

Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
Национальный исследовательский университет
«Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Панов Игнат Константинович

**РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ДЛЯ ПЛАТФОРМЫ
GREEN DATA**

Выпускная квалификационная работа

студента образовательной программы «Программная инженерия»
по направлению подготовки 09.03.04 Программная инженерия

Руководитель
Кандидат технических наук, доцент
кафедры информационных
технологий в бизнесе

О.Л. Викентьева

Пермь, 2025 год

Аннотация

Автор: Панов Игнат Константинович.

Наименование

работы:

«Разработка программных модулей для платформы Green Data».

Работа состоит из глоссария, введения, трех глав, заключения, библиографического списка и приложений.

Работа содержит 67 листов формата А4 и содержит 54 рисунка и 7 таблиц. Также, в работу включено 2 приложения.

Библиографический список состоит из 12 источников.

В первой главе проведен анализ процессов обработки данных с помощью платформы GreenData, описаны проблемы в текущих процессах партнеров, составлены модели рабочих процессов партнеров AS-IS, описаны методы решения проблем и проведен анализ существующих систем с архитектурной парадигмой «Распределенная файловая система». В результате были сформулированы пользовательские требования к доработкам в модули «Бизнес-процессы», «Файлы» и «Алгоритмы», составлены диаграммы прецедентов, выявлены атрибуты качества системы, составлены нефункциональные требования к доработкам в модули «Бизнес-процессы», «Файлы» и «Алгоритмы», а также составлено техническое задание к разрабатываемым дополнениям в эти модули.

Во второй главе представлено описание ключевых прецедентов и диаграммы последовательностей, описана интеграция средства криптографической защиты информации Crypto Pro CSP, спроектированы изменения в базу данных платформы GreenData и спроектированы доработки в модули «Бизнес-процессы», «Файлы» и «Алгоритмы» платформы GreenData.

В третьей главы описаны внесенные дополнения в базу данных платформы GreenData, разработанные дополнения для модулей «Бизнес-процессы», «Файлы» и «Алгоритмы», модификация образа сборки платформы GreenData путем установки на него средства криптографической защиты информации Crypto Pro CSP, а также показаны интеграционные и unit тесты для проверки работоспособности разработанных решений.

Оглавление

Глоссарий	4
Введение	6
Глава 1. Анализ предметной области и разработка требований	9
1.1 Анализ процессов обработки информации с помощью платформы GreenData	9
1.2 Методы решения проблем	15
1.3 Анализ существующих решений	20
1.4 Анализ требования к системе	22
1.5 Результат анализа процессов обработки информации с помощью платформы GreenData	27
Глава 2. Проектирование дополнений в модули «Бизнес-процессы», «Алгоритмы» и «Файлы»	28
2.1 Сценарии использования	28
2.2 Интеграция Crypto Pro CSP	35
2.3 Функциональные требования	35
2.4 Проектирование дополнений в базу данных платформы GreenData	36
2.5 Проектирование дополнений в модули «Бизнес-процессы», «Алгоритмы» и «Файлы»	38
Глава 3. Реализация дополнений в модули «Бизнес-процессы», «Файлы» и «Алгоритмы»	41
3.1 Дополнения в базу данных платформы GreenData	41
3.2 Реализация подмодулей для модулей «Бизнес-процессы», «Файлы» и «Алгоритмы».	46
3.3 Модификация образа сборки платформы GreenData	57
3.4 Интеграционные и unit тесты	57
Заключение	63
Библиографический список	64
Приложение А. Техническое задание	66
Приложение Б. Акт о внедрении	67

Глоссарий

Объект (экземпляр типа объектов) - отдельный элемент соответствующего типа, имеющий его атрибутный состав. Доступен для редактирования если не является системным.

Системный статус - атрибут, который присутствует у каждого объекта и регулирует возможность его редактирования. При установке этого статуса в качестве активного запрещает дальнейшее редактирование объекта пользователем. Любой объект, который является предустановленным на платформе или подготавливается в миграцию, является системным.

Тип объектов - объект типа «Тип объекта», который описывает структуру будущих его экземпляров (объектов). Для этого создаются атрибуты, выражения жизненного цикла, индексы, проверки и ограничения на экземпляры. В качестве дополнительных настроек можно задать иерархию, описание, версионность, запретить наследование и подключить соединение к внешней базе данных.

Атрибут - объект типа «Атрибут», который является характеристикой объекта. Может быть числом, строкой, логическим значением, датой и временем или ссылкой на другой тип объектов. Для объектной ссылки поддерживаются связи «1-к-1», «1-ко-многим» и «многие-ко-многим».

Реестр объектов - объект типа «Настройка базового реестра», который включает себя список объектов определенного типа, в том числе и объекты дочерних (наследованных) типов. В качестве настройки перед его отображением можно задать фильтрацию объектов, добавить виртуальные колонки, которые рассчитываются в момент открытия. Во время работы с реестром можно фильтровать объекты либо по одному, либо по всем атрибутам [1].

Визуал - объект типа «Визуал», который является экранной формой. Он существует для каждого объекта в системе. На визуале можно отобразить атрибуты типов объектов, виджеты, реестры, а также каждый из этих элементов поддерживает настройку.

Виджет - объект типа «Виджет», который является визуальным элементом, позволяющий получить доступ к тому или иному действию для решения отдельных задач. Все виджеты доступны каждому объекту в системе.

Бизнес-процесс - объект типа «ДО. Бизнес-процесс» в котором, с помощью BPMN нотации, предоставляется возможность автоматизации рабочих процесс. Для возможности запуска бизнес-процесса нужно указать обязательные атрибуты: тип маршрутного объекта и тип создаваемого экземпляра процесса. По одному маршрутному объекту можно запускать любое количество бизнес-процессов без ограничений. Во время запуска бизнес-процесса создается один его экземпляр.

Экземпляр бизнес процесса - объект типа «ДО. Экземпляр запущенного процесса» который создается во время инициализации бизнес-процесса по маршрутному объекту. Хранит в себе техническую информацию: версию, признак активности, прикрепленные к процессу файлы, каким образом был запущен и т.д.

Маршрутный объект - существующий в системе объект определенного типа, по которому запускается и выполняется экземпляр бизнес-процесса. В рамках бизнес-процесса на этапах и задачах происходит взаимодействие с значениями его атрибутов.

Этап бизнес-процесса - объект типа «ДО. Этап бизнес-процесса», который является любым исполняемым элементом на схеме бизнес-процесса. Например, задача или сервисное действие. При достижении этапа в экземпляре бизнес-процесса происходит его инициализация.

Задача бизнес-процесса - объект типа «ДО. Задача на процессе». Задача может быть сервисной (выполняться автоматически) или назначаемой на пользователя для ручного выполнения и завершения.

Маршрут бизнес-процесса - элемент системы, который доступен каждому объекту каждого типа, которые указаны как в бизнес-процессе как маршрутные. Включает в себя бизнес-процессы, запущенные по определенному объекту, а также их выполненные и активные задачи.

Excel-фильтр - компонент системы, который позволяет фильтровать объекты по значению их атрибутов.

Алгоритм - объект типа «Алгоритм», является последовательности вычислительных шагов, используемая для выполнения определенной задачи или преобразования входных данных в выходные [2].

Введение

GreenData - IT компания, основанная в 2014 году. Основной распространяемый продукт - low-code платформа, которая позволяет автоматизировать рабочие процессы без знаний в области программирования. Это достигается путем:

- проектирования бизнес-процессы с использованием BPMN нотации,
- представлением информации с помощью интерактивных информационных панелей,
- управлением потоком электронных документов с использованием редактируемых шаблонов и встроенного редактора,
- выполнением настраиваемых алгоритмов и пакетных действий,
- интегрированием сторонних системы,
- и другие возможности.

Информация в платформе хранится в виде настраиваемой объектной модели. Каждый элемент системы является объектом. Для описания структуры объектов используются типы.

Из-за того, что объекты часто подвержены изменениям, их визуальное отображение не может оставаться статичным для пользователя. Для настройки внешнего вида экземпляров используются визуалы. Визуал - объект типа «Визуал», является экранной формой.

Часто встречающиеся действия, например, прикрепления файлов к объекту из системы или с компьютера пользователя, выполняются с помощью виджетов.

Одной из ключевых возможностей автоматизации рабочих процессов пользователей на платформе являются бизнес-процессы. Для их проектирования используется BPMN нотация. Для использования доступны все ее элементы, начиная от задач и заканчивая шлюзами и событиями. Любой выполняемый элемент на схеме бизнес-процесса является этапом.

Для возможности запуска бизнес-процесса нужно указать тип объектов. После этого для объектов появляется возможность запускать связанные с их типом бизнес-процессы. Такие объекты называются маршрутными. Для одного маршрутного объекта можно запустить неограниченное количество бизнес-процессов.

В экземпляре бизнес-процесса при достижении на BPMN схеме этапа, который является пользовательской задачей или сервисным действием, создается его экземпляр. После инициализации, пользовательская задача назначается на определенного пользователя или же их группу. В экземпляре пользовательской задачи можно использовать виджеты.

В свою же очередь, экземпляр сервисного действия выполняется автоматически после их инициализации. Сервисное действие выполняет настроенные на этапе пользователем действия, а именно: меняет значения атрибутов у объектов, копирует или создает дочерние объекты, а также выполняет алгоритмы. Алгоритм - объект типа «Алгоритм», является последовательностью вычислительных шагов, используемой для выполнения определенной задачи или преобразования входных данных в выходные [2].

В совокупности все эти возможности предоставляют широкий функционал, что позволяет снизить количество ошибок в рабочих процессах и затраты времени на выполнение рутинных задач.

Решение проблем партнеров является первостепенной задачей при разработке и сопровождении платформы. Во время эксплуатации платформы GreenData выявились проблемы с доступностью поиска задач на визуале маршрута объекта, а также отсутствием возможности сохранения файлов в несколько хранилищ без использования внешних систем и возможности проверки целостности файлов на этапе бизнес-процесса.

За взаимодействие с файлами отвечает модуль «Файлы». Для хранения файлов можно создавать и настраивать хранилища разных видов. Платформа GreenData поддерживает скачивание, загрузку, удаление, а также обновление содержимого файлов.

Для построение маршрута бизнес-процесса по маршрутному объекту используется модуль «Бизнес-процессы». В маршрут включаются созданные и выполненные экземпляры этапов бизнес-процессов.

Для решения описанных выше проблем на серверной части платформы GreenData необходимо добавить в модуль «Бизнес-процессы» фильтры для визуала маршрута, в модуль «Алгоритмы» функцию для проверки целостности файла и в модуль «Файлы» возможность сохранения файла в несколько файловых хранилищ.

Объектом исследования являются процессы обработки информации с помощью платформы GreenData. Предмет исследования представляет собой модули «Алгоритмы», «Бизнес-процессы» и «Файлы» платформы GreenData.

Цель исследования: расширение функциональных возможностей модулей «Алгоритмы», «Бизнес-процессы» и «Файлы» платформы GreenData.

Для достижения цели необходимо решить следующие задачи:

1. Выполнить анализ требований к необходимым изменениям в модули «Алгоритмы», «Бизнес-процессы» и «Файлы» на серверной части платформы GreenData.
2. Спроектировать дополнение для модулей «Алгоритмы», «Бизнес-процессы» и «Файлы» на серверной части платформы GreenData.
3. Реализовать дополнение модулей «Алгоритмы», «Бизнес-процессы» и «Файлы» на серверной части платформы GreenData.
4. Разработать интеграционные и unit тесты для проверки работоспособности решения.
5. Внедрить разработанные решения в платформу GreenData.

Практической значимостью полученных результатов является расширение функционала платформы GreenData для автоматизации рабочих процессов партнеров и пользователей, снижение затрат времени на выполнение задач и рисков ошибок.

Глава 1. Анализ предметной области и разработка требований

В данной главе будут рассматриваться анализ предметной области, проблемы партнеров и их решения.

1.1 Анализ процессов обработки информации с помощью платформы GreenData

Проектирование и внедрение бизнес-процессов на платформе в рабочие процессы партнеров позволяет снять нагрузку с их работников и снизить риск ошибок в рабочих процессах. Сам бизнес-процесс не заменяет людей и не устраняет ошибки его исполнителей, однако, существенно снижает вероятность возникновения организационных ошибок [3].

Бизнес-процесс представляет из себя схему взаимодействия, по которому действуют пользователи для достижения результата [3]. Для его проектирования используется BPMN нотация (см. Рисунок 1).

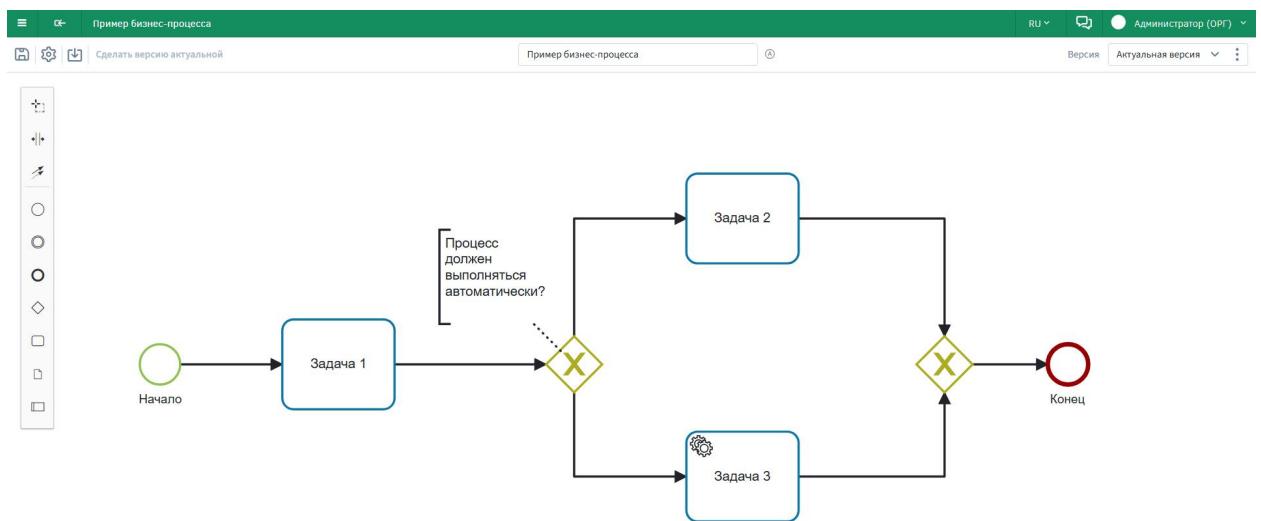


Рисунок 1 - Пример бизнес-процесса на платформе GreenData

«Задача 1» и «Задача 2» представляют из себя этапы, на основе которых будут генерироваться задачи в экземпляре бизнес-процесса. Задачи же в свою очередь будут назначаться на пользователей, которые указаны на этапе как «Исполнитель», а выполняться будут ответственным сотрудником или их группой (см. Рисунок 2).

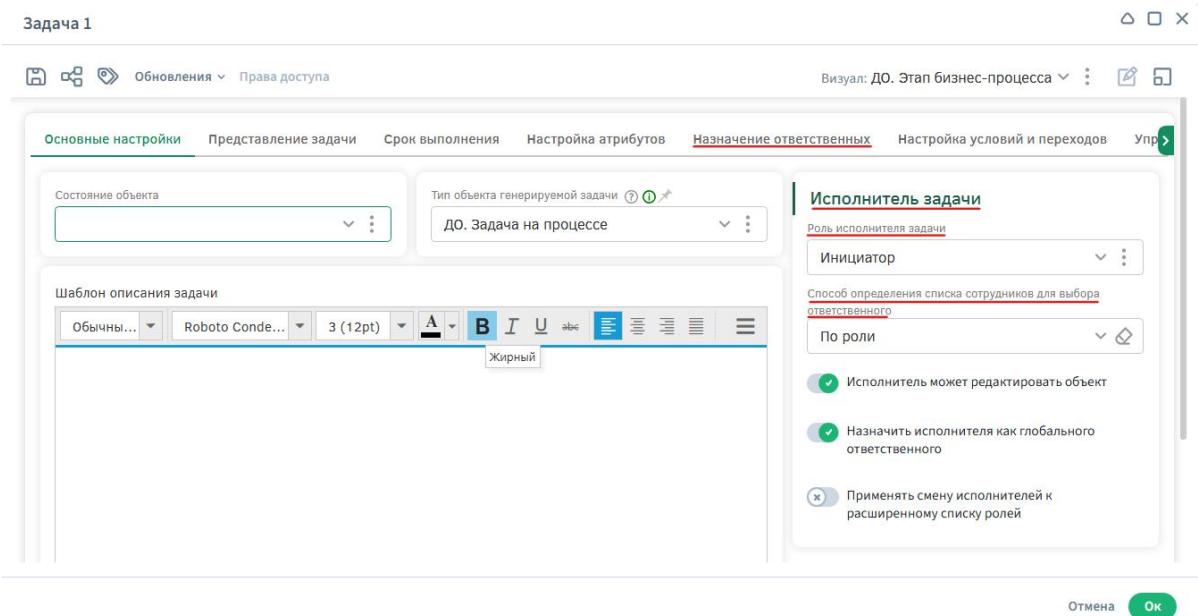


Рисунок 2 - Выбор исполнителя и ответственных сотрудников на этапе бизнес-процесса

«Задача 3» является сервисным действием, которое выполняется автоматически. Оно может быть исполняемым алгоритмом, вычислять и изменять значение атрибута определенного объекта, управлять правами пользователей на бизнес-процесс и так далее. Например, сервисное действие, запускающее алгоритм, можно увидеть ниже (см. Рисунок 3).

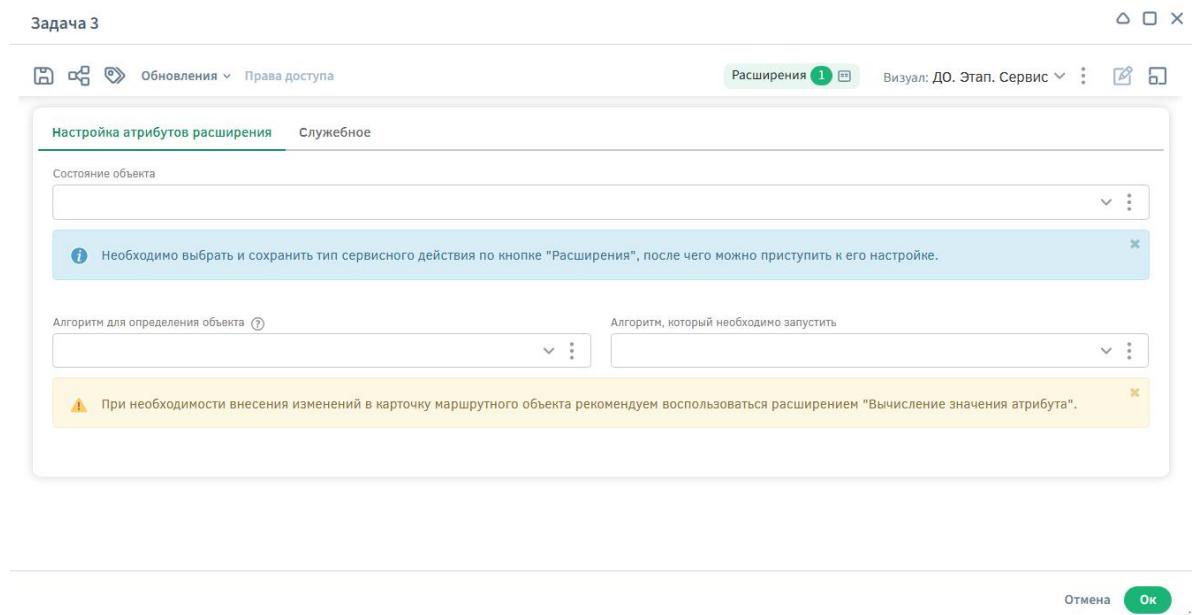


Рисунок 3 - Визуал настройки сервисного действия, запускающего алгоритм

Далее будут рассмотрены проблемы в существующем функционале.

1.1.1 Маршрут объекта

За формирование маршрута объекта отвечает модуль «Бизнес-процессы». Маршрутом объекта является визуал, на котором отображаются бизнес-процессы, запущенные по определенному объекту, а также их активные и завершенные задачи и этапы (см. Рисунок 4).

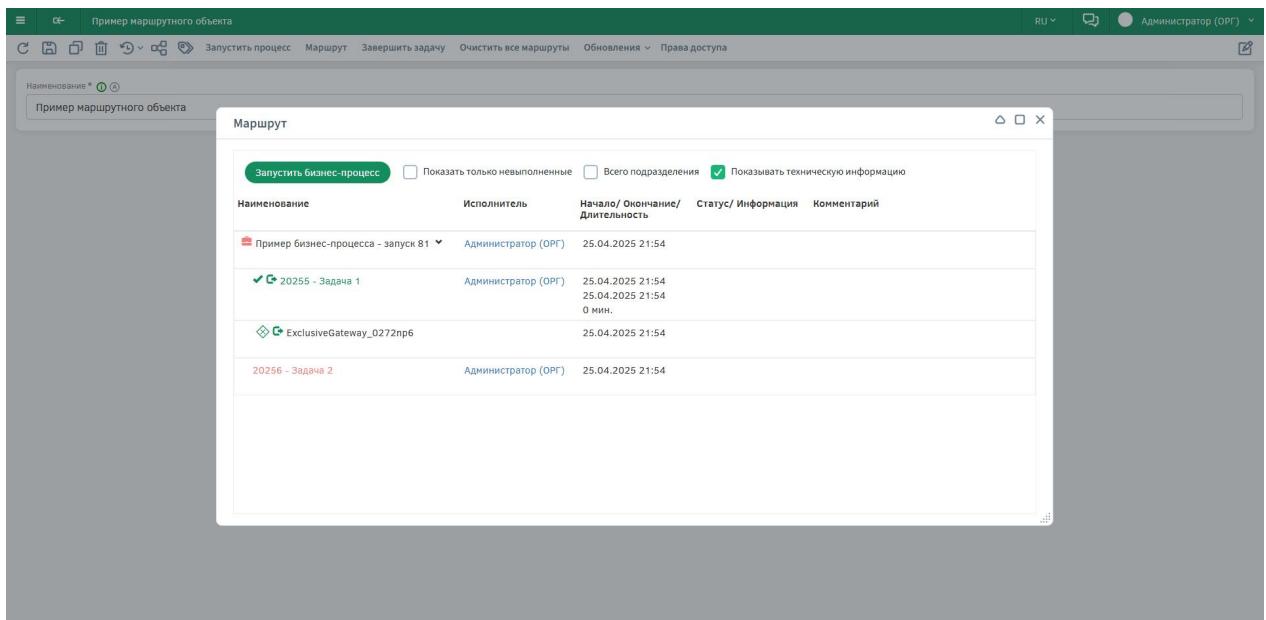


Рисунок 4 - Визуал маршрута объекта

В системе нет ограничений на количество использований типа объектов в качестве маршрутного для бизнес-процессов. Обычно это не является проблемой, так как под каждый бизнес-процесс создается и настраивается определенный тип, экземпляры которого будут выступать маршрутными объектами. Также нет ограничений и на количество повторных запусков бизнес-процессов по маршрутному объекту.

Все вышеперечисленное приводит к тому, что появляется большое количество экземпляров бизнес-процессов, их задач и этапов, что делает затруднительным визуальную ориентацию и поиск информации на данном визуале. Это приводит к ошибкам в рабочих процессах партнеров и пользователей тем, что нужная информация о выполненных задачах может быть пропущена.

1.1.2 Проверка целостности файлов

Документооборот неотъемлемая часть рабочих процессов. Его также можно настроить в рамках бизнес-процессов. Для этого нужно использовать «Входящие» (файлы из предыдущих этапов) и «Исходящие» (файлы, которые будут

использоваться дальше в бизнес-процессе) документы на этапе бизнес-процесса (см. Рисунок 5).

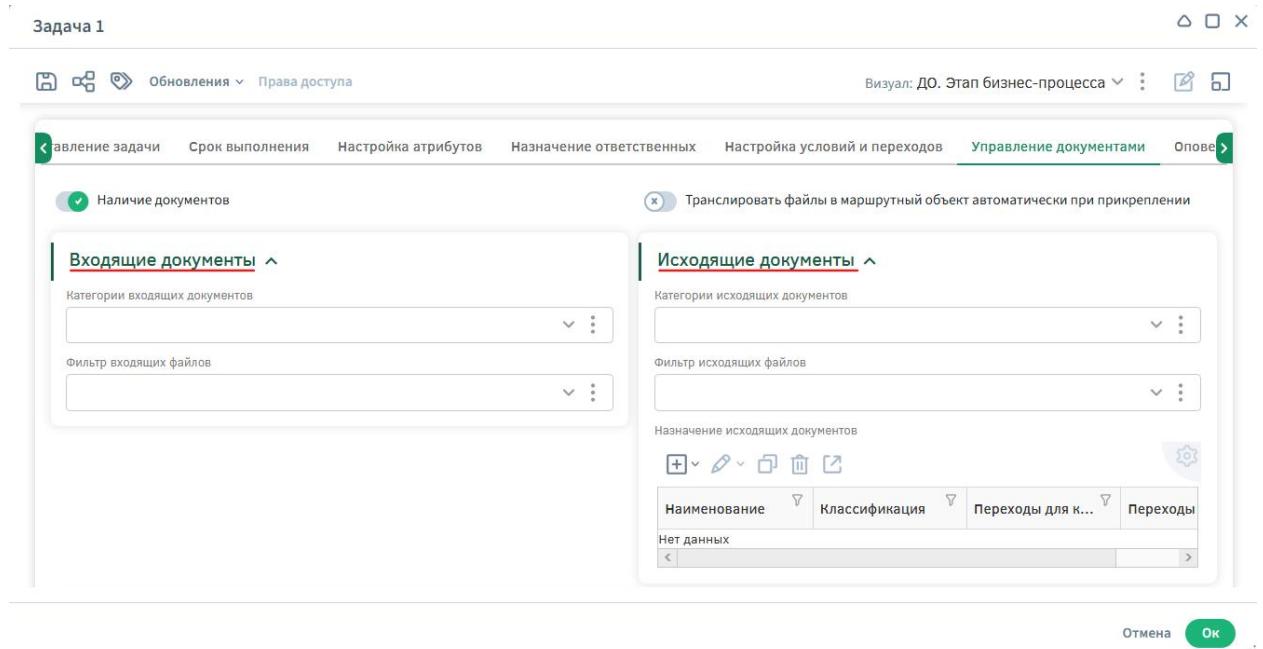


Рисунок 5 - Настройка «Входящих» и «Исходящих» документов

Пользователь, выполняющий задачу, не может влиять на изменения, которые могут быть внесены в прикрепляемые им файлы по мере выполнения бизнес-процессса.

В случае, если на определенном этапе выполнения экземпляра бизнес-процесса нужно проверить целостность файла, его приходится скачивать и проверять через внешние системы или самостоятельно просматривать его содержимое на платформе (см. Рисунок 6).

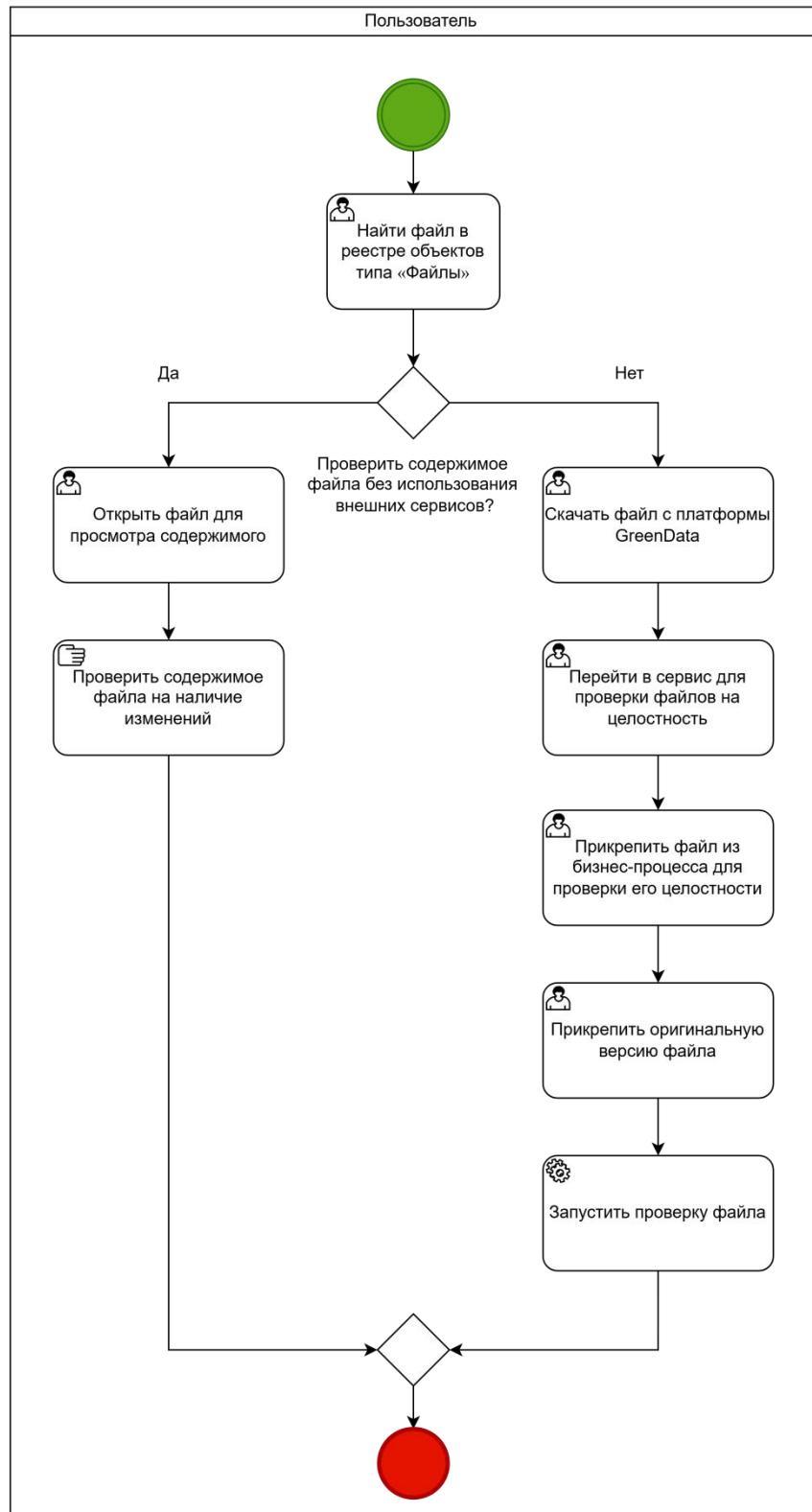


Рисунок 6 - AS-IS описание бизнес-процесса «Проверка целостности файла»

С учетом того, что количество файлов, которые попадают на этап, может достигать десятки, то проверка на целостность каждого из них занимает большое количество времени.

1.1.3 Сохранение файлов в разные хранилища

В бизнес-процессах часто требуется прикрепление файлов, которых нет в системе. Одной из основных возможностей, предоставляющей данный функционал, является виджет «Проводник» (см. Рисунок 7).

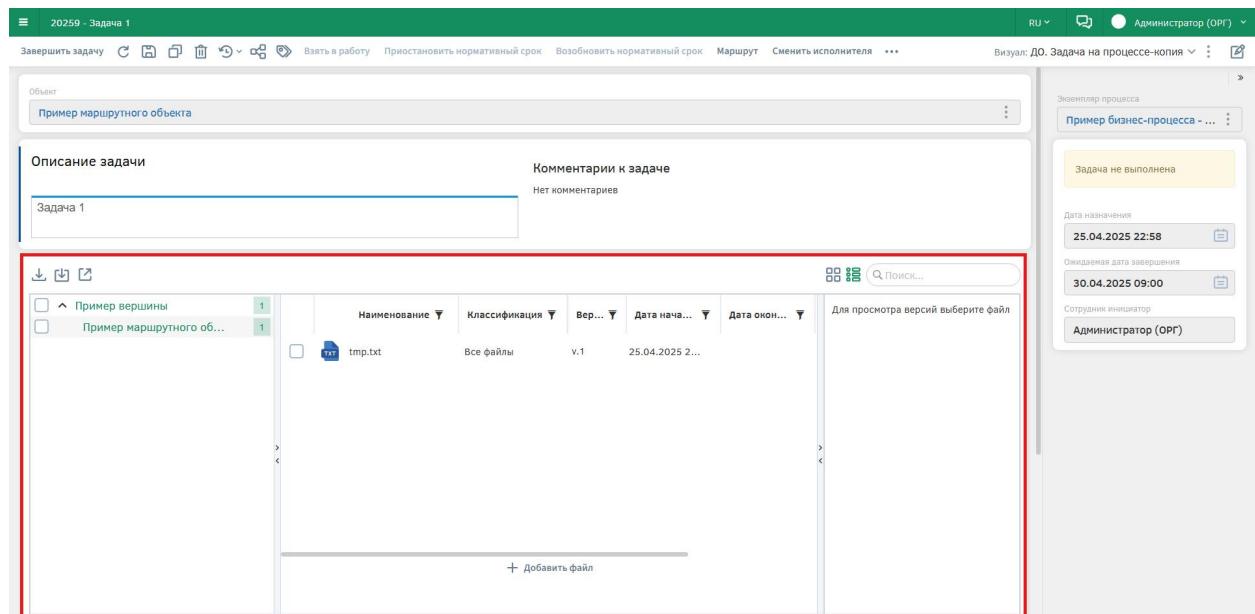


Рисунок 7 - Виджет «Проводник» на визуале задачи в экземпляре бизнес-процесса

Виджет - объект типа «Виджет», является визуальным элементом, позволяющим получить доступ к тому или иному действию для решения отдельных задач. Любой виджет можно добавить на визуал и они доступны каждому объекту в системе.

За хранение документов отвечает модуль «Файлы». Во время прикрепления файл сохраняется в хранилище, которое выставлено в системе по умолчанию. На данный момент существует четыре вида файловых хранилищ: внутри базы данных системы, внешняя сетевая директория, S3 (Simple Storage Service) и редактора документов (см. Рисунок 8).

Хранилище файлов			
Наименование	Идентификатор	Вид хранилища файлов	Перемещать в хранилище при замене содержимого
Хранилище файлов в БД 1	FS_PARTITION_001	0	
Хранилище файлов в БД 2	FS_PARTITION_002	0	
Ссылки на внешние файлы	FS_EXTERNAL_FILES	2	
Хранилище eDoc	EDOC_STORAGE	6	
Хранилище файлов 53	FS_S3	3	

Рисунок 8 - Реестр объектов типа «Хранилище файлов»

Администратор платформы может установить любой объект хранилища файлов по умолчанию и файлы будут сохраняться исходя из его внутренних настроек. Также, есть возможность использовать логическое или физическое удаление. В случае необходимости, администратор также может создать новое хранилище любого из вышеперечисленных видов.

Партнеры могут не хранить файлы во внутренней базе данных системы, исходя из соображений безопасности конфиденциальных данных (152-ФЗ «О персональных данных») [4]. Для бесперебойного доступа к ним сохраняют его в несколько сетевых директорий, находящихся в локальной сети без доступа в интернет.

На платформе на данный момент есть возможность конфигурировать такое хранилище, но файлы одновременно могут сохраняться только в одно. Из-за этого пользователям, имеющим доступ к файлам, содержащих персональные данные, приходится их скачивать и сохранять один файл в разные сетевые директории (см. Рисунок 9).

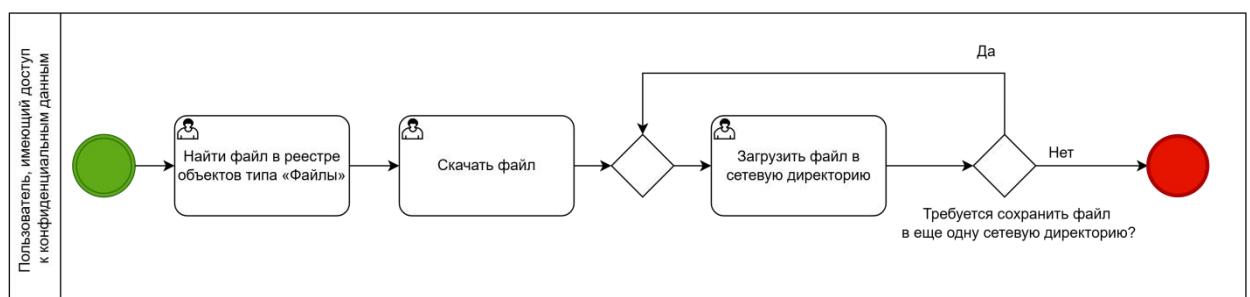


Рисунок 9 - AS-IS описание бизнес-процесса «Сохранение файла в сетевые директории»

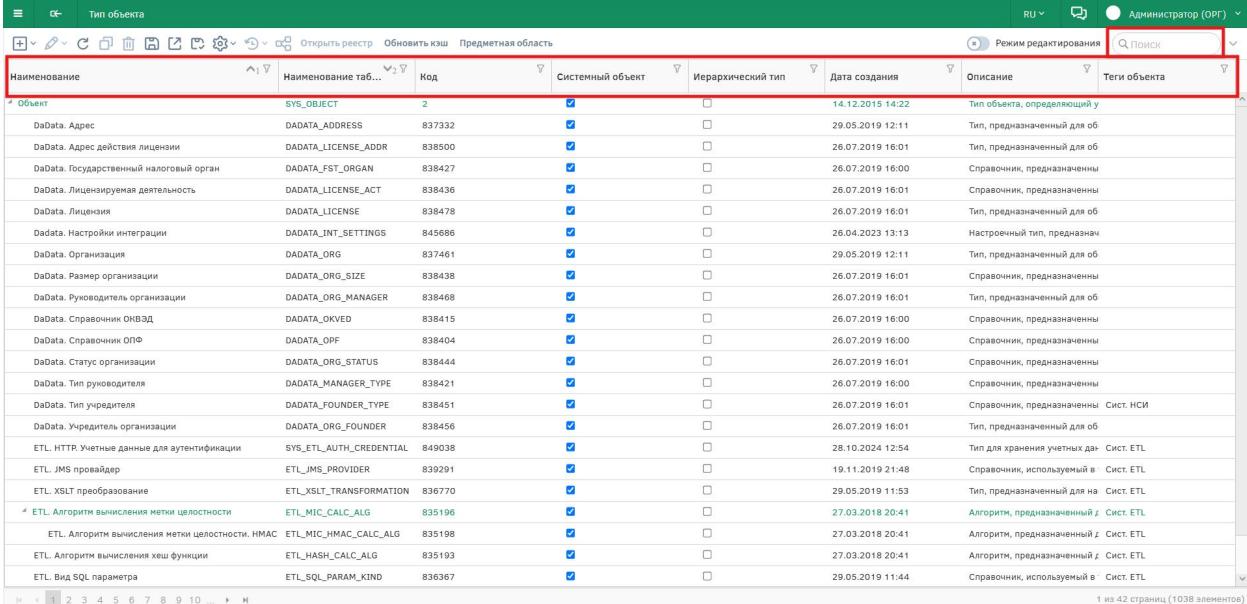
С большим количеством файлов, используемых в бизнес-процессах, и наличие нескольких сетевых директорий приводит к затратам времени пользователей и потенциальным ошибкам.

1.2 Методы решения проблем

Обзор будет посвящен решениям проблемы поиска информации по этапам на визуале маршрута, отсутствием возможности проверки целостности файлов и сохранения файла в несколько сетевых директорий.

1.2.1 Поиск информации на визуале маршрута

Для облегчения поиска информации на платформе GreenData существуют excel-фильтры (см. Рисунок 10).



Тип объекта							
Наименование							
	Наименование таб...	Код	Системный объект	Иерархический тип	Дата создания	Описание	Теги объекта
Объект	SYS_OBJECT	2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	14.12.2015 14:22	Тип объекта, определяющий у	
DaData. Адрес	DADATA_ADDRESS	827332	<input checked="" type="checkbox"/>	<input type="checkbox"/>	29.05.2019 12:11	Тип, предназначенный для об	
DaData. Адрес действия лицензии	DADATA_LICENSE_ADDR	838500	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Справочник, предназначены	
DaData. Государственный налоговый орган	DADATA_FST_ORGAN	838427	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:00	Справочник, предназначены	
DaData. Лицензируемая деятельность	DADATA_LICENSE_ACT	838436	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Справочник, предназначены	
DaData. Лицензия	DADATA_LICENSE	838478	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Тип, предназначенный для об	
Dadata. Настройки интеграции	DADATA_INT_SETTINGS	845686	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.04.2023 13:13	Настроекий тип, предназнач	
DaData. Организация	DADATA_ORG	837461	<input checked="" type="checkbox"/>	<input type="checkbox"/>	29.05.2019 12:11	Тип, предназначенный для об	
DaData. Размер организации	DADATA_ORG_SIZE	838438	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Справочник, предназначены	
DaData. Руководитель организации	DADATA_ORG_MANAGER	838468	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Тип, предназначенный для об	
DaData. Справочник ОКВЭД	DADATA_OKVED	838415	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:00	Справочник, предназначены	
DaData. Справочник ОПФ	DADATA_OPF	838404	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:00	Справочник, предназначены	
DaData. Статус организации	DADATA_ORG_STATUS	838444	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Справочник, предназначены	
DaData. Тип руководителя	DADATA_MANAGER_TYPE	838421	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:00	Справочник, предназначены	
DaData. Тип учредителя	DADATA_FOUNDER_TYPE	838451	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Справочник, предназначены Сист. НСИ	
DaData. Учредитель организации	DADATA_ORG_FOUNDER	838456	<input checked="" type="checkbox"/>	<input type="checkbox"/>	26.07.2019 16:01	Тип, предназначенный для об	
ETL. HTTP. Учетные данные для аутентификации	SYS_ETL_AUTH_CREDENTIAL	849038	<input checked="" type="checkbox"/>	<input type="checkbox"/>	28.10.2024 12:54	Тип для хранения учетных да-	Сист. ETL
ETL. JMS провайдер	ETL_JMS_PROVIDER	839291	<input checked="" type="checkbox"/>	<input type="checkbox"/>	19.11.2019 21:48	Справочник, используемый в	Сист. ETL
ETL. XSLT преобразование	ETL_XSLT_TRANSFORMATION	836770	<input checked="" type="checkbox"/>	<input type="checkbox"/>	29.05.2019 11:53	Тип, предназначенный для на	Сист. ETL
ETL. Алгоритм вычисления метки целостности	ETL_MIC_CALC_ALG	835196	<input checked="" type="checkbox"/>	<input type="checkbox"/>	27.03.2018 20:41	Алгоритм, предназначенный	Сист. ETL
ETL. Алгоритм вычисления метки целостности. HMAC	ETL_MIC_HMAC_CALC_ALG	835198	<input checked="" type="checkbox"/>	<input type="checkbox"/>	27.03.2018 20:41	Алгоритм, предназначенный	Сист. ETL
ETL. Алгоритм вычисления хеш функции	ETL_HASH_CALC_ALG	835193	<input checked="" type="checkbox"/>	<input type="checkbox"/>	27.03.2018 20:41	Алгоритм, предназначенный	Сист. ETL
ETL. Вид SQL параметра	ETL_SQL_PARAM_KIND	836367	<input checked="" type="checkbox"/>	<input type="checkbox"/>	29.05.2019 11:44	Справочник, используемый в	Сист. ETL

Рисунок 10 - Excel-фильтры на платформе GreenData

На данный момент они доступны только для реестров объектов. Реестр объектов - объект типа «Настройка базового реестра», который включает себя список объектов определенного типа, в том числе и объекты дочерних (унаследованных) типов. В качестве настройки перед его отображением можно задать фильтрацию объектов, а также добавить виртуальные колонки, которые рассчитываются в момент открытия реестра [1].

При открытии фильтра для определенного атрибута, открывается модальное окно, в котором отображаются все вхождения в колонке, загружаемые количеством от 10 до 50 штук «ленивой подгрузкой» (пагинацией) для оптимизации (см. Рисунок 11).

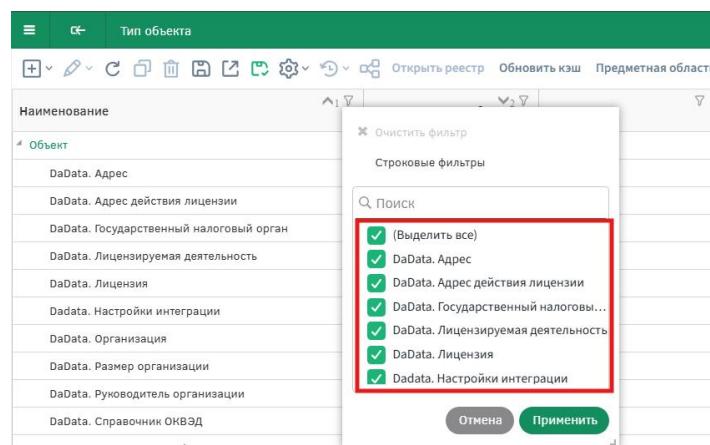


Рисунок 11 - Модальное окно строкового фильтра

Также доступно поле «Поиск» где посредством ввода пользователя фильтруются значения по вхождениям подстроки. Для любого вида атрибута это поведение одинаковое (см. Рисунок 12).

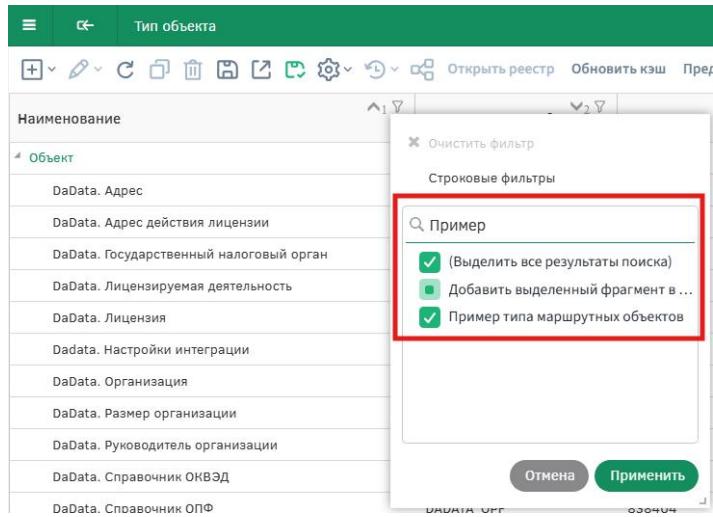


Рисунок 12 - Фильтрация атрибута через поле «Поиск»

При нажатии на кнопку «Применить» происходит фильтрация объектов по данному атрибуту (см. Рисунок 13).

Тип объекта									RU	Администратор (OPF)	Поиск
									Режим редактирования	Поиск	
Наименование	Наименование таб...	Код	Системный объект	Иерархический тип	Дата создания	Описание	Теги объекта				
Объект	SYS_OBJECT	2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	14.12.2015 14:22	Тип объекта, определяющий у...					
Пример типа маршрутных объектов	WAY_OBJECT_TYPE	B50527	<input type="checkbox"/>	<input type="checkbox"/>	25.04.2025 21:07	Тип для хранения данных "При...					

Рисунок 13 - Результатом работы фильтра по атрибуту

При повторном использовании фильтра для другого атрибута предыдущие настройки фильтрации сохраняются.

Для каждого вида фильтров существуют дополнительные настройки - «Пользовательские фильтры». Для строковых это: «Равно», «Не равно», «Начинается с», «Заканчивается на», «Содержит», «Не содержит». Также есть возможность объединить несколько этих операторов через условия «И» или «ИЛИ» (см. Рисунок 14).

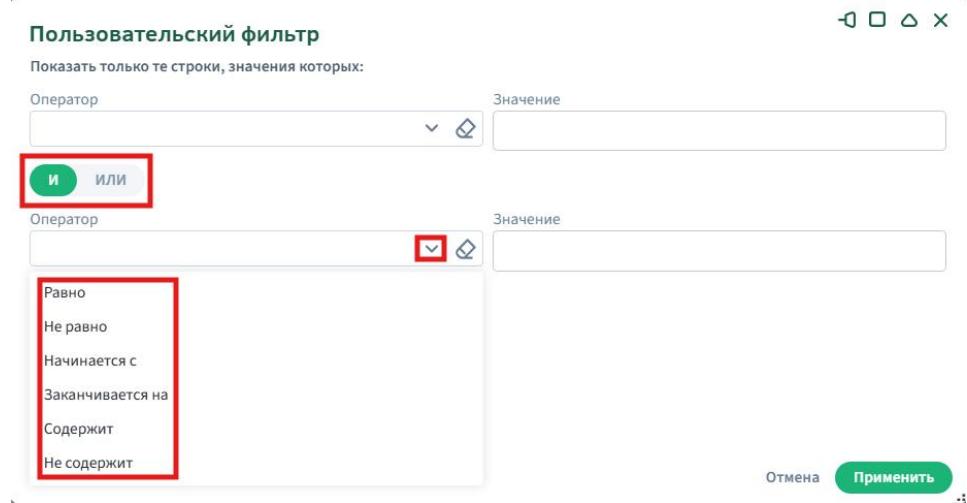


Рисунок 14 - Модальное окно «Пользовательские фильтры» для строковых атрибутов

Для «Пользовательских фильтров» по датам и числам можно задать операторы: «Равно», «Не равно», «Больше», «Больше или равно», «Меньше», «Меньше или равно». Их также можно объединять через условия «И» или «ИЛИ» (см. Рисунок 15).

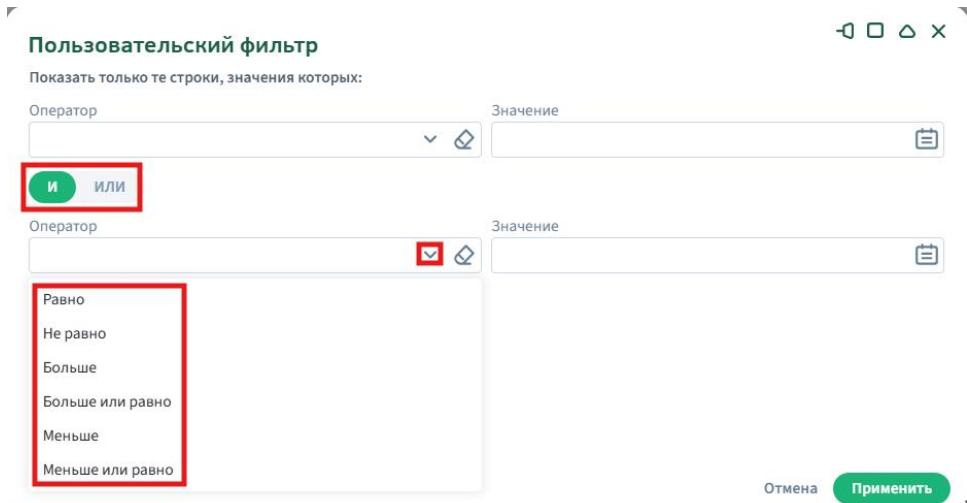


Рисунок 15 - Модальное окно «Пользовательские фильтры» для дат и чисел

Фильтрация поддерживается для каждого из возможных видов атрибутов: строковой, числовой, дата и время, логический, денежный и ссылка на объект.

На визуале маршрута отображаются пять колонок: «Наименование», «Исполнитель», «Начало/ Окончание/ Длительность», «Статус/ Информация»,

«Комментарий». Следовательно, все они являются датой или строковыми атрибутами.

Для построение запроса фильтрации с клиентской части строится с помощью запросов RSQL. RSQL (Role-based SQL) - расширение языка SQL (Structured Query Language), который с помощью динамических типов данных позволяет облегчить разработки и сопровождение системы [6].

Формирование маршрута происходит внутри модуля «Бизнес-процессы» и не использует логику формирования реестров, поэтому надо добавить функционал фильтрации, как в реестрах объектах, и обработку RSQL-запросов с клиентской части платформы GreenData для визуала маршрута.

1.2.2 Проверка целостности файлов

Sadeghi-Nasab и Rafe утверждают, что для проверки целостности данных идеально подходят функции хеширования [7].

Хеширование - процесс смены последовательности символов в более короткую последовательность определенного размера (256 или 512 бит) [7].

У функций хеширования есть стандарты. Один из них - ГОСТ 34.11. Разработан в Российской Федерации и принят как стандарт на национальном уровне [5].

При использовании криптографических функций, ИТ компания обязана пройти аккредитацию от лаборатории Федеральной Службы Безопасности. Интеграция аккредитованного средства криптографической защиты информации позволит облегчить прохождение данной процедуры.

На платформе уже присутствует использование аккредитованных средств криптографической защиты информации от компании Crypto Pro. Однако, используются библиотеки JCP старых версий и нет подключенных для использования функций хеширования.

В рамках решения проблемы проверки целостности файлов предлагается интеграция средства криптографической защиты информации Crypto Pro CSP и библиотеки JCSP последней версии - 5.0 для возможности вызова функции хеширования стандарта ГОСТ 34.11-2018 и добавления этой функции в модуль «Алгоритмы».

1.2.3 Возможность сохранения файла в несколько сетевых директорий

Для сохранения файла в несколько хранилищ используется архитектурная парадигма - «Распределенная файловая система». Согласно Tzong-Jye и др. архитектурная парадигма «Распределенная файловая система» распространена среди информационных систем с использованием электронных документов [8].

Распределенная файловая система - парадигма клиент/серверных информационных систем. Она улучшает возможность доступа к файлу благодаря хранением файлов на разных сетевых серверах, объединенных в одно пространство.

Внедрение архитектурной парадигмы в новый вид составных хранилищ позволит производить операции с файлом в нескольких файловых хранилищах.

1.3 Анализ существующих решений

В качестве существующих решений будут рассматриваться системы, реализующие архитектурную парадигму «Распределенная файловая система» для решения проблемы сохранения файла в несколько сетевых директорий.

1.3.1 Amazon S3 (Simple Storage Service)

Amazon Simple Storage Service или S3 - облачное объектное хранилище, созданное компанией Amazon в 2006 году и распространяемое от Amazon Web Services (AWS). Оно предназначено для хранения любых объемов данных с высоким уровнем доступности, безопасности и масштабируемости (Таблица №1).

Таблица 1 - Amazon S3

Преимущества	Недостатки
Встроенная возможность настройки масштабируемости системы и репликации хранимых данных.	Требуется оплата за хранение данных, выполнение запросов и используемый трафик.
Обеспечивает безопасность путем шифрованием данных на клиентской и серверных частях системы.	Зависимость от наличия доступа в интернет, что делает невозможным использование в локальной сети партнеров.
Версионность данных хранимых объектов.	Нет поддержки блокировок или транзакций.
	Не соблюдает требования 152-ФЗ «О персональных данных».

Amazon S3 не решает поставленную задачу, так как его невозможно использовать в локальной сети партнеров.

1.3.2 Azure Blob Storage

Azure Blob Storage - объектное хранилище, разработанное компанией Microsoft, ориентированное на корпоративных клиентов и облачные решения (Таблица №2).

Таблица 2 - Azure Blob Storage

Преимущества	Недостатки
Интеграция с Microsoft Active Directory.	Не соблюдает требования 152-ФЗ «О персональных данных».
Управление жизненным циклом объектов.	Для возможности настройки масштабирования требуется покупка подписки.
Возможность восстановления данных спустя некоторое время после удаления.	Зависимость от наличия доступа в интернет, что делает невозможным использование в локальной сети партнеров.
Возможность настройки масштабируемости системы и репликации данных.	

Azure Blob Storage не решает поставленную задачу, так как не соблюдает требования 152-ФЗ «О персональных данных».

1.3.3 Selectel Object Storage

Selectel Object Storage - объектное хранилище, разработанное компанией Selectel, для хранения и раздачи неограниченного объема структурированных и полуструктурных данных (Таблица №3).

Таблица 3 - Selectel Object Storage

Преимущества	Недостатки
Соблюдение 152-ФЗ «О персональных данных».	Минимальная поддержка международных стандартов.
Возможность настройки масштабируемости системы и репликации данных.	Центры обработки данных находятся только на территории Российской Федерации.
	Зависимость от наличия доступа в интернет, что делает невозможным использование в локальной сети партнеров.

Selectel Object Storage не решает поставленную задачу, так как центры обработки данных находятся только на территории Российской Федерации.

1.3.4 Соответствие критериям

На основе анализа текущих возможностей настройки файлового хранилища на платформе GreenData, а также процессов партнеров по взаимодействию с сетевыми хранилищами, были составлены критерии для сравнения существующих решений.

Таблица 4 - Сравнение существующих решений по критериям

№	Критерий	Amazon S3	Azure Blob Storage	Selectel Storage Service
1	Соблюдение 152-ФЗ «О персональных данных».	-	-	+
2	Отсутствует зависимость от внешнего интернет соединения.	-	-	-
3	Возможность восстановления данных после удаления.	-	+	-
4	Поддержка масштабируемости.	+	+	+
5	Поддержка репликации данных.	+	+	+

Ни одна из существующих систем не соответствует критериям. Анализ этих решений позволит спроектировать требования для составного хранилища на платформе GreenData.

1.4 Анализ требований к системе

В рамках анализа требований к системе будут выявлены пользовательские требования и нефункциональные требования.

1.4.1 Пользовательские требования

На основе анализа процессов обработки информации на платформе GreenData, методов решения проблем и анализа существующих решений были составлены пользовательские требования.

Для проверки целостности файлов пользователь должен иметь возможность использовать и выполнять вызов функции стандарта ГОСТ 34.11-2018 с использованием средства криптографической защиты информации Crypto Pro CSP в алгоритмах для снятия хеша файла.

Для excel-фильтров в визуале маршрута пользователь должен иметь возможность:

1. Использовать строковые, а также фильтры по дате.
2. Задать «Пользовательские фильтры»: «Равно», «Не равно», «Содержит», «Не содержит», «Начинается с», «Заканчивается на» для строковых атрибутов.
3. Задать «Пользовательские фильтры»: «Равно», «Не равно», «Меньше», «Меньше или равно», «Больше», «Больше или равно» для атрибутов дат.
4. Объединять несколько «Пользовательских фильтров» через условия «И» или «ИЛИ» для строковых, а также атрибутов дат.
5. Отфильтровать определенный атрибут или все этапы по условию «Содержит».

Для использования распределенного файлового хранилища пользователь должен иметь возможность:

1. Указать распределенное файловое хранилище как хранилище по умолчанию.
2. Добавлять уже существующие файловые хранилища, в которые будет сохраняться файл.
3. Задать приоритет чтения файлов для хранилищ, входящих в распределенное файловое хранилище.
4. Указать логический или физический вид удаления файла из хранилищ.
5. Просматривать информацию о статусах операций над файлами в распределенном файловом хранилище.
6. Возможность сохранить, удалить, обновить, а также прочитать содержимое файла внутри распределенного файлового хранилища.

На основе пользовательских требований были составлены диаграммы прецедентов.

Для проверки целостности файлов в алгоритмах с использованием функции хеширования диаграмма прецедентов представлена ниже (см Рисунок 16).

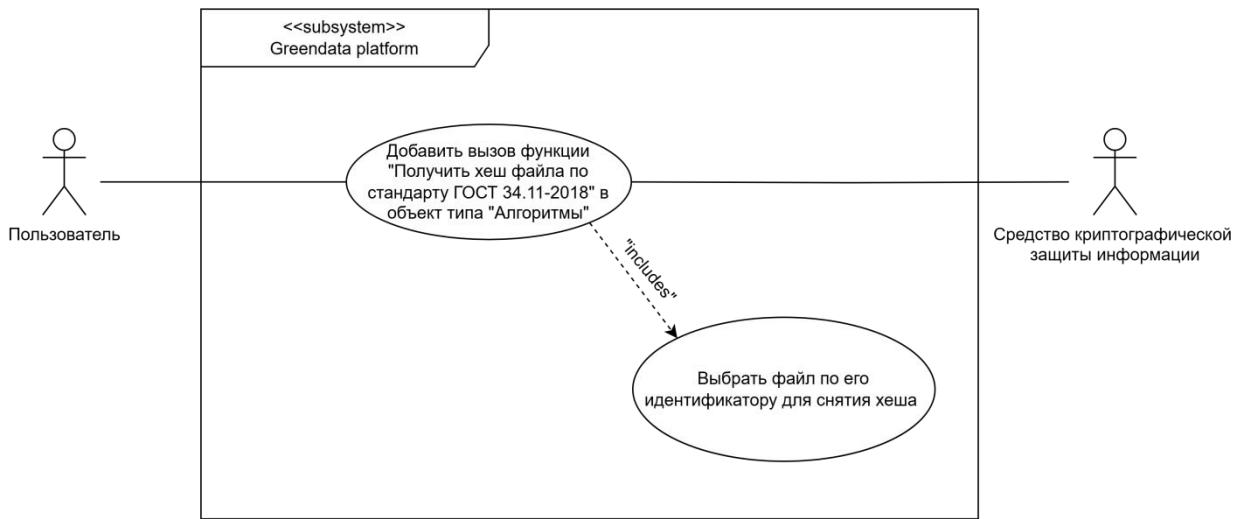


Рисунок 16 - Диаграмма прецедентов для вызова функции хеширования

Для excel-фильтров на визуале маршрута диаграмма прецедентов представлена ниже (см. Рисунок 17).

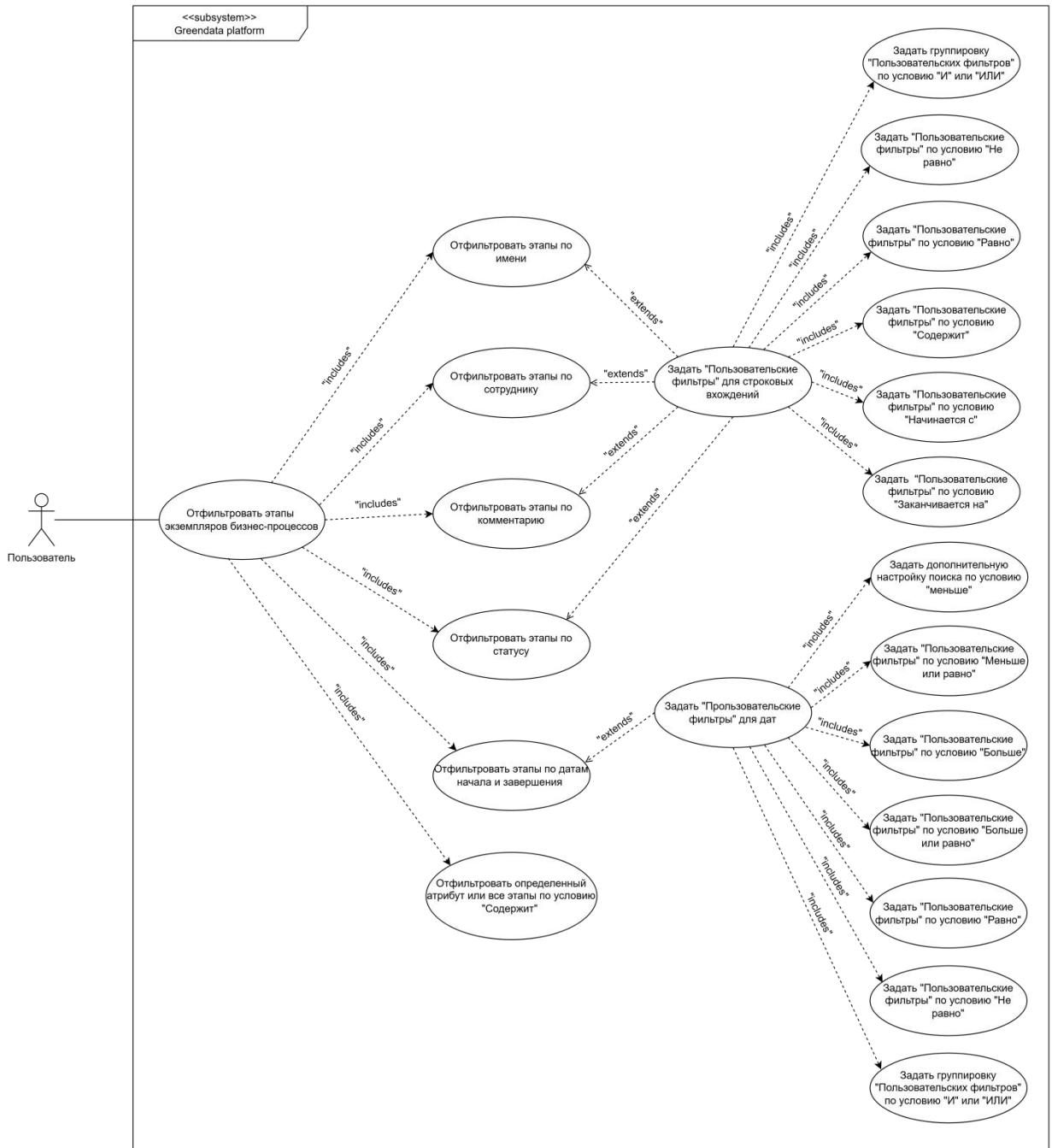


Рисунок 17 - Диаграмма прецедентов для excel-фильтров визуала маршрута

Для распределенного файлового хранилища диаграмма прецедентов представлена ниже (см. Рисунок 18).

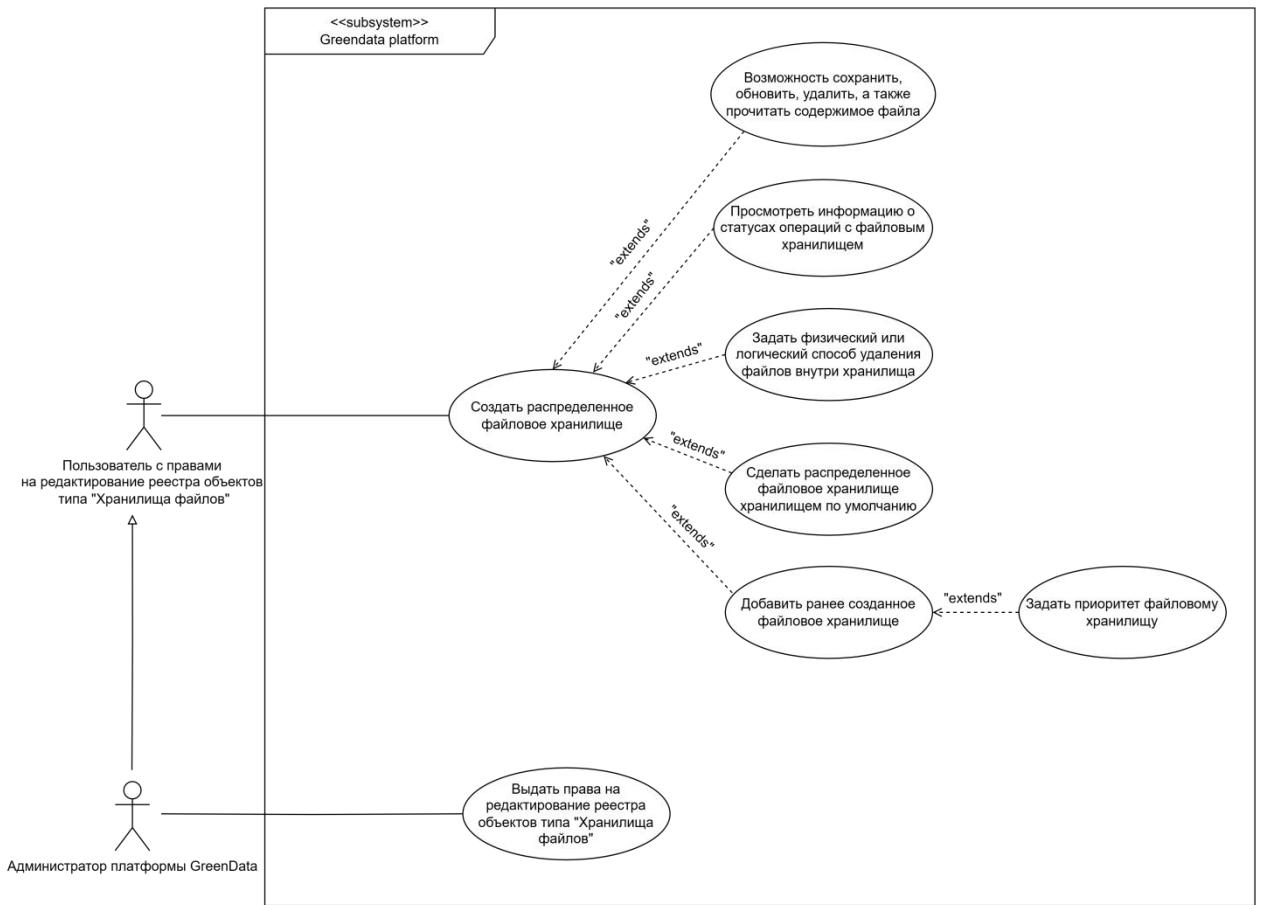


Рисунок 18 - Диаграмма прецедентов для распределенного файлового хранилища

Далее будут рассмотрены нефункциональные требования.

1.4.2 Нефункциональные требования

В результате анализа пользовательских требований были выделены следующие ключевые свойства системы:

1. Доступность.
2. Удобство использования
3. Масштабируемость и репликация данных

На основе этих свойств были выведены критерии оценивания степени соответствия требованиям:

1. Доступность:

- 1.1. Система должна обеспечивать возможность выполнения вызова функции хеширования у средства криптографической защиты информации.

- 1.2. Система должна обеспечивать возможность сохранения и чтения файлов при недоступности некоторых хранилищ внутри составного хранилища.
2. Удобство использования:
 - 2.1. Система должна обеспечивать пользователя текстом ошибки в случае его неправильных действий с настройкой.
 - 2.2. Система должна записывать и предоставлять информацию пользователю об ошибках в операциях с составным хранилищем.
 - 2.3. Система должна соблюдать дизайн систему платформы GreenData.
 - 2.4. Система должна отображать локализированный текст пользователю исходя из доступных языков на платформе GreenData.
3. Масштабируемость и репликация данных:
 - 3.1. Система должна обеспечивать масштабируемость и репликацию данных тела файлов через добавление новых хранилищ в «Составное хранилище».

1.5 Результат анализа процессов обработки информации с помощью платформы GreenData

В процессе анализа были:

1. Рассмотрены процессы управлением информации на платформе GreenData, бизнес-процессы партнеров и пользователей и проблемы в них
2. Рассмотрены методы решения проблемы поиска информации на визуале маршрута, а также проверки целостности файла и сохранения файла в несколько сетевых директорий
3. Рассмотрены существующие системы с архитектурной парадигмой «Распределенное файловое хранилище»
4. Выявлены требования к разрабатываемым дополнениям к модулям «Бизнес-процессы», «Алгоритмы» и «Файлы».
5. На основе пользовательских и нефункциональных требований было составлено техническое задание разрабатываемым дополнениям к модулям «Бизнес-процессы», «Алгоритмы» и «Файлы» (см. Приложение А).

Глава 2. Проектирование дополнений в модули «Бизнес-процессы», «Алгоритмы» и «Файлы»

В результате анализа пользовательских требований для дополнений в модули «Бизнес-процессы», «Алгоритмы» и «Файлы» были выделены следующие ключевые варианты использования:

1. Отфильтровать этапы экземпляров бизнес-процесса.
2. Добавить вызов функции «Получить хеш файла по стандарту ГОСТ 34.11-2018» в объект типа «Алгоритмы».
3. Создать распределенное файловое хранилище.

На основе данных прецедентов будет производиться проектирование дополнений в существующие модули «Бизнес-процессы», «Алгоритмы» и «Файлы». Будут составлены описание данных прецедентов, требование к интеграции Crypto Pro CSP, функциональные требования, дополнения в базу данных платформы GreenData и архитектура дополнений.

2.1 Сценарии использования

Для ключевых вариантов использования было составлено описание и построены диаграммы прецедентов.

2.1.1 Отфильтровать этапы экземпляров бизнес-процесса

Описание прецедента представлено ниже (Таблица №5).

Таблица 5 - Описание прецедента «Отфильтровать этапы экземпляров бизнес-процесса»

Идентификатор и название	UC-1 Отфильтровать этапы экземпляров бизнес-процессов.
Акторы	Основные: Пользователь.
Описание	Прецедент дает возможность пользователю отфильтровать этапы бизнес-процессов по их атрибутам.
Триггер	Пользователь открыл визуал маршрута бизнес-процесса у маршрутного объекта.
Предварительные условия	PRE-1 Пользователь прошел процесс авторизации и аутентификации на платформе GreenData. PRE-2 У пользователя есть права на просмотр маршрута бизнес-процесса у маршрутного объекта.
Выходные условия	POST-1 Этапы бизнес-процессов отфильтрованы исходя из заданных фильтров.

Основной поток	
Пользователь	Платформа GreenData
1. Пользователь выбирает колонку, по которой будет происходить фильтрация: наименование этапа; сотрудник; комментарий к этапу; статус этапа; дата начала и завершения этапа или же по всем колонкам одновременно и вводит значение, по которому будет происходить фильтрация.	
2. В случае необходимости пользователь настраивает «Пользовательские фильтры» и группирует их через условие «И» или «ИЛИ». 2.1. Для строковых атрибутов пользователь выбирает из условий: «Равно», «Не равно», «Содержит», «Не содержит», «Начинается с», «Заканчивается на» и вводит значение, по которому будет происходить фильтрация. 2.2. Для дат атрибутов пользователь выбирает из условий: «Равно», «Не равно», «Меньше», «Меньше или равно», «Больше», «Больше или равно» и вводит значение, по которому будет происходить фильтрация.	
3. Пользователь нажимает кнопку «Применить».	4. Платформа GreenData строит маршрут объекта.
	5. Платформа GreenData фильтрует этапы в маршруте объекта исходя из заданной фильтрации.

Диаграмма последовательности представлена ниже (см. Рисунок 19).

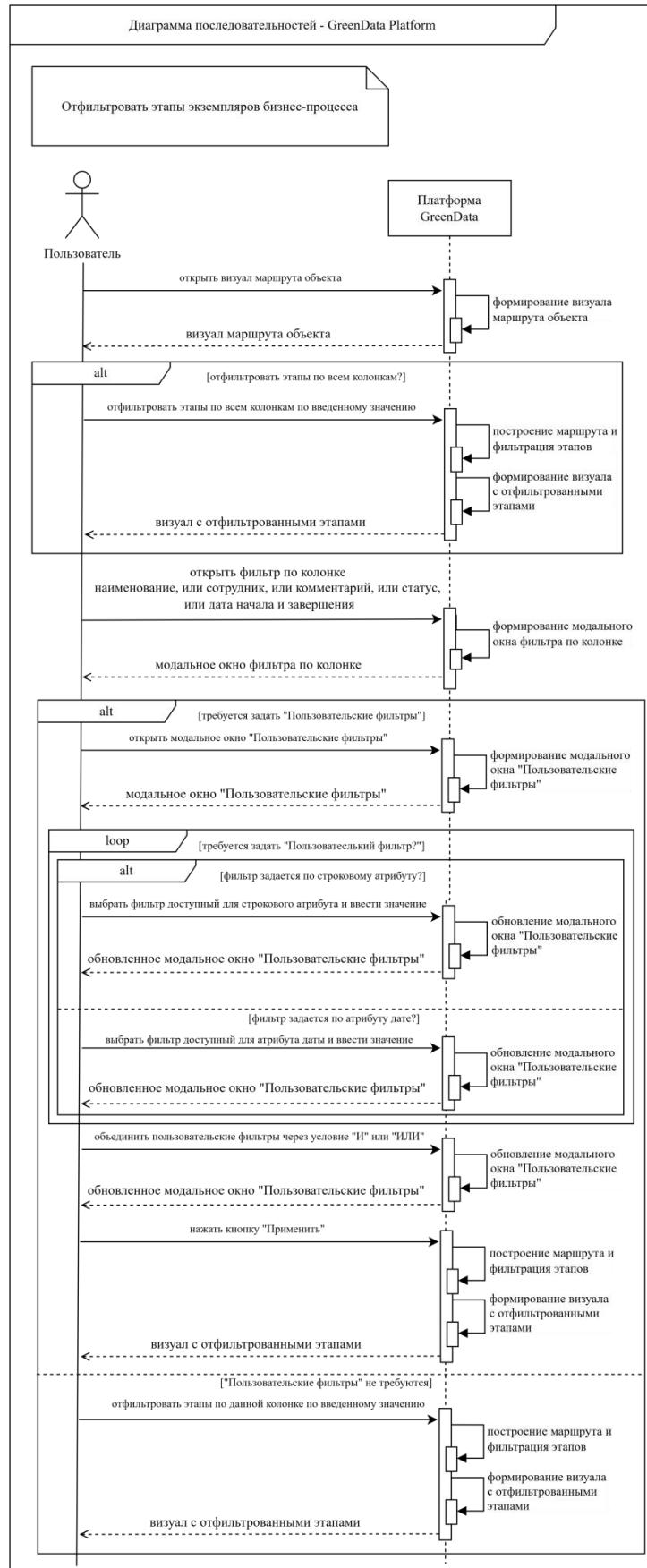


Рисунок 19 - Диаграмма последовательности для прецедента «Отфильтровать этапы бизнес-процессов»

**2.1.2 Добавить вызов функции
«Получить хеш файла по стандарту ГОСТ 34.11-2018» в объект типа
«Алгоритмы»**

Описание прецедента представлено ниже (Таблица №6).

**Таблица 6 - Описание прецедента
«Получить хеш файла по стандарту ГОСТ 34.11-2018»**

Идентификатор и название	UC-2 Добавить вызов функции «Получить хеш файла по стандарту ГОСТ 34.11-2018» в объект типа «Алгоритмы».		
Акторы	Основные: Пользователь. Дополнительные: Средство криптографической защиты информации.		
Описание	Прецедент дает возможность пользователю выполнить вызов хеш функции по стандарту ГОСТ 34.11-2018.		
Триггер	Пользователь открыл или создал объект типа «Алгоритмы».		
Предварительные условия	PRE-1 Пользователь прошел процесс авторизации и аутентификации на платформе GreenData. PRE-2 Средство криптографической защиты информации установлено на образ платформы GreenData или сервер заказчика.		
Выходные условия	POST-1 Алгоритм прошел валидацию и сохранился в системе. POST-2 Алгоритм выполнился и вернул хеш файла.		
Основной поток			
Пользователь	Платформа GreenData	Средство криптографической защиты информации	
1. Пользователь выбирает действие «Добавить функцию получения хеша файла по стандарту ГОСТ 34.11-2018» на визуальном отображение объекта.	2. Платформа добавляет на визуальное представление объекта типа «Алгоритмы» функцию получения хеша файла по стандарту ГОСТ 34.11-2018.		
3. Пользователь выбирает действие: «Сохранить объект».	4. Платформа производит валидацию и сохранение объекта.		
5. Пользователь выбирает действие «Запустить алгоритм».	6. Платформа производит запуск алгоритма и выполняет вызов функции получения хеша у криптографического средства защиты информации.	7. Средство криптографической защиты информации получает на вход поток битов файла и возвращает хеш.	

Диаграмма последовательности представлена ниже (см. Рисунок 20).

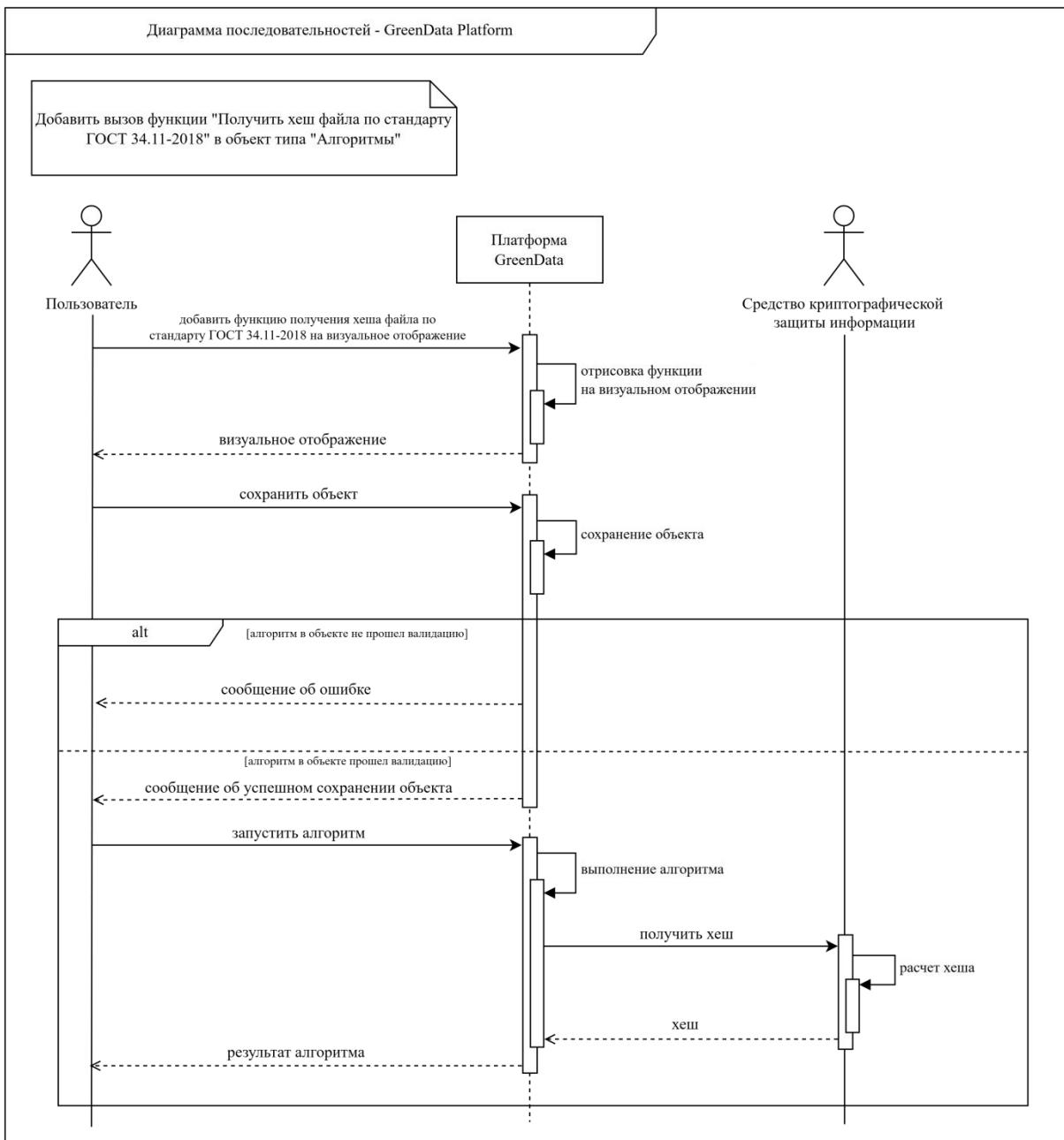


Рисунок 20 - Диаграмма последовательности для прецедента «Получить хеш файла по стандарту ГОСТ 34.11-2018»

2.1.3 Создать распределенное файловое хранилище

Описание прецедента представлено ниже (Таблица №7).

Таблица	7	-	Описание	прецедента
«Создать распределенное файловое хранилище»				
Идентификатор и название			UC-3 Создать распределенное файловое хранилище.	
Акторы			Основные: Пользователь, имеющий права на редактирование реестра объекта типа «Хранилище файлов».	
Описание			Прецедент дает возможность пользователю создать и настроить распределенное файловое хранилище.	

Триггер	Пользователь открыл реестр объектов типа «Хранилище файлов».
Предварительные условия	PRE-1 Пользователь прошел процесс авторизации и аутентификации на платформе GreenData. PRE-2 Пользователю были выданы права на редактирование реестра объектов типа «Хранилище файлов».
Выходные условия	POST-1 Объект прошел валидацию и сохранился в системе. POST-2 Файлы стали сохраняться исходя из настроек «Составного хранилища».
Основной поток	
Пользователь	Платформа GreenData
1. Пользователь выбирает действие «Создать объект» в реестре объектов типа «Хранилище файлов».	
2. Пользователь настраивает объект: определяет тип хранилища как «Составное хранилище», выбирает логическое или физическое удаление, а также является ли хранилищем по умолчанию, добавляет существующие хранилища и задает им приоритет.	
3. Пользователь выбирает действие «Сохранить объект».	4. Платформа производит валидацию и сохранение объекта.

Диаграмма последовательности представлена ниже (см. Рисунок 21).

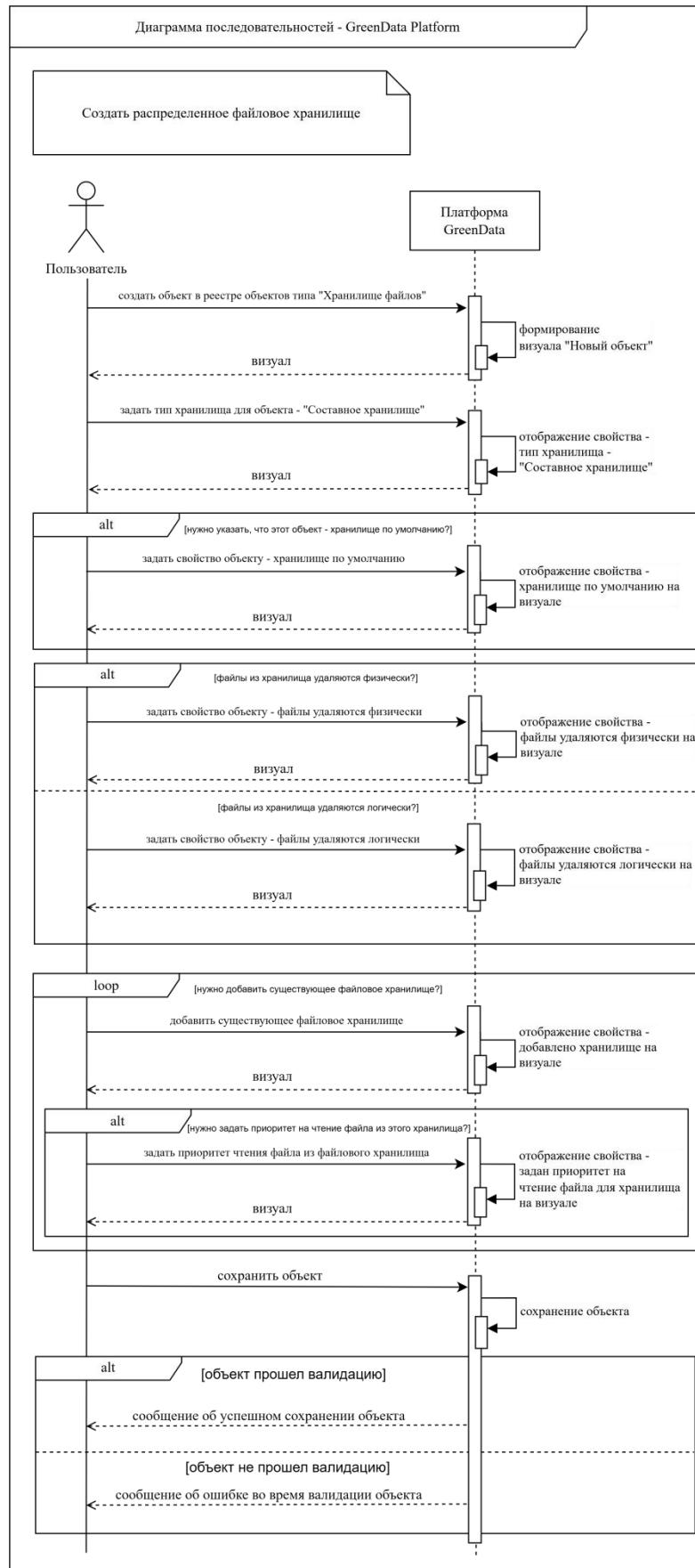


Рисунок 21 - Диаграмма последовательности для прецедента
«Создать распределенное файловое хранилище»

2.2 Интеграция Crypto Pro CSP

Библиотека JCSP версии 5.0 использует стандартный интерфейсы JCA-интерфейса Java - «MessageDigest». На основе данного интерфейса предоставляется реализация вычислителя хеша, который обращается в средство криптографической защиты информации Crypto Pro CSP для вычисления хеша.

Следовательно, для возможности вычисления хеша по стандарту ГОСТ 34.11-2018 средство криптографической защиты информации Crypto Pro CSP должно находиться в зоне видимости платформы GreenData, то есть быть предустановленным на образ платформы GreenData или находиться на сервере партнеров, где запущена платформа GreenData.

Для возможности обращения к средству криптографической защиты информации Crypto Pro CSP должна быть подключена библиотека JCSP версии 5.0.

2.3 Функциональные требования

Исходя из каждого ключевого варианта использования и его описания были составлены следующие функциональные требования:

1. Система должна фильтровать этапы экземпляров бизнес-процесса:
 - 1.1. Система должна фильтровать этапы бизнес-процесса по всем колонкам.
 - 1.2. Система должна фильтровать этапы по колонкам: наименование, сотрудник, комментарий, статус, дата начала и завершения.
 - 1.3. Система должна фильтровать этапы по «Пользовательским фильтрам»:
 - 1.3.1. Система должна объединять «Пользовательские фильтры» через условия «И» или «ИЛИ»
 - 1.3.2. Система должна использовать для строковых колонок в «Пользовательских фильтрах» условия фильтрации: «Равно», «Не равно», «Содержит», «Не содержит», «Начинается с», «Заканчивается на».
 - 1.3.3. Система должна использовать для колонок дат в «Пользовательских фильтрах» условия фильтрации: «Равно», «Не равно», «Меньше», «Меньше или равно», «Больше», «Больше или равно»

- 1.4. Система должна фильтровать этапы по значению колонки с условием «Содержит» если не указаны «Пользовательские фильтры».
2. Система должна создавать распределенные файловые хранилища:
 - 2.1. Система должна предоставлять возможность редактировать значения атрибутов у объекта типа «Хранилища файлов»: «Хранилище по умолчанию», «Вид хранилища», «Физическое или логическое удаление файлов из хранилища».
 - 2.2. Система должна предоставлять возможность добавлять существующие файловые хранилища в «Составное хранилище» и задавать им приоритет для чтения файлов.
3. Система должна предоставлять возможность добавлять вызов функции «Получить хеш файла по стандарту ГОСТ 34.11-2018» в объект типа «Алгоритмы»:
- 3.1. Система должна получать хеш файла из средства криптографической защиты информации Crypto Pro CSP.

2.4 Проектирование дополнений в базу данных платформы GreenData

В рамках разработки excel-фильтров для визуала маршрута и функции «Получить хеш файла по стандарту ГОСТ 34.11-2018» изменения в структуру базы данных платформы внесены не будут.

2.4.1 Проектирование изменений в базу данных платформы GreenData для распределенного файлового хранилища

Тип объекта в базе данных платформы GreenData представляет из себя сущность (таблицу). Атрибуты типа объектов - атрибут (колонка) сущности в базе данных.

Так как атрибуты: «Хранилище по умолчанию», «Вид хранилища», «Физическое или логическое удаление файлов из хранилищ» у сущности «Хранилище файлов» существуют, то требуется добавить атрибут с типом «Дочерний объект» - «Пакет хранилища файлов», который будет содержать хранилища, добавленные в «Составное хранилище».

Атрибут с типом «Дочерний объект» - объектный атрибут, который поддерживает связь «Многие-ко-многим» с другим типом объектов. Этот тип был создан для облегчения настройки атрибутной модели типа.

В атрибуте с типом «Дочерний объект» другой тип объектов можно выбрать, если в нем присутствуют два атрибута:

1. «Ссылка на родительский объект» - объектный атрибут, который указывает на объекты, в тип которого добавляется дочерний атрибут.
2. «Ссылка на объект» - объектный атрибут, который указывает на любой другой тип объектов, с которым формируется связь «Многие-ко-многим».

Атрибут «Пакет хранилища файлов» будет ссылаться на новый тип объектов «Пакет хранилища файлов». У него будет три атрибута:

1. «Порядок» - числовой атрибут. Он будет показывать приоритет хранилища для чтения файла.
2. «Ссылка на родительский объект» - объектный атрибут с ссылкой на тип объектов «Хранилище файлов». В данный атрибут будет записываться ссылка на объект типа «Хранилище файлов» с видом хранилища - «Составное хранилище».
3. «Хранилище файлов» - объектный атрибут с ссылкой на тип объектов «Хранилище файлов». В данный атрибут будет записываться ссылка на объект типа «Хранилище файлов», который будет добавляться в «Составное хранилище».

Для сохранения информации о статусах операций с составным хранилищем и просмотром ее пользователями будет добавлен тип «Информация о составном хранилище» с атрибутами:

1. «Файл» - числовой атрибут, в который будет записываться идентификатор файла.
2. «Пакет хранилища файлов» - объектный атрибут, ссылающийся на тип «Пакет хранилищ файлов», запись в котором будет объектом хранилища, в котором происходит операция с файлом.
3. «Актуальная версия» - логический атрибут. Указывает актуальная ли версия файла находится в определенном хранилище. Если при записи или обновлении файла в хранилище произошла ошибка, то данный атрибут принимает значение «Нет». В ином случае «Да».
4. «Время последнего обновления» - атрибут дата. Указывает на дату последней операции с файлом в хранилище.

5. «Статус последней операции с хранилищем» - строковой атрибут. Может принимать значения: «SAVED», «SAVING_FAILED», «DELETION_FAILED» и «READING_FAILED» в зависимости от того, как прошла последняя операция с файлом в данном хранилище.

В результате также была составлена ER-диаграмма (см. Рисунок 22).

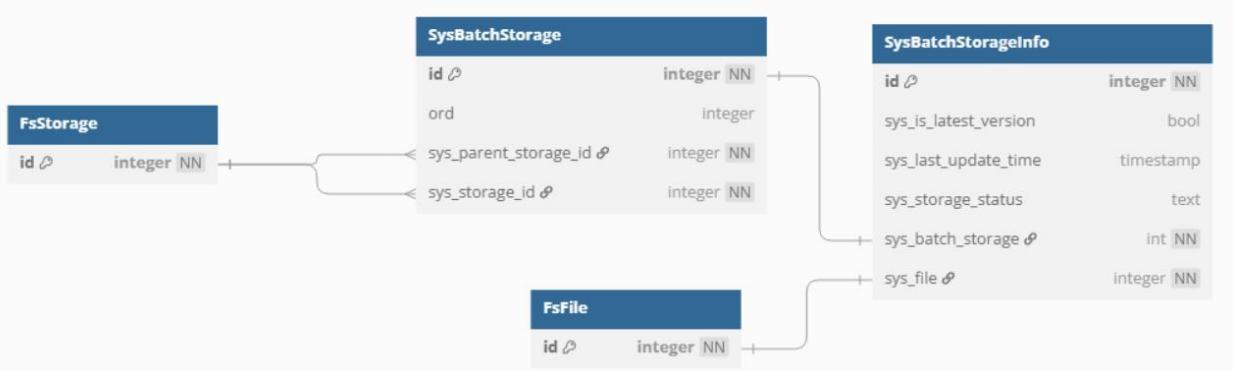


Рисунок 22- ER-диаграмма добавляемых сущностей в базу данных платформы GreenData

Во время разработки вышеописанные изменения будут произведены на платформе GreenData и «свернуты» в файлы формата «.guf» (Greendata Update File). Данные файлы созданы специально для внесения изменений в существующую объектную модель платформы GreenData на стендах партнеров и пользователей.

Данные файлы будут добавлены в список миграций платформы GreenData и при обновлении стендов платформы будет обновлена объектная модель этих стендов.

2.5 Проектирование дополнений в модули «Бизнес-процессы», «Алгоритмы» и «Файлы»

Платформа GreenData представляет из клиент-серверную систему. Дополнения в модули «Бизнес-процессы», «Алгоритмы» и «Файлы» представляют из себя подмодули, взаимодействующие с модулями.

В платформе GreenData создана архитектура с четырьмя уровнями [9]:

1. Фреймворки и драйверы. В данный уровень входят пользовательский интерфейс (клиентская часть платформы), а также базы данных и т.п.
2. Адаптеры интерфейсов. Данный уровень отвечает за взаимодействие с внешними компонентами системы, например, базой данных или клиентской частью.

3. Прикладные бизнес-правила (use-case). Данный уровень отвечает за поведение системы для выполнения определенный сценариев;
4. Бизнес-правила уровня предприятия. Данный уровень отвечает за поведение сущностей предметной области.

Проектируемые дополнения в модули «Бизнес-процессы», «Алгоритмы» и «Файлы» будут находиться на уровне прикладных бизнес-правил, а также были использованы принципы SOLID для облегчения сопровождения и тестирования этих дополнений [10].

2.5.1 Проектирование дополнений для модулей «Бизнес-процессы», «Алгоритмы» и «Файлы»

В рамках проектирования были выделены компоненты, находящиеся на уровне «Прикладные бизнес-правила» (см. Рисунок 23).

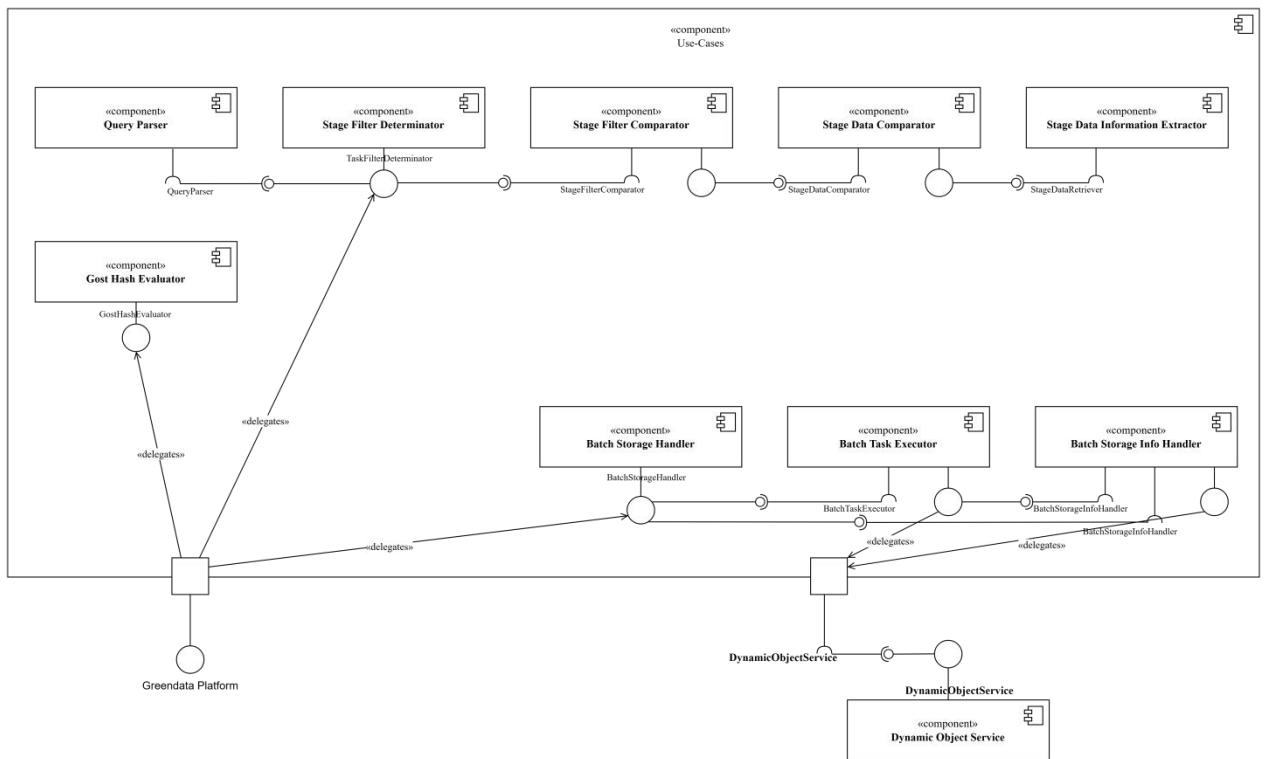


Рисунок 23 - Диаграмма компонентов для дополнения модулей «Бизнес-процессы», «Файлы» и «Алгоритмы»

Для возможности вычисления хеша служит компонент «Gost Hash Evaluator». Реализация данного компонента будет получать байты тела файла и вызывать функцию для вычисления хеша стандарта ГОСТ 34.11-2018 с помощью средства криптографической защиты информации Crypto Pro CSP.

За операции с файлом в «Составном хранилище» будет отвечать компонент «Batch Storage Handler». Реализация компонента будет отвечать за сохранение,

чтение, удаление и обновление тела файла. Реализация компонента «Batch Task Executor» будет в многопотоковом режиме выполнять операции сохранения и удаления файлов из «Составного хранилища». Реализация компонента «Batch Storage Info Handler» будет предоставлять возможность записи информации об операциях над файлами в «Составном хранилище».

Первым этапом для фильтрации этапов бизнес-процессов является определение RSQL-запроса. Реализация компонентов «Stage Filter Determinator» и «Query Parser» будут отвечать за это. Затем, реализация компонента «Stage Filter Comparator» будет определять для какого атрибута происходит фильтрация - строковой или атрибут дата. Реализация компонента «Stage Data Comparator» будет получать значение определенного атрибута у этапа с помощью реализации компонента «Stage Data Information Retriever» и определять проходит ли этап по заданному фильтру.

Глава 3. Реализация дополнений в модули «Бизнес-процессы», «Файлы» и «Алгоритмы»

В рамках данной главы будут рассмотрены дополнения, внесенные в базу данных платформы GreenData, реализованные подмодули для модулей «Бизнес-процессы», «Файлы» и «Алгоритмы», а также модификация образа сборки платформы GreenData с установкой средства криптографической защиты информации Crypto Pro CSP.

3.1 Дополнения в базу данных платформы GreenData

Хоть и для возможности использования вызова функции вычисления хеша стандарта ГОСТ 34.11-2018 в объектах типа «Алгоритмы» изменения в структуру базы данных платформы GreenData не требуются, но для отображения в списке доступных функций требуется добавить в миграции объекты.

Для распределенного файлового хранилища будут «свернуты» в миграцию два новых типов объектов и добавлен один новый атрибут в тип объектов «Хранилище файлов», а также стандартный объект с наименованием «Составное хранилище».

3.1.1 Дополнения в базу данных платформы GreenData для отображения функции вычисления хеша стандарта ГОСТ 34.11-2018 в объекта типа «Алгоритм»

Для отображения функции вычисления хеша стандарта ГОСТ 34.11-2018 в списке функций в объекте типа «Алгоритмы» нужно создать объекты в типах «Функция алгоритма».

Для функции вычисления хеша стандарта ГОСТ 34.11-2018 размером в 256 бит был добавлен следующий объект в тип «Функция алгоритма» (см. Рисунок 24).

The screenshot shows a form for creating a new algorithm object. The fields are as follows:

- Название *: gost3411_512
- Порядок: 868874
- Идентификатор *: GOST3411_512
- Описание: Результатом функции является 512 бит хеш строка
- Хint: Результатом функции является 512 бит хеш строка

Рисунок 24 - Объект функции вычисления хеша 256 бит стандарта ГОСТ 34.11-2018

Для функции вычисления хеша стандарта ГОСТ 34.11-2018 размером в 512 бит был добавлен следующий объект в тип «Функция алгоритма» (см. Рисунок 25).

Наименование *	<input type="text" value="gost3411_512"/>
Порядок	<input type="text" value="868874"/>
Идентификатор *	<input type="text" value="GOST3411_512"/>
Описание	<input type="text" value="Результатом функции является 512 бит хэш строки"/>
Хеш	<input type="text" value="Результатом функции является 512 бит хэш строки"/>

Рисунок 25 - Объект функции вычисления хеша 512 бит стандарта ГОСТ 34.11-2018

Оба этих объекта были «свернуты» в файл формата «.guf» и добавлен в миграцию (см. Рисунок 26).

```
</> master.xml </> 00000000000821_ADD_GOST_3411_AND_HASH_FUNC_GROUP_LINK.xml </> 1

1  <?xml version="1.0" encoding="UTF-8"?>
2  <databaseChangeLog
3      xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.4.xsd">
6
7      <changeSet id="00000000000821_CREATE_ALG_GOST3411_256" author="panov.ik">
8          <customChange class="pro.greendata.migration.guf.StandaloneGufChange">
9              <param name="fileName" value="classpath:config/liquibase/data/00000000000821_CREATE_ALG_GOST3411_256.guf"/>
10         </customChange>
11     </changeSet>
12
13     <changeSet id="00000000000821_CREATE_ALG_GOST3411_512" author="panov.ik">
14         <customChange class="pro.greendata.migration.guf.StandaloneGufChange">
15             <param name="fileName" value="classpath:config/liquibase/data/00000000000821_CREATE_ALG_GOST3411_512.guf"/>
16         </customChange>
17     </changeSet>
18
19 </databaseChangeLog>
```

Рисунок 26 - Инструкции по установке миграции для функций вычисления хеша

Миграция была добавлена в общий список миграций (см. Рисунок 27).

```
<!-- master.xml x 00000000000821_ADD_GOST_3411_AND_HASH_FUNC_GROUP_LINK.xml -->
2 <databaseChangeLog>
2141 <include file="classpath:config/liquibase/changelog/
2142 <include file="classpath:config/liquibase/changelog/
2143 <include file="classpath:config/liquibase/changelog/
2144 <include file="classpath:config/liquibase/changelog/
2145 <include file="classpath:config/liquibase/changelog/
2146 <include file="classpath:config/liquibase/changelog/
2147 <include file="classpath:config/liquibase/changelog/
2148 <include file="classpath:config/liquibase/changelog/
2149 <include file="classpath:config/liquibase/changelog/
2150 <include file="classpath:config/liquibase/changelog/00000000000821_ADD_GOST_3411_AND_HASH_FUNC_GROUP_LINK.xml"/>
2151 <include file="classpath:config/liquibase/changelog/
2152 <include file="classpath:config/liquibase/changelog/
2153 <include file="classpath:config/liquibase/changelog/
2154 <include file="classpath:config/liquibase/changelog/
2155 <include file="classpath:config/liquibase/changelog/
2156 <include file="classpath:config/liquibase/changelog/
2157 <include file="classpath:config/liquibase/changelog/
2158 <include file="classpath:config/liquibase/changelog/
2159 <include file="classpath:config/liquibase/changelog/
2160 <include file="classpath:config/liquibase/changelog/
2161 <include file="classpath:config/liquibase/changelog/
2162 <!-- <include file="classpath:config/liquibase/char
2163 <include file="classpath:config/liquibase/changelog/
2164 <include file="classpath:config/liquibase/changelog/
2165 <include file="classpath:config/liquibase/changelog/
2166 <include file="classpath:config/liquibase/changelog/
2167 <include file="classpath:config/liquibase/changelog/
2168 <include file="classpath:config/liquibase/changelog/
2169 <include file="classpath:config/liquibase/changelog/
2170 <include file="classpath:config/liquibase/changelog/
2171 <include file="classpath:config/liquibase/changelog/
2172 <include file="classpath:config/liquibase/changelog/
2173 <include file="classpath:config/liquibase/changelog/
2174 </databaseChangeLog>
```

Рисунок 27 - Миграция для функций вычисления хеша в списке миграций

3.1.2 Дополнения в базу данных платформы GreenData для распределенного файлового хранилища

Для возможности настройки распределенного файлового хранилища был добавлен новый атрибут «Пакет хранилища файлов» в тип объектов «Хранилище файлов» (см. Рисунок 28).

Наименование	Наименование поля для таблицы	Контекст (Тип)	Тип значения	Ссылка на тип	Обязательность	Владелец (Тип)
ETL для получения содержимого файла	DOWNLOAD_ETL	Хранилище файлов	Объект	Объект ETL	Нет	Хранилище файлов
ETL для сохранения содержимого файла	UPLOAD_ETL	Хранилище файлов	Объект	Объект ETL	Нет	Хранилище файлов
Java обработчик	STORAGE_JAVA_HANDLER	Хранилище файлов	Строка		Нет	Хранилище файлов
URL хранилища файлов	STORAGE_URL	Хранилище файлов	Строка		Нет	Хранилище файлов
Вид хранилища файлов	STORAGE_KIND	Хранилище файлов	Целое число		Да	Хранилище файлов
Идентификатор	IDENT	Хранилище файлов	Строка		Да	Хранилище файлов
Место хранения файлов	STORAGE_PLACE	Хранилище файлов	Строка		Нет	Хранилище файлов
Описание	NOTE	Хранилище файлов	Строка		Нет	Хранилище файлов
Пакет хранилища файлов	SYS_BATCH_STORAGE_ID	Хранилище файлов	Дочерний объект	Пакет хранилищ файлов	Нет	Хранилище файлов
Перемещать в хранилище при замене с	MOVE_TO_STORAGE	Хранилище файлов	Объект	Хранилище файлов	Нет	Хранилище файлов
По умолчанию	IS_DEFAULT	Хранилище файлов	Логический		Да	Хранилище файлов
Удалять файлы без возможности восснти	SYS_FS_IS_FILE_DEL_PHYS	Хранилище файлов	Логический		Нет	Хранилище файлов

Рисунок 28 - Атрибут «Пакет хранилища файлов» в типе объектов «Хранилище файлов»

Атрибут «Пакет хранилища файлов» содержит следующие настройки (см. Рисунок 29).

Наименование *	SYS_BATCH_STORAGE_ID
Тип *	Пакет хранилищ файлов
Обязательность *	Нет
Ссылка на родительский объект	

Рисунок 29 - Настойки атрибута «Пакет хранилища файлов»

Атрибут «Пакет хранилища файлов» ссылается на тип объектов «Пакет хранилищ файлов» (см. Рисунок 30).

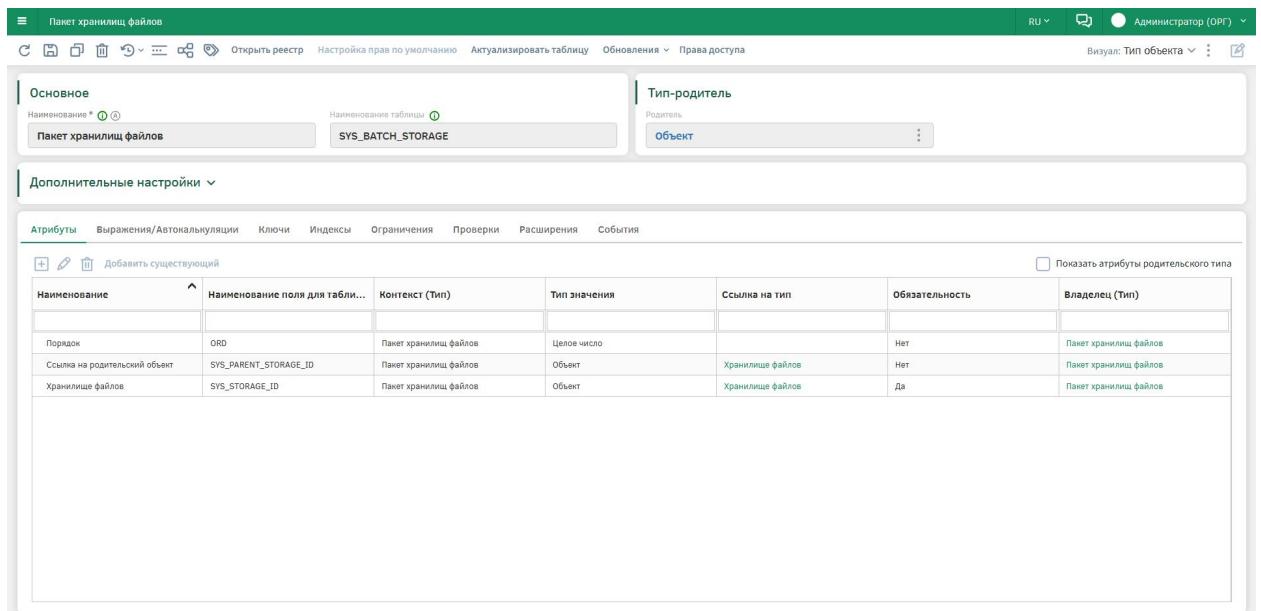


Рисунок 30 - Настройка типа объектов «Пакет хранилищ файлов»

Также, для возможности просмотра информации об операциях с файлом в распределенном хранилище был создан тип объектов «Информация о составном хранилище» (см. Рисунок 31).

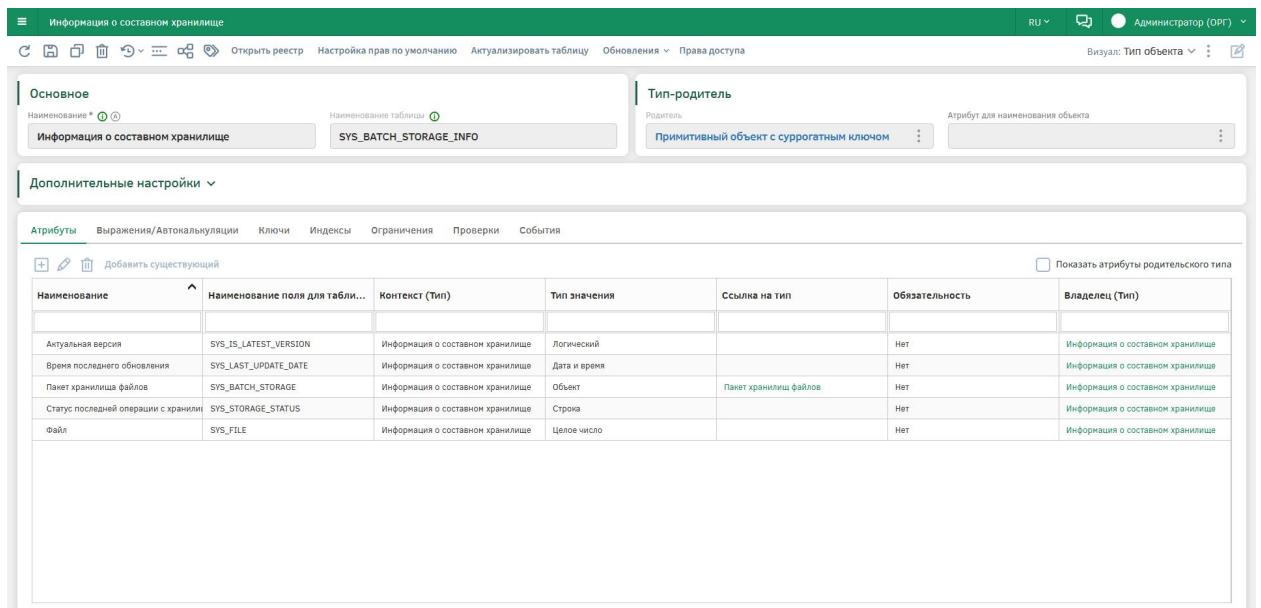


Рисунок 31 - Настройка типа объектов «Информация о составном хранилище»

Для примера настройки «Составного хранилища» был создан объект с наименованием «Составное хранилище» и добавлен в миграцию (см. Рисунок 32).

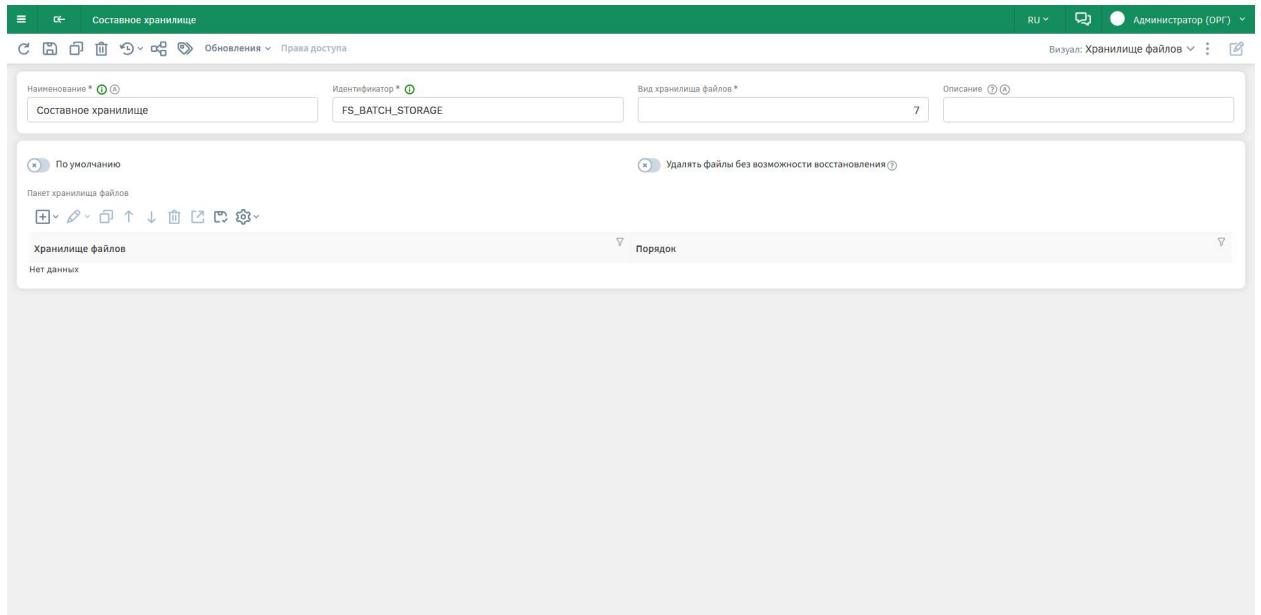


Рисунок 32 - Объект с наименованием «Составное хранилище» типа «Хранилище файлов»

Для установки на все стенды платформы GreenData все вышеописанные объекты были «свернуты» в миграцию и добавлены в список миграций (см. Рисунок 33).

```
</> master.xml ✘  console [postgres]
  2   <databaseChangeLog
  3116    <include file="classpath:config/liquibase/changeLog/00000000001024_ADD_BATCH_STORAGE.xml"/>
```

Рисунок 33 - Миграция для распределенного файлового хранилища в списке миграций

Миграция выглядит так (см. Рисунок 34).

The screenshot shows a terminal window titled 'console [postgres]' displaying an XML file named '00000000001024_ADD_BATCH_STORAGE.xml'. The XML content is a database change log with several change sets. Change set 00000000001024_ADD_BATCH_STORAGE_TYPE adds a column 'java_handler' to the 'sys_obj_type' table. Change set 00000000001024_ADD_BATCH_STORAGE_INFO_TYPE adds a column 'batch_storage' to the 'sys_obj_type' table. Change set 00000000001024_ADD_TO_FS_STORAGE_TYPE_BATCH_STORAGE_ATTRS adds a column 'batch_storage' to the 'sys_obj_type' table. Change set 00000000001024_ADD_COMPOSITE_STORAGE_OBJ adds a column 'batch_storage' to the 'sys_obj_type' table.

```
<databaseChangeLog>
    <changeSet id="00000000001024_ADD_BATCH_STORAGE_TYPE" author="panov.ik" contextFilter="all-dbs">
        <customChange class="lowcode.migration.guf.StandaloneGufChange">
            <param name="fileName" value="classpath:config/liquibase/data/00000000001024_ADD_BATCH_STORAGE_TYPE.guf"/>
        </customChange>
    </changeSet>

    <changeSet id="00000000001024_ADD_BATCH_STORAGE_TYPE_JAVA_HANDLER" author="panov.ik" contextFilter="all-dbs">
        <preConditions onFail="MARK_RAN">
            <tableExists tableName="sys_obj_type"/>
            <columnExists tableName="sys_obj_type" columnName="java_handler"/>
            <columnExists tableName="sys_obj_type" columnName="ident"/>
        </preConditions>

        <sql><![CDATA[
            UPDATE sys_obj_type
            SET java_handler = 'sysBatchStorageObjTypeHandler'
            WHERE ident = 'BATCH_STORAGE';
        ]]></sql>
    </changeSet>

    <changeSet id="00000000001024_ADD_BATCH_STORAGE_INFO_TYPE" author="panov.ik" contextFilter="all-dbs">
        <customChange class="lowcode.migration.guf.StandaloneGufChange">
            <param name="fileName" value="classpath:config/liquibase/data/00000000001024_ADD_BATCH_STORAGE_INFO_TYPE.guf"/>
        </customChange>
    </changeSet>

    <changeSet id="00000000001024_ADD_TO_FS_STORAGE_TYPE_BATCH_STORAGE_ATTRS" author="panov.ik" contextFilter="all-dbs">
        <customChange class="lowcode.migration.guf.StandaloneGufChange">
            <param name="fileName" value="classpath:config/liquibase/data/00000000001024_ADD_TO_FS_STORAGE_TYPE_BATCH_STORAGE_ATTRS.guf"/>
        </customChange>
    </changeSet>

    <changeSet id="00000000001024_ADD_COMPOSITE_STORAGE_OBJ" author="panov.ik" contextFilter="all-dbs">
        <customChange class="lowcode.migration.guf.StandaloneGufChange">
            <param name="fileName" value="classpath:config/liquibase/data/00000000001024_ADD_COMPOSITE_STORAGE_OBJ.guf"/>
        </customChange>
    </changeSet>
</databaseChangeLog>
```

Рисунок 34 - Инструкции по установке миграции для распределенного файлового хранилища

3.2 Реализация подмодулей для модулей «Бизнес-процессы», «Файлы» и «Алгоритмы».

Далее будет рассмотрена реализация дополнений в модули «Бизнес-процессы», «Файлы» и «Алгоритмы».

3.2.1 Реализация подмодуля «Алгоритмы»

Все криптографические функции, вызов которых осуществляется через средство криптографической защиты информации Crypto Pro CSP, интегрированы нативно в модуль безопасности «java.security», входящий в стандартный набор модулей языка программирования Java.

Для возможности загрузки криптопровайдера от Crypto Pro в видимость программы Java во время выполнения нужно добавить в список подключенных библиотек - зависимостей (см. Рисунок 35).

```
<dependency>
    <groupId>ru.CryptoPro</groupId>
    <artifactId>JCSP</artifactId>
</dependency>
```

Рисунок 35 - Подключение библиотеки JCSP с функциями хеширования от Crypto Pro

Также, скомпилированная библиотека была загружены в корпоративный JFrog Artifactory - менеджер репозитория артефактов, используемый для хранения скомпилированных библиотек и фреймворков, а также других компонентов для Java-проектов.

Далее, в список криптографических провайдеров нужно загрузить JCSP криптопровайдер для возможности вызова функций хеширования стандарта ГОСТ 34.11-2018 (см. Рисунок 36).

```
public void loadSCPPProvider() { 2 usages ▾ Панов Игнат Константинович
    try {
        Security.addProvider(new JCSP());
    } catch (Exception error) {
        throw new RuntimeException("Unable to find JCSP library or CSP application", error);
    }
}
```

Рисунок 36 - Загрузка криптопровайдера JCSP в криптопровайдеры Java

После этого уже можно вычислять хеш тела файла (см. Рисунок 37).

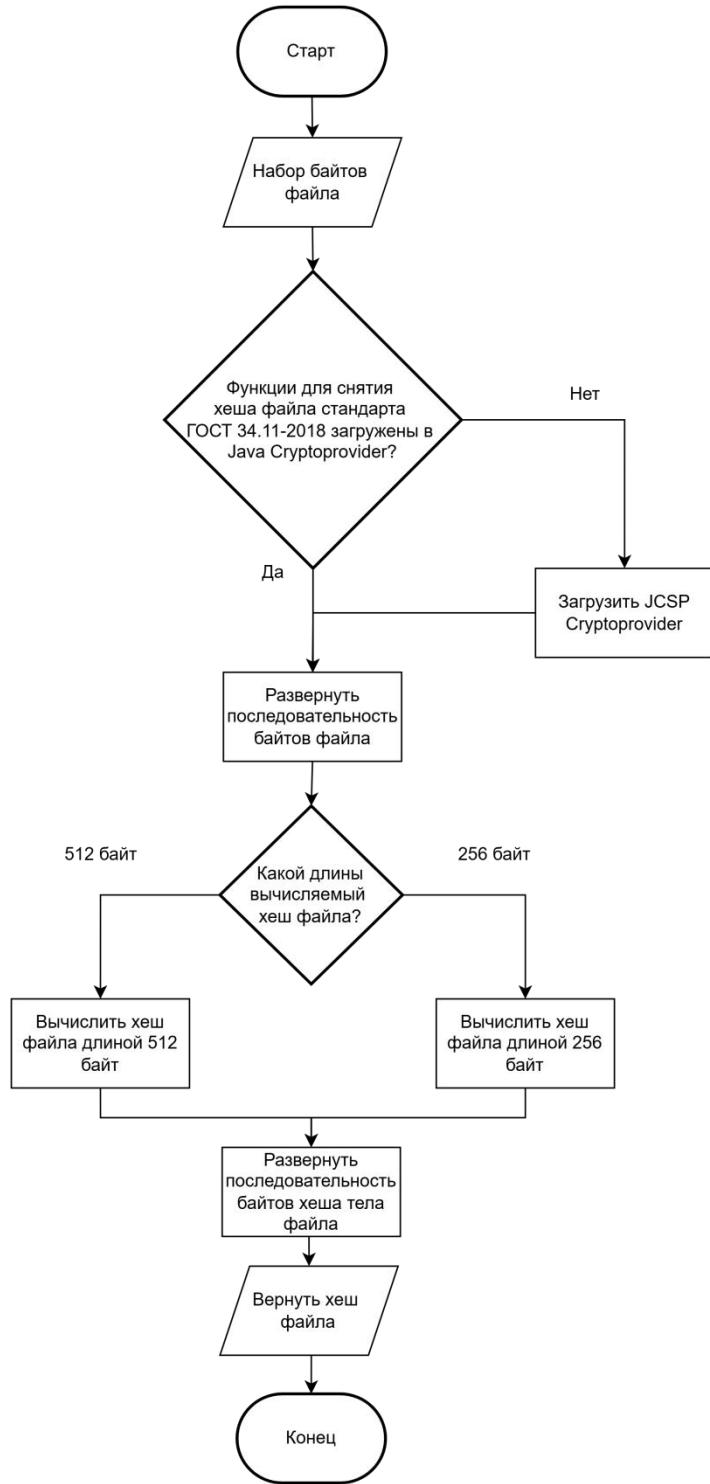


Рисунок 37 - Схема вычисления хеша стандарта ГОСТ 34.11-2018

Реализация представлена ниже (см. Рисунок 38).

```
@Override 2 usages ▾ Панов Игнат Константинович
public String calculate512BitHash(InputStream input) {
    Assert.notNull(input, message: "Hashed value must not be null");

    byte[] message = CSPUtils.convertInput(input);

    byte[] hash = calculateHash(CSPUtils.GOST_3411_512_DIGEST, CSPUtils.JCSP_PROVIDER, message);
    return Hex.encodeHexString(hash);
}

private byte[] calculateHash(String algorithmName, String providerName, byte[] message) { 2 usages ▾ Панов Игнат Константинович
try {
    if (!initializer.checkProvider()) {
        initializer.loadSCPPProvider();
    }

    MessageDigest digest = MessageDigest.getInstance(algorithmName, providerName);
    reverseMessage(message);
    byte[] hash = digest.digest(message);
    reverseMessage(hash);
    return hash;
} catch (NoSuchAlgorithmException error) {
    throw new RuntimeException("Unable to find " + algorithmName + " algorithm", error);
} catch (NoSuchProviderException error) {
    throw new RuntimeException("Unable to find " + providerName + " provider", error);
}
}

/**
 * Have to reverse the input message and hash, because the input message is in big-endian byte order,
 * but we need it in little-endian.
 * @param message byte sequence to reverse
 */
private void reverseMessage(byte[] message) { 2 usages ▾ Панов Игнат Константинович
for (int i = 0; i < message.length / 2; ++i) {
    byte rememberedByte = message[message.length - 1 - i];
    message[message.length - 1 - i] = message[i];
    message[i] = rememberedByte;
}
}
```

Рисунок 38 - Реализация вычисления хеша стандарта ГОСТ 34.11-2018

3.2.2 Реализация подмодуля «Файлы»

Распределенное файловое хранилище работает по принципу паттерна «Декоратор» [11]. Распределенное файловое хранилище использует API-интерфейс других хранилищ для сохранения, удаления, обновления и чтения тела файла внутри них.

Рассмотрим процесс сохранения файла (см. Рисунок 39).



Рисунок 39 - Процесс сохранения файла в распределенном файловом хранилище

Реализация представлена ниже (см. Рисунок 40).

```
@Override + Ignat Panov +1
public int putFile(FsFile file, FsStorage fsStorage, Resource resource) {
    List<BatchStorage> batchStorages = this.getBatchStorages(fsStorage);

    TransactionUtils.doAfterCommitIfTransactionActive(() ->
        this.batchStorageTaskService.saveFile(file, batchStorages, resource)
    );

    Long fileBodySize = file.getBodySize();
    return !Objects.isNull(fileBodySize)
        ? fileBodySize.intValue()
        : 0;
}
```

Рисунок 40 - Реализация сохранения файла в распределенном файловом хранилище

Для сохранения тела файла используется многопоточность. Сохранение в хранилище, находящегося в составе распределенного файлового хранилища, происходит в отдельном потоке (см. Рисунок 41).

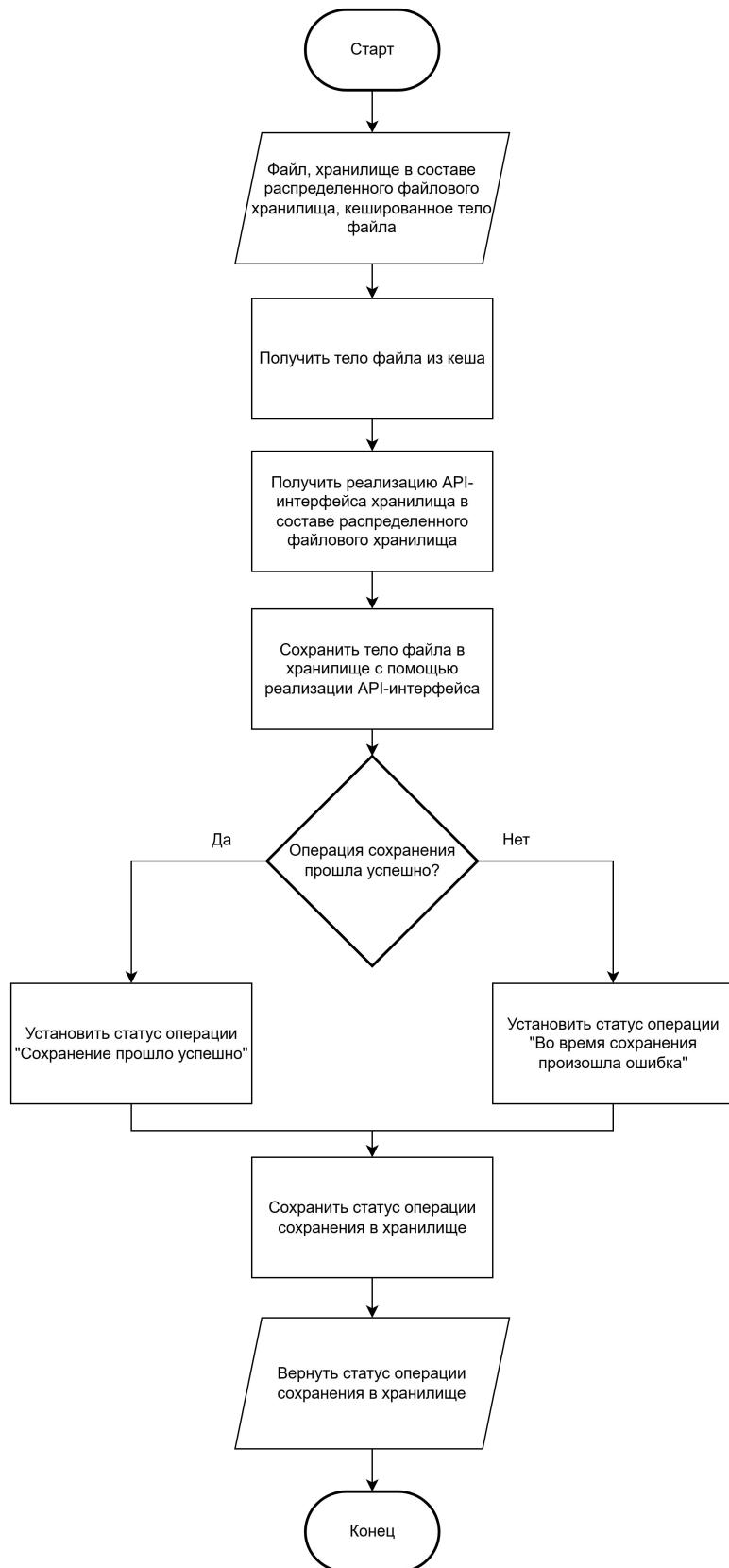


Рисунок 41 - Описание процесса сохранения файла в хранилище

Реализация представлена ниже (см. Рисунок 42).

```
private Callable<BatchOperationResult> createSaveTask(FsFile file, BatchStorage batchStorage, FileUpload tmpFile) { 1 usage  ± Ignat Panov +1
    return () -> {
        Long fileId = file.getId();

        BatchOperationResult operationResult = new BatchOperationResult()
            .set fileId(fileId)
            .setBatchStorage(batchStorage);

        try {
            FsStorage storage = batchStorage.getStorageOrNull();

            if (Objects.isNull(storage)) {
                throw new RuntimeException("Cant get fsStorage from batchStorage");
            }

            Resource resource = this.fsFileService.getTempFileResource(tmpFile.getUuid());

            if (Objects.isNull(resource)) {
                throw new RuntimeException("Cant get file resource for saving");
            }

            this.fileStorageServiceProvider.getFileStorageService(storage).putFile(file, storage, resource);

            operationResult
                .setOperationStatus(BatchOperationStatus.SAVED);
        } catch (RuntimeException e) {
            // TODO: need to log error in db
            operationResult
                .setOperationStatus(BatchOperationStatus.SAVING_FAILED);
        }

        this.fakeAuthenticationService.execAsAdmin(forceAuthentication: false, () -> {
            this.batchStorageInfoService.updateBatchStorageStatus(operationResult);
        });
    }

    return operationResult;
};
}
```

Рисунок 42 - Реализация операции сохранения файла в хранилище

В процессе сохранения заменяется предыдущее тело файла или же в случае с первым сохранением объекта типа «Файл» тело файла сохраняется в первый раз в каждое хранилище. Удаление и чтение тела файла используют такой же подход взаимодействия с хранилищами в составе распределенного файлового хранилища, а также удаление тела файла из хранилища использует подобную многопоточную операцию.

3.2.3 Реализация подмодуля «Бизнес-процессы»

Для того, чтобы отфильтровать этапы бизнес-процессов используются RSQL-фильтры. Они формируются на клиентской части в зависимости от действий пользователя с таблицей маршрута. Для преобразования фильтров в java-объекты используется библиотека парсера RSQL-фильтров [12].

Верхнеуровневый процесс фильтрации этапов бизнес-процессов выглядит так (см. Рисунок 43).

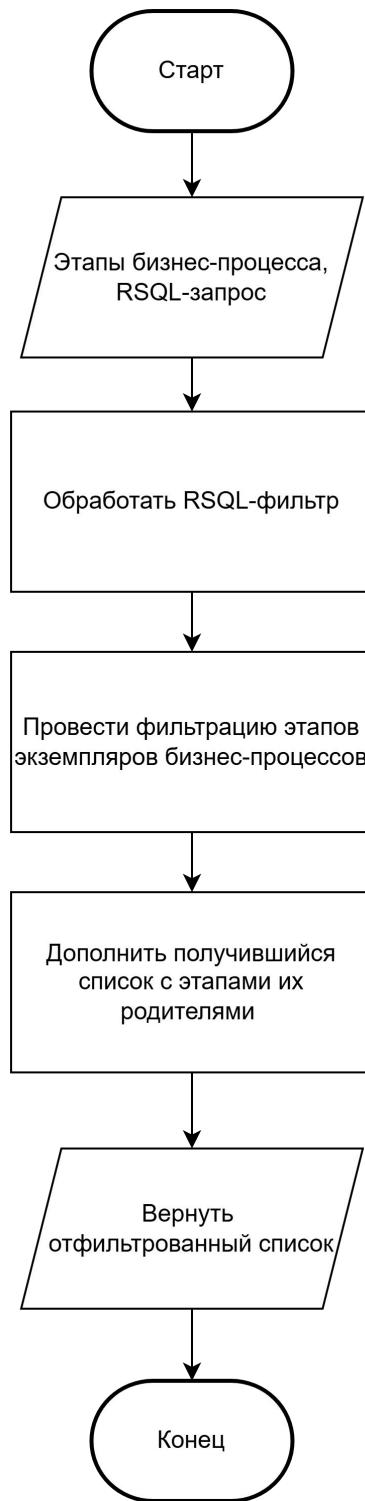


Рисунок 43 - Верхнеуровневый процесс фильтрации этапов бизнес-процессов

Значения атрибута этапа (колонки в таблице маршрута) бывают двух видов: строковыми и датами. Из этого следует, что есть две реализации фильтров этапов: строковой фильтр и фильтр по дате. Процесс фильтрация по дате представлен ниже

(см. Рисунок 44). Фильтрация по строковой колонке реализована подобным образом.

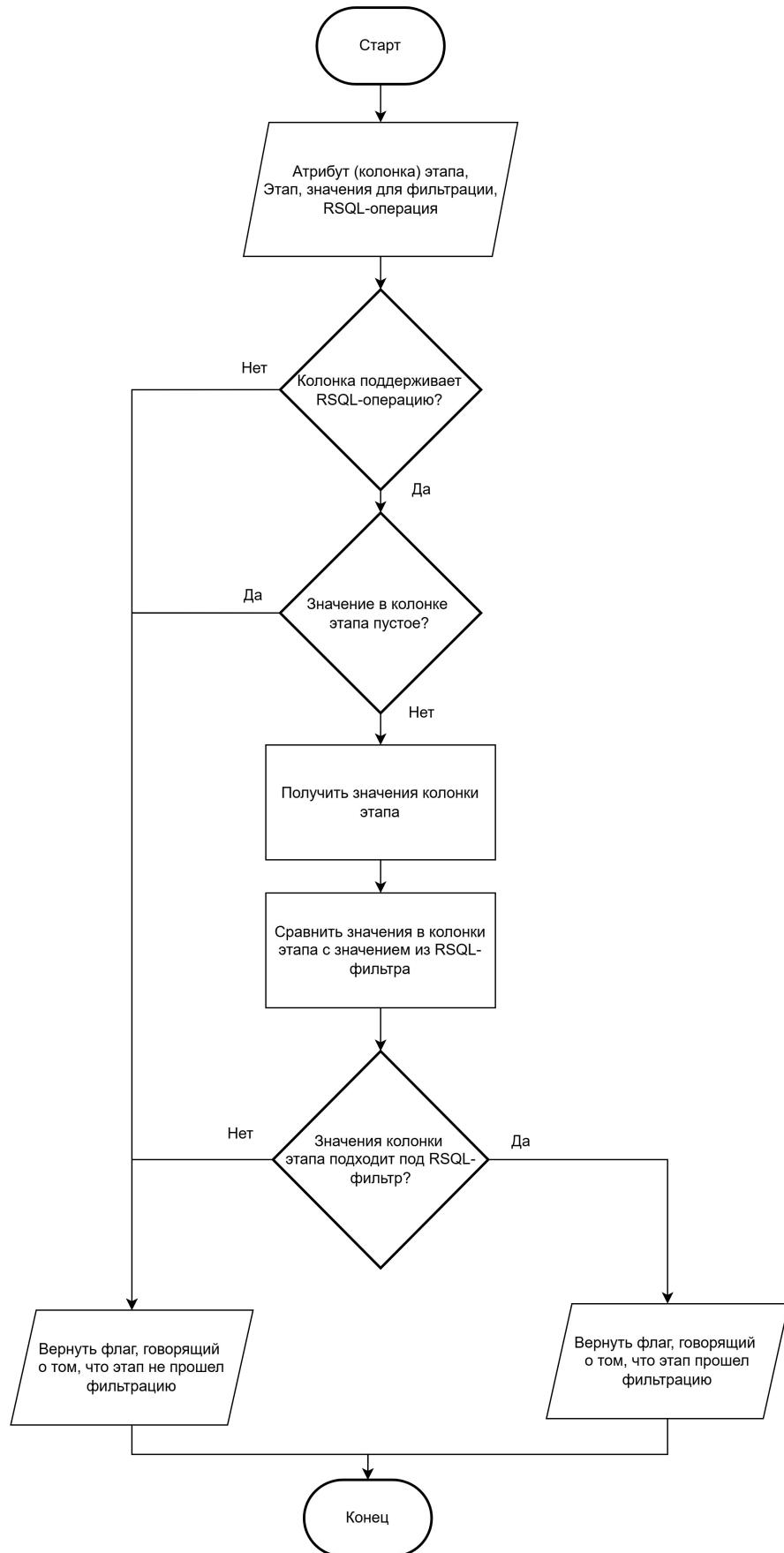


Рисунок 44 - Процесс фильтрации колонки этапа с датой в виде значение

Реализация представлена ниже (см. Рисунок 45).

```
@Override 1 usage  ± Панов Игнат Константинович
protected Boolean filterItemInternal(WayColumn wayColumn,
                                     WfWayItem item,
                                     List<String> args,
                                     RsqlSearchOperation rsqlSearchOperation) {
    Boolean supportsNumberComparison = wayColumn.supportsNumberComparison();
    if (BooleanUtils.IsFalse(supportsNumberComparison)) {
        return Boolean.FALSE;
    }

    return args.stream()
        .anyMatch( String arg -> this.filterItem(wayColumn, item, arg, rsqlSearchOperation));
}

private Boolean filterItem(WayColumn wayColumn, 1 usage  ± Панов Игнат Константинович
                           WfWayItem item,
                           String arg,
                           RsqlSearchOperation rsqlSearchOperation) {
    Boolean isValuesNull = wayColumn.checkIsNull(item);
    if (BooleanUtils.IsTrue(isValuesNull)) {
        return Boolean.FALSE;
    }
    List<String> values = wayColumn.getValues(item);

    return values.stream()
        .anyMatch( String value -> this.compareValueWithArg(value, arg, rsqlSearchOperation));
}

private Boolean compareValueWithArg(String value, String arg, RsqlSearchOperation rsqlSearchOperation) {
    WayIntegerComparisonService wayIntegerComparisonService = this.wayIntegerComparisonFactory
        .getWayIntegerComparisonByValue(value);

    return wayIntegerComparisonService.compare(value, arg, rsqlSearchOperation);
}
```

Рисунок 45 - Реализация фильтрации колонки этапа с датой в виде значения

3.3 Модификация образа сборки платформы GreenData

Для возможности вызова функции хеширования стандарта ГОСТ 34.11-2018 был модифицирован docker-образ, на котором происходит сборка платформы GreenData (см. Рисунок 46).

```
FROM [REDACTED] /greendata/openjdk-gd:14-updated-libs as fullimage

WORKDIR /csp-download

RUN apt-get update

RUN apt-get install -y wget

RUN wget http://[REDACTED] /linux-amd64_deb.tgz

RUN tar -xvzf linux-amd64_deb.tgz

FROM [REDACTED] /greendata/openjdk-gd:14-updated-libs

COPY --from=fullimage /csp-download/linux-amd64_deb /csp-download/linux-amd64_deb

RUN /csp-download/linux-amd64_deb/install.sh

RUN rm -rf /csp-download
```

Рисунок 46 - Модификация docker-образа сборки платформы GreenData

После этого сборка docker-контейнеров, в которых находится платформа GreenData, была переведена на модифицированный выше docker-образ.

3.4 Интеграционные и unit тесты

Далее будут рассмотрены тесты для проверки работоспособности разработанных решений.

3.4.1 Unit тесты для проверки работоспособности вычисления хеша по стандарту ГОСТ 34.11-2018

Для проверки работоспособности вычисления хеша стандарта ГОСТ 34.11-2018 размером в 512 бит был написан следующий тест (см. Рисунок 47).

```
@Test // Панов Игнат Константинович
public void GostHashCalculation_Check512BitHash_ReturnStringHashNotNull() {

    try (MockedStatic<MessageDigest> messageDigest = mockStatic(MessageDigest.class)) {
        MessageDigest digest = mock(MessageDigest.class);

        when(digest.digest(any(byte[].class)))
            .thenReturn(new byte[] { 0x1, 0x2, 0x3, 0x4, 0x5 });

        messageDigest
            .when(() -> MessageDigest.getInstance(any(), anyString()))
            .thenReturn(digest);

        byte[] input = new byte[] {
            0x5, 0x4, 0x3, 0x2, 0x1
        };
        InputStream message = new ByteArrayInputStream(input);

        String resHash = gostHashCalculation.calculate512BitHash(message);
        String expectedHash = "0504030201";

        assertEquals(expectedHash, resHash);
    }
}
```

Рисунок 47 - Unit тест для вычисления хеша в 512 бит

Для проверки работоспособности вычисления хеша стандарта ГОСТ 34.11-2018 размером в 256 байт был написан следующий тест (см. Рисунок 48).

```
@Test // Панов Игнат Константинович
public void GostHashCalculation_Check256BitHash_ReturnStringHashNotNull() {

    try (MockedStatic<MessageDigest> messageDigest = mockStatic(MessageDigest.class)) {
        MessageDigest digest = mock(MessageDigest.class);

        when(digest.digest(any(byte[].class)))
            .thenReturn(new byte[] { 0x1, 0x2, 0x3, 0x4, 0x5 });

        messageDigest
            .when(() -> MessageDigest.getInstance(any(), anyString()))
            .thenReturn(digest);

        byte[] input = new byte[] {
            0x5, 0x4, 0x3, 0x2, 0x1
        };
        InputStream message = new ByteArrayInputStream(input);

        String resHash = gostHashCalculation.calculate256BitHash(message);
        String expectedHash = "0504030201";

        assertEquals(resHash, expectedHash);
    }
}
```

Рисунок 48 - Unit тест для вычисления хеша в 256 бит

3.4.2 Интеграционные тесты для проверки работоспособности распределенного файлового хранилища

Перед написанием интеграционных тестов нужно добавить установку файлов формата «.guf», с которыми будет происходить взаимодействие в самих тестах (см. Рисунок 49).

```
@BeforeEach // Ignat Panov
fun initGufs() {
    if (init)
        return
    }

    fakeAuthenticationService.execAsFakeAdminIfNotAuthenticated {
        updateService.applyUpdate(updateService.zipToUpdate(ClassPathResource(path: "guf/BatchStorageServiceImplTest/i98057/i98057_fs_storage.guf")))
        updateService.applyUpdate(updateService.zipToUpdate(ClassPathResource(path: "guf/BatchStorageServiceImplTest/i98057/i98057_partition_001_batch_storage.guf")))
        updateService.applyUpdate(updateService.zipToUpdate(ClassPathResource(path: "guf/BatchStorageServiceImplTest/i98057/i98057_partition_002_batch_storage.guf")))
    }

    init = true
}
```

Рисунок 49 - Установка файлов формата «.guf» в тесте

В этих файлах лежит объект «Составного хранилища» и 2 объекта типа «Пакет хранилища файлов», которые будут записаны в атрибут «Пакет хранилища файлов» объекта «Составное хранилище».

Первый тест проверяет сохранение файла в 2 хранилища, которые являются таблицами в базе данных платформы GreenData с именами «fs_partition_001» и «fs_partition_002» на основе настроек объекта «Составное хранилище». По условию теста нужно, чтобы файл сохранился в обе таблицы (см. Рисунок 50).

```
@Test + Ignat Panov +1
@Order(1)
fun `save file in batch storage`() {
    fakeAuthenticationService.execAsFakeAdminIfNotAuthenticated {
        val file: FsFile = createFile()
        Assertions.assertNotNull(file)

        val batchStorage: FsStorage = getBatchStorage()
        Assertions.assertNotNull(batchStorage)

        file.storage = batchStorage

        clearCreatedFileBody(file.id)

        batchStorageService.putFile(file, batchStorage, ByteArrayResource(this.body))

        val partition001SaveResult: ByteArray = namedParameterJdbcTemplate.queryForObjectOrNull(
            sql: "SELECT file_body FROM fs_partition_001 WHERE file_id = :file_id;",
            ByteArray::class.java,
            file.id
        )
        Assertions.assertNotNull(partition001SaveResult)

        val partition002SaveResult: ByteArray = namedParameterJdbcTemplate.queryForObjectOrNull(
            sql: "SELECT file_body FROM fs_partition_002 WHERE file_id = :file_id;",
            ByteArray::class.java,
            file.id
        )
        Assertions.assertNotNull(partition002SaveResult)

        fsFile = file
        fsStorage = batchStorage

        Assertions.assertEquals(text, partition001SaveResult.toString(DEFAULT_DECODING))
        Assertions.assertEquals(text, partition002SaveResult.toString(DEFAULT_DECODING))
    }
}
```

Рисунок 50 - Интеграционный тест на проверку сохранения файла в «Составном хранилище»

Второй тест проверяет чтение файла из «Составного хранилища». По условиям теста нужно, чтобы операция чтения файла вернуло его тело (см. Рисунок 51).

```
@Test * IgnatPanov *
@Order(2)
fun `read file in batch storage`() {
    fakeAuthenticationService.execAsFakeAdminIfNotAuthenticated {
        val resource: Resource = batchStorageService.getFileResource(fsFile, fsStorage)
        Assertions.assertNotNull(resource)

        resource.inputStream.use {
            Assertions.assertEquals(this.text, it.readAllBytes().toString(DEFAULT_DECODING))
        }
    }
}
```

Рисунок 51 - Интеграционный тест на проверку чтения файла из «Составного хранилища»

Третий тест проверяет удаление файла из обоих хранилищ в «Составном хранилище». По условиям теста нужно, чтобы файл удалился из обоих хранилищ (см. Рисунок 52).

```
@Test * IgnatPanov
@Order(5)
fun `check deleting file body in batch storage`() {
    fakeAuthenticationService.execAsFakeAdminIfNotAuthenticated {
        updateStoragesPhysDeletion( doPhysDeletion: 1, STORAGES)

        batchStorageService.deleteFile(fsFile, fsStorage)

        val partition001: ByteArray? = namedParameterJdbcTemplate.queryForObjectOrNull(
            sql: "SELECT file_body FROM fs_partition_001 WHERE file_id = :fileId;",
            ByteArray::class.java,
            fsFile.id
        )

        val partition002: ByteArray? = namedParameterJdbcTemplate.queryForObjectOrNull(
            sql: "SELECT file_body FROM fs_partition_001 WHERE file_id = :fileId;",
            ByteArray::class.java,
            fsFile.id
        )

        updateStoragesPhysDeletion( doPhysDeletion: 0, STORAGES)

        Assertions.assertNull(partition001)
        Assertions.assertNull(partition002)
    }
}
```

Рисунок 52 - Интеграционный тест на проверку удаления файла из «Составного хранилища»

3.4.2 Unit тесты для проверки работоспособности фильтрации в визуале маршрута бизнес-процесса

Для проверки работоспособности решения каждый Unit тест проверяет фильтрацию по одной или нескольким колонок в визуале маршрута бизнес-процесса (см. Рисунок 53).

```
@Test * Панов Игнат Константинович *
public void FilterByNameAndEmployee_ReturnsThreeItems() {
    WayFilterOptionsDto dto = new WayFilterOptionsDto();
    dto.setFilter("NAME=in=(item1);EMPLOYEE==test");

    List<Long> expIds = new ArrayList<>(List.of( 1L));
    this.filterAndAssert(dto, expIds);
}

@Test * Панов Игнат Константинович *
public void FilterByInStatusAndCommentary_ReturnsTwoItems() {
    WayFilterOptionsDto dto = new WayFilterOptionsDto();
    dto.setFilter("STATUS=in=('some state name'),COMMENTARY=in=('item1')");

    List<Long> expIds = new ArrayList<>(List.of(1L, 5L));
    this.filterAndAssert(dto, expIds);
}
```

Рисунок 53 - Unit тесты для проверки фильтрации по колонкам визуала бизнес-процесса

Остальные тесты написаны подобным образом. Они проверяют все условия, используемые в RSQL-фильтрах: содержит, не содержит, равно, не равно и т.д.; и комбинируют их между собой.

Заключение

В рамках выпускной квалификационной работы были решены проблемы, возникающие на платформе GreenData, и реализованы функции поиска этапов на визуале маршрута объекта, возможность сохранения файлов в несколько хранилищ без использования внешних систем и возможность проверки целостности файлов на этапе бизнес-процесса.

Для этого был проведен анализ требований к необходимым изменениям в модули «Алгоритмы», «Бизнес-процессы» и «Файлы» на серверной части платформы GreenData. В процессе выполнения анализа были разработаны модели процессов партнеров AS-IS в нотации BPMN, разработаны пользовательские и нефункциональные требования. Требования были формализованы с помощью диаграммы вариантов использования, а также составлено техническое задание к разрабатываемым дополнениям в эти модули.

На этапе проектирования была разработаны сценарии вариантов использования, а также диаграммы последовательности для описания взаимодействия пользователя и платформы, а также были спроектированы изменения в базе данных платформы GreenData и описана диаграмма компонентов подмодулей.

Были реализованы дополнения модулей «Алгоритмы», «Бизнес-процессы» и «Файлы» на серверной части системы и добавлены изменения базу данных платформы GreenData. Для доступности средства криптографической защиты информации Crypto Pro CSP был модифицирован образ платформы GreenData.

Было разработано 6 интеграционных и 27 unit теста для проверки работоспособности решения.

Разработанные решения были внедрены в платформу GreenData (см. ПРИЛОЖЕНИЕ Б).

Библиографический список

1. Документация GreenData. Базовый и пользовательские реестры // Общество с ограниченной ответственностью «Гриндата» [Электронный ресурс]. Режим доступа: <https://docs.greendata.ru/platform/ru/basic-and-custom-registries> (дата обращения 10.04.2025).
2. Документация GreenData. Алгоритмы // Общество с ограниченной ответственностью «Гриндата» [Электронный ресурс]. Режим доступа: <https://docs.greendata.ru/platform/ru/purpose-of-algorithms> (дата обращения 10.04.2025).
3. Документация GreenData. Бизнес-процессы // Общество с ограниченной ответственностью «Гриндата» [Электронный ресурс]. Режим доступа: <https://docs.greendata.ru/platform/ru/business-processes> (дата обращения 10.04.2025).
4. Федеральный закон от 27 июля 2006 г. №152-ФЗ «О персональных данных» // Национальная система аккредитации [Электронный ресурс]. Режим доступа: <https://fsa.gov.ru/documents/9478/> (дата обращения 10.04.2025).
5. ГОСТ 34.11-2018 // Федеральное агентство по техническому регулированию и метрологии [Электронный ресурс]. Режим доступа: <https://protect.gost.ru/document.aspx?control=7&id=232143> (дата обращения 10.04.2025).
6. Jäkel, T. et al. RSQL - a query language for dynamic data types // Proceedings of the 18th International Database Engineering & Applications Symposium. – 2014. – С. 185-194.
7. Sadeghi-Nasab A., Rafe V. A comprehensive review of the security flaws of hashing algorithms // Journal of Computer Virology and Hacking Techniques. – 2023. – Т. 19. – № 2. – С. 287-302.
8. Liu T. J., Chung C. Y., Lee C. L. A high performance and low cost distributed file system // 2011 IEEE 2nd International Conference on Software Engineering and Service Science. – IEEE, 2011. – С. 47-50.

9. Мартин Р. Чистая архитектура // В кн.: Чистая архитектура. Искусство разработки программного обеспечения / ред.: Мартин Р. СПб.: Питер, 2023. С. 202-209.
10. Мартин Р. Принципы принципы дизайна SOLID // В кн.: Чистая архитектура. Искусство разработки программного обеспечения / ред.: Мартин Р. СПб.: Питер, 2023. С. 75-104.
11. Фримен Э., Робсон Э., Съерра К., Бейтс Б. Паттерн Декоратор // В кн.: Head First. Паттерны проектирования. 2-е изд. / ред.: Фримен Э., Робсон Э., Съерра К., Бейтс Б. СПб.: Питер, 2024. С. 112-138.
12. RSQL-parser [Электронный ресурс]. Режим доступа: <https://github.com/jirutka/rsql-parser> (дата обращения 20.04.2025).

Приложение А. Техническое задание

Техническое задание для разрабатываемых дополнений в модули «Бизнес-процессы», «Файлы» и «Алгоритмы» см. в файле: «Техническое_задание_для_разрабатываемых_дополнений_в_модули_Бизнес-процессы_Файлы_и_Алгоритмы.pdf».

Приложение Б. Акт о внедрении

Акт о внедрении представлен ниже (см. Рисунок 54).

АКТ О ВНЕДРЕНИИ

Панов Игнат Константинович, студент группы ПИ-21-2 выполнил выпускную квалификационную работу на тему «РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ДЛЯ ПЛАТФОРМЫ GREENDATA». Цель работы, реализация серверной части распределенного файлового хранилища, поисковых фильтров для визуального отображения маршрута и возможность вызова функции хеширования ГОСТ 34.11-2018, были выполнены. В ходе работы студентом были решены следующие задачи:

- анализ предметной области,
- проектирование,
- разработка в соответствии с постановками задач №84639, №98057 и №90832,
- разработка unit и интеграционных тестов,
- тестирование и отладка.

Разработанный функционал уже внедрен на платформе GreenData.

Руководитель центра разработки

 Невоструев Р.А.



Рисунок 54 - Акт о внедрение разработанного функционала в платформу GreenData