КОРПОРАЦИЯ Галактика

Система «Галактика ECM.CORP»

Руководство системного администратора

galaktika.ru

galaktika.ru/ecm/

2019

Оглавление

Введение

Область применения

Цель данного руководства — предоставить информацию по установке, настройке и использованию системы управления корпоративными данными **«ГАЛАКТИКА ECM.CORP»** в режиме системного администратора.

Система «ГАЛАКТИКА ECM.CORP предназначена для решения задач, связанных с управлением, хранением и интеграцией данных.

Основные функциональные возможности

- Автоматизация процесса импорта документов из разнородных источников
- Реализация хранения, редактирования и поиска документов
- Предоставление возможности настройки модели данных для хранимых документов
- Возможность построения пользовательских интерфейсов для работы с документами
- Возможность настраивать поиски, табличные представления, карточки, закладки, инструменты, связные объекты, справочники и др.
- Интеграция с существующими системами заказчика
- Автоматизация, мониторинг и контроль документо-ориентированных бизнес-процессов
- Сервисы для доступа к данным, работе с мета информацией, ведения истории, версионирования, доступа к контенту
- Управление правами доступа в систему и к документам – поддержка локальных учетных записей, интеграция с одним или многими LDAP каталогами, возможность интеграции с любым другим источником
- Сервис для работы с отчетами
- Сервис для интеграции с BPM движками

Уровень подготовки системного администратора

Системный администратор должен иметь базовые знания Java, Groovy, JSON, а также иметь представление о нотации BPMN для настройки бизнес-процессов.

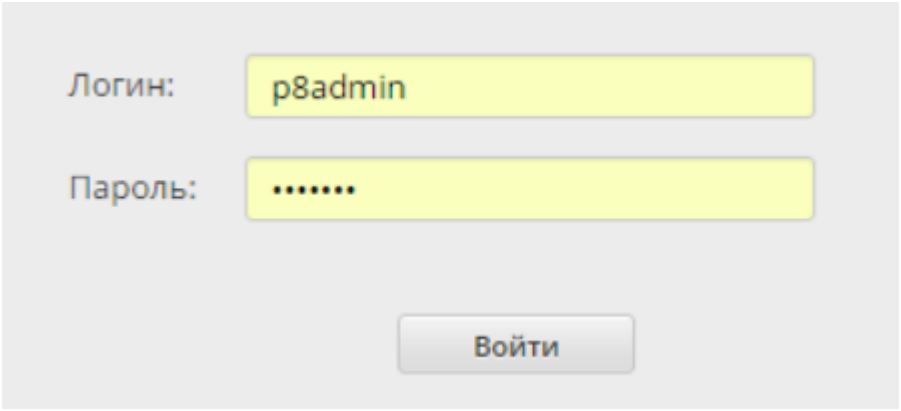
Интерфейс системного администратора

Вход в систему

Для доступа пользователя в систему необходимо пройти идентификацию и авторизацию пользователя через страницу входа в систему.

Для этого необходимо:

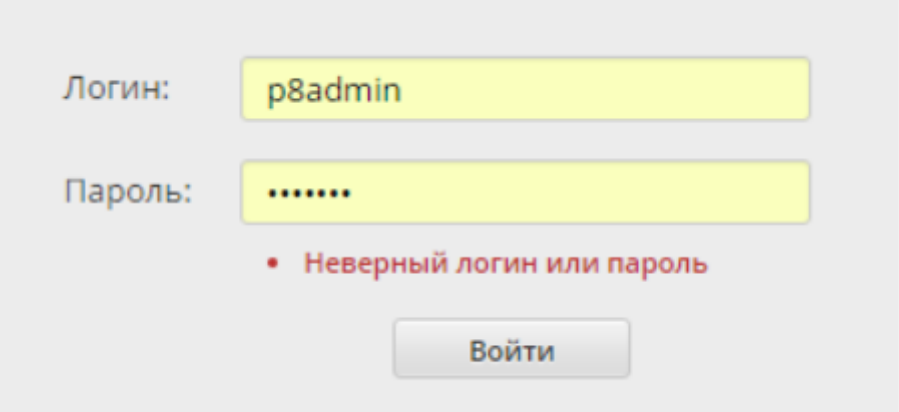
1. Ввести адрес в адресную строку браузера;
2. Ввести логин;
3. Ввести пароль;
4. Нажать кнопку «Войти».



Компоненты системы обеспечивают:

- идентификацию пользователя
- разграничение прав доступа в системе
- проверку полномочий пользователя в системе

В случае, если пользователь неверно указал данные учётной записи, возникает следующее предупреждение.



Системная роль определяет вид пользовательского интерфейса при работе с системой. Каждому пользователю должна быть настроена хотя бы одна системная роль.

Система поддерживает три вида системных ролей:

- «Пользователь»;
- «Прикладной администратор»;
- «Системный администратор».

Возможно совмещать несколько системных ролей и переключения между различными интерфейсами.

В интерфейсе реализовано выпадающее меню, где пользователь может увидеть все доступные ему роли, а также переключаться между ними.

Для того чтобы перейти в интерфейс “Системный администратор” необходимо на информационной панели нажать на кнопку  и выбрать “Рабочий стол системного администратора”.

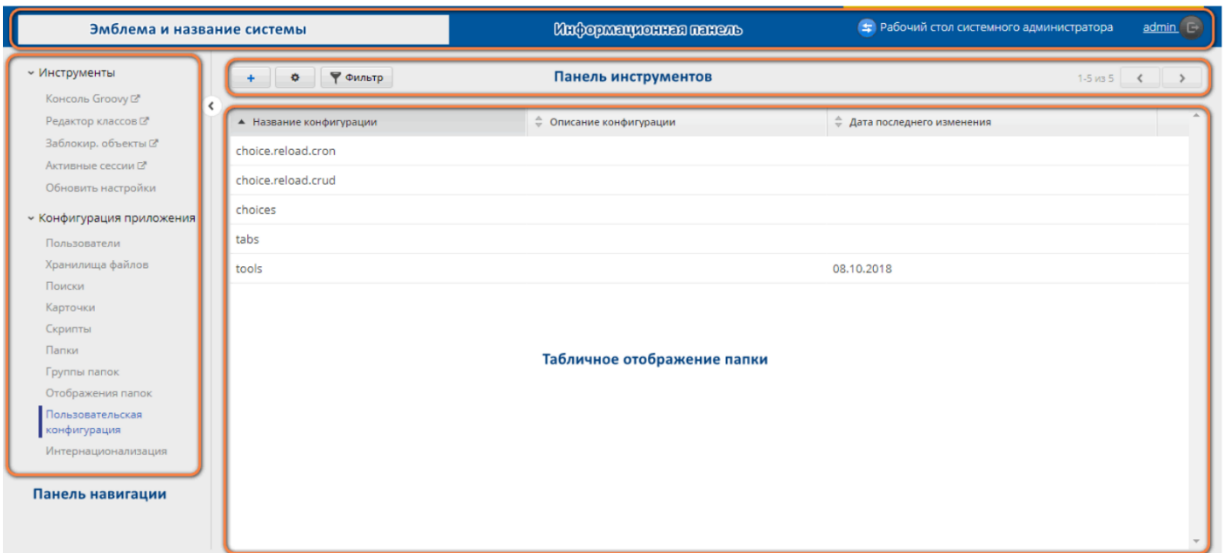
Интерфейс рабочей области системного администратора разделен на несколько зон (см. Рис. “Интерфейс общей рабочей области системного администратора”):

- Информационная консоль – служит для быстрого перехода к общей рабочей области и отображает информацию о пользователе

- Консоль инструментов – консоль с кнопками, которые позволяют выполнять различные действия с объектами;

- Консоль навигации – содержит разделы, доступные системному администратору. При необходимости консоль можно свернуть

- Табличное отображение папки – отображает список элементов выбранного объекта.



Информационная консоль

Консоль содержит информацию об имени и правах доступа пользователя. В случае совмещения нескольких ролей – элемент переключения между соответствующими интерфейсами.

На информационной консоли отображены следующие инструменты:

- Логотип системы. При нажатии осуществляется переход на главную страницу

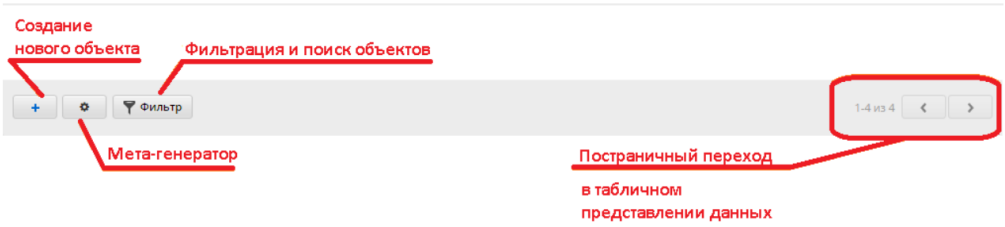
- Название интерфейса. При нажатии отображаются доступные пользователю рабочие столы и может быть выполнен переход на другой рабочий стол

- Имя текущего пользователя. При нажатии отображается информация о пользователе и его ролях в системе

- Кнопка “Выход”.

Консоль инструментов

Консоль инструментов представлена на Рис. **“Консоль инструментов”**.

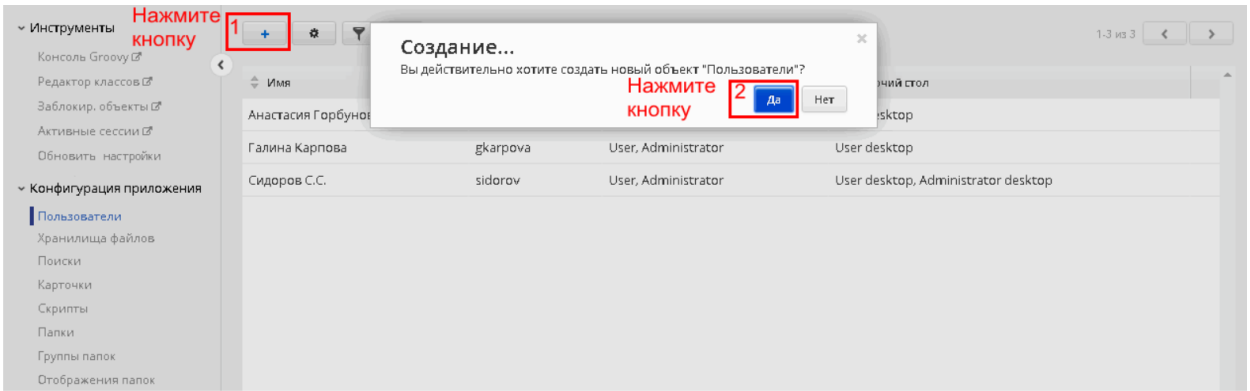


На консоли отображены следующие инструменты:

- Создание нового объекта
- Мета-генератор
- Фильтр
- Переход между страницами

Создание нового объекта

Для создания нового объекта требуется перейти в необходимую папку и нажать на кнопку “Создать новый объект”, далее следует подтвердить создание, нажав кнопку “Да” (см. Рис. **“Создание нового объекта”**).

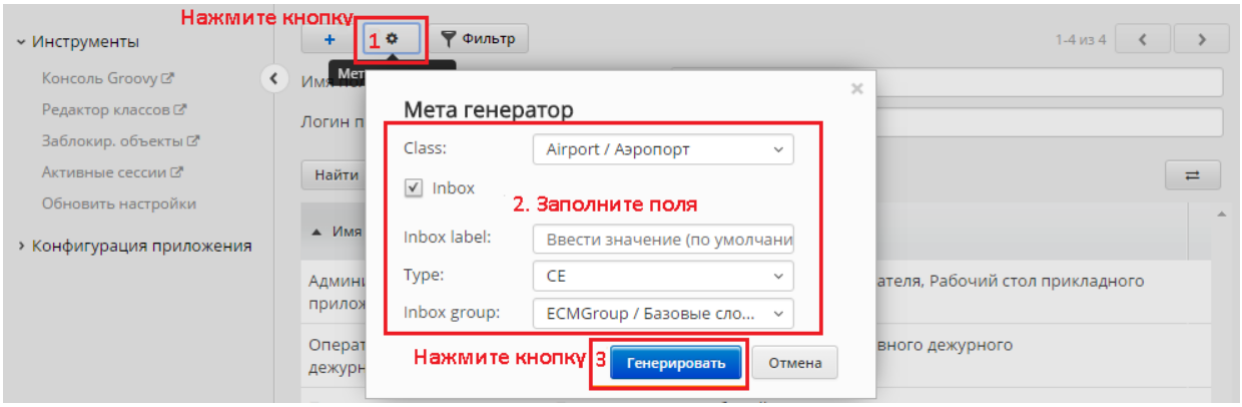


В результате карточка объекта откроется для заполнения. После заполнения нужных полей, сохраните объект, нажав кнопку “Сохранить”. В итоге, в выбранной папке раздела будет создан новый объект.

Мета-генератор

Данный инструмент автоматически генерирует соответствующие конфигурации в разделах «Папки», «Карточки», «Отображения папок» для выбранного класса объектов, а также прописывает местонахождения папки для объектов класса в соответствующей конфигурации в разделе «Группы папок».

Для запуска генератора необходимо нажать на кнопку «Мета-генератор» и заполнить поля (см. Рис. “Мета-генератор”).
В поле «Тип документа» требуется выбрать имя класса из списка доступных классов.
В поле «Тип» указать тип «CE» или «PE» (тип «PE» предназначен для бизнес-процессов).
В поле «Папка» необходимо ввести наименование, которое будет использоваться в системе, если название отлично от названия класса в «Редакторе классов». Если поле «Папка» оставить пустым, то будет использовано наименование выбранного класса в «Редакторе классов» в качестве наименования папки в системе.
В поле «Название рабочего стола» выбрать необходимый рабочий стол.
В поле «Группа папок» необходимо выбрать из списка одну из имеющихся групп папок в системе, к которой будет добавлена новая папка объектов.



Фильтрация и поиск

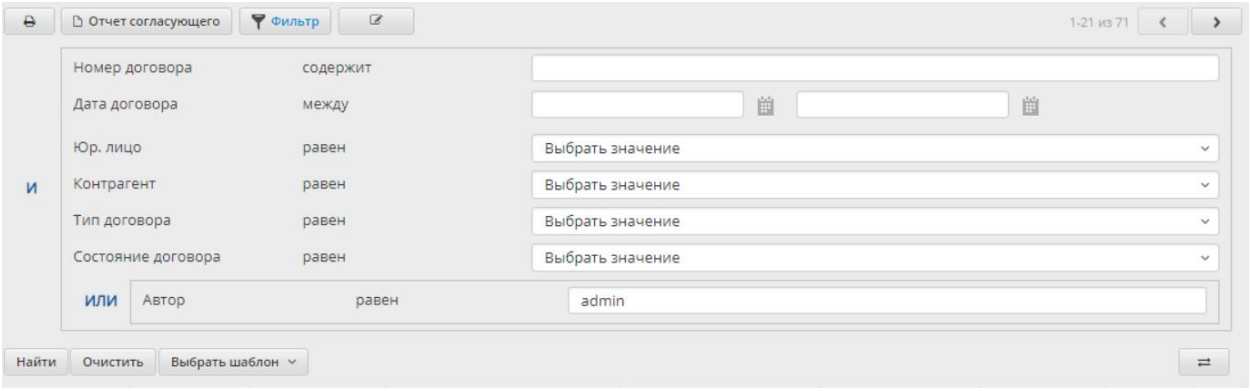
В Системе реализован алгоритм поиска и фильтрации данных, состоящий из 3 этапов, представленных ниже в таблице:

Наименование шага	**Описание**
Отбор объектов	Выбор всех объектов системы соответствующего типа, согласно объектной модели приложения
Предустановленный фильтр	Выбор из результатов предыдущего шага тех объектов, которые удовлетворяют обязательным критериям поиска
Пользовательский фильтр	Выбор из результатов предыдущего шага тех объектов, которые удовлетворяют пользовательским критериям поиска

Механизм поиска поддерживает два основных способа фильтрации данных:

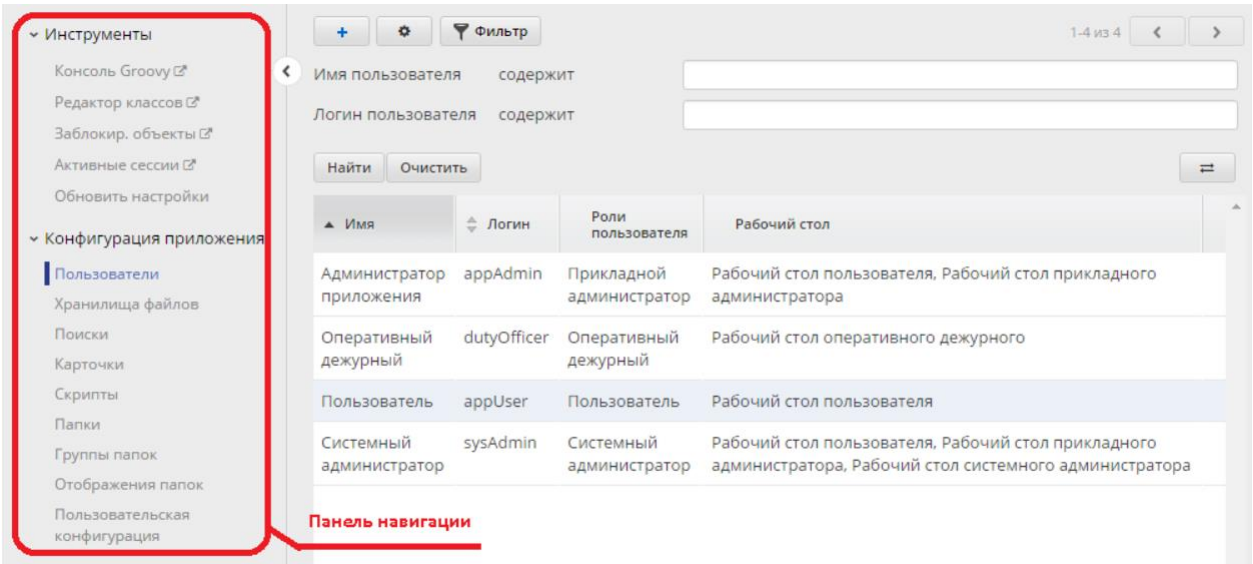
1. логическое «И» (используется между критериями поиска);
2. логическое «ИЛИ» (используется между значениями критериев поиска).

Поиск записей в разделе осуществляется с помощью панели фильтрации данных.



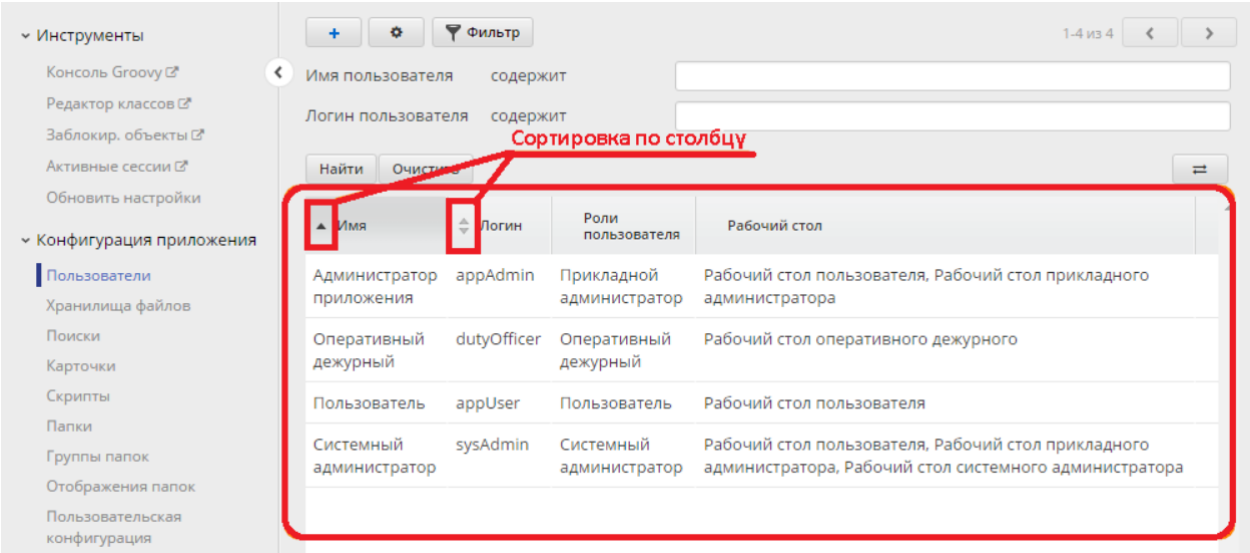
Консоль навигации

Консоль навигации представлена на Рис. **“Консоль навигации”**. На консоли расположены раздели в которых отображаются папки, которые содержат инструменты для администрирования системы.



Табличное отображение данных

В табличном отображении папок пользователь может изменить сортировку списка по столбцам.



Инструменты

Раздел “Инструменты” состоит из следующих папок:

- Консоль Groovy;
- Редактор классов;
- Заблокированные объекты;
- Активные сессии;
- Обновить настройки.

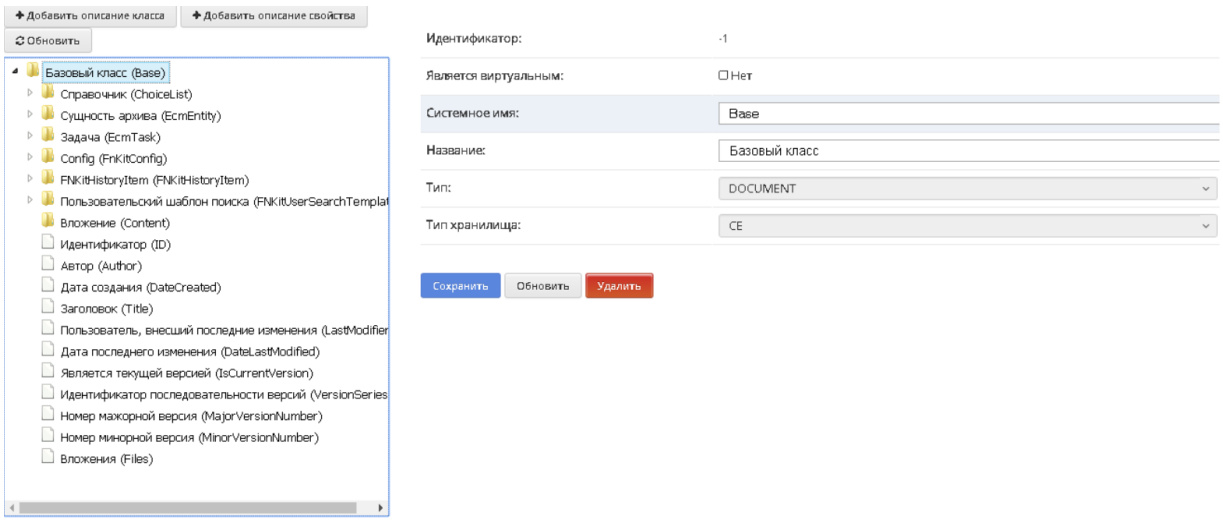
Инструмент “Консоль Groovy”

Инструмент позволяет открыть Grails Debug Console и создать конфигурацию на языке groovy.

Инструмент “Редактор классов”

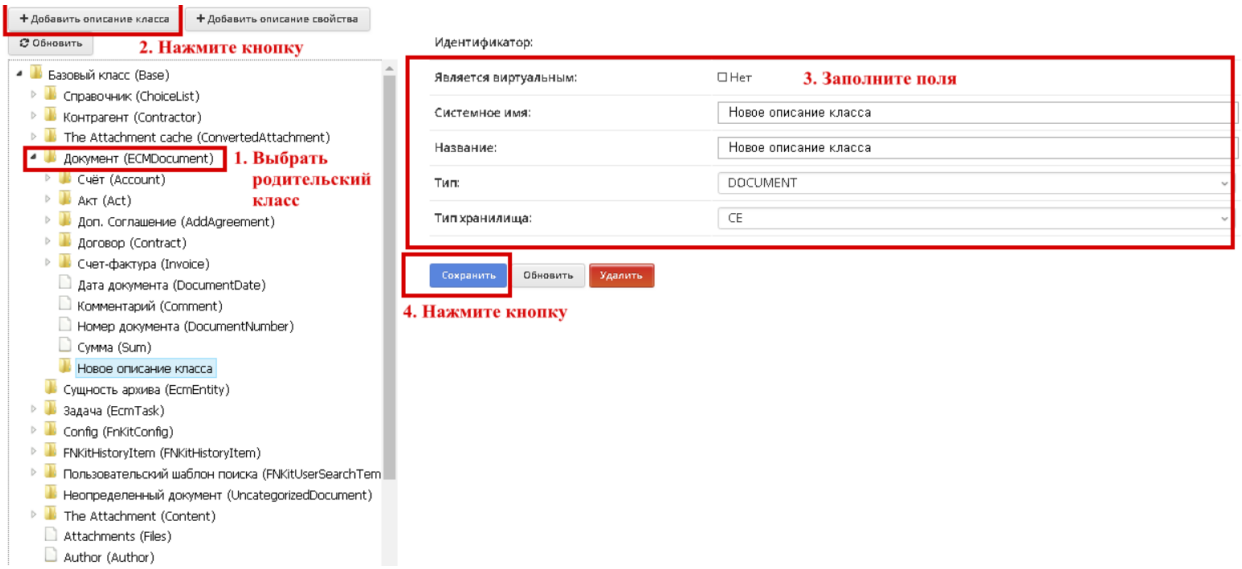
Инструмент “Редактор классов” предназначен для создания и редактирования модели данных системы.

Модель данных – это логическая структура, которая содержит набор классов и свойств объектов, которые будут использоваться в системе.
Отображение модели данных в редакторе классов.



Описание класса

Для создания нового класса требуется выбрать родительский элемент, в котором будет создан класс, и нажать кнопку “Добавить описание класса”. Заполнить информацию о классе и нажать кнопку “Сохранить” (см. рисунок **“Создание нового описания класса”**).



Описание параметров класса:

- **Идентификатор** - внутренний ID класса в базе данных;
- **Является виртуальным** – обозначает, что класс не создается и не хранится в базе данных;
- **Системное имя** - имя таблицы в БД;
- **Название** - имя класса, которое будет отображаться в интерфейсе системы для пользователей;
- **Тип** - тип класса. Доступно одно из следующих значений:

  − *DOCUMENT* – документ (с поддержкой версионирования);
  − *OBJECT* – объект (без поддержки версионирования);
  − *FOLDER* – папка;
  − *LINK* – ссылка.

- **Тип хранилища** - определяет базу данных, используемую для хранения документов. Доступно одно из следующих значений:
  − *CE* – Content Engine (хранится в объектном хранилище);
  − *PE* – Process Engine (хранится на бизнес-процессе).

  Если сущность хранит состояние процесса, то необходимо использовать тип хранилища PE. Объекты с типом хранилища PE создаются при запуске бизнес-процесса и существуют до завершения бизнес-процесса. В остальных случаях необходимо использовать тип хранилища CE.

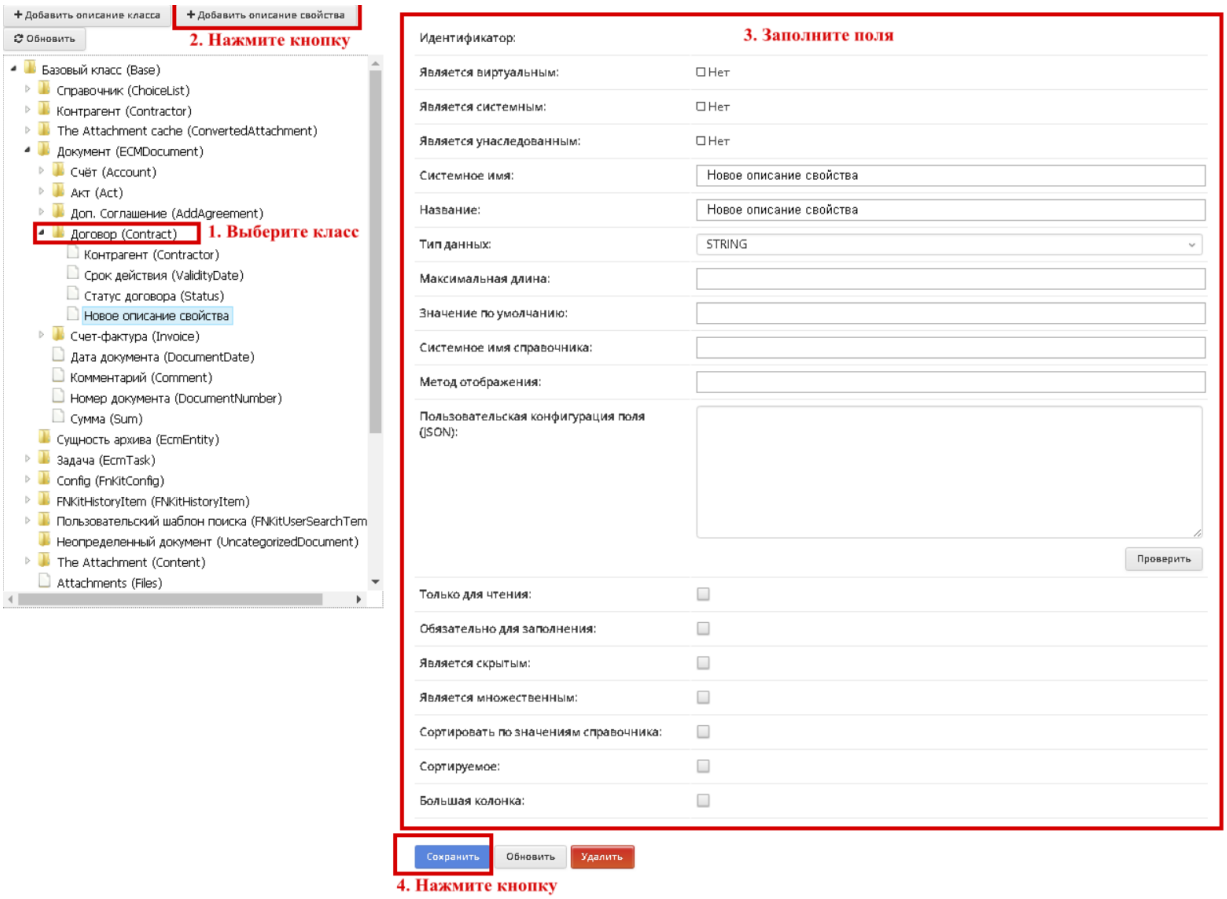
Описание свойства класса

«Базовый класс (Base)» - это базовый класс, от которого наследуются все типы объектов, которые будут храниться в системе. В базовом классе есть свойства, которые наследуются всеми созданными классами.

Системные свойства:

- Attachments(Files) вложения документа;
- Author(Author) – пользователь системы, который создал объект в системе;
- Created date(DateCreated) – дата создания объекта;
- Identifier(ID) – идентификатор объекта;
- Is current version(IsCurrentVersion) – текущая версия объекта;
- Last modified date(DateLastModified) – дата последней редакции объекта;
- Last modifier(LastModifier) – пользователь, внесший последние изменения в карточку объекта;
- Major version number(MajorVersionNumber) – старшая версия документа;
- Minor version number(MajorVersionNumber) – младшая версия документа;
- Title(Title) – название объекта;
- Version series id(VersionSeriesId) – идентификатор версии объекта.

Для создания нового свойства класса требуется выбрать класс и нажать кнопку “Добавить описание свойства”. Далее заполнить поля, характеризующие свойство, и нажать кнопку “Сохранить” (см. рисунок **“Создание нового описания свойства”**.



Описание параметров свойства класса:

- **Идентификатор** - внутренний ID свойства класса в базе данных;
- **Является виртуальным** – обозначает, что свойство класса не создается и не хранится в базе данных;
- **Системное имя** - имя поля в БД;
- **Название** - имя свойства, которое будет отображаться в интерфейсе системы для пользователей;
- **Тип данных** - тип данных свойства. Доступно одно из следующих значений:
  − *BOOLEAN* – логический тип данных (значения true/false);
  − *INTEGER* – целое число;
  − *FLOAT* – тип вещественного числа с плавающей запятой;

− *DATA* – тип данных дата;
− *STRING* – строковый тип данных;
− *JSON* – тип используется для доп. настройки полей;
− *POINT* – тип данных точка. Используется для полей с геоданными;
− *LINE* –тип данных линия. Используется для полей с геоданными;
− *POLYGON* – тип данных полигон. Используется для полей с геоданными;
− *PASSWORD* – тип используется для хранения пароля в зашифрованном виде;
− *TSVECTOR* – тип используется для полнотекстового поиска (FullTextSearch).

- **Максимальная длина** - максимальное количество символов допустимых в поле;

- **Значение по умолчанию** - значение, которое будет отображаться в поле при создании карточки объекта;
  - Можно передавать значение справочника, указав значение поле по которому настроен поиск в справочнике (напр.: 123).
  - Можно передавать массив значений справочника, в формате:

```
Значение по умолчанию: [1,2,3] - числовые значения (ID)
Значение по умолчанию: ["admin", "user"] - строковые значения
```

Примечание: Поле должно являться множественным (чек-бокс) и метод отображения должен быть “multiselect”.

- **Системное имя справочника** - имя справочника, из которого необходимо отобрать значения. Используется для полей, содержащих список значений для выбора;

- **Метод отображения** - тип отображения данных.

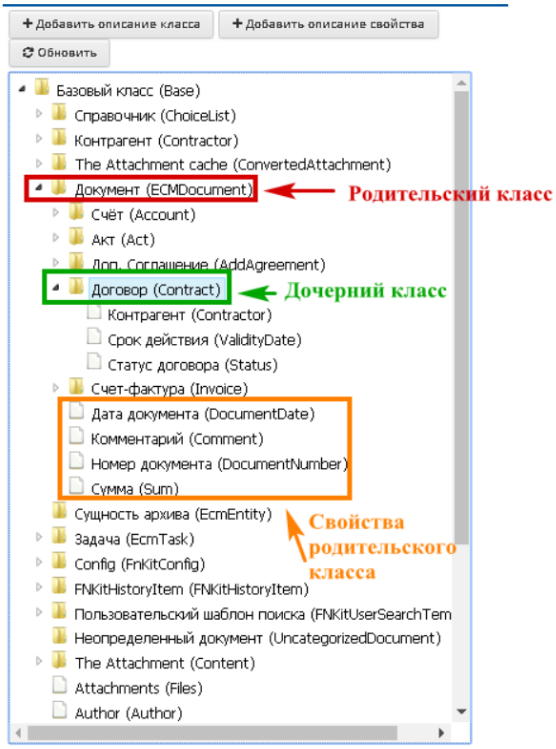
  - Доступно одно из следующих значений:
    − *dateTimePicker* – вывод даты и времени;
    − *textarea* – многострочное текстовое значение;
    − *currency* – отображение валют. Разделитель тысяч - пробел, разделитель дробной части - запятая, отображение 2-х знаков после запятой (для типа данных FLOAT, DOUBLE).
    − *select* – выбор одного значения из справочника;

− *multiselect* – выбор одного или нескольких значений из справочника;
и другие см. раздел **“Стандартные методы отображения данных”**

- **Пользовательская конфигурация поля (JSON)** - поддерживает json-скрипт, определяет поведение поля (например, задается правило, по которому вычисляется поле);
- **Только для чтения** - установленный флаг означает, что поле доступно только для чтения;
- **Обязательно для заполнения** - при установленном флаге, поле в карточке объекта помечается как обязательное для заполнения;
- **Является скрытым** - установленный флаг означает, что поле не будет отображаться в карточке объекта;
- **Является множественным** - установленный флаг означает, что в поле доступно введение нескольких значений установленного типа;
- **Сортируемое** - в табличном отображении папки для столбца с данными добавляется кнопка, позволяющая сортировать значения по возрастанию/убыванию, если флаг установлен;
- **Большая колонка** - установленный флаг означает, что поле является многострочным.

В системе поддерживается наследование свойств классов. Если для родительского класса создать дочерний класс, то дочерний класс будет наследовать все свойства

родительского класса (см. рисунок **“Наследование свойств”)**.



В приведенном примере, класс “Документ” является родительским, а класс “Договор” - дочерним классом, созданным к родительскому классу. Дочерний класс “Договор” наследует свойства родительского класса “Документ” и, соответственно, свойства родительского класса могут быть использованы для объектов дочернего класса.

Удаление классов и свойств

Для удаления класса или свойства класса требуется выбрать нужный элемент модели данных и нажать кнопку «Удалить».
Доступны следующие варианты удаления:
– частичное удаление класса или свойства класса - удаление из дерева модели данных;
– полное удаление класса или свойства класса - удаление из дерева модели данных и в базе данных.
Для полного удаления по умолчанию установлен флаг “Удалить все связанные данные”.

Инструмент “Заблокированные объекты”

Инструмент “Заблокированные объекты” содержит информацию о заблокированных объектах с возможностью принудительной разблокировки.

По умолчанию для объектов используется пессимистическая блокировка, т.е. механизм, при котором объект доступен для сохранения только открывшим его пользователям, остальным пользователям объект доступен только в режиме чтения.

Блокировка объекта выполняется по его идентификатору.

Инструмент “Активные сессии”

В данном инструменте содержится информация о пользователях, которые в данный момент находятся в системе, с возможностью принудительного завершения сессии.

Инструмент “Обновить настройки”

Данный инструмент предназначен для обновления мета-информации в кэше после внесения изменений в модель данных.

Конфигурация приложения

Раздел “Конфигурация приложения” состоит из следующих папок:

- Пользователи;
- Хранилища файлов;
- Поиски;
- Карточки;
- Скрипты;
- Папки;

- Группы папок;
- Отображения папок;
- Пользовательская конфигурация;
- Интернационализация.

Пользователи

В папке “Пользователи” представлен список пользователей в Системе

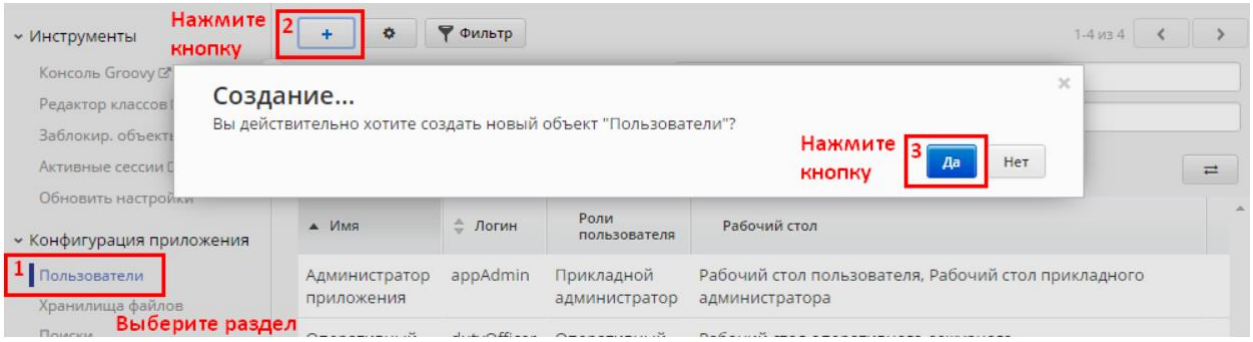
Системная роль определяет вид пользовательского интерфейса при работе с **«ГАЛАКТИКА ECM.CORP»**. Каждому пользователю присваивается хотя бы одна системная роль. На основании системной роли система определяет доступ в соответствующий интерфейс пользователя.

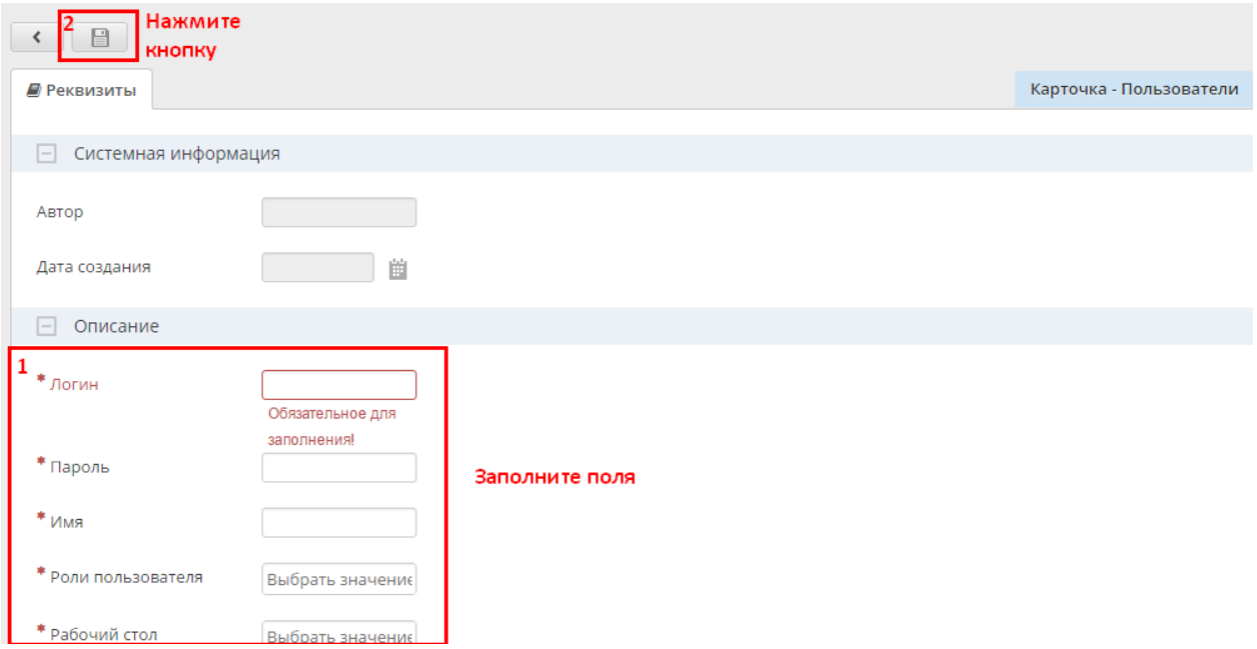
Система поддерживает три вида системных ролей:
– «Системный администратор» отвечает за настройку платформы;
– «Прикладной администратор» отвечает за настройку и работу со справочниками системы;
– «Пользователь» работает в системе.

Пользователю может быть настроено несколько системных ролей.
В системе реализовано выпадающее меню, где пользователь может увидеть все доступные ему интерфейсы и выполнить переключение в другой интерфейс.

Создание нового пользователя

Для того чтобы создать нового пользователя, необходимо выполнить следующие действия.





В результате новый пользователь будет создан.

Для редактирования информации о пользователе выберите необходимый элемент справочника, откройте карточку, внесите изменения и нажмите кнопку “Сохранить”.

Удаление элемента справочника “Пользователи”

Для того чтобы удалить элемент справочника, выберите необходимый элемент справочника и нажмите кнопку «удалить».

В результате выбранный элемент справочника будет удален.

Хранилища файлов

Для хранения файлов, загружаемых в систему (например, файлы добавленные на вкладку “Вложения” в карточки объектов) необходимо создать конфигурацию с указанием пути к хранилищу файлов.

Хранилище файлов – это область на файловой системе для хранения бинарных данных (файлов).

Для того чтобы создать хранилище файлов необходимо выполнить следующие действия:
1. Перейти в раздел «Конфигурация приложения», папку «Хранилища файлов»;
2. Нажать на кнопку «Новый документ» и подтвердить создание нового документа;
3. В поле «Путь» указать путь к папке, где будут храниться файлы и установить флаг «Используется по-умолчанию»;
4. Нажать кнопку «Сохранить».

Конфигураций с указанием пути к хранилищу файлов может быть создано несколько, но в одной из конфигураций обязательно должен быть установлен флаг “Используется по-умолчанию”. Новые файлы в системе, будут храниться именно в хранилище, где установлен флаг “Используется по-умолчанию”.

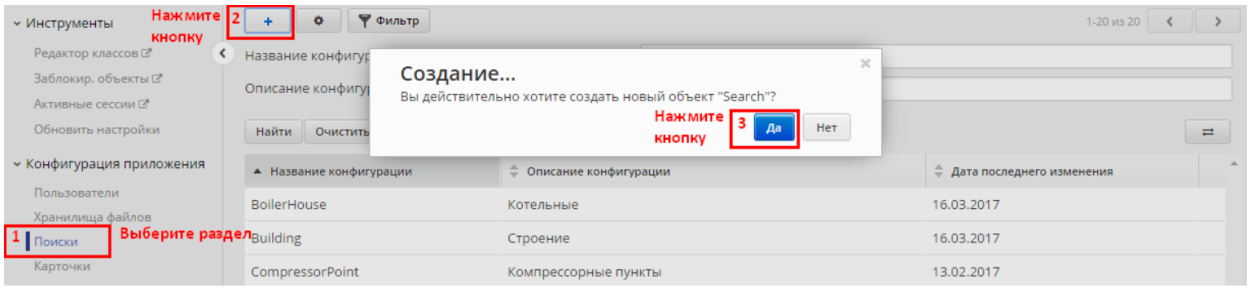
При необходимости хранилище может быть изменено.
Для этого необходимо отредактировать путь к хранилищу или установить флаг “Используется по-умолчанию” в другой конфигурации.

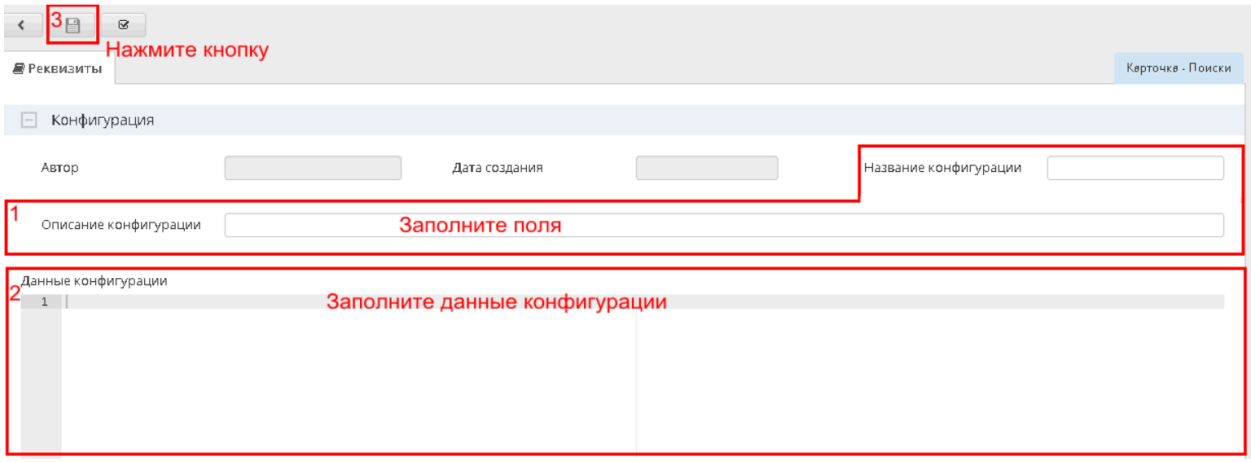
Поиски

Запросы поиска - это программные фильтры, которые позволяют задавать условия поиска объектов в папках или в карточке объекта по какому-то признаку.

Создание запроса поиска

Для того чтобы создать запрос поиска, необходимо выполнить следующие действия (см. Рис. **“Создание запроса поиска”**, **“Карточка создания нового поиска”**).





В результате пользовательский запрос поиска будет создан.

Формирование данных конфигурации

Поле «Данные конфигурации» содержит JSON-блок настройки поиска. Ниже в таблицах приведены атрибуты (см. Табл. **“Атрибуты данных конфигурации”**) и условия (см. Табл. **“Условия данных конфигурации”**), используемые при формировании данных конфигурации.

Таблица 2. Атрибуты данных конфигурации

Атрибут	**Значения**	**Используемое значение по умолчанию (если значение не выбрано)**

visible	true, false	true
binding	AND, OR	AND
readonly	true, false	false
value	Может содержать как константу, так и стандартный макрос: ${user.id}, ${user.name}, ${role.id}, ${role.name} или разработанный свой макрос.	

Таблица 3. Условия данных конфигурации

Значение условия	**Описание**	**Примечание**
fieldName, (обязательное условие)	Имя поля объекта	
	EQUAL	Равно
	GREATER_EQUAL	Больше или равно

	GREATER_THAN	Больше чем
	LESS_EQUAL	Меньше или равно
	LESS_THAN	Меньше чем
	NOT_EQUAL	Не равно
	LIKE	Как
	BETWEEN	Между
	EMPTY	Пусто
	NOT EMPTY	Не пусто
	IS_NULL	Не заполнено
	NOT_NULL	Заполнено
	CONTAINS_ALL	Содержит все

	NOT_CONTAINS_ALL	Не содержит все
	CONTAINS_ONE_OF	Содержит один из
	SQL	Условие SQL
	IN	Входит в
	NOT_IN	Не входит в

Пример конфигурации:

```
1. {"criteria":
2. {"id": "criteria_Mediator", "binding": "AND", "conditions": [
3. {"id": "FolderName", "field": "FolderName", "label": "Номер","readonly": false,
"visible": true, "condition": "EQUAL", "maxlength": 255 },
4. {"id": "PackageNumber", "field": "PackageNumber", "readonly": false,
"label":"Номер агентского договора", "visible": true, "condition": "LIKE" },
5. {"id": "Branch", "field": "Branch", "label":"Филиал", "readonly": false,
"visible": true, "condition": "IN", "maxlength": 255 },
6. {"id": "DateCreated", "field": "DateCreated", "label": "Дата создания",
"readonly": false, "visible": true, "condition": "BETWEEN" },
7. {"id": "Author", "field": "Author", "readonly":false, "visible": true,
"condition": "EQUAL", "value": "${user.login}"},
8. ]}
9. }
```

Описание скрипта

1 строка – начало скрипта;

2 строка – наименование (id) скрипта, условие объединения запросов (binding);

3 – 7 строка – условия поиска:

- id – уникальный идентификатор запроса;
- field – наименование поля для поиска (как в “Редакторе классов”);
- label – наименование поля для пользователя, если отличается от заданного в “Редакторе классов”;
- readonly – признак редактирования условия поиска (см. Таблицу 2);
- visible – признак видимости условия поиска (см. Таблицу 2);
- condition – условия фильтрации (см. Таблицу 3);
- maxlength – максимальное количество символов для ввода в условие;
- value - дополнительное условие для поиска. Может содержать константу или макрос ($макрос).
  Где макрос — это код groovy, возвращающий строку.
  После создания шаблона поиска его необходимо связать с папкой, в которой он будет использоваться по умолчанию. Об этом написано в разделе «Папки».

  Так же допустима настройка сложных поисков с несколькими вложенными критериями, в таком случае атрибут И/ИЛИ будет отображаться в фильтре. Пример конфигурации приведен ниже:

  Пример конфигурации:

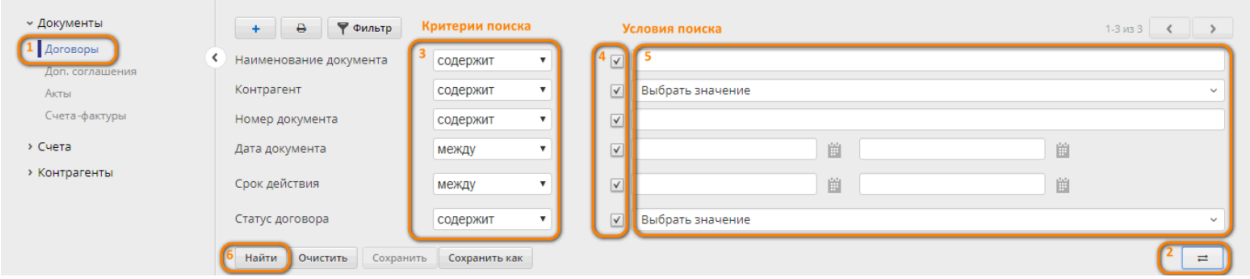
```
{
"criteria": {
"id": "criteria_1",
    "binding": "AND",
        "conditions": [
            {"id":"IsCurrentVersion", "field":"IsCurrentVersion", "visible":false,
"condition": "EQUAL", "value": true},
            {"id": "ContractNumber", "field": "ContractNumber", "visible": true,
"condition": "LIKE"},
            {"id": "ContractDate", "field": "ContractDate", "visible": true,
"condition": "BETWEEN"},
            {"id": "Performer", "field": "Performer", "visible": true,
"condition": "EQUAL"},
            {"id": "ContractState1", "field": "ContractState", "visible": false,
"condition":  "IN", "value": [175, 187, 181, 177]},

                {"id": "criteria_2",
```

```
                    "binding": "OR",
                        "conditions": [
                         {"id": "Author", "field": "Author", "visible": true,
"condition": "EQUAL", "value": "${user.login}"},
                         {"id": "ApprovalDepartments", "field":
"ApprovalDepartments", "visible": false, "condition": "OVERLAP", "value":
$user.params.Department}
                        ]}
                ]}
}
```

Фильтрация записей раздела

Поиск записей в папке осуществляется с помощью настроенной панели фильтрации. Панель фильтрации может быть отредактирована. Для этого необходимо выполнить следующие действия:

1. Перейти в интерфейс «Пользователь», открыть необходимый раздел и папку, нажать кнопку «Фильтр»;
2. Нажать кнопку в панели фильтрации «Сменить режим»;
3. Отредактировать критерии поиска (выпадающий список);
4. Выбрать условия поиска (чек-боксы), которые будут отображаться в панели фильтрации (если
   чек-бокс не установлен, то поле не будет отображаться на панели фильтрации);
5. Нажать кнопку «Сменить режим» еще раз.



В результате набор полей и критерии поиска для полей на панели фильтрации будут изменены в соответствии с новыми настройками.

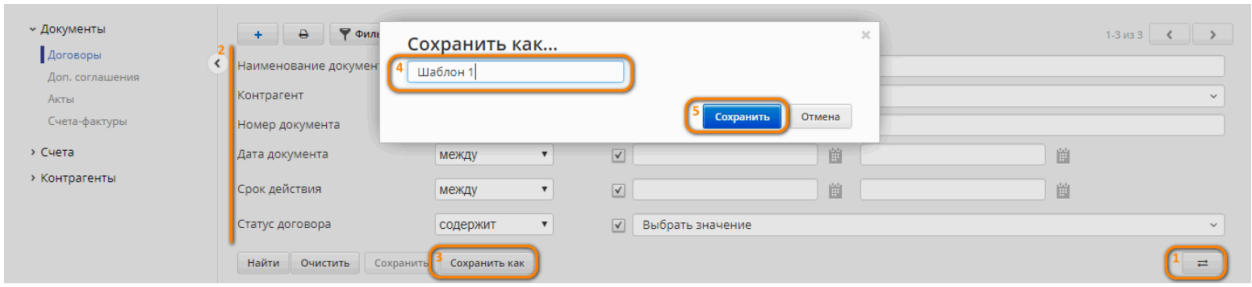
Управление шаблонами поиска

Для удобства поиска по записям разделов пользователь может создать собственный шаблон поиска, состоящий из набора условий и критериев поиска.

Создание шаблона поиска

Для того чтобы создать шаблон поиска, необходимо выполнить следующие действия:

1. В папке нажать кнопку «Сменить режим»;
2. Настроить критерии поиска;
3. Нажать кнопку «Сохранить как»;
4. В окне сохранения шаблона поиска ввести имя шаблона поиска;
5. Нажать кнопку «Сохранить».



Удаление запроса поиска

Для того чтобы удалить запрос поиска, необходимо открыть карточку объекта и нажать кнопку “Удалить”. А также в конфигурации папки, параметре “Search” указать значение по умолчанию “DefaultSearch” (в данном случае, для папки панель фильтрации отображаться не будет) или другое существующее наименование конфигурации поиска, подходящего для папки.

Отображения папок

Отображение папок создается на основе типа объекта и определяет набор столбцов, видимых пользователю во время просмотра папки.

Создание отображения папки

Для того чтобы создать отображение папки, необходимо перейти в раздел “Конфигурация приложения”, папку “Отображения папки” и нажать на кнопку “Создать

новый документ”. Заполнить поля “Название конфигурации “, “Данные конфигурации”, при необходимости добавить описание в поле “Описание конфигурации”.
В результате отображение папки будет создано. После создания отображения папки его необходимо связать с папкой, в которой оно будет использоваться по умолчанию. Об этом написано в разделе “Папки”.

Формирование данных конфигурации

Конфигурация представляет собой json-скрипт, состоящий из двух обязательных блоков *class* и *fields* и дополнительных: *rowsPerPage*, *params* и *orderBy*.

Блок *class* содержит название используемого класса представления, доступен один класс *com.galantis.ecm.model.inbox.InboxView*.

Блок *fields* содержит поля, которые будут отображаться в виде столбцов в таблице. В каждом элементе блока необходимо указать обязательный параметр *fieldName*:

- **fieldName** – имя свойства объекта, в соответствии с системным именем свойства в редакторе классов;

Для дополнительной настройки формата вывода значения поля, в блоке *fields*, доступны следующие необязательные параметры:

- **label** – название поля, отображаемое в табличном представлении папки. Если значение не задано, устанавливается название в соответствии с названием свойства в редакторе классов;

- **renderer** – компонент, отвечающий за вывод значения. Доступны следующие компоненты:

  - *checkbox* – компонент выбора для полей типа «флаг», значение установлено/значение не установлено. Значение по умолчанию для полей типа Boolean;

  - *date* – вывод даты в формате «dd.mm.yyyy». Значение по умолчанию для полей с типом дата;

  - *dateTime* – вывод даты в формате «dd.mm.yyyy hh:mm:ss»;

- *double* – вывод вещественных чисел в формате «#.##»;
- *number* – вывод числовых значений, ограничивает количество выводимых символов до 100;
- *text* – вывод текстового значения, ограничивает количество выводимых символов до 100;
- *choiceIcon* - используется для отображения иконок, в соответствии с выбранным значением из справочника.

- **hidden** – позволяет скрыть поле. Принимает значения true - поле скрыто, false - поле не скрыто. По умолчанию принимает значение false;
- **sortable** – позволяет отсортировать значения. Принимает значения true - сортировка разрешена, false - запрещена;
- **width** - позволяет задать ширину колонки. В формате:”200px”.

Блок *rowsPerPage* содержит количество строк выводимых на одной странице в таблице.

Блок *orderBy* предназначен для вывода объектов в отсортированном виде. Параметры:

- *fieldName* - системное имя свойства по которому будут сортироваться объекты;
- *direction*- направление сортировки: ASC – по возрастанию, DESC – по убыванию.

Блок *params* предназначен для дополнительной настройки табличного отображения папки. Параметры и значения:

- *newWindow* – тип boolean, параметр отвечает за открытие карточки объекта в новом окне. Принимает два значение: true - открывать карточку в новом окне, false - не открывать (значение по умолчанию).

- *availableViewConfigs* – параметр отвечает за отображение данных.
  Возможны следующие значения параметра *availableViewConfigs*: dataTable -

отображает объекты в табличном виде и map - позволяет отобразить объекты на карте. При использовании двух параметров одновременно данные отображаются в соответствии с первым указанным значением, а также добавляется переключатель между табличным представлением данных и картой.

Пример конфигурации:

```
{
    "class" : "com.galantis.ecm.model.inbox.InboxView",
    "params": {"newWindow": true},
    "rowsPerPage":20,
    "fields" : [
    {
      "fieldName" : "ContractNumber"
    }, {
      "fieldName" : "ContractDate",
      "renderer":"dateTime"
    },
    {
      "fieldName" : "Customer"
    },
    {
      "fieldName" : "Performer"
    },
    {
      "fieldName" : "ContractState"
    },
    {
      "fieldName" : "Urgent"
    }
    ],
    "orderBy" : {
      "fieldName" : "DateCreated",
      "direction" : "DESC"
    }
  }
```

Настройка отображения иконок в зависимости от значения справочника

В системе есть возможность отображать в табличном отображении иконки в соответствии с выбранным значением из справочника.

Для добавления иконки в табличное отображение папки необходимо выполнить следующие действия:

1. В редакторе классов в классе “Справочник(ChoiceList)” создать свойство со следующими атрибутами:
   Системное имя: IconCssClass
   Название: Css класс иконки
   Тип данных: STRING
   Максимальная длина: 64
   Сортируемое: установить чек-бокс

2. В карточку справочника для которого необходимо добавить иконку требуется добавить поле и “IconCssClass” и указать CSS класс. Ссылки на **доступные иконки** и **цвета**.

3. Переопределить справочник в конфигурации “choices”, добавив в параметр “itemFields” значение “IconCssClass”.
   Пример:

```
MissionStatus     : new SimpleChoice('MissionStatus',
'criteriaChoiceListProcessor', [
        index     : 'ID',
        label     : '${fields.Title}',
        search    :
Search.from('MissionStatus').whereCurrentVersion().orderBy('Title'),
        fields    : ['ID','Title'],
        itemFields: ['Title','IconCssClass']
```

4. В конфигурации табличного отображения добавить в блок *fields* компонент “renderer”: “choiceIcon”

   Пример конфигурации:

```
{
    "class" : "com.galantis.ecm.model.inbox.InboxView",
    "fields" : [
     {
      "fieldName" : "MissionStatus",
      "renderer": "choiceIcon",
      "label":"",
      "width":"25px"
     },
     {
      "fieldName" : "Executor"
     }
    ],
    "orderBy" : {
      "fieldName" : "DateCreated",
      "direction" : "ASC"
    }
  }
```

Удаление отображения папки

Для того чтобы удалить табличное отображение папки, необходимо нажать на кнопку “Удалить”. А также необходимо заменить наименование конфигурации отображения папки во всех конфигурациях папок (раздел «Папки»), где использовалась данная конфигурация и заменить на другое существующее наименование конфигурации.

Папки

Папка – это сущность, определяющая группировку объектов по каким-то критериям. В папке отображаются объекты.

Создание папки

Для того чтобы создать папку, необходимо выполнить следующие действия. Перейти в раздел “Конфигурация приложения”, папку “Папки” и нажать на кнопку “Создать новый документ”. Заполнить поля “Название конфигурации “, “Данные конфигурации”, при необходимости добавить описание в поле “Описание конфигурации”.

Формирование данных конфигурации

Конфигурация представляет собой json-скрипт, в котором указываются следующие параметры:

- **label** – имя папки в системе;
- **caseName** – название класса объектов в редакторе классов, для которых создается папка;
- **type** – тип папки (CE или PE);
- **inboxView** – наименование конфигурации табличного отображения папки;
- **roles** – список ролей, для которой будет доступна папка;
- **search** – наименование конфигурации поиска для папки;
- **showCount** - отображение количества объектов рядом с названием папки. Доступно два значения: true - количество объектов отображается, false - не отображается (значение по умолчанию).

Пример конфигурации:

```
{
    "label": "Поиск",
    "caseName": "Contract",
    "type": "CE",
    "inboxView": "Contract",
    "roles": ["appUser", "curator"],
    "search": "ArchiveSearch",
    "showCount":true
  }
```

Удаление папки

Для того чтобы удалить папку, необходимо нажать кнопку “Удалить”. А также удалить название конфигурации папки из конфигураций групп папок, где было указано название данной конфигурации папки.

Группы папок

Группы папок и папки – это логические разделы, в которых отображаются бизнес-объекты.

Создание группы папок

Для настройки групп папок необходимо перейти в раздел “Пользовательская конфигурация”, папку “Группы папок” и нажать на кнопку “Создать новый документ”. Выбрать в поле “Название рабочего стола“ рабочий стол, для которого необходимо создать группы папок, и заполнить “Данные конфигурации”, при необходимости добавить описание в поле “Описание конфигурации”.

Конфигурация создается один раз для каждого рабочего стола.

Формирование данных конфигурации

Конфигурация представляет собой json-скрипт, в котором указываются следующие параметры:

- **class** – класс, который реализует логику группировки папок;
- **label** – название группировки(раздела);
- **inboxes** – список названий папок входящих в группировку(раздел).

Пример конфигурации:

```
{
"ECMDocument" : {
    "class" : "com.galantis.ecm.model.inbox.InboxGroup",
    "label" : "Документы",
    "inboxes" : [ "NewContract", "OnAgreement", "Approved" ]
  }
}
```

Удаление группы папок

Для того чтобы удалить группу папок, необходимо нажать кнопку “Удалить”. В результате группа папок будет удалена.

Карточки

Карточка - это набор полей с информацией об объекте, расположенной в определенном порядке. Поля в карточке - это визуальное отображение свойств объектов.

Создание карточки

Для того чтобы создать карточку объекта необходимо перейти в раздел «Конфигурация приложения», папку «Карточки», нажать кнопку “Создать новый документ”. Или открыть карточку, котороя была сгенерирована при помощи иструмента «Мета-генератор». Конфигурация карточки объекта была сформирована только с системными атрибутами такие как автор, пользователь, внесший последние изменения, дата создания и дата последнего изменения.

Формирование данных конфигурации

Конфигурация данных представляет json-скрипт.

panels – это секции на которые разбиваются поля на карточке, секции можно сворачивать/разворачивать.
Каждая секция имеет заголовок (элемент *legend*) и набор выводимых компонентов *componentsRows* – это строки в рамках панели.

Для описания полей в карточке предназначен элемент конфигурации *components*. Для настройки расположения, формата и других параметров доступны следующие параметры:

- **cssClass** – название css классов применяемых к полю (разделитель пробел). Доступные классы можно посмотреть в файлах с расширением css проекта. Наиболее часто используемые классы это span1 – span12;
- **propName** – название свойства объекта. Обязательный параметр;
- **readonly** – тип boolean, позволяет сделать поле доступным или недоступным для редактирования (true - нередактируемое поле, false - редактируемое, является значением по умолчанию);

- **required** – тип boolean, позволяет указать что свойство является обязательным для заполнения (true - обязательное, false - необязательное, является значением по умолчанию);
- **koBinding** – используется для добавления дополнительной не универсальной и не настраиваемой логики к полю. В поле требуется указать название предварительно разработанного биндинга и его параметры;
- **maxLength** – ограничивает максимально допустимое количество символов для поля с текстовым значением;
- **label** – название поля, выводится перед значением. По умолчанию используется название свойства свойства;
- **choiceName** – название справочника для полей с выбором значений из справочника. По умолчанию используется справочник, указанный в атрибуте choice в Редакторе классов;
- **templateName** – название компонента для вывода значения поля. Доступны следующие значения:
  - *text* – нередактируемый текст;
  - *input* – однострочное текстовое значение. Значение по умолчанию для большинства полей;
  - *textarea* – многострочное текстовое значение;
  - *select* – выбор значения из справочника. Значение по умолчанию для полей по справочнику;
  - *multiselect* – выбор значения из справочника, позволяет выбрать несколько значений;
  - *attachmentList* - отображение списка вложений (для свойства ‘Files’) в карточке объекта;

- *checkbox* – для полей типа «флаг», значение установлено/значение не установлено. Значение по умолчанию для полей типа Boolean;
- *datePicker* – дата в формате «dd.MM.yyyy». Значение по умолчанию для полей с типом дата;
- *dateTimePicker* – дата в формате «dd.MM.yyyy HH:mm:ss»;
- *hidden* – скрытое поле, на карточке не отображается;
- *staticText* – отображение статического текста с сообщением пользователю. Дополнительный параметр params: messageCode - сообщение, с поддержкой интернационализации;
- *code* – для полей, содержащих конфигурационный скрипт. Для полей с таким templateName доступен дополнительный параметр «mode»;
- *table* – для полей, содержащих объекты другого класса объектов. Для полей с таким templateName требуется дополнительно указывать следующие атрибуты:
  - *view* – название табличного представления;
  - *className* – класс выводимых объектов.
- *choiceIcon* – используется для полей, содержащих значения из справочника. Рядом с полем выводится иконка, соответствующая выбранному значению в справочнике. В справочнике для каждого значения должен быть указан CSS класс иконки.

- **params** – дополнительные параметры. Доступны следующие дополнительные параметры:
  - *linkedChoice* - тип boolean, реализует предзаполнение полей на основе данных других полей (см.раздел **Настройка связанных справочников**). Значения true - предзаполнять поле, false - не предзаполнять (значение по умолчанию);
  - *hideLabel* – не выводить название поля. Принимает два значения, true (не выводить) и false (выводить). По умолчанию для полей название выводиться;

- *mode* – доступно для templateName «code»; Позволяет управлять подсветкой синтаксиса для кода скриптов. Доступны два значения: groovy или json;

- *link* – используется для отображения наименования поля (‘label’) в виде ссылки на связанный объект. Для настройки используется два параметра:
  - ‘to’ - название класса;
  - ‘binding’ - используется для задания доп. поисков, позволяющих найти связанный объект.
    Доп. параметры для ‘binding’: “IsCurrentVersion” - указатель на версию объекта (‘true’ - последняя версия объекта, ‘false’ - все версии объекта);
    “VersionSeriesId” - идентификатор последовательности версий объекта.

- **mask** – позволяет задать правило для проверки формата вводимого значения. Для параметра требуется указать следующие атрибуты;
  - *pattern* – паттерн для проверки значения, более подробно можно прочитать в документации по JavaScript;
  - *msg* – сообщение для некорректных значений.

Пример конфигурации:

```
{
    "panels": [
    { "legend": "Документы",
        "componentsRows": [
          {"components": [ {"cssClass": "span8", "propName": "Files",
"templateName": "attachmentList","required": true}]}
        ]
    },

    { "legend": "Реквизиты",
        "componentsRows": [
          {"components": [ {"cssClass": "span4", "propName": "ContractNumber",
"label":"Номер договора субподряда", "required":true},{"cssClass": "span4",
"propName": "ContractDate","label":"Дата договора
субподряда","required":true},{"cssClass": "span4", "readonly": true, "propName":
"ContractType"}  ]},
          {"components": [ {"cssClass": "span4", "propName": "ContractAmount"},
```

```
{"cssClass": "span4", "propName": "Currency"},
          {"cssClass": "span4", "propName": "DocumentLink", "readonly": true,
"label":"Договор",
          "params":{"link": { "to": "Contract", "binding": {"IsCurrentVersion":
true, "VersionSeriesId": "{params.DocumentLink}"}}}}]},
          {"components": [ {"cssClass": "span8", "propName":
"Customer","required": true}]},
          {"components": [ {"cssClass": "span10", "propName": "INN",
"label":"ИНН", "params": {"mask": {"pattern": "^\\d{13}$", "msg": "Поле должно
содержать 13 цифр!"}}}]}
          {"components": [ {"cssClass": "span4", "propName": "Urgent"}]}
        ]
      }
    ]
  }
```

Удаление карточки

Для того чтобы удалить карточку, необходимо нажать на кнопку “Удалить”.

Скрипты

Раздел предназначен для изменения базового поведения системы. Например, отображение инструментов, вкладок, полей в карточках в зависимости от требований, накладываемых системой (должностная роль, рабочий стол, статус документа и др.) Для переопределения поведения системы необходимо создать одну из следующих конфигураций и переопределить методы

Доступные конфигурации:

- **tabsScript** - предназначен для управления набором вкладок в карточке объекта;
- **toolsScript** - предназначен для управления инструментами в папке и в карточке объекта, а также для управления приложенными файлами к карточке объекта;
- **layoutsScript** - предназначен для управления отображением карточки объекта, а также переопределением карточки в зависимости от параметров;
- **inboxesScript** - предназначен для управления табличным представлением папки;
- **activitiScript** - предназначен для описания методов и вызова их в Workflow engine.

Описание скрипта tabsScript

Для того чтобы создать tabsScript необходимо перейти в раздел “Конфигурация приложения”, папку “Скрипты”, нажать кнопку “Создать новый документ”, заполнить поле “Название конфигурации” значением tabsScript.
В поле “Данные конфигурации” создать новый класс, реализующий методы интерфейса или наследующий класс, и переопределить требуемые методы.

Свойства входящего параметра **CalculatorContextForTab context**:
user - пользователь;
desktopName - название рабочего стола пользователя;
tabName - название текущей вкладки;
vo - объект.

Пример конфигурации tabsScript:

```
import com.galantis.ecm.meta.DefaultTabsCalculator
import com.galantis.ecm.api.object.model.ValueObject
import com.galantis.ecm.meta.context.CalculatorContextForTab
import org.springframework.web.context.request.RequestContextHolder
import com.galantis.ecm.auth.type.Desktop

class CommonTabCalculator extends DefaultTabsCalculator {

@Override
Collection<String> getCaseTabs(CalculatorContextForTab context) {
        def isNew = context.vo.id == null
        def session = RequestContextHolder.currentRequestAttributes().getSession()
        def tabs = []
                switch (context.vo.classType) {
                                case 'Contract':
                                        tabs << 'details'

                                if (!isNew) {
                                                                tabs << 'content'
                                                                tabs << 'Act'
                                                                tabs << 'history'
```

```
                                                    }
                                break

                        default:
                                tabs << 'details'
                                if (!isNew) {
                                                                tabs << 'content'
                                                                tabs << 'history'
                                                    }
                                                }
                        tabs
                }
}
```

Описание скрипта toolsScript

Для того чтобы создать toolsScript необходимо перейти в раздел “Конфигурация приложения”, папку “Скрипты”, нажать кнопку “Создать новый документ”, заполнить поле “Название конфигурации” значением toolsScript.
В поле “Данные конфигурации” создать новый класс, реализующий методы интерфейса или наследующий класс, и переопределить требуемые методы.

Свойства входящего параметра **ToolCalculatorContextForInbox context**:
user пользователь;
desktopName - название рабочего стола пользователя;
inboxConfig - название папки;
inboxViewType - название табличного отображения папки.

Свойства входящего параметра **CalculatorContextForTab context**:
user - пользователь;
desktopName - название рабочего стола пользователя;
tabName - название текущей вкладки;
vo - объект.

Свойства входящего параметра **ToolCalculatorContextForContent context**:
user - пользователь;

desktopName - название рабочего стола пользователя;

tabName - название текущей вкладки.

Пример конфигурации toolsScript:

```
import com.galantis.ecm.meta.*
import com.galantis.ecm.BeanUtils
import com.galantis.ecm.api.auth.model.User
import com.galantis.ecm.api.content.model.Content
import com.galantis.ecm.api.meta.ToolsCalculator
import com.galantis.ecm.api.object.model.ValueObject
import com.galantis.ecm.auth.type.Desktop
import com.galantis.ecm.meta.context.CalculatorContext
import com.galantis.ecm.meta.context.CalculatorContextForTab
import com.galantis.ecm.meta.context.ToolCalculatorContextForContent
import com.galantis.ecm.meta.context.ToolCalculatorContextForInbox
import com.galantis.ecm.model.inbox.Inbox
import com.galantis.ecm.model.inbox.InboxViewType

class ToolsScript extends DefaultToolsCalculator {

        Collection<String> getTools(ToolCalculatorContextForInbox context) {
                def result =super.getTools(context)
                result << 'exportToFile'
                result
        }

        Collection<String> getTools(CalculatorContextForTab context) {
                def tools =super.getTools(context)
                def isNew = context.vo.id == null

                if (context.vo.classType == 'Contract' && context.tabName ==
'Act'){
                        tools << 'newAct'
                }
                tools
        }
        Collection<String> getTools(ToolCalculatorContextForContent context) {
```

```
                def tools =super.getTools(context)
                        tools << 'updateContentTool'
                tools
        }
}
```

Описание скрипта layoutsScript

Для того чтобы создать layoutsScript необходимо перейти в раздел “Конфигурация приложения”, папку “Скрипты”, нажать кнопку “Создать новый документ”, заполнить поле “Название конфигурации” значением layoutsScript.
В поле “Данные конфигурации” создать новый класс, реализующий методы интерфейса или наследующий класс, и переопределить требуемые методы.

Свойства входящего параметра **LayoutCalculatorContext context**:
user - пользователь;
desktopName - название рабочего стола пользователя;
tabName - название текущей вкладки;
vo - объект;
layout - базовая карточка объекта.

Свойства входящего параметра **CalculatorContextForTab context**:
user - пользователь;
desktopName - название рабочего стола пользователя;
tabName - название текущей вкладки;
vo - объект.

Пример конфигурации layoutsScript:

```
import com.galantis.ecm.BeanUtils
import com.galantis.ecm.api.auth.model.User
import com.galantis.ecm.api.meta.LayoutsCalculator
import com.galantis.ecm.api.meta.extension.ComponentsLookup
import com.galantis.ecm.api.object.model.ValueObject
import com.galantis.ecm.meta.context.CalculatorContextForTab
import com.galantis.ecm.meta.context.LayoutCalculatorContext
import com.galantis.ecm.model.meta.Layout
import com.galantis.ecm.auth.type.Desktop
```

```
import com.galantis.ecm.meta.*

class LayoutsScript extends DefaultLayoutsCalculator {
    @Override
    Layout processLayout(LayoutCalculatorContext context) {
        super.processLayout(context)
        def isNew = context.vo.id == null

        if (Desktop.USER == context.desktopName) {
            if (context.vo.classType == 'Contract'){
                if (!isNew && (context.user.login != context.vo.params.Author ||
(context.vo.params.ContractState != 173 && context.vo.params.ContractState !=
179))) {
                    context.layout.readonly = true
                }
            }
        }
        context.layout
    }

    @Override
    String getLayoutName(CalculatorContextForTab context) {
        def layoutName = super.getLayoutName(context)
        def isNew = context.vo.id == null

        if (context.vo.classType == 'Contract') {
            if (isNew) {
                if (context.vo.params.ContractType==231) {
                    layoutName = 'PrepareAddAgreement'
                }
                if (context.vo.params.ContractState==177){
                    layoutName = 'ApprovedContract'
                }
            }
        }
        if(Desktop.APP_ADMIN==context.desktopName && context.vo.classType ==
'Contract'){
```

```
            layoutName = 'AdminContract'
        }
        layoutName
    }
}
```

Описание скрипта inboxesScript

Для того чтобы создать inboxesScript необходимо перейти в раздел “Конфигурация приложения”, папку “Скрипты”, нажать кнопку “Создать новый документ”, заполнить поле “Название конфигурации” значением inboxesScript.
В поле “Данные конфигурации” создать новый класс, реализующий методы интерфейса или наследующий класс, и переопределить требуемые методы.

Доступный параметр *inboxView* - табличное оторбражение папки;

Пример конфигурации inboxesScript:

```
import com.galantis.ecm.meta.DefaultInboxesCalculator
import com.galantis.ecm.model.inbox.InboxView

class CommonInboxesCalculator extends DefaultInboxesCalculator {

  @Override
  InboxView processInboxView(InboxView inboxView) {
    def diffField = ["fieldName": "DocumentStatus" ]
    def fields = inboxView.fields
    if (inboxView.name == "Document"){
        fields<<diffField
        }
    inboxView
  }
}
```

Описание скрипта activitiScript

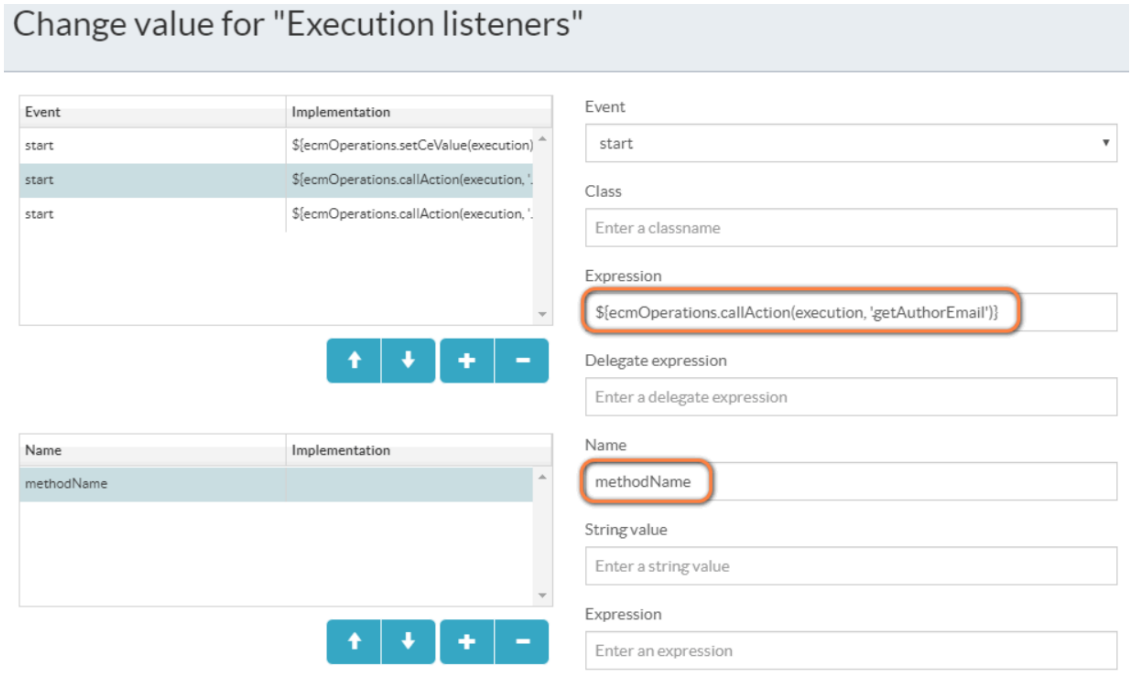
Для того чтобы создать activitiScript необходимо перейти в раздел “Конфигурация приложения”, папку “Скрипты”, нажать кнопку “Создать новый документ”, заполнить поле “Название конфигурации” значением activitiScript.

В поле “Данные конфигурации” описать методы, которые необходимо будет вызывать в Workflow engine.

Пример конфигурации activitiScript:

```
import com.galantis.ecm.bpm.activiti.meta.BaseActivitiScript
import com.galantis.ecm.api.search.type.ConditionType
import org.activiti.engine.delegate.VariableScope
import com.galantis.ecm.api.search.model.Search

class ActivitiScript extends BaseActivitiScript {

    public void getAuthorEmail(VariableScope execution) {
        def user = userService.findByLogin(execution.getVariable("Author"))
        if (user != null && user.params.Email != null)
            execution.setVariable("AuthorEmail", user.params.Email)
        else
            execution.setVariable("AuthorEmail", null)

    }
}
```

Пример: Для вызова методов, из activiti-app необходимо выполнить следующие действия



1.Создать”Execution listeners”, добавить в поле “Expression” выражение:

${ecmOperations.callAction(execution, ‘getAuthorEmail’)}

где callAction вызывает методы из ActivitiScript, execution - параметр, куда передаются значения, ‘getAuthorEmail’ - название метода в ActivitiScript.

2. В поле “Name” добавить значение “MethodeName”
3. Сохранить “Execution listeners”

Если необходимо создать “Task listeners”, то в поле “Expression”, требуется указать парамтер “task”: ${ecmOperations.callAction(**task**, ‘getAuthorEmail’)}.

Пользовательская конфигурация

Раздел содержит настройки пользовательской конфигурации системы. Система распознает следующие файлы конфигурации (название конфигурации):

1. choice.reload.cron - конфигурация для обновления справочников по cron;

2. choice.reload.crud - конфигурация обновления справочников по crud;

3. choices - конфигурация описания справочников;

4. tabs - конфигурация описания вкладок;

5. tools - конфигурация описания инструментов.

Формирование данных конфигурации “choice.reload.cron”

Конфигурация представляет собой json-скрипт для обновления справочников по cron.

В конфигурации указываются следующие параметры:

```
- interval – интервал для задания времени выполнения, используется синтаксис cron:
* * * * *
- - - - -
| | | | |
| | | | ----- День недели (0 - 7) (Воскресенье =0 или =7)
| | | ------- Месяц (1 - 12)
| | --------- День (1 - 31)
| ----------- Час (0 - 23)
------------- Минута (0 - 59);
```

- initialDelay – задержка перед первым выполнением в миллисекундах;
- choices – список в формате: [‘название справочника’, …].

Пример конфигурации:

```
[
    [
        interval    : '0 0/5 * * * *',
        initialDelay: 1000 * 60,
        choices     : ['Users']
    ]
]
```

Примеры использования интервалов в формате по cron:

```
* * * * *                               Каждую минуту

59 23 31 12 5                           За минуту до конца года, если последний
```

```
день года - пятница

59 23 31 Dec Fri                          За минуту до конца года, если последний
день года - пятница
                                              (еще один вариант записи)

0 12 * * 1-5 (0 12 * * Mon-Fri)           В полдень по рабочим дням

* * * 1,3,5,7,9,11 *                     Каждую минуту в январе, марте, мае, июле,
сентябре и ноябре

0 9 1-7 * 1                               Первый понедельник каждого месяца, в 9
утра

0 0 1 * *                                 В полночь, первого числа, каждый месяц

* 0-11 * *                               Каждую минуту до полудня

* * * 1,2,3 *                            Каждую минуту в январе, феврале и марте

* * * Jan,Feb,Mar *                      Каждую минуту в январе, феврале и марте

0 0 * * *                                 Каждый день в полночь

0 0 * * 3                                 Каждую среду в полночь
```

Формирование данных конфигурации “choice.reload.crud”

Конфигурация представляет собой скрипт для обновления справочников по событиям в хранилище объектов:

1. событие «создан» – создан новый объект в хранилище;

2. событие «обновлен» – существующий объект обновлён в хранилище;

3. событие «удален» – объект удален из хранилища;

Конфигурация обновления справочников имеет следующий формат:

```
[Системное имя класса: [‘название справочника’, …]].
```

Пример конфигурации:

```
[
'FileStores': [ 'FileStores'] ,
'DocStatusList': [ 'DocStatusList']
]
```

Формирование данных конфигурации “choices”

Описание справочника с контроллером criteriaChoiceListProcessor соответствует схеме:

```
Наименование справочника : new Реализация справочника(’Системное имя справочника’, ’название контроллера’, [
        index : '' - название поля используемого в качестве идентификатора элемента справочника,
        label : '' - название элемента справочника, которое выводится пользователю,
        search : '' sql запрос,
        fields: ['','',...] - набор полей для формировании описания элемента справочника,
        itemFields: ['','',...] -  набор кэшируемых полей
])
```

Доступные реализации справочников:

- *SimpleChoice* - базовая реализация справочника;
- *LinkedChoice* - связанный справочник, содержит дополнительный параметр в поле params: linkedChoice - название связанного справочника, обязательный для заполнения параметр;
- *ParametrizedChoice*
- *TimeCacheChoice* - временный справочник. Справочника переформировывается, через заданный промежуток времени.

Список контроллеров:

- CompositeChoiceListProcessor;
- CriteriaChoiceListProcessor;

- DataSourceChoiceListProcessor;
- GrailsApplicationChoiceListProcessor;
- HierarchyChoiceListProcessor;
- JsonURLChoiceListProcessor;
- PropertiesChoiceListProcessor;
- RESTChoiceListProcessor.

Описание параметров контроллеров см. в разделе **“Стандартные контроллеры для справочников”**

Пример конфигурации:

```
[
    Desktops : new SimpleChoice('Desktops',
'grailsApplicationChoiceListProcessor', [
        values: 'repo.meta.desktops.available'
      ]),
      DesktopsWOAdmin  : new SimpleChoice('DesktopsWOAdmin',
grailsApplicationChoiceListProcessor', [
        values: 'repo.meta.desktops.available',
        excludeParams: ['console']
      ]),
          Department  : new SimpleChoice('Department',
'criteriaChoiceListProcessor', [
        index     : 'ID',
        label     : '${fields.Title}',
        search    :
Search.from('Department').whereCurrentVersion().orderBy('Title'),
        fields    : ['ID', 'Title'],
        itemFields: ['ID', 'Title']
      ])
]
```

Формирование данных конфигурации “tabs”

Конфигурация описывает вкладки в карточках объектов.

Объявление новой вкладки в скрипте “tabs” происходит следующим образом:

```
Название вкладки: new Tab(label: '', controller: '', classStyle: '', params: [])
```

- label - название вкладки. Выводится пользователю в интерфейсе системы;
- controller - название контроллера, реализующего бизнес-логику вкладки. Является обязательным для заполнения. Контроллер выбирается из базового набора контроллеров вкладок (см. раздел **“Стандартные контроллеры для вкладок”**);
- classStyle - название класса иконки, которая будет отображаться перед названием вкладки. Используются иконки Glyphicons, доступные названия можно посмотреть перейдя по ссылке **http://getbootstrap.com/2.3.2/base-css.html#icons**;
- params - дополнительные параметры. Параметры определяются контроллером, реализующим вкладку;
- view - табличное представление InboxView, которое используется для вывода информации пользователю. Параметр доступен только для контроллеров: historyTab, linkedTab, linkedTabWithSearch.

Пример конфигурации:

```
[
    details: new Tab(label: 'Реквизиты', controller: 'detailsTab', classStyle:
'icon-book', params: [processLayout: true]),
    geodata: new Tab(label: 'Геоданные', controller: 'detailsTab', classStyle:
'icon-book', params: [processLayout: true]),
    history: new Tab(label: 'История', controller: 'historyTab', params: [view:
'index'],classStyle: 'icon-list'),
    content: new Tab(label: 'Вложения', controller: 'contentDetailsTab',
classStyle: 'icon-file', params: [tabLayout: 'default']),

    AddAgreement: new Tab(label: 'Доп. соглашения', controller:
'linkedTabWithSearch',
        classStyle: 'icon-list-alt', params: [
        tabView: 'Contract',
        linkedCaseClass: 'Contract',
        linkedBy: 'DocumentLink',
        valueParamName:'VersionSeriesId',
        currentVersionOnly: true,
        search:'AddAgreementSearch'
```

```
                ])
]
```

Формирование данных конфигурации “tools”

Инструмент – это модуль расширения платформы, который позволяет выполнять различные действия над объектом или папками.

Конфигурация описывает “tools” инструменты в папках и карточках объектов. Объявление инструмента в скрипте tools’ происходит следующим образом:

```
Название инструмента: new Tool(label: '', icon: '', controller: '', params: [])
```

- label - название инструмента. Выводится пользователю в интерфейсе системы;
- icon - название класса иконки, которая будет отображаться перед названием инструмента. Используются иконки Glyphicons, доступные названия можно посмотреть перейдя по ссылке **http://getbootstrap.com/2.3.2/base-css.html#icons**;
- controller - название контроллера, реализующего бизнес-логику инструмента. Является обязательным для заполнения. Контроллер выбирается из базового набора контроллеров инструментов (см. раздел **“Стандартные контроллеры для инструментов”**);
- params - дополнительные параметры. Параметры определяются контроллером, реализующим инструмент.

Пример конфигурации:

```
[
    newCase        : new Tool(label: 'Новый документ', icon: 'new-icon-add',
controller: 'createCaseTool'),

    deleteCase     : new Tool(label: 'Удалить', icon: 'new-icon-remove',
controller: 'deleteCaseTool',
    params: [
                hotKeys: ['Ctrl', 'Shift', 'D'],
                safeDelete: true
                        ]),

    newSubContract: new Tool(label: 'Новый договор субподряда', icon: 'new-icon-
add', controller: 'createCaseTool',
```

```
    params: [
        classType : 'Contract',
        defaultParams: '{DocumentLink: "$vo.params.VersionSeriesId", ContractType:
"233"}'
                        ])
]
```

Интернационализация

Интернационализация позволяет адаптировать систему для потенциального использования практически в любом месте. В данном разделе содержится список доступных языков в системе.

Создание объекта интернационализации

Для того чтобы создать интернационализацию необходимо перейти в раздел “Конфигурация приложения”, папку “Интернационализация” и создать новый документ.
В поле “Наименование конфигурации” требуется указать код языка в соответствии с ISO 639-1 (например: ru, en, it, es, fr и т.д.). В поле “Данные конфигурации” указать перевод объектов системы на требуемый язык.

Для применения нового языка, необходимо чтобы данный язык был указан как основной язык браузера (настройка языка браузера отличается в зависимости от браузера).
Например, для браузера Google Chrome необходимо выбрать из меню пункт “Настройки”, перейти в раздел “Дополнительные” и найти настройку языков. Далее необходимо добавить желаемый язык (для которого создана конфигурация в системе), удалить все остальные языки и перезапустить браузер.

Для применения настроек языка в Системе требуется перезайти в Систему. Все наименования объектов Системы указанные в конфигурации будут изменены.

Формирование данных конфигурации

Поле «Данные конфигурации» представляет собой json-скрипт содержащий перевод компонентов и названий объектов на другой язык.

Пример конфигурации:

```
[
    'application.header.title'    : 'GALAKTIKA',
    // Inboxes
    'Search'                      : 'Поиски',
    'Layout'                      : 'Карточки',
    'Scripts'                     : 'Скрипты',
    'Folders'                     : 'Папки',
    'Folder group'                : 'Группы папок',
    'Folder view'                 : 'Отображения папок',
    'Users'                       : 'Пользователи'
]
```

Стандартные контроллеры и методы отображения

Стандарные методы отоброжения данных

Метод отображения данных можно указать редакторе классов в поле “Метод отображения” для свойства объекта или переопределив параметр “templateName” в карточке объекта, указав необходимый метод отображения.

Список параметров:

- input – однострочное текстовое значение. Значение по умолчанию для большинства полей;
- textarea – многострочное текстовое значение (соответствует установленному чек-боксу “Большая колонка” в редакторе классов для свойства класса);
- select – выбор одного значения из справочника;

- multiselect – выбор одного или нескольких значений из справочника. Для свойства в редакторе классов должен быть установлен чек-бокс “Является множественным”;

- datePicker – дата в формате «dd.MM.yyyy». Значение по умолчанию для полей с типом дата;

- dateTimePicker – дата в формате «dd.MM.yyyy HH:mm:ss»;

  Пример отображения данных для следующих параметров:

  input, textarea, select, multiselect, datePicker, dateTimePicker.

- choiceIcon – отображение иконки, рядом с полем, соответствующая выбранному значению в справочнике;

- password - поле для ввода пароля, вводимые символы отображаются звездочками, точками или другими знаками (в зависимости от используемого браузера);

- checkbox – поле типа «флаг», значение установлено/значение не установлено. Значение по умолчанию для полей типа Boolean;
- currency – отображение валют. Разделитель тысяч - пробел, разделитель дробной части - запятая, отображение 2-х знаков после запятой (для типа данных FLOAT, DOUBLE).
  Дополнительный параметр “null_value” - который позволяет задать значение по умолчанию, если поле пусто (например: -, 0).

  Поле “Пользовательская конфигурация поля (JSON) ”

```
{
    "currency" :
        {
              "null_value": "-"
         }
}
```

- loadData - быстрый поиск значений справочника. В основном используется для справочников, которые содержат большое количество данных.

Для реализации необходимо добавить в редакторе классов в поле “Пользовательская конфигурация поля (JSON)” следующий код:

```
{
  "dataController": "ChoiceLiveSearch",
  "dataService": "choiceLiveSearchService"
}
```

Примечание: Если требуется добавить быстрый поиск по справочнику в фильтр в папке, то необходимо в конфигурацию (Поиски) добавить параметр “renderer”: “loadData”.

- attachmentList отображение списка вложений (для свойства ‘Files’) в карточке объект

Пример конфигурации:

```
{    "panels": [

     { "legend": "Значения параметра templateName в карточке объекта",
        "componentsRows": [
          {"components": [ {"cssClass": "span6", "propName":
"DocumentStatus","label":"choiceIcon","templateName": "choiceIcon"},
          {"cssClass": "span4", "propName": "checkbox",
"templateName":"checkbox"}]},
          {"components": [ {"cssClass": "span6", "propName":
"Test","label":"password", "templateName": "password"}, {"cssClass": "span3",
"propName": "currency", "templateName":"currency"}]}
        ]
      },
      { "legend": "Отображение параметра attachmentList в templateName",
        "componentsRows": [
         {"components": [ {"cssClass": "span8", "propName": "Files",
"templateName": "attachmentList","required": true}]}

        ]
      }
```

```
    ]
  }
```

- staticText - отображение статического текста с сообщением пользователю. Дополнительный параметр params: messageCode - сообщение, с поддержкой интернационализации.

  Пример конфигурации:

```
{"components": [{"cssClass": "span6", "propName": "Code", "templateName":
"staticText",
                                "params": {"messageCode": "Штрих код не
распознан, повторите сканирование"}}]}
```

- hidden – скрытое поле, на карточке не отображается;

- code – отображение конфигурационных скриптов. Дополнительный параметр “mode”, который отвечает за подсветку синтаксиса для кода скриптов. Доступны два значения: groovy или json;

  Пример конфигурации:

```
"components": [ {"cssClass": "span8", "propName": "TestCode", "templateName":
"code", "params":{"mode":"json"}}]}
```

Стандартные контроллеры для инструментов

Для выбора доступны контроллеры общего и административного назначения (пакет `com.galantis.ecm.tool`).

Для настройки инструментов доступны следующие общие параметры:

- *withText* - добавляет название инструмента после иконки. Доступные значения:
  - true - выводить название инструмента;
  - false - не выводить название инструмента. По умолчанию значение параметра false.

- *hotKeys* - позволяет задать комбинацию клавиш для быстрого вызова инструмента.

- *checkinType* - параметр используется для версионируемых объектов. Параметр может принимать следующие значения:

- *MAJOR* - изменение старшей версии;
- *MINOR* - изменение младшей версии.

Параметр *checkinType* используется в следующих контроллерах: saveCaseTool, deleteContentTool, uploadContentTool, updateContentTool, reportTool, parameterReportTool, revokeProcessTool.

- *onComplete* - определяет тип действия после использования инструмента. Значения параметра:
- ToolCompleteAction.RELOAD – выполняется обновление вкладки;
- ToolCompleteAction.TO_CASE – открывается новая карточка, если создается новая версия;
- ToolCompleteAction.TO_INBOX – выполняется переход в папку, название папки в параметре inboxName;
- ToolCompleteAction.TO_BACK_URL – выполняется действие назад;
- ToolCompleteAction.NONE – никакие действия не выполняются.

Параметр *onComplete* используется в следующих контроллерах: saveCaseTool, launchProcessTool, routeTool, revokeProcessTool.

Контроллеры общего назначения

1. **createCaseTool** - обеспечивает создание новых объектов в системе. Инструмент, может быть добавлен как на карточку объекта, так и в папку. Для настройки инструмента доступны следующие дополнительные параметры:

- *classType* - тип строка, обязательный параметр для инструмента, добавляемого на карточку объекта. Если инструмент добавляется в папку, то параметр должен отсутствовать. Содержит название класса объекта, который требуется создать.

- *defaultParams* - тип строка, используется для задания начальных значений параметров объекта. Параметр содержит json объект, в котором ключ - это название свойства нового объекта, значение - значение свойства нового объекта. В значении можно использовать макросы:

  ${макрос}

где макрос — это код groovy, возвращающий строку. В макросе доступны следующие переменные: *vo* - родительский объект (если новый объект создается с карточки объекта), *user* - текущий пользователь, *current* - текущая дата.

Пример использования параметров:

```
newSubContract: new Tool(label: 'Новый договор субподряда', icon: 'new-icon-add',
controller: 'createCaseTool',
params: [
    classType : 'Contract',
    defaultParams: '{DocumentLink: "$vo.params.VersionSeriesId", ContractType:
"233"}'])
```

2. **deleteCaseTool** - обеспечивает удаление последней версии объекта системы. Инструмент может быть добавлен только на карточку объекта и не предназначен для работы из папки.
   Параметры:
   - *recursive* - тип boolean, значение по умолчанию true - выполняется удаление всех версий документа и всей информации связанной с документом, false - удаление только последней версии документа.

   Пример использования параметров:

```
deleteCase     : new Tool(label: 'Удалить', icon: 'new-icon-remove', controller:
'deleteCaseTool',
                 params: [hotKeys: ['Ctrl', 'Shift', 'D' ]])
```

3. **saveCaseTool** - обеспечивает сохранение свойств объекта. Инструмент может быть добавлен только на карточку объекта. Для настройки инструмента доступны следующие дополнительные параметры:
   - *checkinType;*
   - *onComplete;*
   - *inboxName;*

   - *checkExist* - параметр типа список строк, позволяет проверить что созданный объект не является дубликатом.В проверке участвуют только свойства, перечисленные в параметре *checkExist*. Если найден объект с идентичными значениями, указанным в параметре *checkExist*, то объект не будет сохранен. На экране отобразится

предупреждение: “Объект с такими же свойствами уже существует”. По умолчанию проверка не выполняется.

- *rewriteParams* - тип параметра строка, позволяет переинициализировать свойства объекта перед сохранением. Содержит json объект, в котором: ключ - это название свойства объекта, значение - это значение параметра объекта. Поддерживает макросы на языке groovy, формат макроса: ${макрос}

```
saveAddAgreementAsMajor: new Tool(label: 'Сохранить Доп. соглашение', icon: 'new-icon-save',controller:'saveCaseTool',
                         params: [
checkinType: CheckInType.MAJOR,
checkExist: ['ContractNumber'],
rewriteParams: '{ ContractDate: "${current.date}"}',
                                  ])
```

4. **commentTool** - обеспечивает добавление комментария к текущему объекту.Инструмент может быть добавлен только в карточку объекта. Комментарий отображается на вкладке “История” или на дополнительно настроенной вкладке “comments”.

   Пример использования:

```
addComment   : new Tool(label: 'Добавить комментарий', icon: 'icon-comment',
controller: 'commentTool',
                       params: [hotKeys: ['Shift', 'Q']])
```

5. **reloadCurrentTabTool** – обеспечивает обновление активной вкладки объекта.

   Пример использования:

```
reloadCurrentTab: new Tool(label: 'Обновить', icon: 'icon-refresh', controller: 'reloadCurrentTabTool')
```

6. **changeClassTool** - обеспечивает смену класса объекта.

   Пример использования:

```
changeClass    : new Tool(label: 'Поменять класс', icon: 'icon-leaf', controller: 'changeClassTool',
                          params: [:])
```

7. **printMapTool** - обеспечивает печать видимой области карты.

   Пример использования:

```
printMap       : new Tool(label: 'Распечать видимую область карты', icon: 'icon-
print', controller: 'printMapTool',
                                params: [selector: 'canvas'])
```

8. **uploadContentTool** - обеспечивает добавление вложений к объекту. Может использоваться только на карточке объекта.

- *checkinType;*
- *acceptedFiles* - допустимые расширения фалов или mime type (**https://www.dropzonejs.com/#configacceptedFiles)**;
- *saveData* - параметр типа boolean, предназначен для проверки сохранения данных перед выполнением действий с вложениями.
  -true - отображает предупреждение пользователю “Перед выполнением действия сохраните изменения”, если пользователь внес изменения и не сохранил карточку. Значение по умолчанию;
  -false - не отображает предупреждение, изменения в карточке объекта будут потеряны.
- *maxFiles* - максимальное количество файлов для загрузки.

  Пример использования параметров:

```
uploadContent  : new Tool(label: 'Загрузить вложения', icon: 'icon-upload',
controller: 'uploadContentTool',
                                params: [
checkinType: CheckInType.MINOR,
saveData: true,
maxFiles: 10,
acceptedFiles: 'application/pdf, .docx'
                                        ])
```

9. **deleteContentTool** - обеспечивает удаление файлов, приложенных к объекту. Инструмент может быть добавлен только на вкладку управления вложениями объекта. Для настройки инструмента доступны следующие дополнительные параметры:

- *checkinType;*

- *saveData*;

- *mode* - тип строка, определяет, какие вложения требуется удалить. Доступны два значения:
  - *current* - удаление одного выбранного вложения;
  - *common* - удаление нескольких выбранных вложений объекта.

- *enableContentCheck* - тип boolean, отображение чек-боксов для удаления нескольких вложений, значение по умолчанию false – чек-боксы не отображаются на вложении, true - отображаются.

  Пример использования параметров:

```
deleteContent  : new Tool(label: 'Удалить вложение', icon: 'icon-remove-sign',
controller: 'deleteContentTool',
                params: [mode: 'current', checkinType: CheckInType.MINOR])

deleteContents : new Tool(label: 'Удалить вложения', icon: 'icon-remove-sign',
controller: 'deleteContentTool',
                            params: [mode: 'common', checkinType:
CheckInType.MINOR, enableContentCheck:true])
```

10. **updateContentTool** - обеспечивает обновление выбранного вложения. Инструмент может быть добавлен только на вкладку управления вложениями объекта.Поддерживает параметр:
- *checkinType;*
- *saveData*;
- *acceptedFiles.*

  Пример использования:

```
  updateContentTool: new Tool(label: 'Обновить документ', icon: 'icon-refresh',
controller: 'updateContentTool',
  params: [
      checkinType: CheckInType.MINOR,
acceptedFiles: 'application/pdf, .docx',
      withText : true
      ])
```

11. **downloadTool** - обеспечивает загрузку выбранного вложения на персональный компьютер. Инструмент может быть добавлен только на вкладку управления вложениями объекта.

    Пример использования:

```
downloadContent: new Tool(label: 'Скачать', icon: 'icon-download-alt', controller:
'downloadTool')
```

12. **printTool** - обеспечивает печать выбранного вложения. Инструмент может быть добавлен только на вкладку управления вложениями объекта.

    Пример использования:

```
printContent: new Tool(label: 'Распечатать файл', icon: 'icon-print', controller:
'printTool'),
```

Контролеры для работы с бизнес-процессом

13. **launchProcessTool** - обеспечивает запуск бизнес-процесса в используемом Workflow engine. Инструмент можно добавить только на карточку объекта. Для настройки инструмента доступны следующие параметры:

- *processName* - тип строка, идентификатор запускаемого процесса в используемом Workflow engine;
- *peClassType* - тип строка, название класса с типом PE;
- *taskParams* - тип строка, содержит json объект, позволяет проинициализировать свойства задачи начальными значениями при запуске бизнес-процесса. В значении можно использовать макросы;
- *onComplete* - параметр определяет тип действия после запуска процесса;
- *isUnique* - параметр принимает два значения true/false.

Пример использования параметров:

```
approvalProcess: new Tool(label: 'Запустить согласование', icon: 'icon-play',
controller: 'launchProcessTool',
params: [
        isUnique: true,
        processName: 'CARTApprovalProcess',
        peClassType: 'ApprovalProcess',
```

```
        taskParams: JsonOutput.toJson([
            ParentContract: '${vo.params.VersionSeriesId}'
            ContractNumber: '${vo.params.ContractNumber}'
                                    ]),
        onComplete: ToolCompleteAction.TO_INBOX,
        inboxName: 'NewContract',
        withText : true
        ]),
```

14. **routeTool** - обеспечивает выполнение шагов(действий) по бизнес-процессу. Инструмент можно добавить только на карточку с типом объекта PE. Для настройки инструмента доступны следующие дополнительные параметры:
- *onComplete* - параметр определяет тип действия после выполненного действия;
- *taskParams* - тип строка, содержит json объект, позволяет проинициализировать свойства задачи начальными значениями при запуске бизнес-процесса. В значении можно использовать макросы;
- *requiredFields* - параметр типа список строк, список обязательных свойств объекта, которые должны быть заполнены перед выполнением действия;
- *stepResponses* - параметр типа список строк, который содержит названия шагов (действий) в бизнес-процессе. Если не указывать данный параметр, то инструмент отобразит все шаги из бизнес-процесса;
- *needComment* – параметр типа список строк, в котором перечисляются названия шагов (stepResponses) для которых комментарий является обязательным для заполнения.

Пример использования параметров:

```
approveCase : new Tool(label: 'Выполнить действие', icon: 'icon-check',
controller: 'RouteTool',
    params: [
    withText : true,
    stepResponses: ['Approve', 'Reject', 'GetBack'],
    needComment: ['Reject', 'GetBack'],
            ])
```

15. **revokeProcessTool** - обеспечивает удаление запущенного процесса. Инструмент может быть добавлен только на карточки объектов, по которым запускается процесс. Для настройки инструмента доступны следующие параметры:

- *checkinType;*
- *onComplete;*
- *updateParams* – тип строка, содержит json объект, позволяет проинициализировать свойства объекта по которому был запущен бизнес-процесс. В значении можно использовать макросы.

Пример использования параметров:

```
revokeProcess  : new Tool(label: 'Отозвать процесс', icon: 'icon-stop',
controller: 'revokeProcessTool',
        params: [
            checkinType: CheckInType.MAJOR,
            updateParams: JsonOutput.toJson([ContractState:'173']),
            onComplete: ToolCompleteAction.TO_CASE
                ])
```

Контролеры для работы с отчетами

1. **reportTool** - обеспечивает формирование отчета в формате xlsx по объектам в папке. Для настройки инструмента доступны следующие дополнительные параметры:

- *template* - объект типа {@link com.galantis.ecm.model.report.XlsxReportTemplate} с параметрами формирования отчета.

Параметры для формирования отчета:

- *sheetName* - название листа xlsx файла. Необязательный параметр;
- *headerRow* - номер строки, в которой содержится шаблон заголовка и в которой необходимо сформировать заголовок отчета;
- *contentRow* - номер строки, в которой содержится шаблон для колонок с содержимым отчета. Формирование данных отчета будет выполняться, начиная с указанной строки;
- *sheetIndex* - индекс листа в шаблоне. При формировании отчета по шаблону параметр обязателен для заполнения;

- *attachmentName* - название файла шаблона, по которому необходимо сформировать отчет. Файл шаблона должен быть прикреплен к карточке с описанием папки на вкладке “Вложения” в интерфейсе системного администратора в меню “Конфигурация приложения -> Папки”. Т.е. шаблон должен быть прикреплен к той папке из которой вызывается инструмент. Параметр является необязательным.
  Пример использования параметров:

```
exportToFile   : new Tool(label: 'Сформировать отчет в формате xlsx', icon: 'icon-print', controller: 'reportTool',
                              params: [
template: new XlsxReportTemplate(headerRow: 0, contentRow: 1, attachmentName: 'inbox-report-template.xlsx', sheetIndex: 0)])
```

2. **parameterReportTool** - обеспечивает формирование BIRT-отчета из заданного шаблона в указанном формате с пользовательскими параметрами. Инструмент может быть добавлен в карточку объекта или в папку.
   Для настройки инструмента доступны следующие дополнительные параметры:
- *withText;*
- *templateChoice* - тип строка, название справочника в котором содержится список наименований шаблонов отчётов (ReportTemplate.Title) используемых для генерации отчета;
- *templateVsId* - содержит VersionSeriesId шаблона. Если указан параметр templateVsId, то параметр templateChoice игнорируется, а также не отображается список для выбора шаблона отчёта.
- *attachToObject* - boolean, определяет нужно ли добавить сформированный отчёт в текущий открытый объект, если инструмент добавлен в карточку объекта;
- *downloadAttach* - boolean, определяет нужно ли предлагать пользователю скачать отчёт после генерации;
- *defaultFormat* – тип строка, формат генерации отчета по умолчанию;
- *availableFormats* - парамерт тип список строк, список возможных форматов генерации отчета. Отображается в виде выпадающего списка возможных форматов перед генерацией отчета;
- *checkinType*. Если инструмент добавлен в карточку объекта.

Пример использования параметров для инструмента, который добавлен в карточку объекта:

```
 approvalList: new Tool(label: 'Лист согласования', icon: 'icon-ok-sign',
controller: 'parameterReportTool',
params: [
    templateVsId: 4813,
    attachToObject: true,
    downloadAttach: true,
    checkinType: CheckInType.MINOR,
    availableFormats: [ReportFormat.PDF],
    defaultFormat: ReportFormat.PDF,
    withText: true
])
```

Пример использования параметров для инструмента, который добавлен в папку:

```
approverReport: new Tool(label: 'Отчет согласующего', icon: ' icon-file',
controller: 'parameterReportTool',
params: [
    templateChoice: 'ParameterTemplate',
    attachToObject: false,
    downloadAttach: true,
    availableFormats: [ReportFormat.PDF, ReportFormat.XLS],
    defaultFormat: ReportFormat.PDF,
    withText: true
])
```

Контролеры административного назначения

1. **userSearchTemplateTool** - системный контроллер, обеспечивает сохранение пользовательских шаблонов поиска, для использования в только папках.
   Пример использования:

```
searchTemplate : new Tool(label: 'Сохранить как', controller:
'userSearchTemplateTool'),\
```

2. **reloadMetaTool** - системный контроллер, отвечает за обновление кэша мета-данных системы. Обновляются все конфигурации из интерфейса системного администратора (классы (из редактора классов), папки, карточки и т.д.) Для использования в папках и

на карточке объекта не предназначен.

Пример использования:

```
reloadMeta     : new Tool(label: 'Обновление мета информации', icon: 'icon-refresh', controller: 'reloadMetaTool')
```

3. **metaGeneratorTool** - вспомогательный контроллер для интерфейса системного администратора. Обеспечивает формирование конфигураций типов «Папка», «Табличное представление» и «Карточка».

   Пример использования:

```
metaGenerator  : new Tool(label: 'Мета-генератор', icon: 'icon-cog', controller: 'metaGeneratorTool'),
```

4. **configValidationTool** - системный контроллер, обеспечивает валидацию содержимого JSON-блоков. Если в результате были проверки конфигурации были найдены ошибки, то список ошибок будет выведен на экран.При успешном завершение проверки будет отображено сообщение “Проверка успешно завершена, ошибок не найдено”.

   Пример использования:

```
validate       : new Tool(label: 'Проверить конфигурацию', icon: 'icon-check', controller: 'configValidationTool',
                              params: [hotKeys: ['Ctrl', 'Shift', 'V']]),
```

Стандартные контроллеры для вкладок

Для выбора доступны следующие контроллеры (пакет `com.galantis.ecm.tab`).

1. **detailsTab** - обеспечивает отображение и редактирование свойств объекта. Свойства на вкладке отображаются в соответствии с настройками карточки объекта Layout.Form - тип java.util.Map, форма, в которой выводятся параметры объектов. Позволяет при вызове инструментов выполнять отправку формы вместо ajax запроса.

   Пример использования:

```
details: new Tab(label: 'Реквизиты', controller: 'detailsTab', classStyle: 'icon-book', params: [processLayout: true]),
```

2. **contentDetailsTab** - обеспечивает способ отображения прикрепленных к объекту файлов.

   Для настройки инструмента доступны следующие дополнительные параметры:

- *tabLayout*- тип строка, обеспечивает способ вывода файлов. Доступны следующие значения:

- default - отображение списка загруженных файлов с возможностью предварительного просмотра содержимого;
- list - отображение списка загруженных файлов без возможности просмотра содержимого.

Пример использования параметра:

```
content: new Tab(label: 'Attachments', controller: 'contentDetailsTab',
classStyle: 'icon-file',
                params: [tabLayout: 'default'])
```

3. **historyTab** - обеспечивает отображение действий связанных с изменением объекта. Параметры:

- *view* - тип строка, имя представления (gsp) для отображения, которое содержится в папке historyTab в проекте;
- *searchCriteria* - дополнительное условие поиска истории;
- *customClassType* - тип класса объектов истории, по умолчанию - FNKitHistoryItem;
- *customProperties* - массив свойств, которые нужно вычитать из объекта истории (пр.: [“Duration”,”CustomName”]). Используется только вместе с параметром *customClassType*

Пример использования стандартной вкладки “История”:

```
history: new Tab(label: 'История', controller: 'historyTab',
                                params: [view: 'index'],classStyle: 'icon-list')
```

Пример использования вкладки “Комментарии” с дополнительными параметрами:

```
comments: new Tab(
                label: 'Комментарии',
                controller: 'historyTab',
                params: [
                        view: 'index_comments',
                        searchCriteria: new Criteria(Criteria.AND, [new
Condition("OperationType", ConditionType.EQUAL, "comment.added")])],
                classStyle: 'icon-comment-alt'
                )
```

4. **linkedTab** - обеспечивает отображение в табличном виде связанных объектов. Для данного контроллера необходимо заполнить три обязательных

   параметра *tabView*, *linkedCaseClass* и *linkedBy*.

   Параметры:

- *tabView* - тип строка, название табличного представления InboxView. Обязательный параметр. Используется контроллером для вывода связанных объектов;
- *linkedCaseClass* - тип строка, название класса связанных объектов. Параметр обязателен для заполнения.
- *linkedBy* - тип строка, название свойства у связанного объекта (по которому связанны дочерний объект с родительским объектом). Параметр обязателен для заполнения;
- *valueParamName* - тип строка, название свойства класса родительского объекта по которому осуществляется поиск в параметре linkedBy. По умолчанию используется свойство класса «ID».
- *condition* - тип строка, условие поиска для связи между родительским и дочерними объектами. По умолчанию используется значение equal. Таким образом, на вкладке в таблице связанных объектов отображаются дочерние объекты, для которых выполняется условие: значение свойства класса заданного для параметра ‘linkedBy’ равно значению свойства класса указанного для параметра ‘valueParamName’.
- *currentVersionOnly* - тип boolean, принимает два значения:
  - true - в таблице будут отображаться только последние актуальные версии связанных объектов;
  - false - будут отображаться все версии связанных объектов.
- *canDelete* - тип boolean. Принимает два значения:
  - true - для каждой строки в таблице отображается кнопка «Удалить», позволяющая выполнить удаление выбранного объекта;
  - false - кнопка «Удалить» для каждой строки в таблице не отображается. По умолчанию значение параметра false.
- *activeTab* - тип строка, название вкладки, которая будет открыта при переходе в карточку связанного объекта.

Пример использования параметров:

```
Act: new Tab(label: 'Акты', controller: 'linkedTab', classStyle: 'icon-
screenshot',
```

```
                        params: [
                                        tabView: 'Act',
                                        linkedCaseClass: 'Act',
                                        linkedBy: 'ContractLink',
                                        valueParamName: 'VersionSeriesId',
                                        currentVersionOnly: true,
                                        canDelete: true,
                                        activeTab: 'history'])
```

5. **linkedTabWithSearch** - обеспечивает отображение в табличном виде связанных объектов и панель фильтрации данных. Контроллер наследует все параметры контроллера linkedTab и обязательный параметр *search*.

   Параметры:

- *search* - тип строка, название поиска (FNKitSearch) для настройки панели фильтрации данных на вкладке. Обязательный параметр;
- *tabView;*
- *linkedCaseClass;*
- *linkedBy;*
- *valueParamName;*
- *currentVersionOnly;*
- *canDelete;*
- *activeTab;*
- *condition.*

Пример использования параметров:

```
Act: new Tab(label: 'Акты', controller: 'linkedTab', classStyle: 'icon-
screenshot',
                        params: [
                                        tabView: 'Act',
                                        linkedCaseClass: 'Act',
                                        linkedBy: 'ContractLink',
                                        valueParamName: 'VersionSeriesId',
                                        currentVersionOnly: true,
```

```
                                        canDelete: false,
                                        search: 'ActTabSearch'])
```

6. **linkedMultipleTab** - обеспечивает отображение нескольких типов связанных объектов на одной вкладке родительского объекта. Например в карточке “Договор” на вкладке “Связанные документы” могут отображаться связанные объекты типа “Акты”, “Доп.соглашения”, “Счета-фактуры”. Доступный параметр:
   - *tabConfigs* - тип массив строк, который содержит названия вкладок (tabName). Параметр обязателен для заполнения.

   Пример использования параметров:

```
Documnets   : new Tab(label: 'Связанные документы', controller:
'linkedMultipleTab', classStyle: 'icon-copy',
                                        params: [
                                                tabConfigs: ['Act',
'AddAgreement', 'Bill']
                                                ])
```

7. **HTMLTemplateTabController**- обеспечивает отображение HTML страницы, в которой можно использовать вставки для вывода свойств объекта. Дополнительно требуется создать класс “HTMLTemplateDocument” со свойством “HTMLTemplateBody” типа String и указать чеб-бокс “Большая колонка”. В карточке в поле “HTMLTemplateBody” внести HTML шаблон. Доступный параметр:
   - *docVsId* - тип строка, значение свойства “VersionSeriesId” объекта “HTMLTemplateDocument”.

   Пример использования параметров:

```
invoice: new Tab(label: 'Счет-фактура', controller: 'HTMLTemplateTab', classStyle:
'icon-list',
                        params: [docVsId: 141])
```

8. **linkedTasksTab** - обеспечивает вывод задач, запущенных по текущему объекту в Workflow engine. Параметры:
   - *tabView* - тип строка, название табличного представления InboxView. Обязательный параметр.

- *linkedCaseClass* - тип строка, название класса с типом хранилища PE, который используется для поиска в Workflow engine.

Пример использования параметров:

```
linkedEcmTasks : new Tab(label: 'Задачи', controller: 'linkedTasksTab',
classStyle: 'icon-table',
                        params: [
                                                tabView : 'ApprovalProcessView',
                                                linkedCaseClass:
'ApprovalProcess'
                                        ])
```

Стандартные контроллеры для справочников

1. **criteriaChoiceListProcessor** – обеспечивает поиск в основном хранилище по sql запросу.Параметры:

- *index* - тип строка, название поля используемого в качестве идентификатора элемента справочника;
- *label* - тип строка, содержит описания элемента справочника, которое выводится пользователю. Имеет следующий формат:
  ‘${<название поля с префиксом fields>} …’ .
  Пример: ‘${fields.Title} - ${fields.Number}’
- *search* - sql запрос;
- *fields* - набор полей для формирования описания элемента справочника;
- *itemFields* - набор кэшируемых полей.

Примеры:

```
Department  : new SimpleChoice('Department', 'criteriaChoiceListProcessor', [
     index     : 'ID',
     label     : '${fields.Title}',
     search    : Search.from('Department').whereCurrentVersion().orderBy('Title'),
     fields    : ['ID', 'Title'],
     itemFields: ['ID', 'Title']
   ]),
```

```
  DepartmentUsers : new SimpleChoice('Users', 'criteriaChoiceListProcessor', [
    index     : 'Login',
    label     : '${fields.Name}',
    search    : Search.from('Users').where('Department', ConditionType.CONTAINS,
'${vo.params.AssigneeDepartment}').orderBy('Name'),
    fields    : ['Login', 'Name'],
    itemFields: ['Login', 'Name']
  ])
```

2. **grailsApplicationChoiceListProcessor** – формирует справочник по конфигурации приложений. Параметры:

- *values* - тип строка, название конфигурации;
- *excludeParams* - параметр типа список строк, содержащий параметры, которые не должны использоваться в choice;
- *label*;
- *index*;
- *fields*;
- *itemFields*;

Примеры:

```
Desktops      : new SimpleChoice('Desktops',
'grailsApplicationChoiceListProcessor', [
       values: 'repo.meta.desktops.available'
]),

DesktopsWOAdmin  : new SimpleChoice('DesktopsWOAdmin',
'grailsApplicationChoiceListProcessor', [
values: 'repo.meta.desktops.available',
excludeParams: ['console']
])
```

4. **CompositeChoiceListProcessor** - объединяет справочники в один справочник. Параметры:

- *composite* - параметр типа список строк, содержащий список названий справочников;

```
EmployeeAndCandidate: new SimpleChoice('EmployeeAndCandidate',
'compositeChoiceListProcessor', [
    composite: ['Employee', 'Candidate']
])
```

5. **DataSourceChoiceListProcessor** - осуществляет вычитку элементов в справочник из произвольного DataSource. DataSource должен быть предварительно зарегестрирован в файле конфигурации grails. Параметры:

- *dataSourceName* - тип строка, название DataSource;
- *query* - sql запрос;
- *label*;
- *index*;
- *fields*;
- *itemFields*;

Доступные DataSource настраиваются в grails, в конфигурации справочника указывается название. Пример:

```
dataSources {
  dataSource {
    <параметры>
  }
  orgstructure {
    <параметры>
  }
  <название 1> {
    <параметры>
  }
  ...
  <название n> {
    <параметры>
  }
}
```

Пример:

```
PersNoSearch    : new TimeCacheChoice('PersNoSearch',
'dataSourceChoiceListProcessor', 4, TimeUnit.HOURS, [
```

```
        index          : 'TAB_NUM',
        label          : '${fields.TAB_NUM}',
        query          : 'SELECT TAB_NUM FROM GLX#PERSONS',
        fields         : ['TAB_NUM'],
        dataSourceName: 'dataSource_orgstructure'
    ])
```

6. **HierarchyChoiceListProcessor** - обеспечивает поиск по иерархии классов справочников доступных в системе.

7. **JsonURLChoiceListProcessor** - разбивает и выводит json по ссылке. Параметры:
   - *url* - тип строка, ссылка на json;
   - *label*
   - *fields*;

   Пример json:

```
[
  {
    "index": "484",
    "label": "X5",
    "param: {
      "Title": "X5"
    }
  },
  ...
  {
    "index": "500",
    "label": "Mondeo 4",
   "param: {
      "Title": "Mondeo 4"
    }
  }
]
```

7. **PropertiesChoiceListProcessor** - обеспечивает поиск по свойствам классов доступных в системе. Параметры:
   - *class* - тип строка, системное имя класса в системе;

- *itemFields*;

8. **RESTChoiceListProcessor** - вычитывает справочник из другого экземпляра по REST API и преобразует его в соответствии с конфигурацией. Параметры:
- *choiceName* - тип строка, название справочника;
- *processingEnabled* - тип boolean, true - преобразует значения, с помощью макросов, указанных в ‘label’, false - не преобразует.;
- *items*;
- *label*;
- *index*;
- *fields*;
- *itemFields*;

9. **ldapChoiceListProcessor** – формирует справочник по пользователям из AD. Параметры:
- *label* - свойство с именем пользователя;
- *index* - свойство с идентификатором пользователя.

Настройка связанных справочников

Настройка связанных справочников с выбором значений

В системе есть возможность при выборе определённого значения отображать в другом поле список значений связанных с этим полем.

Краткое описание действий:

1. Создать классы в классе “Справочники”;
2. Установить связь между классами по свойству класса;
3. Заполнить значениями справочники;
4. Добавить в “choices” параметризованный справочник (ParametrizedChoice) и в “choice.reload.crud” (название параметризованного справочника в ту таблицу, по которой осуществляется поиск значений);
5. Создать в произвольном классе свойства:
- Свойство от которого будут зависеть значения в другом поле;

- Свойство в которое будут подтягиваться значения в зависимости от выбранного значения;
  Указать в поле “Системное имя справочника” - название параметризированного справочника;
  Указать в поле “Пользовательская конфигурация поля (JSON):”

```
{"parametrized": {
  "parameters": {
    "название свойства в справочнике": "название свойства в карточке"
  },
  "required": [список обязательных параметров для формирования
параметризованнового справочника]}
}
```

6. В конфигурацию карточки объекта, в поле добавить параметр “templateName”: “dynamicSelect” в которое будут подтягиваться значения в зависимости от выбранного значения в другом поле.

   Параметр “templateName”: “dynamicMultiselect” - предоставляет выбор нескольких значений, который вычисляется повторно после изменения связанных атрибутов в карточке объекта.

   Важно! Для корректного отображения значений из связанных справочников в табличном отображении папок необходимо в конфигурации отображения папки принудительно задать наименование родительского справочника, который не является связанным.

```
 {
      "fieldName" : "OffInDep",
      "choiceName": "Office"
}
```

   Пример

Рассмотрим создание связанного справочника на примере департаментов и отделов. При выборе департамента в другом поле будет отображаться список отделов, которые состоят в выбранном департаменте.

1. В редакторе классов в классе “Справочники” создать класс “Departments”(Департаменты);
2. В редакторе классов в классе “Справочники” создать класс “Offices”(Офисы);
3. Добавить классы “Offices” и “Departments” на рабочий стол прикладного администратора;
4. В редакторе классов в класс “Offices” добавить свойство “DepartmentID”:

```
- Системное имя: DepartmentID
- Название: Департамент
- Тип данных: INTEGER
- Метод отображения:select
- Системное имя справочника: Departments
```

4. Добавить свойство “DepartmentID” в карточку “Offices”;

5. Создать несколько департаментов;
6. Создать несколько офисов, выбрав значение в поле “DepartmentID”;

7. Добавить в “choices” параметризованный справочник (ParametrizedChoice):

```
OfficesInDepartment : new ParametrizedChoice('OfficesInDepartment',
'criteriaChoiceListProcessor', [
    index     : 'ID',
    label     : '${fields.Title}',
    search    : Search.from('Offices').where('DepartmentID',
ConditionType.EQUAL).orderBy('Title'),
    fields    : ['ID', 'Title', 'DepartmentID'],
    itemFields: ['DepartmentID']
    ])
```

8. Добавить в конфигурацию “choice.reload.crud”

```
Offices:['Offices', 'OfficesInDepartment']
```

9. Создать произвольный класс, добавить свойства:

Свойство Департамент

```
- Системное имя: Department
- Название: Департамент
- Тип данных: INTEGER
- Системное имя справочника: Departments
- Метод отображения:select
```

Свойство Офис в департаменте

```
- Системное имя: OffInDep
- Название: Офис в департаменте
- Тип данных: INTEGER
- Системное имя справочника: OfficesInDepartment
- Метод отображения:select
- Пользовательская конфигурация поля (JSON):
                        {
                         "parametrized": {
                                "parameters": {"DepartmentID": "Department"},
                                "required": ["DepartmentID"]
                                }
                        }
```

где в поле в “Пользовательская конфигурация поля (JSON)” конфигурация заполнятся следующим образом:

```
{"parametrized": {
  "parameters": {
    "название свойства в справочнике": "название свойства в карточке"
  },
  "required": [список обязательных параметров для формирования справочника]}
}
```

10. Добавить в карточку объекта свойства:

```
{"components": [ {"cssClass": "span8", "propName": "Department"} ]},
{"components": [ {"cssClass": "span8", "propName": "OffInDep", "templateName":
"dynamicSelect"} ]}
```

где параметр **“templateName”:”dynamicSelect”** - это справочник, который вычисляется повторно после изменения связанных атрибутов в карточке объекта.

В результате в карточке при выборе департамента, в поле “Офис в департаменте” будут отображаться только те офисы, которые состоят в этом департаменте (см. рис.):

Настройка связанных справочников с предзаполнением полей

В системе есть возможность предзаполнять поля на основании данных других полей.

Краткое описание:

1. Создать классы в классе “Справочники” и установить связь между ними по свойству.
2. Заполнить справочники значениями.
3. Добавить в класс два свойства (указав соответствующие справочники). Связать их между собой, в свойстве, от которого будет предзаполняться поле, заполнить поле “Пользовательская конфигурация поля (JSON)”:

```
{
  "linked": {
    "Название свойства в карточке объекта": "Название свойства в справочнике",
    ...
  }
}
```

4. Добавить в конфигурацию карточки объекта, параметр “params”: {“linkedChoice”: true} в components поля от которого будет предзаполняться другое поле.

Пример:
Рассмотрим создание связанных справочников на примере выбора департамента и входящих в него отделов.

Структура:

Департамент управления экономики и развития

- Отдел экономического анализа и прогнозирования
- Отдел бюджетного планирования
- Отдел труда и заработной платы

Департамент финансового управления

- Отдел финансовых расчётов и платежей
- Отдел сводной отчётности
- Отдел бухгалтерского учёта

Для того чтобы настроить связанный справочник необходимо выполнить следующие действия:

1. В редакторе классов создать классы “Departments” и “Offices” в классе “Справочники”.

2. В классе “Departments” создать свойство “OfficeInDep” с параметрами:

```
- Тип данных: INTEGER
- Метод отображения: multiselect
- Системное имя справочника: Offices
- Является множественным: +
```

3. Создать в классе свойство “Department” с параметрами:

```
- Системное имя: Department
- Название: Департамент
- Тип данных: INTEGER
- Системное имя справочника: Departments
- Метод отображения: select
- Пользовательская конфигурация поля (JSON):
{
  "linked": {
    "OfficesInDepartment": "OfficeInDep"
  }
}
```

где поле “Пользовательская конфигурация поля (JSON)” заполняется следующим образом:

```
{
  "linked": {
    "Название свойства в карточке объекта": "Название свойства в справочнике",
    ...
  }
}
```

Создать свойство “OfficesInDepartment” в котором будут автоматически отображаться значения в зависимости от выбора значения из поля “Department”:

```
- Системное имя: OfficesInDepartment
- Название: Офисы департамента
- Тип данных: INTEGER
- Системное имя справочника: Offices
- Метод отображения: multiselect
- Является множественным: установить чек-бокс
```

5. Объявить справочники в “choices” и “choice.reload.crud”;
6. Добавить справочники на рабочий стол прикладного администратора;
7. Заполнить справочники данными, добавив в карточку “Departments” поле “OfficeInDep”;
8. Добавить в карточку объекта поля “Department”,”OfficesInDepartment” и параметр **“linkedChoice”: true”** для “Department”:

```
{"components": [ {"cssClass": "span8", "propName": "Department","params":
{"linkedChoice": true}}]},
{"components": [ {"cssClass": "span8", "propName": "OfficesInDepartment"} ]},
```

В результате в поле “OfficesInDepartment” будут автоматически подставляться значения, в зависимости от выбора значения в поле “Department”.