЭД. Задаётся название сущности, требуемые атрибуты и ограничения по стандартам организации, устанавливаются связи с другими сущностями, если это необходимо. К тому же необходимо произвести необходимые доработки на экранах документа.

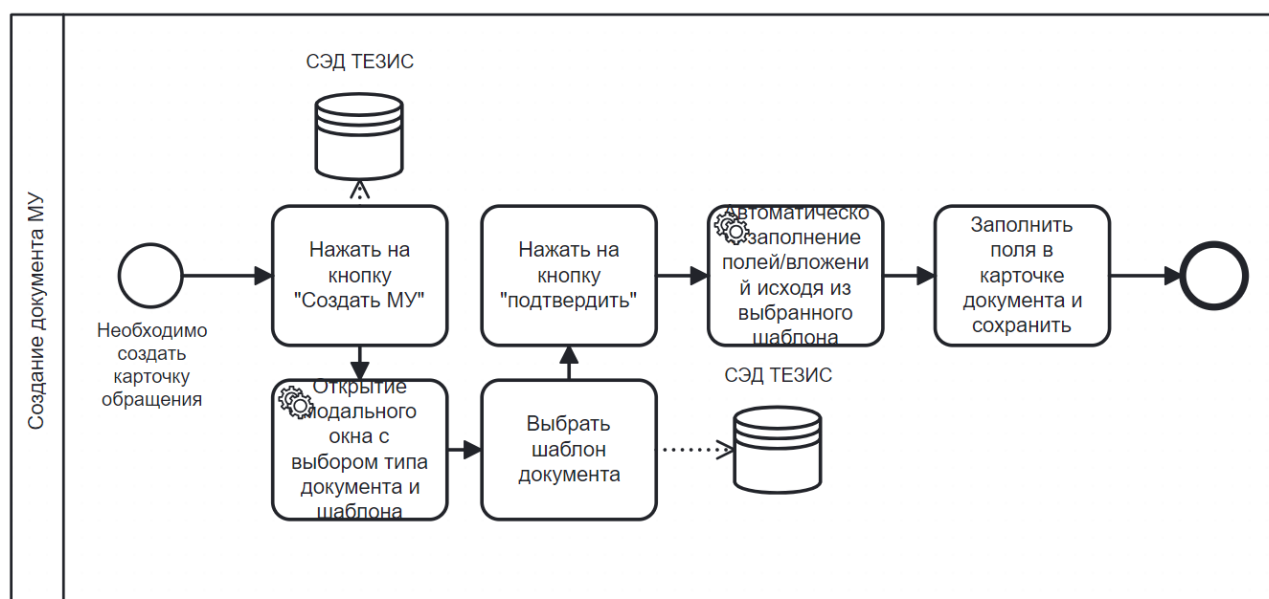


Рисунок 11 – Создание карточки документа муниципальные услуги

Источник: составлено автором

Для создания экземпляра документа пользователю необходимо кликнуть на кнопку «Создать МУ» во вкладке документы на верхней панели главного экрана СЭД ТЕЗИС. Для просмотра списка документов МУ аналогично создана

кнопка «Список МУ». После того, как пользователь нажмёт на создание документа, появится модальное окно с просьбой выбрать шаблон документа. После выбора откроется экран редактирования документа с выбранным шаблоном.

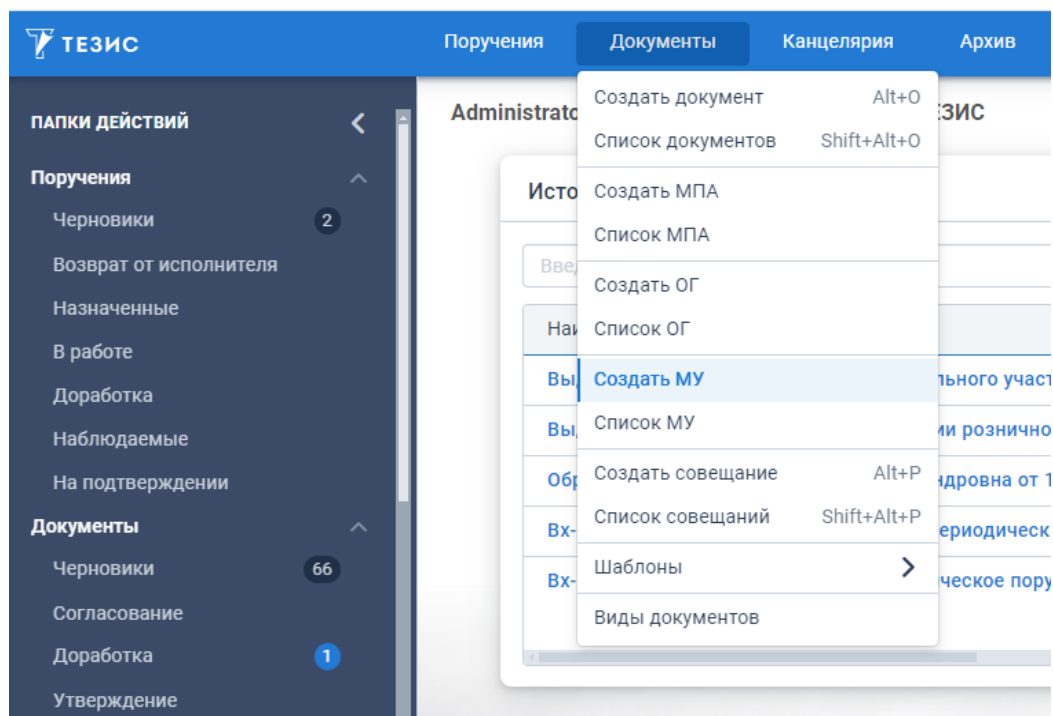


Рисунок 12 – Вкладка для создания документов

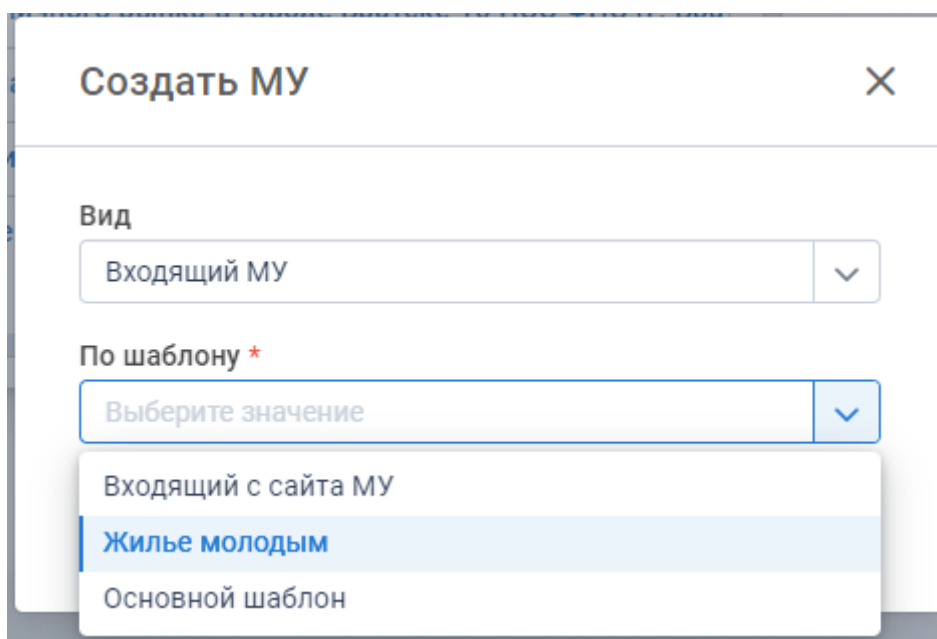


Рисунок 13 – Выбор шаблона документа

Некоторые поля заполняются автоматически. В задании это будет реализовано для поля owner. На экране это поле скрыто. После заполнения необходимых полей и нажатия кнопки «Сохранить» создастся экземпляр документа

МУ список × Создание документа ×

Детали Вложения Канцелярия Межведомственные запросы Рассылка Иерархия Связанные карточки Обсуждения Журнал действий

☆ Вид: Входящий МУ

Запросы предоставлены  
Выберите значение

Дата подачи заявителем  
ДД.ММ.ГГГГ

Дата поступления  
ДД.ММ.ГГГГ

Дата поступления результата в МКУ «ЦРДИМТО»  
ДД.ММ.ГГГГ

Вид МУ \*  
Выберите значение

Тематика обжалуемых решений  
Выберите значение

Штрихкод  
Введите текст

Содержание  
Введите текст

Уникальный идентификатор сайта  
Введите текст

Регистрационный номер (МФЦ)  
Введите текст

☐ Нарушение срока передачи МФЦ

Отдел МФЦ  
Выберите значение

Срок исполнения: \*  
ДД.ММ.ГГГГ

Дата исполнения услуги  
ДД.ММ.ГГГГ

Дата выдачи результата  
ДД.ММ.ГГГГ

Ввел данные  
Выберите значение

Организация  
МКУ "Многофункциональный центр обслу

Орган-получатель \*  
Выберите значения

Категория гражданина

Свернуть панель

Сохранить и закрыть

Сохранить

Переназначить

Зарегистрировать

Отправить на регистрацию

Рассылка

Отправить МЗ

Заполнить по шаблону

+ Создать

Печать

Сформировать email

Рисунок 14 – Экран редактирования документа МУ

### Задачи:

1. Создать сущность `municipalService`, необходимые атрибуты и связанные сущности.
2. Добавить в меню верхней панели вкладки «Документ» кнопки «Создать МУ» и «Список МУ».
3. Настроить автозаполнение поля «Исполнитель» (owner) на экране редактирования документа.
4. Настроить русификацию.
5. В контроллере экрана редактирования настроить метод, проверяющий, является ли документ просроченным.
6. Виды документа в выплывающем списке должны быть отсортированы по алфавиту.



7. Добавить метод для вывода сообщения о состоянии документа, если документ просрочен. В шапке документа указать, на сколько дней просрочен документ, добавив поле «Срок превышен на».

Атрибуты документа муниципальных услуг **municipalService**:

- sheetsCount [тип Integer] – количество листов в документе;
- dateInMKU [тип Date] – дата поступления в MKU;
- attachmentsCount [тип Integer] - количество вложений к документу;
- generalCountSheetsAttachments [тип Integer] - общее количество листов во всех вложениях к документу;
- note [тип String] - примечание к документу;
- municipalServiceType [тип сущность municipalServiceType, связь "многие к одному"] - тип муниципальной услуги;
- docReceiver [тип сущность Employee, связь "многие к одному"] - сотрудник, получивший документ;
- serviceExecutionDate [тип Date] - дата исполнения услуги;
- urlFromSite [тип String, ограничение 1000 символов] - URL с сайта, связанный с документом;
- dateOfIssueResult [тип Date] - дата выдачи результата;
- requestType [тип сущность requestType, связь "многие к одному"] - тип запроса;
- regNoMfc [тип String, ограничение 127 символов] - регистрационный номер в МФЦ;
- dateApplyByApplicant [тип Date] - дата подачи заявителем;
- isTransferMfcDeadline [тип Boolean] - признак передачи срока выполнения МФЦ;
- isExternalExecutor [тип Boolean] - признак внешнего исполнителя;
- subjAppealedDecisions [тип сущность subjAppealedDecisions, связь "многие к одному"] - тематика обжалуемых решений;

- departmentMfc [тип сущность departmentMfc, связь "многие к одному"]  
- отдел МФЦ;
- citizenCategory [тип сущность citizenCategory, связь "многие к одному"]  
- категория граждан;
- aisRegNo [тип String, ограничение 127 символов] - регистрационный номер в АИС;
- aisNumberDirection [тип String] - номер направления в АИС;
- aisDate [тип Data, ограничение 50 символов] - дата в АИС;
- direction [тип Integer] - направление;
- group [тип Integer] - группа;
- muStayMode [тип Integer] - режим пребывания в МУ;
- muEducProgram [тип Integer] - образовательная программа в МУ;
- reviewResult [тип Integer] - результат рассмотрения;
- notificationForKgs [тип Integer] - уведомление для КГС;
- noteFromSite [тип String, ограничение 511 символов] - примечание с сайта;
- barcode [тип String, ограничение 127 символов] - штрихкод;
- receiveDate [тип Data] - дата получения;
- uniqueWebsiteId [тип Long] - уникальный идентификатор веб-сайта.

Некоторые из атрибутов документа являются ссылками на другие сущности, а значит их нужно создать в первую очередь.

Сущность **Employee** уже есть в штатной версии СЭД ТЕЗИС. Её можно расширять, если организация попросит добавления атрибутов. В данном случае будем пользоваться изначальной версией. Остальные же сущности в штатной версии отсутствуют, их создаём вручную:

#### **municipalServiceType:**

- name [тип String, уникальное] – название;
- useCalenderDays [тип Boolean] – признак использования календарных дней;

- daysCount [тип Integer] – количество дней.

#### **requestType:**

- name [тип String, уникальное] – название.

#### **subjAppealedDecisions:**

- name [тип String, уникальное] – название.

#### **departmentMfc:**

- name [тип String, уникальное, обязательное] – название.

#### **citizenCategory:**

- name [тип String, уникальное, обязательное] – название.

**Замечание:** атрибут **owner** наследуется от системного класса Doc, от которого наследуются все сущности документов, поэтому добавлять его не нужно.

## **2.2 Настройка поиска штрихкода при сканировании документов с помощью плагина ТЕЗИС Помощник.**

Для начала работы необходимо установить плагин ТЕЗИС Помощник. На главном экране СЭД ТЕЗИС во вкладке «Помощь» нажать на кнопку Настройки. В разделе «Дистрибутивы» выбрать подходящую операционную систему.

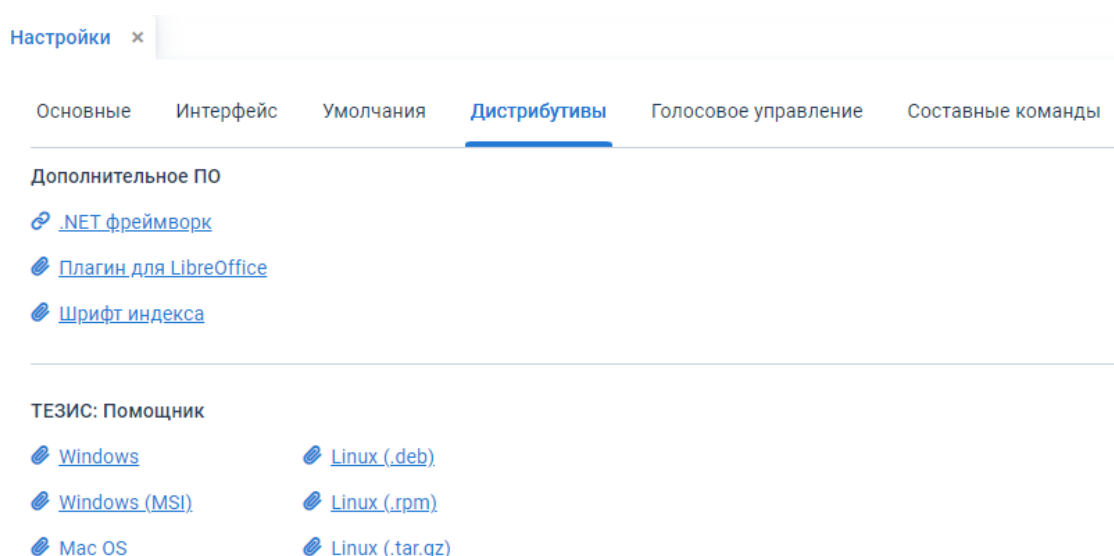


Рисунок 15 – Экран загрузки плагина ТЕЗИС Помощник

Скачиваем файл и производим установку. Теперь в каждом поручении во вкладке «Вложения» мы можем добавить файл-изображение отсканированного документа. Сканирование документа после установки плагина уже настроено и готово к использованию, однако такой стандартный функционал не всегда удовлетворяет требованиям организации. Иногда требуется сканирование определенного участка листа, а листы не всегда соответствуют стандарту А4.

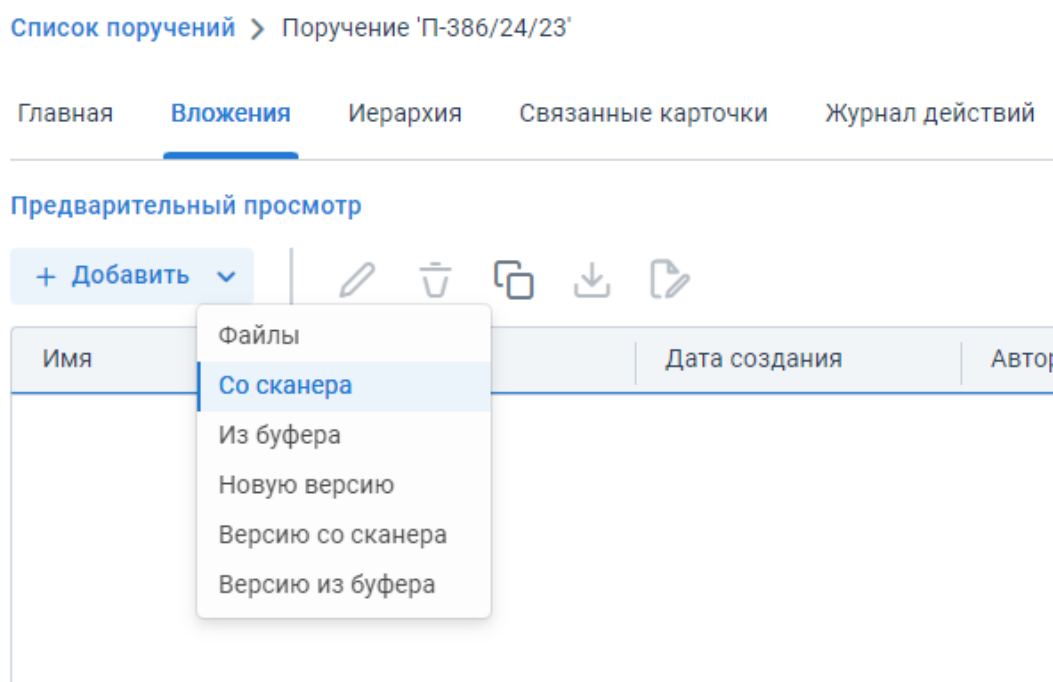


Рисунок 16 – Возможность добавить вложение в поручение через сканирование документа

### Задачи:

Необходимо переписать стандартную функциональность для считывания штрихкода при потоковой регистрации. При разных разрешениях изображения требуются разные размеры области штрихкода. Это связано с тем, что при разных разрешениях изображения могут требоваться разные размеры области с штрих-кодом для корректного считывания. Возможные разрешения документов: 150, 200, 300 dpi (точек на дюйм).

Длина и ширина листов при различных dpi в пикселях (px):

- При 300 dpi – от 2300 до 2600 px.

- При 200 dpi – от 1600 до 2000 px.
- При 150 dpi – от 550 до 700 px.

Для реализации необходимо написать метод считывания подизображения, содержащего штрих-код из исходного скана документа.

Отступы от краёв листа подизображения со штрихкодом рассчитываются так, чтобы они составляли около 40% и 15% от ширины и высоты исходного изображения соответственно.

### **2.3 Группа периодических поручений**

В штатной версии СЭД ТЕЗИС существует такое понятие как периодическое поручение. Это поручение, которое создается с определенной периодичностью. Допустим, сотруднику нужно каждую неделю по пятницам отправлять отчет о выполненных им услугах. Таким образом, в системе можно настроить автоматическое создание документа в каждую пятницу недели.

Для создания периодического поручения нужно навести курсор на вкладку «Поручения» и кликнуть на кнопку «Создать периодическое поручение».

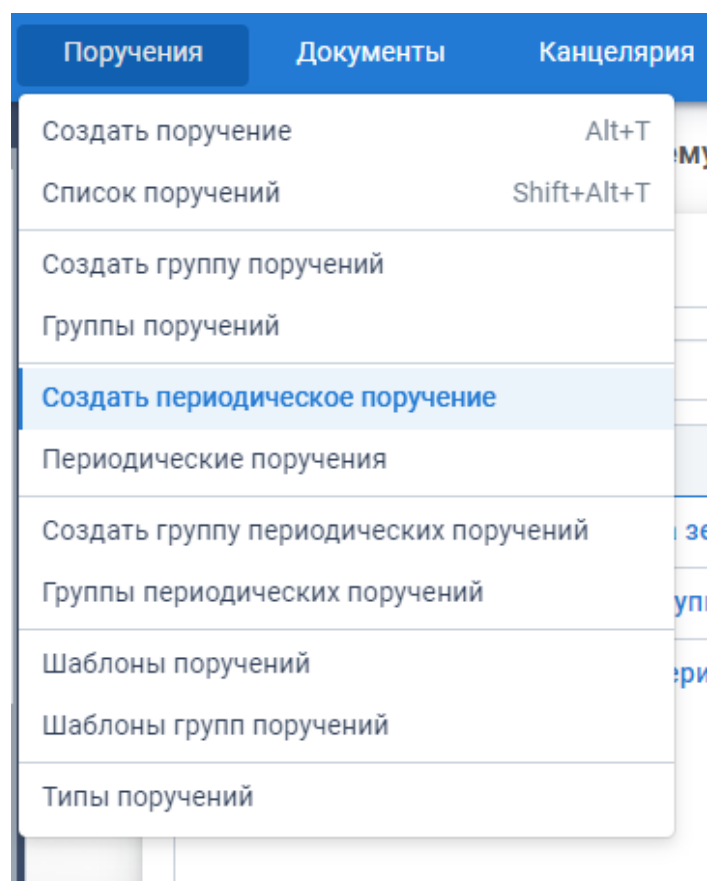


Рисунок 17 – Вкладка «Поручения»

Откроется экран редактирования периодического поручения. Пользователь заполняет необходимые поля. Указывает исполнителя, контроллеров, документ, на основании которого строятся поручения, действия (это дополнительные функции, которые будут работать при работе с поручением, например: каждый раз при создании поручения создатель будет получать уведомления о создании периодического поручения). Также указывается расписание, по которому поручение будет создаваться автоматически. Далее поручение сохраняется и идёт в работу.

Редактирование периодического поручения ✕

Главная

Журнал

История изменений

Созданные поручения

Документ \*

Развернуть

Введите текст

Текст поручения

Развернуть

Введите текст

Организация

МКУ "Центр молодежных инициатив"

🔍

▼

Инициатор \*

Выберите значение

▼

Основание

Выберите значение

🔍

≡

Группа контролеров

Выберите значение

≡

Исполнитель \*

Выберите значение

▼

Рисунок 18 – Экран редактирования периодического поручения

Действия

Создать

Тип	Название

Расписания

Создать

Тип	Описание	Дата след. запуска	Активно	Использовать рабочий

Рисунок 19 – Поля «Действия» и «Расписание» экрана редактирования периодических поручений

Однако у этой системы есть существенный недостаток. Поручение о создании отчета может относиться сразу ко многим сотрудникам. Рассмотрим BPMN-диаграмму процесса:

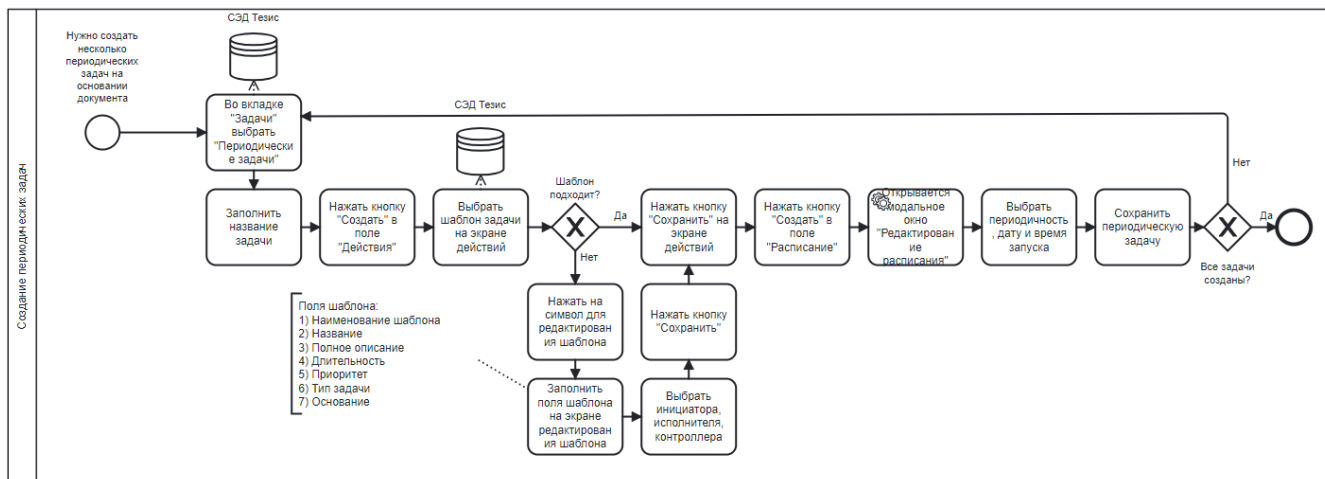


Рисунок 20 – BPMN-диаграмма процесса создания периодических поручений на несколько человек в штатной версии СЭД ТЕЗИС

Источник: составлено автором

Как видно по диаграмме, если нам нужно создать поручение на нескольких исполнителей, то действие с созданием периодического поручения и



заполнением полей нужно повторять для каждого такого исполнителя. Более того, каждый исполнитель может создавать подпоручение на основании полученного периодического поручения для своих подчиненных, которых также может быть много. Чем длиннее иерархия, тем больше однотипных действия придется совершать пользователям. В системе отсутствует возможность создания поручения для нескольких сотрудников сразу. Для решения этой проблемы я принял участие в производстве доработки «Группа периодических поручений» в составе программистов Сибирского центра информационных технологий. Рассмотрим BPMN-диаграмму доработки:

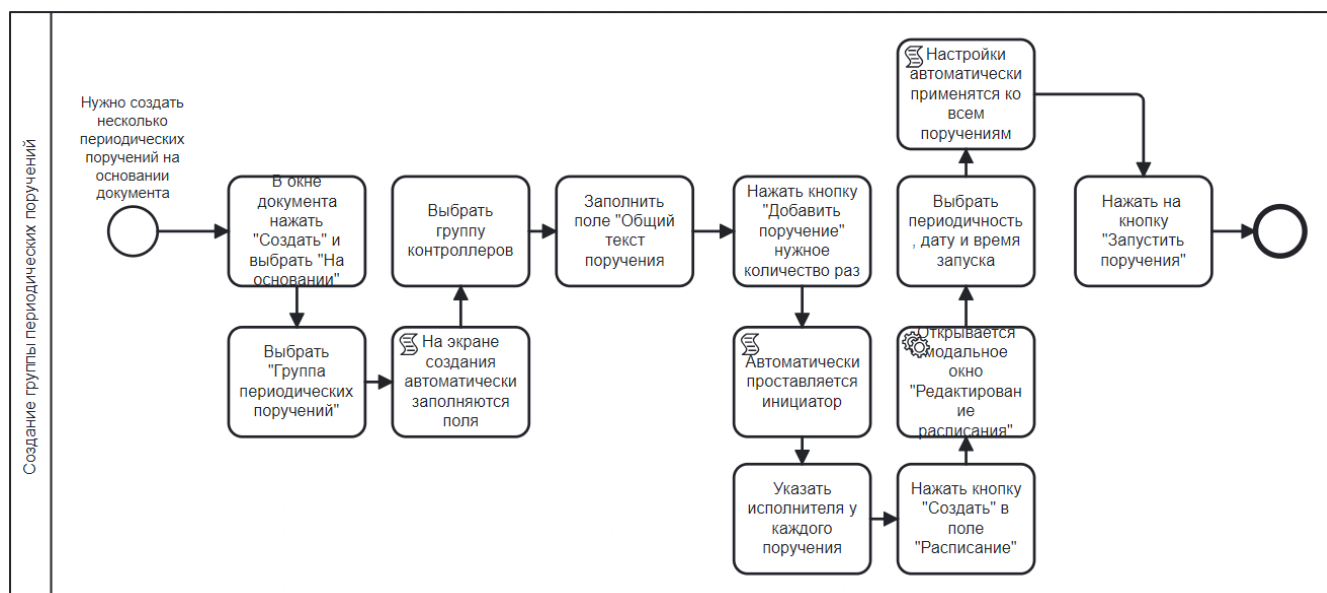


Рисунок 21 – BPMN-диаграмма процесса создания группы периодических поручений

Источник: составлено автором

Добавлена возможность создания периодического поручения прямо на экране документа. После нажатия кнопки «На основании» и «Группа периодических поручений» автоматически откроется экран редактирования группы периодических поручений.

Обращение Плотникова Наталия Александровна... x

ОГ список > Обращение Плотникова Наталия Александровна от 19.10.2023 (ОГ)

Главная Вложения 1 Канцелярия Рассылка Иерархия Связанные карточки Обсуждения Журнал действий

☆ Обращение Плотникова Наталия Александровна от 19.10.2023

Дата подачи: 19.10.2023

Тип обращения: Выберите значение

Вид обращения: Выберите значение

Содержание: [Развернуть](#)

Здравствуйте. Я одна воспитываю ребенка, как родила встала на очередь Помощь молодой семье в жилье в 2019 году. Живем мы с бабушкой, квартира 31 квадрат общей площади не жилой... 5 лет мы простояли на очереди и пришло письмо, что нас с

Количество листов: Введите значение

Штрихкод: Введите текст

☐ Поступило из «Приемной мэра»  
☐ Коллективное обращение  
☐ Повторное обращение

Ввел данные: Выберите значение

Организация \*: МКУ "Многофункциональный центр обслу

Орган-получатель \*: Ревизионный отдел

Тематика обращения: [+](#) [-](#)

Наименование

Свернуть панель

Сохранить и закрыть  
 Сохранить  
 Переназначить

Отменить процесс

Рассылка

+ Создать  
 На основании

Скопировать ссылку  
 Сформировать email

Рисунок 22 – Экран документа. Создание на основании

К тому же некоторые поля группы будут автоматически заполнены, исходя из данных документа и пользователя.

Редактирование группы периодических пору... x

ОГ список > Редактирование группы периодических поручений

Детали Вложения 1

Документ \* [Развернуть](#)

Обращение Плотникова Наталия Александровна от 19.10.2023

Инициатор \*: Administrator

Основание: Обращение Плотникова Наталия Александров [Q](#) [≡](#)

Организация: МКУ "Многофункциональный центр обслу [Q](#) [≡](#) [v](#)

Группа контролёров: Выберите значение [≡](#)

Общий текст поручения: Введите текст [Применить](#)

Расписания

[Создать](#) [✎](#) [🗑](#)

Тип	Описание	Дата след. старта	Активн

Длительность (дней): Введите значение

Рисунок 23 – Экран редактирования группы периодического поручения

После заполнения полей пользователь может создавать поручения на вкладке «Поручения». У каждого из поручения можно указать исполнителя и всё это на одном экране. При создании поручения в него автоматически занесется текст поручения, инициатор (автоматически подставится имя пользователя, создающего группу периодических поручений) и указанное до этого расписание создания поручений.

Периодические поручения

Добавить поручение    Запустить поручения    Отменить поручения

Дата создания	Содержание документа	Инициатор	Исполнитель	Ответственный исполн...	Дата след. старта	Распи
	Обращение Плотнико	Administrator	Артименко Г. В. (	<input type="checkbox"/>		
	Обращение Плотнико	Administrator	Беляева Н. А. (Н	<input type="checkbox"/>		
	Введите текст	Administrator	Белезов П. Е. (П	<input type="checkbox"/>		

Рисунок 24 – Таблица для создания поручений

Данная доработка не только решает основную проблему, из-за которой приходилось создавать отдельно однотипные поручения для каждого работника, но и ускоряет процесс за счёт автоматического заполнения полей на основании документа и добавления в окно документа функции создания на основании группы ПП.

### Задачи:

#### 1. Создание основных сущностей доработки:

- ScheduleTasksGroup - группа периодических поручений. Основной новый объект.
- ExtScheduleTask - расширение периодического поручения. Добавление новых полей, необходимых для функционирования доработки.

- ExtScheduleTrigger – сущность, создающая связь между периодическим поручением и группой, к которой оно принадлежит.

- ScheduleTaskGroupAttachment – сущность, создающая связь между группой периодических поручений и вложениями, если таковые имеются.

2. Обработка удаления поручений из группы периодических поручений. В контроллере экрана удаления поручений был разработан класс CancelTasksAction, содержащий методы для удаления поручений. При выборе поручений и нажатии кнопки «Удалить» экран инициализируется. Метод actionPerform в классе CancelTasksAction обрабатывает данные при нажатии кнопки «Удалить». В нем содержится обращение к методу cancelTasks, содержащему логику удаления выбранных поручений. Но необходимо также закрыть все дочерние подпоручения. Для этого был реализован сервис TaskControllerServiceBean с методом cancelTaskWithSubTasks, закрывающим подпоручения, выбранного пользователем периодического поручения.

Атрибуты сущностей:

**ScheduleTaskGroup:**

- name [тип String] - название группы поручений;
- duration [тип Integer] - срок выполнения;
- scheduleTasks [тип List<ExtScheduleTask>] – список поручений в группе;
- scheduleTriggers [тип List<ExtScheduleTrigger>] – триггеры связи;
- description [тип String] - описание;
- organization [тип Organization] - организация;
- initiator [тип User] - инициатор;
- parentCard [тип Card] - родительская карта;
- groupControllers [тип UserGroup] – группа контроллеров;

- attachments [тип List<ScheduleTaskGroupAttachment>] – вложения.

**ExtScheduleTask:**

- scheduleTasksGroup [тип ScheduleTasksGroup] – группа периодических поручений, к которой принадлежит поручение;
- initiator [тип TsUser] - инициатор;
- executor [тип TsUser] - исполнитель;
- groupControllers [тип UserGroup] – группа контроллеров;
- parentCard [тип Card] - родительская карточка;
- responsibleExecutor [тип Boolean] – флаг, указывающий на ответственного исполнителя.

**ExtScheduleTrigger:**

- scheduleTasksGroup [тип ScheduleTasksGroup] – группа периодических поручений, к которой принадлежит поручение.

**ScheduleTaskGroupAttachment:**

- taskGroup [тип ScheduleTasksGroup] – группа периодических поручений, к которой принадлежит вложение.

## ГЛАВА 3. ПРОГРАММНЫЙ КОД ДОРАБОТОК

### 3.1 Документ муниципальных услуг

#### 3.1.1 Создание сущностей

С помощью инструмента Designer платформы Cuba создадим документ MunicipalService. В дереве проекта во вкладке DataModel, где хранятся все сущности системы, выбираем создание документа.

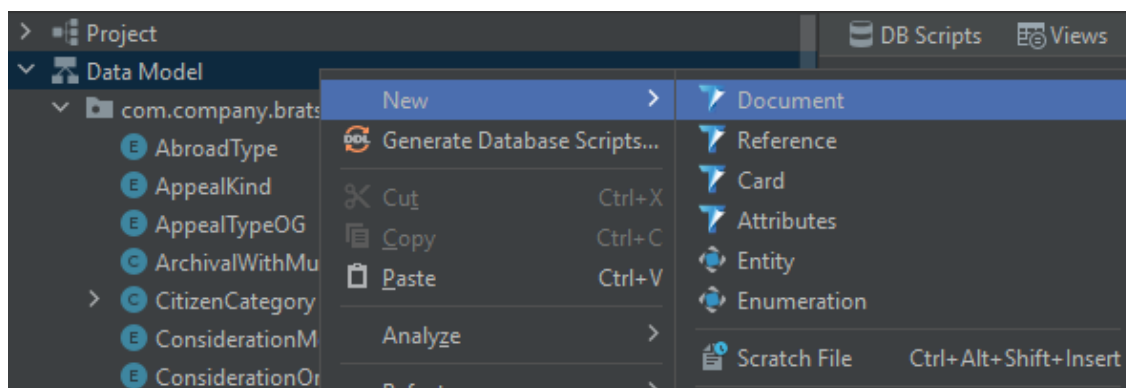


Рисунок 25 – Создание документа в дереве проекта

Откроется окно с данными сущности. Заполняем название и задаем локализацию для русского и английского языка. Локализация отвечает за то, как пользователь будет видеть название документа в системе на выбранном им языке при авторизации.

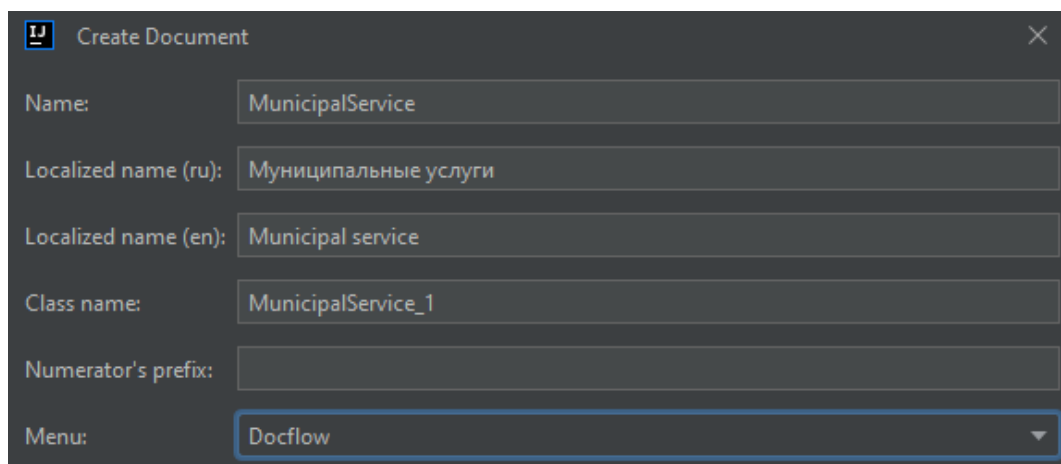


Рисунок 26 – Шапка меню создания сущности

Нажимаем кнопку с плюсом для открытия модального окна с параметрами атрибута. Заполним необходимые поля для создания атрибута sheetsCount типа Integer.

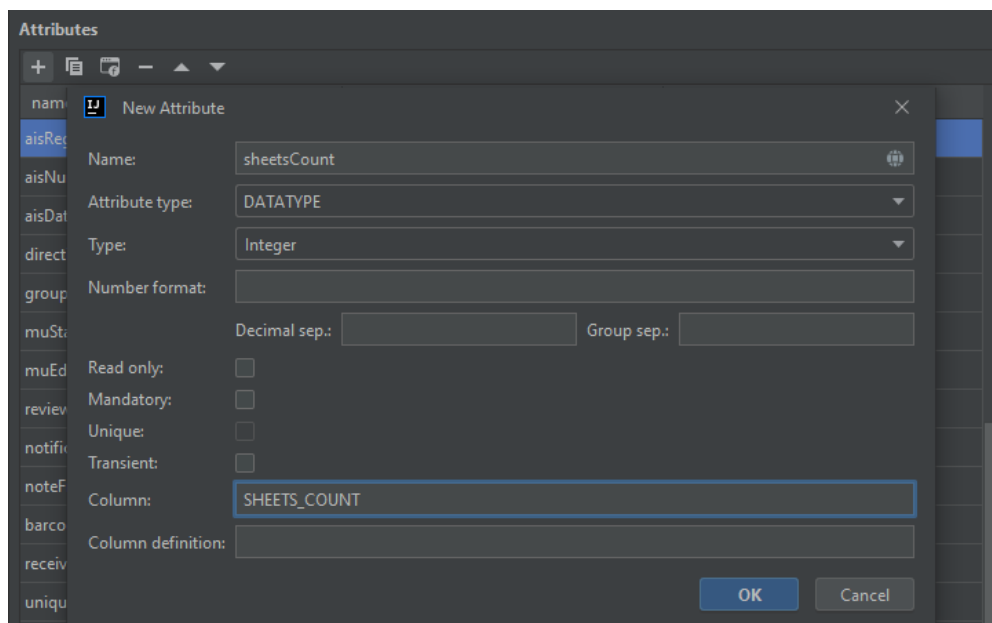


Рисунок 27 – Модальное окно создания атрибута

Аналогично добавляем все остальные атрибуты, требуемые по условию задания. После нажатия кнопки подтверждения генерируются скрипты, описывающие созданный объект.

name	type	column
aisRegNo	String	
aisNumberDirection	String	
aisDate	Date	
direction	MuDirection	DIRECTION
group	MuGroup	GROUP_
muStayMode	MuStayMode	MU_STAY_MODE
muEducProgram	MuEducProgram	MU_EDUC_PROGRAM
reviewResult	ReviewResultMu	REVIEW_RESULT
notificationForKgs	MuNotificationTypeKgs	NOTIFICATION_FOR_KGS
noteFromSite	String	NOTE_FROM_SITE
barcode	String	BARCODE

Рисунок 28 – Атрибуты сущность MunicipalService

В дереве проекта в модуле Screens автоматически создаются xml и java-файлы, содержащие разметку, описывающую внешний вид экрана, и контроллер со стандартными методами инициализации соответственно:

- **MunicipalServiceBrowse** – экран со списком всех документов созданного типа.
- **MunicipalServiceEdit** – экран редактирования документа, содержащий поля атрибутов.

Также создается папка Resource Bundle `messages` с локализацией.

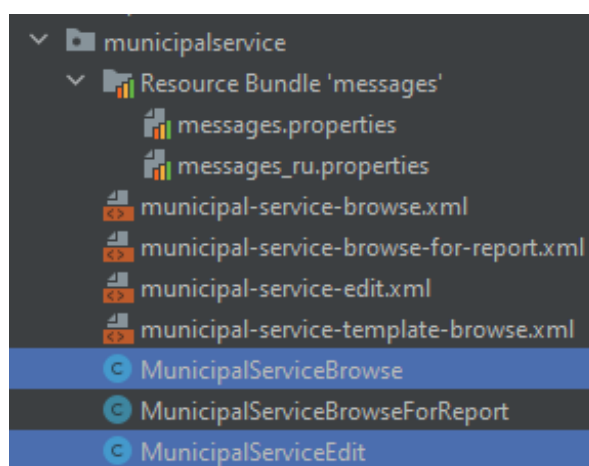


Рисунок 29 - Создание экранов документа

В листинге 1 представлен фрагмент, созданного скрипта для документа **MunicipalService**. Он содержит различные системные настройки, импорты библиотек, аннотации, описание каждого атрибута, его ограничения и тип, а также методы `get`, `set`.

#### Листинг 1 – Фрагмент скрипта сущности **MunicipalService**

```
import java.util.Date;
@PublishEntityChangedEvents
@DiscriminatorValue("2200")
@Table(name = "EKATERINBURG_MUNICIPAL_SERVICE")
@Entity(name = "ekaterinburg$MunicipalService")
@Listeners("thesis_DocEntityListener")
@EnableRestore
@TrackEditScreenHistory
@PrimaryKeyJoinColumn(name = "CARD_ID", referencedColumnName = "ID")
```



## Продолжение листинга 1

```
@NamePattern("%s|description")
public class MunicipalService extends Doc implements
HasDetailedDescription, HasSubTasks {

    private static final long serialVersionUID = -
        5100318791654827840L;

    @Column(name = "SHEETS_COUNT")
    protected Integer sheetsCount;

    . . .

    public Integer getSheetsCount() {
        return sheetsCount;
    }

    public void setSheetsCount(Integer sheetsCount) {
        this.sheetsCount = sheetsCount;
    }
}
```

Оставшиеся сущности создаются аналогично. В дальнейших задачах внимание на этом заостряться не будет.

### 3.1.2 Добавление в меню «Документ» на главном экране вкладок «Список МУ», «Создать МУ»

Для выполнения следующей задачи в модуле web обратимся к файлу `ekaterinburg-web-menu.xml`, содержащему разметку меню главного экрана. Добавим туда следующий код, чтобы нужные нам вкладки правильно отображались (см. листинг 2).

#### Листинг 2 – Добавление вкладок документа в файл `ekaterinburg-web-menu.xml`

```
<item id="municipalServiceDocCreatorItem"
class="com.haulmont.thesis.web.ui.common.CreateDocMenuItemController">
    <param name="docType" value="ekaterinburg$MunicipalService"/>
    <permissions>
        <permission type="ENTITY_OP"
target="ekaterinburg$MunicipalService:create"/>
    </permissions>
```

## Продолжение листинга 2

```
</item>  
<item id="MunicipalService.lookup"  
screen="ekaterinburg$MunicipalService.lookup"/>
```

Для создания кнопки «Создать МУ» используем тег `item` и зададим ему атрибуты:

- `id="municipalServiceDocCreatorItem` - идентификатор элемента;
- `class="com.haulmont.thesis.web.ui.common.CreateDocMenuItemController"` - класс контроллера, который будет обрабатывать нажатие на кнопку пользователем;
- `<param name="docType" value="ekaterinburg$MunicipalService"/>` - параметр, передающий тип документа;
- `<permissions>` - тег определяет, права доступа к операции создания документа;
- `type="ENTITY_OP"` – тип операции, операция над сущностью;
- `target="ekaterinburg$MunicipalService:create"` – операция, к которой определяется право доступа. В данном случае создание документа `MunicipalService`.

Для создания кнопки «Список МУ» создаём новый `item`, добавляем `id` и прописываем ссылку на экран со списком документов типа `MunicipalService`:  
`screen="ekaterinburg$MunicipalService.lookup"`.

### 3.1.3 Автозаполнение поля «Исполнитель» в экране редактирования документа

Для реализации логики автозаполнения необходимо обратиться к контроллеру экрана `MunicipalServiceEdit.java`. В нём переопределяем метод `postInit` (см. листинг 3), который выполняет логику после создания нового объекта, в данном случае нашего документа. Импортируем сущность `employee`, чтобы обратиться к элементу `owner`, соответствующему этому типу. Инжектируем из

разметки экрана поле owner, чтобы вставить в него нашего исполнителя после сохранения документа.

### Листинг 3 – Метод postInit в контроллере экрана MunicipalServiceEdit

```
import com.haulmont.thesis.core.entity.Employee;
@Inject
private LookupPickerField<Employee> owner;

@Override
protected void postInit() {
    super.postInit();
    Employee employee =
orgStructureService.getEmployeeWithDepartmentByUser(userSession.ge
tCurrentOrSubstitutedUser());
    if (employee != null) {
        owner.setValue(employee);
    }
}
```

Теперь можно приступить к основной логике метода. Строка `super.postInit()` отвечает за стандартную логику инициализации, присутствует по умолчанию всегда. Без неё метод не заработает.

Создаем объект `employee` и присваиваем ему значение текущего пользователя с помощью сервиса `orgStructureService`, который находит сотрудника, соответствующего переданному пользователю, в данном случае пользователю текущей сессии.

#### 3.1.4 Настройка локализации

Локализация отвечает за то, как пользователь видит отображение полей, вкладок, меню и прочих объектов на экранах. В модулях системы есть файл `messages.properties` для хранения локализации объектов на английском и русском языках. При создании сущности с помощью инструмента `designer` такие файлы создаются автоматически как для сущности, так и для экранов и уже содержат данные для локализации атрибутов, если те были введены. Пример содержания такого файла для атрибутов документа `MunicipalService` представлен в листинге 4.

#### Листинг 4 – Содержание файла message\_ru.properties для локализации документа

```
MunicipalService.sheetsCount=Количество листов
MunicipalService.attachmentsCount=Количество приложений
MunicipalService.note=Примечание
MunicipalService.theme=Содержание
MunicipalService.needIndRequest=Нужна ли вкладка МВЗ
MunicipalService.municipalServiceType=Вид МУ
MunicipalService.docReceiver=Кому
MunicipalService.owner=Ввел данные (исполнитель)
MunicipalService.serviceExecutionDate=Дата исполнения услуги
MunicipalService.dateOfIssueResult=Дата выдачи результата
MunicipalService.requestType=Запросы предоставлены
MunicipalService.regNoMfc=Регистрационный номер (МФЦ)
MunicipalService.dateApplyByApplicant=Дата подачи заявителем
MunicipalService.isTransferMfcDeadline=Нарушение срока передачи
МФЦ
MunicipalService.isExternalExecutor=Внешний исполнитель
MunicipalService.subjAppealedDecisions=Тематика обжалуемых решений
MunicipalService.departmentMfc=Отдел МФЦ
```

Не всегда атрибуты добавляются с помощью инструментария платформы. Если элемент создавался вручную, то и локализацию для него придется добавлять самостоятельно. В предыдущей задаче мы создали кнопки во вкладке «Документы»: «Создать МУ» и «Список МУ». Для них также необходимо добавить локализацию. Перейдем в файл messages.properties в модуле «Generic UI». Там содержится локализация для элементов главного экрана. Добавим её и для наших кнопок (см. листинг 5).

#### Листинг 5 – Локализация вкладок в messages.properties

```
menu-config.OG.lookup=Список ОГ
menu-config.MunicipalService.lookup=Список МУ
```

Локализация обязательна, так как без неё объекты будут иметь системные названия, не дающие пользователю представление о своём предназначении, что проиллюстрировано на рисунке 30.



Рисунок 30 – Вид кнопок без локализации

### 3.1.5 Работа с просроченным документом

В контроллере экрана `MunicipalServiceEdit` реализуем метод, проверяющий, просрочен ли документ (см. листинг 6).

Листинг 6 – Метод `isDocOverdue`, выясняющий просрочен ли документ

```
@Inject private CardExpiredService cardExpiredService;
protected boolean isDocOverdue(MunicipalService doc) {
    if (!isNewItem((T) doc) && doc.getServiceExecutionDate()
        != null && doc.getFinishDatePlan() != null) {
        return
            cardExpiredService.calculateNumberOfDays(doc.getFinishDatePlan(),
            doc.getServiceExecutionDate()) > 0;
    }
    return false;
}
```

У каждого документа есть атрибут, содержащий дату выполнения требований `FinishDatePlan`, а также атрибут, содержащий дату, на которое выполнение планировалось изначально `ServiceExecutionDate`. Зная это, сделаем так, чтобы метод возвращал `true`, то есть документ просрочен, если, вычитая количество дней запланированных на выполнение из количества дней фактически ушедших на выполнение, получаем число положительное, то есть число дней, которое прошло с момента просрочки документа.

Добавим условие, что документ не должен быть новым, то есть уже должен быть в базе данных, иначе он точно не просрочен, а также проверку ненулевых

значений дат. Для того, чтобы получить разницу дней между датами, реализуем методы `calculateNumberOfDays`, `getResult`, а также сервис `cardExpiredService` для работы с просроченными документами, в которой и поместим их. В модуле `Global` в папке `Services` создадим файл с основным интерфейсом (см. листинг 7), а инициализируем логику методов в `Bean`-файле в модуле `Services` (см. листинг 8). В дальнейшем сервис также пригодится, чтобы вывести строку с числом и словом «день» в корректном падеже.

#### Листинг 7 – Создание интерфейса сервиса `cardExpiredService`

```
import java.util.Date;
public interface CardExpiredService {
    String NAME = "ekaterinburg_CardExpiredService";

    int calculateNumberOfDays(Date begin, Date end);
    String getResult(Date begin, Date end);
}
```

#### Листинг 8 – Инициализация метода в файле `cardExpiredServiceBean`

```
@Service(CardExpiredService.NAME)
public class CardExpiredServiceBean implements CardExpiredService
{
    @Override
    public int calculateNumberOfDays(Date begin, Date end) {
        Days d = Days.daysBetween( new DateTime(begin), new
DateTime(end));
        return d.getDays();
    }

    @Override
    public String getResult(Date deadline) {
        int days = calculateNumberOfDays(deadline);
        String res = ""+days;
        String r = "Просрочено на " + days;
        if (4 < days && days < 21) {
            return res + " дней";
        }
        switch (getLastDigit(days)) {
            case 1:
                res += " день";
                break;
            case 2:
                res += " дня";
                break;
        }
    }
}
```

## Продолжение листинга 8

```
        case 3:
            res += " дня";
            break;
        case 4:
            res += " дня";
            break;
        default:
            res += " дней";
    }
    return res;
}
}
```

Следующим шагом будет инициализация шапки документа и настройка отображения информации в ней. Если документ просрочен на 10 дней, то в шапке должна отобразиться надпись «Просрочен на 10 дней». В методе `initHeaderTitleTags` (см. листинг 9) мы получаем объект `MunicipalService`, проверяем, является ли он просрочен. Если просрочен, то обращаемся к полю шапки через переменную `fragment` и присваиваем ей нашу строку «Просрочен на », которая хранится в папке `messages_ru.properties` как «`header.overdue=Срок превышен на` ». А потом присваиваем ей подсчитанное с помощью сервиса `cardExpiredService` количество дней.

## Листинг 9 – Настройка отображаемых данных в шапке документа

```
protected void initHeaderTitleTags() {
    MunicipalService item = (MunicipalService) getItem();
    if (isDocOverdue(item)) {
        Label<String> fragment =
cardHeaderFragment.getTitleTag("headerTitileTagOverdue");
        String msg = getMessage("header.overdue");
        msg +=
cardExpiredService.getResult(item.getFinishDatePlan(),
item.getServiceExecutionDate());
        fragment.setValue(msg);
cardHeaderFragment.setTitleTagVisible("headerTitileTagOverdue",
true);
    }
}
}
```

Теперь осталось инициализировать саму шапку документа, так как по умолчанию её нет (см. листинг 10). Делаем это с помощью системного метода `initHeaderFragmentContent`. Добавляем в шапку тэг красного цвета и скрываем его по умолчанию (до этого в методе `initHeaderTitleTags` мы прописали, что если документ просрочен, то и тег станет видимым).

В методе `postInit`, вызываемом после инициализации экрана, пропишем вызов метода `initHeaderTitleTags` для формирования строки (см. листинг 10).

Листинг 10 – Инициализация шапки документа через `initHeaderFragmentContent` и вызов метода `initHeaderTitleTags` после инициализации экрана в `postInit`

```
@Override
protected void postInit() {
    super.postInit();
    initHeaderTitleTags();

    //другая логика, не относящаяся к задаче
    . . .
}

@Override
protected void initHeaderFragmentContent(CardHeaderFragment
cardHeaderFragment) {
    cardHeaderFragment.addTitleTag("headerTitileTagOverdue",
getMessage("header.overdue"), NEGATIVE);
    cardHeaderFragment.setTitleTagVisible("headerTitileTagOverdue",
false);
    super.initHeaderFragmentContent(cardHeaderFragment);
}
```

Теперь в шапке документа будет отображаться, на сколько дней просрочен документ (см. рис. 31)

[МУ список](#) > [Выдача градус](#)

[Детали](#)   [Вложения](#)   [Ка](#)

СРОК ПРЕВЫШЕН НА 207 ДНЕЙ

Рисунок 31 – Отображение превышения срока



## 3.2 Считывание штрихкода

### 3.2.1 Имеющийся функционал считывания изображения

В системе присутствует сервис ExtHoldingReservationNumberService, суть которого в предоставлении методов считывания различной информации с документов при сканировании. В нём уже есть метод recognizeBarcode для поиска штрихкода (см. листинг 11).

Листинг 11 – Метод для поиска штрихкода на скане документа

```
protected boolean recognizeBarcode(PDPage page,
                                   BufferedImage image,
                                   ExtResultBuilder
resultBuilder)
    throws DocumentException, IOException {
    BufferedImage subImage = null;

    try {
        BufferedImage bufImage = getGrayScaleImage(image);

        subImage = getSubImageWithBarcode(bufImage);
        subImage = subImage == null ? bufImage : subImage;

        Result result = decodeBarcodeImage(subImage);
        ExtReservationNumber resNum =
findReservationNumber(result.getText());
        resultBuilder.addPage(page, bufImage, resNum);
        return true;
    } catch (NotFoundException exc) {
        resultBuilder.addPage(page, subImage);
        resultBuilder.qrNotFound = true;
    } catch (Exception e) {
        resultBuilder.addPage(subImage, e);
    }
    return false;
}
```

Метод getGrayScaleImage возвращает черно-белый скан нашего документа. Далее с помощью метода getSubImageWithBarcode передается подизображение с искомым штрихкодом. Если штрихкод не найден, декодировано будет все изображение. Наша задача реализовать метод getSubImageWithBarcode.

### 3.2.2 Поиск подизображения со штрих-кодом

Метод `getSubImageWithBarcode` (см. листинг 12) работает с полученными с помощью методов `getWidth` и `getHeight` значениями длины и ширины сканируемого документа. В зависимости от этих значений устанавливаются размеры подизображения, где должен располагаться штрихкод. На случай, если скан документа не подходит под условия, по умолчанию размеры подизображения будут установлены как 40% и 15% от ширины и длины соответственно.

Листинг 12 – Метод `getSubImageWithBarcode`, извлекающий подизображение со штрихкодом со скана документа

```
protected BufferedImage getSubImageWithBarcode(BufferedImage
image) {

    int width = image.getWidth();
    int height = image.getHeight();

    int subWidth = (int) Math.round(width / 2.5);
    int subHeight = (int) Math.round(height / 6.5);

    if (width >= 2300 && width <= 2600) { // 300 dpi
        subWidth = 780;
        subHeight = 320;
    } else if (width >= 1600 && width <= 2000) { // 200 dpi
        subWidth = 510;
        subHeight = 210;
    } else if (width >= 550 && width <= 700) { // 150 dpi
        subWidth = 392;
        subHeight = 192;
    }

    int x = width - subWidth;
    int y = height - subHeight;

    log.info("IMAGE params: " + width + "x" + height + " |
штрихкод " + subWidth + "x" + subHeight);
```

## Продолжение листинга 12

```
        log.warn("IMAGE params: " + width + "x" + height + " |  
        штрихкод " + subWidth + "x" + subHeight);  
        log.error("IMAGE params: " + width + "x" + height + " |  
        штрихкод " + subWidth + "x" + subHeight);  
  
        return image.getSubimage(x, y, subWidth, subHeight);  
    }
```

После того как размеры подсчитаны, определяется координата левого верхнего угла документа. Все значения выводятся в логи, а параметры передаются в метод `getSubimage`, возвращающий искомое изображение штрихкода.

### 3.3 Группа периодических поручений

#### 3.3.1 Кратко о сущностях

Все сущности доработки являются расширениями штатных сущностей системы, так как модификация базируется на уже существующем функционале группы поручений:



- `ScheduleTasksGroup` – оригинальная сущность.
- `ExtScheduleTask` – расширение сущности `ScheduleTask`.
- `ExtScheduleTrigger` – расширение сущности `ScheduleTrigger`.
- `ScheduleTaskGroupAttachment` – расширение сущности `Attachment`.

Создание сущностей происходит аналогично созданию документа муниципальных услуг в разделе 3.1.1. Атрибуты описаны в разделе 2.3

#### 3.3.2 Отмена периодических поручений в группе

Работа с основными методами будет происходить в контроллере экрана `ScheduleTaskGroupTaskFrame`. Этот экран не относится ни к одной из сущностей доработки, он был создан вручную для удобства работы и представляет собой таблицу со списком поручений и их атрибутами (см. рис. 32).

Периодические поручения

Добавить поручение | **Запустить поручения** |   | Отменить поручения

Дата создания	Содержание документа	Инициатор	Исполнитель	Ответственный исполни...	Дата след. старта	Распи
	<input type="text" value="Введите текст"/>	Administrator ▾	Выберите значе ▾	<input type="checkbox"/>		
	<input type="text" value="Введите текст"/>	Administrator ▾	Выберите значе ▾	<input type="checkbox"/>		

Рисунок 32 – Экран ScheduleTaskGroupTaskFrame

В контроллере экрана ScheduleTaskGroupTaskFrame уже есть метод init, инициализирующий экран. Создаём действие new CancelTasksAction(), выполняющееся при нажатии кнопки удаления.

Был создан класс CancelTasksAction, в котором и будут основные прописаны методы задачи. В первую очередь создаём конструктор (см. листинг 14). Именно с помощью этого конструктора мы до этого (см. листинг 13) создали action. При вызове конструктора происходит инициализация полей dataManager и taskControllerService с помощью метода AppBeans.get(): первое поле нужно для обращения к атрибутам сущностей в базе данных, второе поле – для обращения к сервису, который будет описан в дальнейшем.

Листинг 13 – Инициализация экрана ScheduleTaskGroupTaskFrame с помощью метода init, создание действия удаления

```
public void init(Map<String, Object> params) {
    super.init(params);
    tasksTable.addAction(new CancelTasksAction());
    tasksTable.addAction(new RemoveAction(tasksTable, false) {
        //Логика метода
    });
    //Прочая логика
}
```

#### Листинг 14 – Создание класса CancelTasksAction, содержащего основную логику задачи

```
protected class CancelTasksAction extends AbstractAction {

    protected DataManager dataManager;
    protected TaskControllerService taskControllerService;

    public CancelTasksAction() {
        super("cancelTasks");
        dataManager = AppBeans.get(DataManager.class);
        taskControllerService = AppBeans.get(TaskControllerService.class);
    }
    //Методы, которые будут описаны далее
}
```

При нажатии на кнопки экрана следующая логика определяется в первую очередь методом `actionPerform` (см. листинг 15). Обращаемся к поручениям группы и сохраняем их в коллекции.

#### Листинг 15 – Метод `actionPerform`, вызывающий модальное окно подтверждения удаления

```
public void actionPerform(Component component) {

    Collection<ExtScheduleTask> taskGroupTasks =
taskGroupTaskDs.getItems();

    if (!validateData(taskGroupTasks)) return;

    taskGroupTasks.stream().filter(t ->
!CollectionUtils.isEmpty(t.getTasks())).count();
    showOptionDialog(
        getMessage("dialogs.Confirmation"),
        String.format("Вы уверены, что хотите отменить
поручения?", taskGroupTasks.size()),
        MessageType.CONFIRMATION,
        new DialogAction[]{new
DialogAction(DialogAction.Type.OK) {
            public void actionPerform(Component
component) {
                cancelTasks(taskGroupTasks);
                taskGroupTaskDs.refresh();
            }
        }
    }
}
```

## Продолжение листинга 15

```
        }, new DialogAction(DialogAction.Type.CANCEL)
    {
        public void actionPerform(Component
component) {
        }
    }
    };
}
```

Теперь во избежание ошибок нужно убедиться, что поручения готовы к закрытию, для этого реализован метод `validateData` (см. листинг 16). Метод возвращает `True` в том случае, если коллекция не пуста, если все поручения принадлежат данной группе и в системе отсутствуют активные процессы, связанные с этими поручениями. Для поиска незавершенных процессов у поручений в группе реализован метод `doesTaskHaveActiveUnfinishedProc` (см. листинг 17).

Если проверка пройдена успешно, происходит подсчет всех задач и пользователю выводится диалоговое окно с просьбой подтвердить удаление (см. листинг 15). Если пользователь нажимает «Ок», то дальше выполнится логика вызова `actionPerform` уже внутри, содержащего метод `CancelTasks`, который и занимается удалением поручений. Иначе окно просто закроется и ничего не произойдет. Также обновляются данные группы.

Листинг 16 – Метод `validateData`, проверяющий готовность поручений к удалению

```
public boolean validateData(Collection<ExtScheduleTask> taskGroupTasks)
{
    if (CollectionUtils.isEmpty(taskGroupTasks)) {
        showNotification(getMessage("taskGroupTasksEmpty.Msg"),
NotificationType.WARNING);
        return false;
    } else if (!taskGroupTasks.stream().anyMatch(t ->
t.getTasks() != null)) {
        showNotification(getMessage("createdTasksEmpty.Msg"),
NotificationType.WARNING);
        return false;
    }
}
```

```
}
```

### Продолжение листинга 16

```
int countOfScheduledTasksWithActiveProc = 0;
for (ExtScheduleTask scheduleTask : taskGroupTasks) {
    if (scheduleTask.getTasks().stream().anyMatch(t
-> doesTaskHaveActiveUnfinishedProc(t))) {
        countOfScheduledTasksWithActiveProc++;
    }
}
if (countOfScheduledTasksWithActiveProc == 0) {
showNotification(getMessage("createdActiveTasksEmpty.Msg"),
NotificationType.WARNING);
    return false;
}
return true;
}
```

### Листинг 17 – Метод doesTaskHaveActiveUnfinishedProc, проверяющий, есть ли у поручения незавершенные процессы

```
protected boolean doesTaskHaveActiveUnfinishedProc(Task task) {
    if (!TsPersistenceHelper.isPropertiesAccessible(task,
"proc")) {
        LoadContext<Task> context = new
LoadContext<>(Task.class);
        context.setId(task.getId());
        task = dataService.load(context);
    }
    return BooleanUtils.isTrue(task.getProc() != null &&
!WfUtils.isCardInStateList(task, "Finished",
"FinishedByInitiator", "FinishedByController"));
}
```

Метод `cancelTasks` получает коллекцию поручений. Для каждого поручения при условии, что оно не равно `null` и определен ответственный исполнитель, флаг присутствия ответственного исполнителя меняется на `false`, данные сохраняются в базу данных. Далее для каждого поручения проверяется условие, есть ли у поручения подпоручения. Если есть они есть, вызывается сервис `taskControllerService` (см. листинг 19, 20), отменяющий эти подпоручения. Данные о количестве отмененных подпоручений выводятся пользователю.

### Листинг 18 – Метод cancelTasks, реализующий удаление поручений

```
protected void cancelTasks(Collection<ExtScheduleTask>
taskGroupTasks) {
    int countOfCanceledTasks = 0;
    for (ExtScheduleTask it: taskGroupTasks) {
        if (it != null && it.getResponsibleExecutor() != null &&
it.getResponsibleExecutor()) {
            it.setResponsibleExecutor(false);
            CommitContext commitContext = new CommitContext();
            commitContext.addInstanceToCommit(it);
            dataManager.commit(commitContext);
        }

        if (it != null && it.getTasks() != null) {
            for (Task t: it.getTasks()) {
                countOfCanceledTasks +=
taskControllerService.cancelTaskWithSubTasks(t);
            }
        }
    }

    if (countOfCanceledTasks > 0) {
        showNotification("Отменено поручений: " +
countOfCanceledTasks, NotificationType.HUMANIZED);
    } else {
        showNotification("Отсутствуют активные незавершенные
поручения", NotificationType.WARNING);
    }
    isEditable = true;
}
```

### Листинг 19 – Инициализация сервиса TaskControllerService, удаляющего подпоручения родительского поручения

```
import com.haulmont.thesis.core.entity.Task;
import java.util.UUID;

public interface TaskControllerService {
    String NAME = "ekaterinburg_TaskControllerService";
    int cancelTaskWithSubTasks(Task task);
}
```

Опишем функционал сервиса. В файле Services модуля Global создадим интерфейс будущего сервиса TaskControllerService (см. листинг 19), в котором



определим переменную имени и метод `cancelTaskWithSubTasks`, содержащий логику закрытия подпоручений.

В модуле `Services` создадим файл `TaskControllerServiceBean`, в котором и будет определен метод `cancelTaskWithSubTasks` (см. листинг 20). Удаление совершается при условии, что к поручению имеется доступ. Загружаем данные карточки поручения и, если найден незавершенный процесс, завершаем его. Далее циклом проходим по всем подпоручениям родительского поручения. Аналогично обращаемся к их данным и завершаем процессы, если есть незавершенные. У подпоручения могут быть свои подпоручения, их также завершаем. Метод возвращает количество отмененных поручений.

Проверка существования незавершенных процессов осуществлялась с помощью метода `hasActiveUnfinishedProc` (см. листинг 21), возвращающего `True`, если у поручения нет подпоручений или если подпоручения имеют флаг завершенности.

#### Листинг 20 – Логика сервиса `TaskControllerService`

```
@Service(TaskControllerService.NAME)
public class TaskControllerServiceBean implements
TaskControllerService {

    @Override
    public int cancelTaskWithSubTasks(Task card) {
        int countOfCanceledTasks = 0;
        if (!TsPersistenceHelper.isPropertiesAccessible(card,
"subCards", "proc")) {
            LoadContext<Task> context = new
LoadContext<>(Task.class);
            context.setId(card.getId());
            context.setView("with-subcards");
            card = dataService.load(context);
        }
        if (hasActiveUnfinishedProc(card)) {
            wfService.cancelProcess(card);
            countOfCanceledTasks++;
        }
        for (Card it: card.getSubCards()) {
            if (it instanceof Task) {
                Task subTask = (Task) it;
                if (!TsPersistenceHelper.isPropertiesAccessible
(subTask, "subCards", "proc")) {
```

```

(Task.class)      LoadContext <Task> context=new LoadContext <>
                  context.setId(subTask.getId());

```

#### Продолжение листинга 20

```

        context.setView("with-subcards");
        subTask = dataService.load(context);
    }
    if (hasActiveUnfinishedProc(subTask)) {
        wfService.cancelProcess(subTask);

        countOfCanceledTasks++;
    }

    if
(!CollectionUtils.isEmpty(subTask.getSubCards())) {
        countOfCanceledTasks +=
cancelTaskWithSubTasks(subTask);
    }
}
return countOfCanceledTasks;
}
}

```

#### Листинг 21 – Проверка незавершенных процессов поручения

```

protected boolean hasActiveUnfinishedProc(Task task) {
    return task.getProc() != null &&
!WfUtils.isCardInStateList(task, "Finished",
"FinishedByInitiator", "FinishedByController", "Canceled");
}
}

```

Разработка основного функционал закончена. Для наглядного представления алгоритма удаления поручений составим блок-схему.

Взаимодействие методов в процессе удаления поручений можно увидеть на блок-схеме (см. рис. 33).

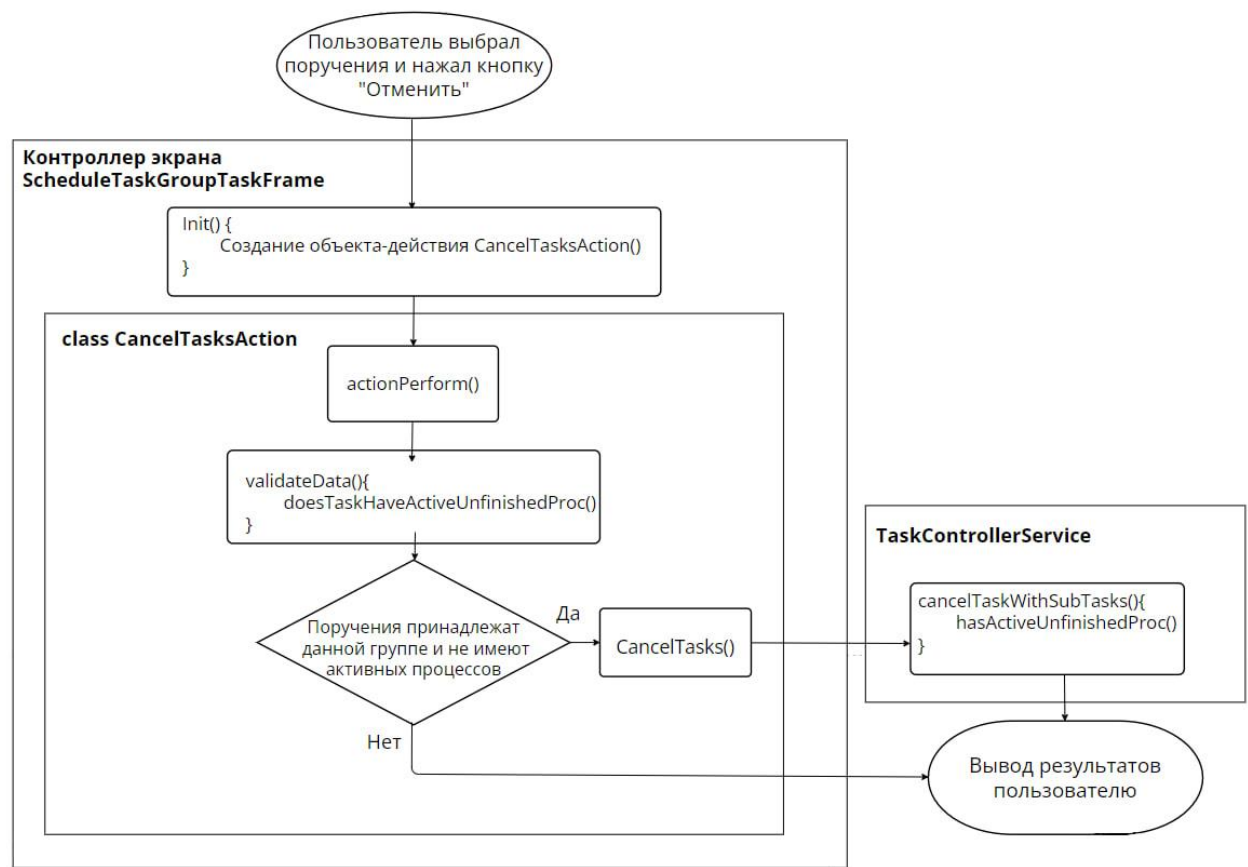


Рисунок 33 – Вызов методов в процессе удаления поручений

Источник: составлено автором

Таким образом, был реализован функционал, позволяющий безопасно и корректно закрывать поручения в группе периодических поручений. Стоит отметить, что сервис может использоваться в каких-либо других задачах, связанных с каскадным закрытием поручений, независимо от изначального объекта.

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы выполнено обследование проектной области, связанное с проектированием системы электронного документооборота на платформе Cuba. Изучен инструментарий платформы и технологии, позволяющие реализовать основные процессы жизнедеятельности документа на предприятии. На примере СЭД ТЕЗИС описаны задачи, решение которых необходимо для корректного функционирования предприятия. На основе описанных задач произведены доработки.

В ходе доработки был создан документ «Муниципальные услуги», произведена его настройка, корректное отображение в веб-приложении, автозаполнение полей и контроль за сроком сдачи. Создан метод для поиска штрихкода на скане документа. При разработке группы периодических поручений реализован сервис, позволяющий корректно закрывать поручения, созданные на основе родительских поручений.

В результате доработки рассмотрены некоторые основные операции, с которыми может столкнуться разработчик систем электронного документооборота, получен практический опыт в работе с платформой и её инструментами. Рассмотрены примеры бизнес-процессов, протекающих на предприятии и как они выглядят в системе, а также способ упростить такой процесс за счет уменьшения количества операций, автоматизации некоторых действий и расширения стандартного функционала.

В итоге цели выпускной квалификационной работы были достигнуты, поставленные задачи выполнены.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Автоматизация документооборота // ТЕЗИС система документооборота : [официальный сайт]. – URL: <https://www.tezis-doc.ru/features/workflow/> (Дата обращения: 24.03.2024)
2. Беляев А. Разработка на CUBA – большой шаг в сторону от Spring? // Jmix : [официальный сайт]. – 2018. – URL: <https://www.jmix.ru/cuba-blog/developing-with-cuba-a-big-shift-from-spring/> (Дата обращения: 20.03.2024)
3. НИИ «Восход» разработает типовую СЭД для госструктур на основе системы ТЕЗИС // Haulmont : [официальный сайт]. – 2020. – URL: <https://www.haulmont.ru/blog/nii-voskhod-razrabotaet-tipovuyu-sed-dlya-gosstruktur-na-osnove-sistemy-tezis/> (Дата обращения: 25.04.2024)
4. Платформа CUBA. Руководство по разработке приложений. Версия 7.2 : сайт. – URL: <https://doc.cuba-platform.com/manual-latest-ru/index.html> (Дата обращения: 05.02.2024)
5. Раис А. Электронный документооборот: переход к полной цифровизации документов и процессов // ELMA : [сайт]. – 2023. – URL: <https://www.elma-bpm.ru/journal/edo-perehod-k-polnoj-cifrovizacii-dokumentov-processov/> (Дата обращения: 14.04.2024)
6. Репин В. Моделирование бизнес-процессов в нотации BPMN : Пособие для начинающих. Часть I / Владимир Репин. – [б. м.] : Издательские решения, 2019. – 84 с. ISBN 978-5-4496-6989-6
7. ТЕЗИС. База знаний // Confluence : [сайт]. – URL: <https://confluence.haulmont.com/> (Дата обращения: 05.02.2024). – Режим доступа: для зарегистрированных пользователей.
8. ТЕЗИС Система управления документами и задачами // TADVISER Государство. Бизнес. Технологии : [сайт]. – 2023. – URL: [https://www.tadviser.ru/index.php/Продукт:ТЕЗИС\\_Система\\_управления\\_документами\\_и\\_задачами?erid=-](https://www.tadviser.ru/index.php/Продукт:ТЕЗИС_Система_управления_документами_и_задачами?erid=-) (Дата обращения: 10.03.2024)

9. Цифровая экономика РФ // Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации : [официальный сайт] . – 2023. – URL: <https://digital.gov.ru/ru/activity/directions/858/> (Дата обращения: 25.04.2024)

# ПРИЛОЖЕНИЯ

## Приложение 1









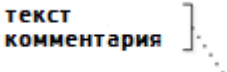



### Глоссарий

Сокращение/ термин	Расшифровка
СЭД	Система электронного документооборота
ERP	Enterprise resource planning - стратегия интеграции производства и операций, управления трудовыми ресурсами и финансового менеджмента, целью которой является оптимизация ресурсов
Github	Сервис, основанный на системе контроля версий Git, позволяющий создавать программные репозитории для совместной работы с проектом
Java EE	Платформа, которая предоставляет API и среду времени выполнения для разработки и запуска крупномасштабных приложений
Apache Tomcat	Контейнер сервлетов с открытым исходным кодом
API	Описание способов взаимодействия одной компьютерной программы с другими
Сервлет	Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ
Развертывание (приложения)	Все действия, которые делают программную систему готовой к использованию
Масштабирование (приложения)	Обработка растущего числа одновременных пользователей без ущерба для производительности веб-приложения
Vaadin	Фреймворк для создания RIA-веб-приложений

JVM	Java Virtual Machine - виртуальная машина Java - основная часть исполняющей системы Java, исполняющая байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java
JSON	Текстовый формат обмена данными, основанный на JavaScript
DOM	Независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-файлам
Gradle	Система автоматической сборки проекта
jBPM	Платформа для выполнения бизнес-процессов, управления бизнес-правилами и комплексной обработки событий.
Elasticsearch	Программная поисковая система
IntelijIDEA	Интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python.
BPMN	Система условных обозначений и их описания в XML для моделирования бизнес-процессов.
МУ	Муниципальные услуги
ПО	Программное обеспечение



Символы описания бизнес-процессов в BPMN-диаграмме

Графическое изображение символа	Описание
	Начало процесса
	<p>Пул – границы, в рамках которых протекает один процесс</p> <p>Дорожка – отображает разделение ролей пользователей</p>
	Конец процесса
	Задача или событие
	Шлюз – разбиение пути процесса, развилка
	Оператор И
	Оператор ИЛИ
	Поток операций - отображает порядок действий.
	Ассоциация – указывает на связь между элементами потока и информацией
	Аннотация (Комментарий)
	Хранилище данных
	Открытие модального окна
	Задача выполняется автоматически