Инструкция по «Кейс-менеджменту»

Пошаговые действия пользователя.

СОДЕРЖАНИЕ

1. Создание и настройка кейса 3
2. Роли 5
3. Активности 7
3.1. Действие 8
3.2. Этап 12
3.3. Задача 15
3.4. Процесс 22
4. События 23
5. Условия событий 34
6. Чек-листы 38
7. Шаблон кейса 49
8. Продвинутые темы 51
8.1. Расширение списка доступных процессов кейса на базе Activiti 51

1. Создание и настройка кейса

С точки зрения системы кейс – это объект с аспектом **icase:case** или «Является кейсом».

Для наглядной демонстрации зайдем в репозиторий и создадим обычную папку в каталоге гостя с названием «CaseTest».

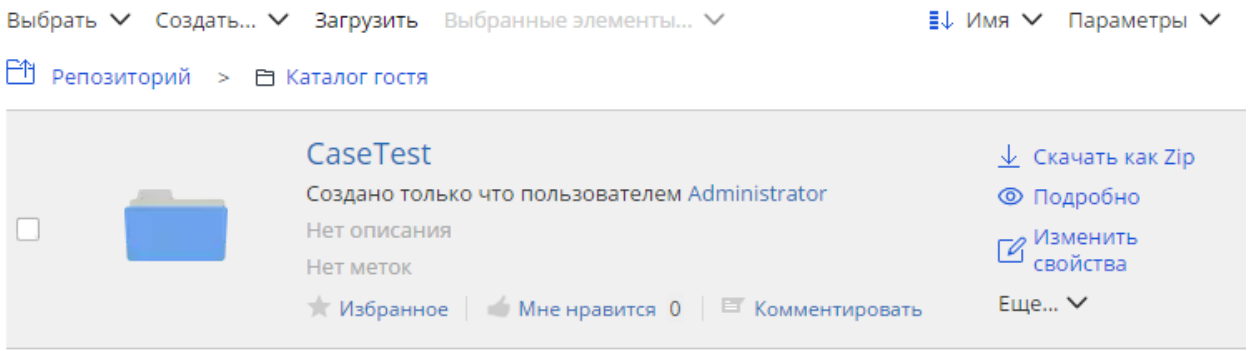


Рисунок 1 – Просто папка

Справа откроем дополнительный список действий (Еще…) и выберем «Настроить аспекты» как показано на рисунке 2.

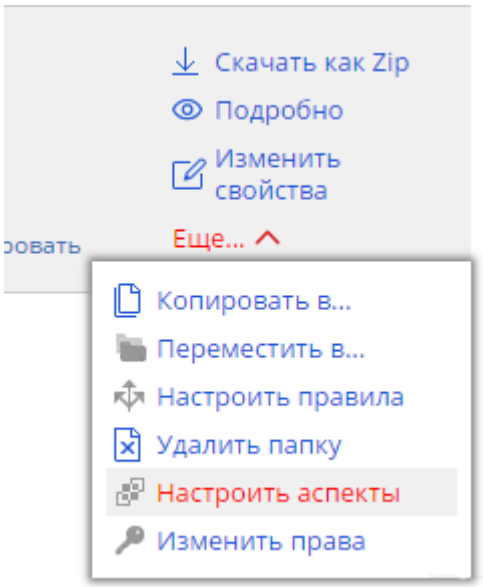


Рисунок 2 – Выбор настройки аспектов

В появившемся окне нам необходимо выбрать аспект «Является кейсом».

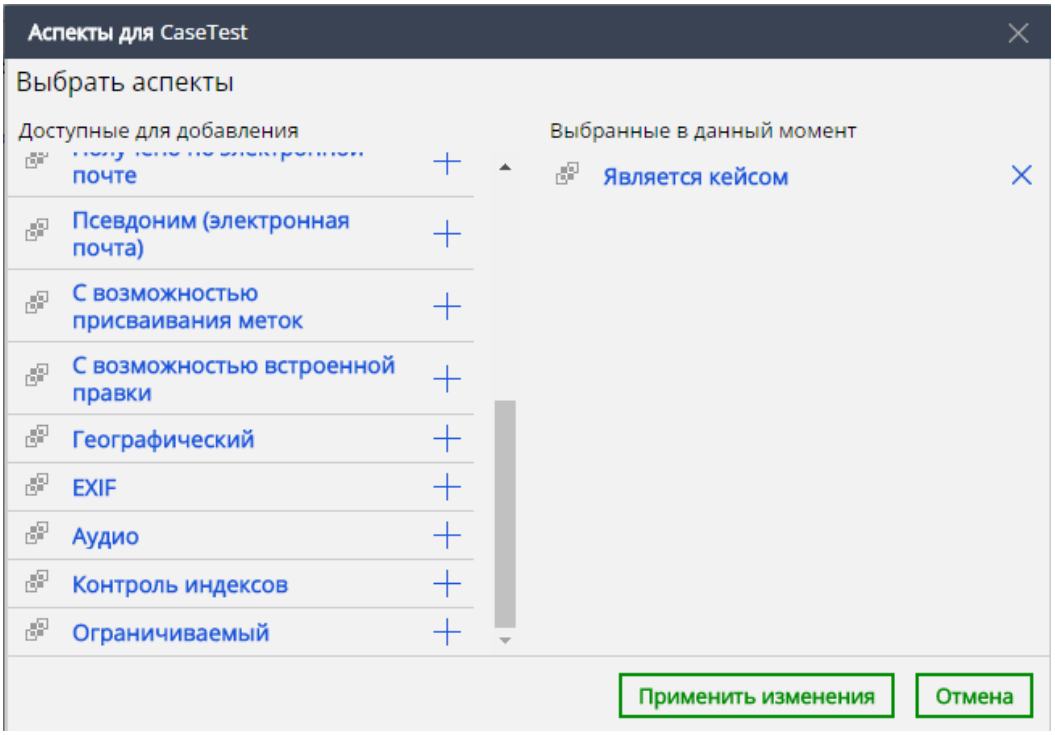


Рисунок 3 – Настройка аспектов

Нажимаем «Применить изменения» и переходим на карточку созданной папки. Это можно сделать, нажав в списке действий «Подробно».

Переходим на вкладку «Процесс кейса»:

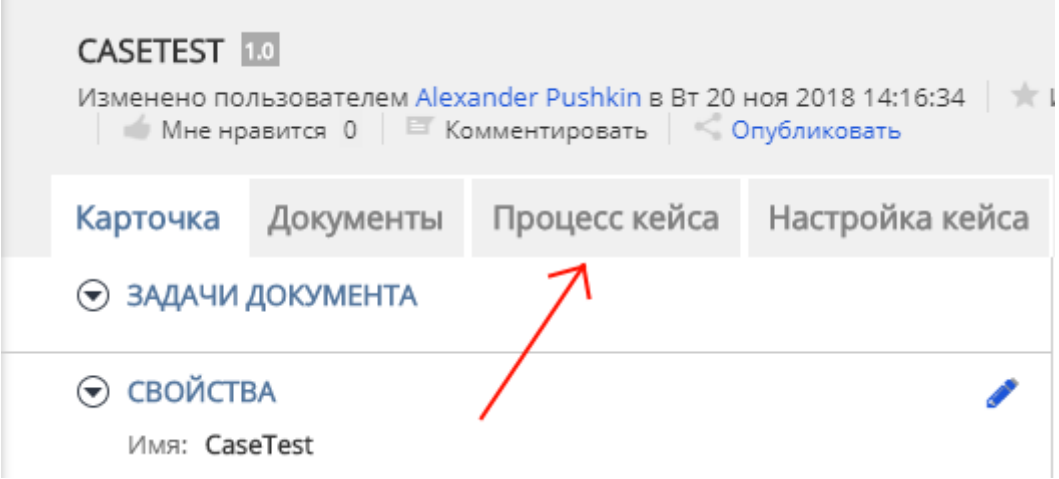


Рисунок 4 – Карточка кейса

И видим конструктор процесса кейса и роли кейса:

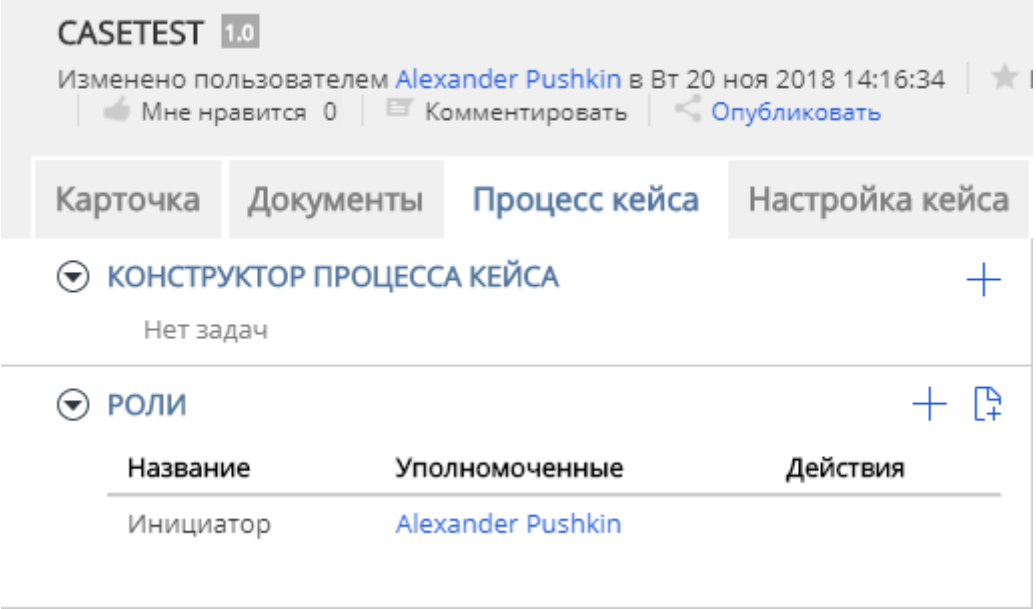


Рисунок 5 – Вкладка «Процесс кейса»

2. Роли

Роли предназначены для выбора людей или групп, относящихся к кейсу. В активности кейса «Задача» всегда участвует одна или несколько ролей. Создадим новую роль нажав кнопку, отмеченную на рисунке 6.

Рисунок 6 – Кнопка для создания новой роли

На появившейся форме мы видим три поля:

- Название для вычислений – идентификатор для чтения или редактирования роли системой. Рекомендуется задавать латинскими символами без пробелов;
- Заголовок – Имя роли. Может быть произвольным;
- Представители – Люди или группы, представляющие данную роль. Выбор представителей осуществляется из оргструктуры.

После заполнения полей форма принимает вид, представленный на рисунке 7.

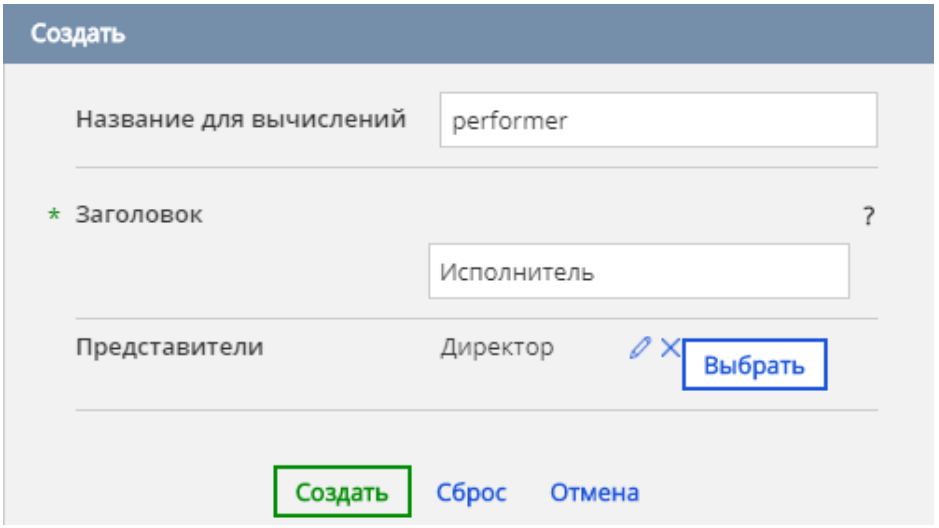


Рисунок 7 – Заполненная форма создания новой роли

Нажимаем «Создать» и видим, что в кардлете «Роли» появилась новая роль (рисунок 8). Таким образом можно создавать неограниченное количество ролей, которым можно назначать задачи.

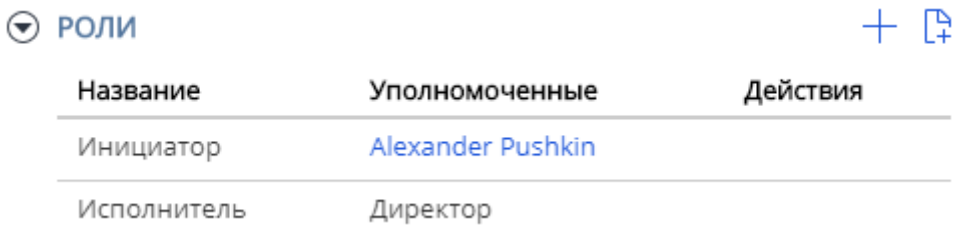


Рисунок 8 – Добавленная роль

Представителей роли можно менять в любое время, но надо учитывать, что все уполномоченные лица из роли считываются в момент старта задачи кейса. Если поменять уполномоченных лиц после старта задачи, то это никак на уже запущенную задачу не повлияет.

3. Активности

Если нажать на синий **+** в кардлете «Конструктор процесса кейса», то можно выбрать одну из пяти видов активностей (см. рисунок 9).

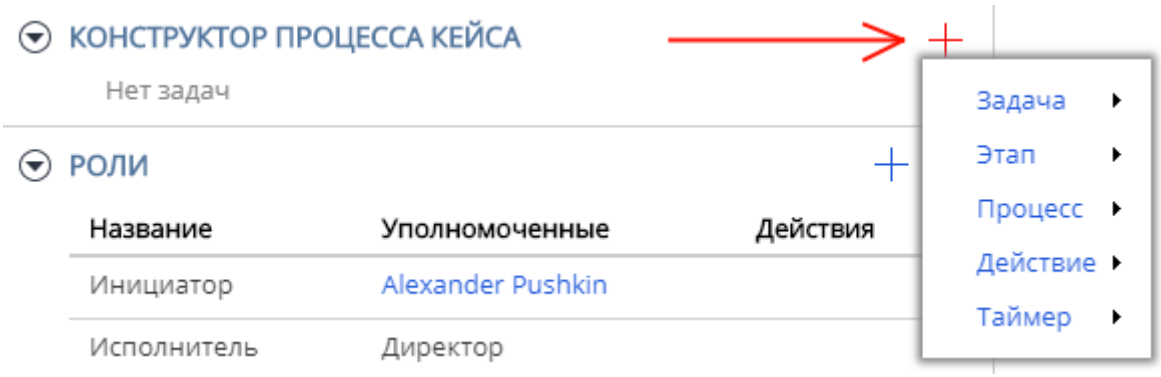


Рисунок 9 – Создание новой активности

У активностей есть ряд общих параметров:

- Повторяемый

Если стоит данный флаг, то активность можно запустить даже если она уже находится в состоянии «выполняется» или «завершена». Если активность при повторном запуске была в состоянии «выполняется», то происходит сброс в состояние «не начато» и последующий запуск. Если при сбросе с активностью связан активный бизнес-процесс, то он отменяется.

- Автоматическое управление событиями

Если стоит данный флаг, то активность при создании и при перемещении будет автоматически связываться с предыдущей активностью или кейсом. Возможные случаи:

Если активность первая в кейсе, то её запуск связывается с событием «кейс создан»;

Если активность первая в этапе, то её запуск связывается с событием «родительский этап начался»;

Если активность следует за какой-то другой активностью, то её запуск связывается с событием «предыдущая активность завершилась»;

Если флаг стоит для этапа, то его завершение связывается с событием «все активные дочерние активности завершились».

Подробнее про события можно прочитать в разделе 4.

- Начинается вручную

Если стоит данный флаг, то активность можно запустить вручную.

3.1. Действие

Действие – это автоматическая активность, которая начинается и заканчивается в ходе одной транзакции. Используются в случаях, когда участие человека не требуется.

Реакция на события: запуск действия (событие для начала)

Виды действий:

- **Отправить e-mail**

Отправить e-mail с заданными темой и текстом указанному получателю.

Свойства: Тема, Получатель, Текст

- **Установить свойство документа**

Записать в указанное свойство текущего документа (кейса) заданное значение. Пример свойства – **cm:title**.

Свойства: Свойство, Значение

- **Установить переменную процесса**

Установить переменную процесса.

Свойства: Переменная, Значение

- **Запустить javascript**

Выполнить произвольный javascript код. В контексте есть доступ к следующим объектам: **document** – текущий документ (кейс), **event** – событие, которое привело к запуску действия. **process** – переменные из предыдущей задачи если такая имеется (данный объект существует только если запуск текущего действия вызван завершением задачи).

Свойства: Заголовок, javascript-код

- **Показать сообщение об ошибке**

При старте данной активности бросается исключение. Пользователю показывается ошибка и транзакция откатывается.

Свойства: Сообщение об ошибке

Пример:

Создадим для примера действие «установить свойство документа» (рисунок 10).

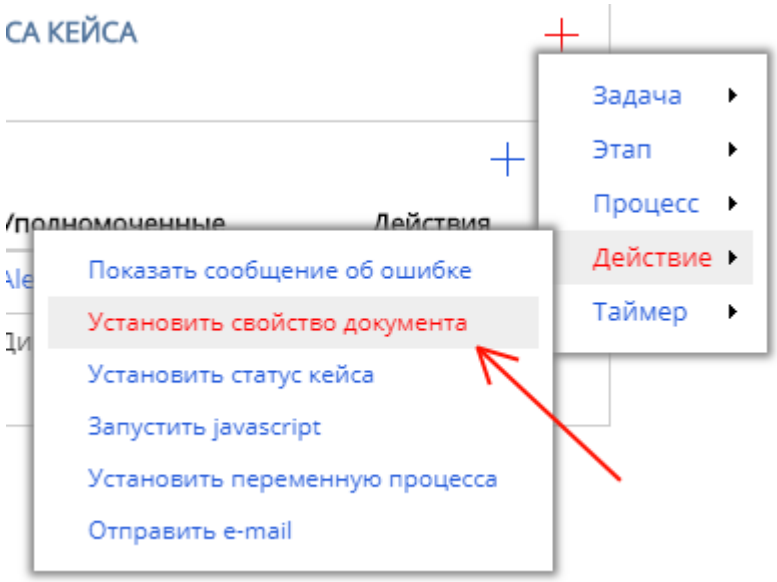


Рисунок 10 – Создание действия «Установить свойство документа»

Заполняем форму как показано на рисунке 11. (Свойство: **cm:title**, Значение: **NewTitle**).

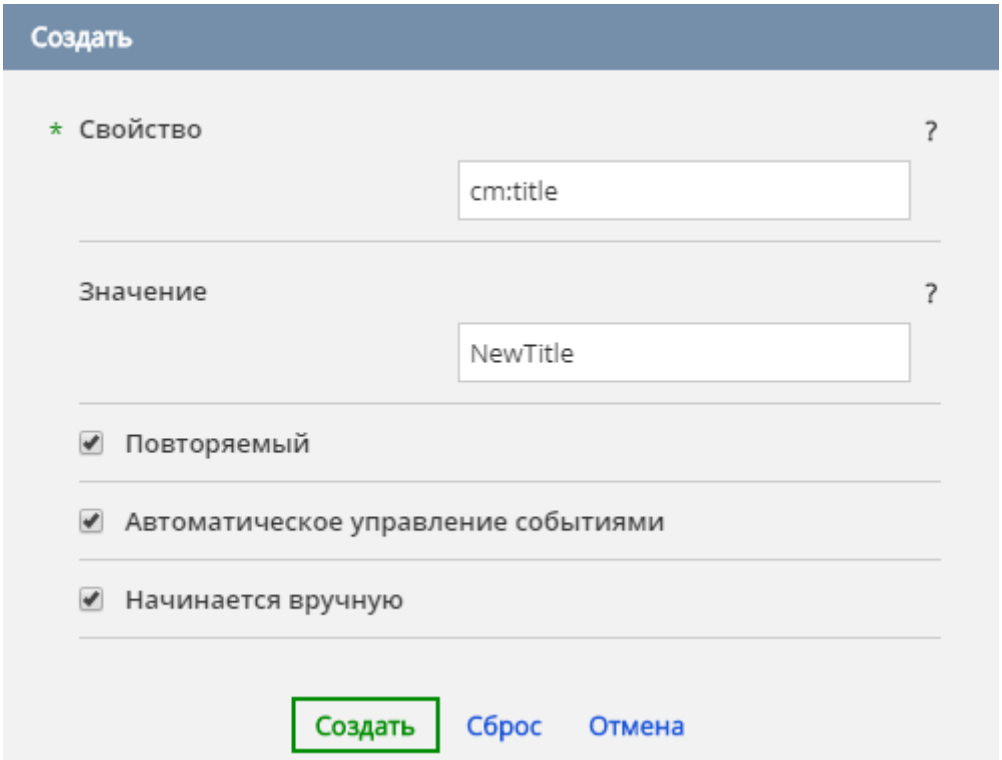


Рисунок 11 – Заполнение формы создания действия

Нажимаем зеленую кнопку «Создать» внизу формы и видим, что наше действие появилось в кардлете «Конструктор процесса кейса» (рисунок 12).

Рисунок 12 – Новое действие

Запускаем действие как показано на рисунке 13.

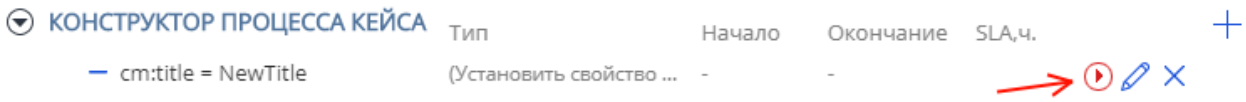


Рисунок 13 – Запуск действия

На вопрос системы «Запустить задачу?» нажимаем кнопку «Да» (рисунок 14).

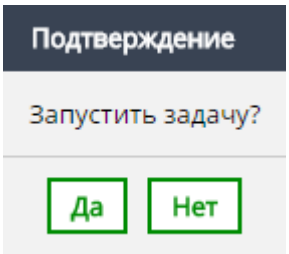


Рисунок 14 – Подтверждение запуска задачи

После подтверждения, мы видим следующие результаты:

Рисунок 15 – Появились даты начала и завершения активности

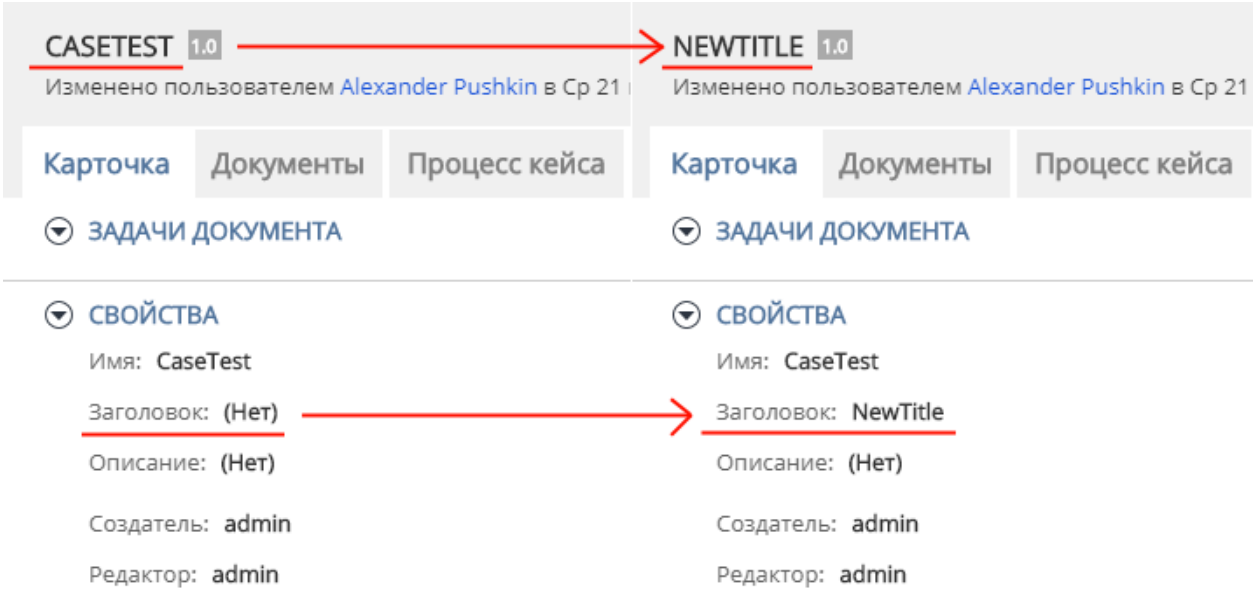


Рисунок 16 – Свойство документа изменилось

Во время разработки часто возникает необходимость сбрасывать состояние процесса. Для этого создаем новое действие «Запустить javascript» и на форме создания в поле javascript-код вводим **caseActivityService.reset(document)**; (рисунок 17). Данное действие сбрасывает состояние всех активностей в документе на исходное.

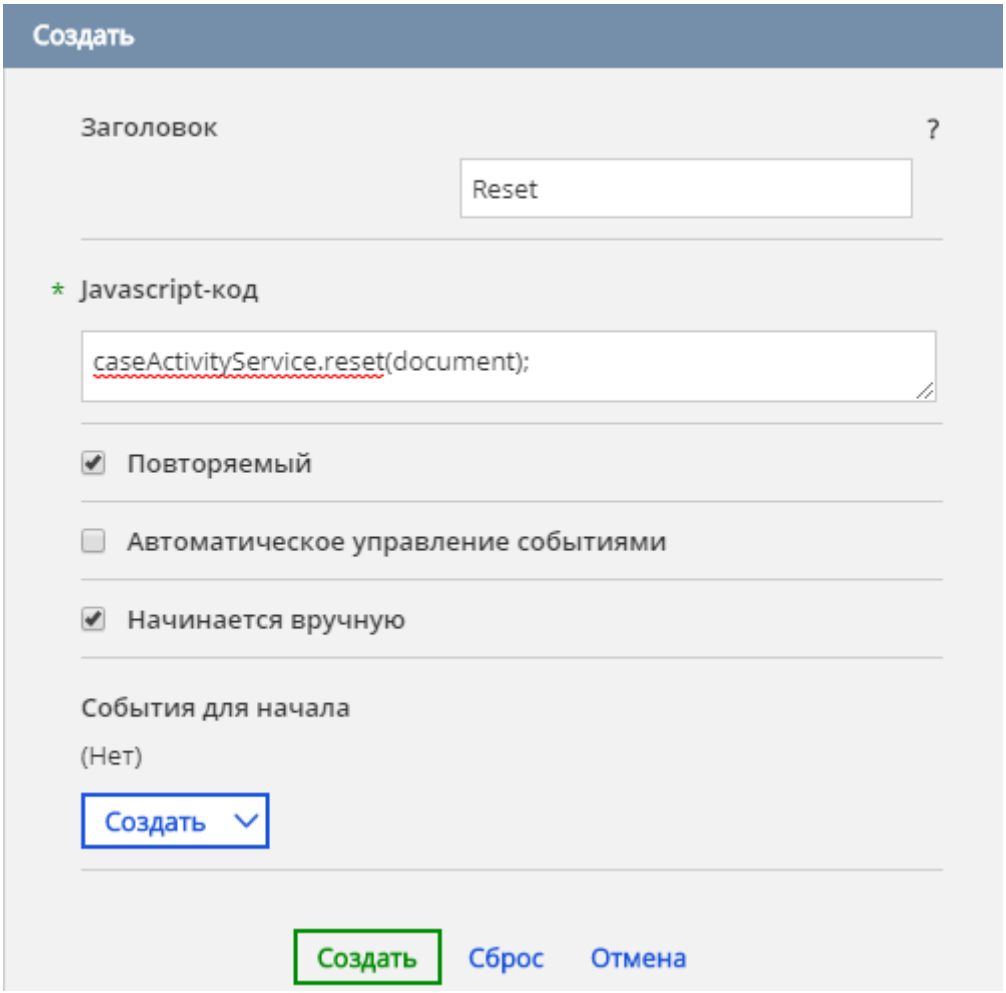


Рисунок 17 – Создание действия для сброса активностей

Запускаем javascript действие, перезагружаем страницу и видим, что у нашей первой активности исчезли даты начала и окончания (рисунок 18).

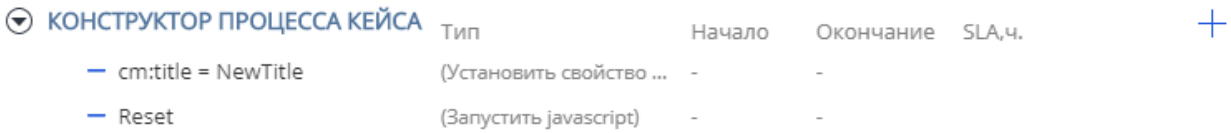


Рисунок 18 – Состояние активностей сбросилось

Отредактируем действие «установить свойство документа». Для этого нажмем кнопку, отмеченную на рисунке 19.

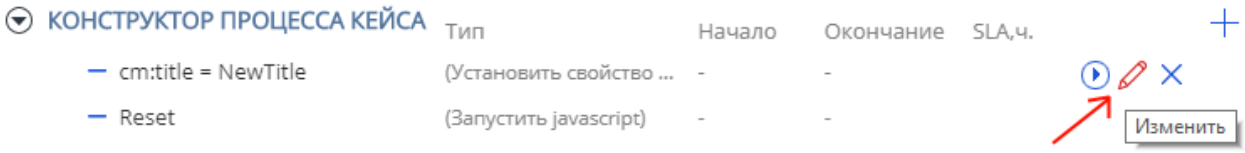


Рисунок 19 – Кнопка «Изменить»

В поле «Значение» выставляем новое произвольное значение, нажимаем кнопку «Сохранить» и запускаем действие. В результате видим, что поле «Заголовок» изменилось на новое значение.

С помощью продемонстрированного выше действия «Reset» очень удобно производить отладку существующих и создавать новые процессы на кейс-менеджменте.

3.2. Этап

Этап – это активность, главной отличительной особенностью которой является возможность содержать в себе другие активности (в том числе и другие этапы). В этапе можно указать параметр «Статус документа», который установится у текущего документа (кейса) при старте этапа.

Свойства: Заголовок, Описание, Статус документа, Планируемая дата начала, Планируемая дата завершения.

Реакция на события: запуск этапа (событие для начала), перезапуск этапа (событие для перезапуска), остановка этапа (событие для завершения).

Чек-листы: Если в документе (кейсе) есть чек-листы, то можно выбрать необходимые требования для начала или завершения. Если указанные требования не выполнены, то этап не может перейти в другое состояние.

Пример:

Из меню создания новой активности выберем этап (рисунок 20).

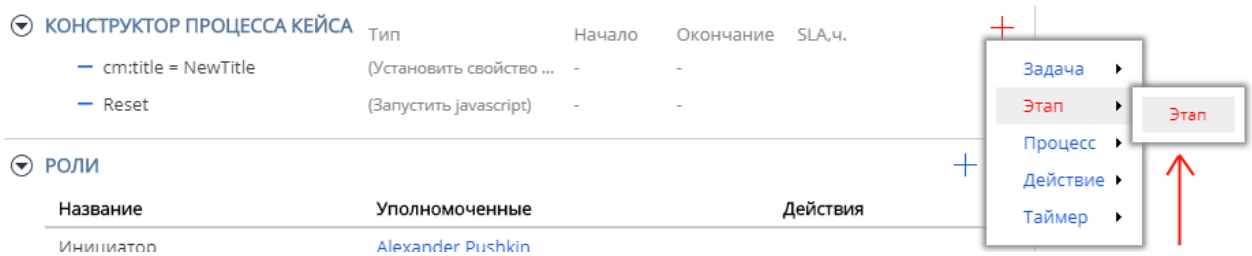


Рисунок 20 – Создание этапа

На появившейся форме произвольно заполняем поле заголовок и нажимаем зеленую кнопку «Создать» внизу формы (рисунок 21).

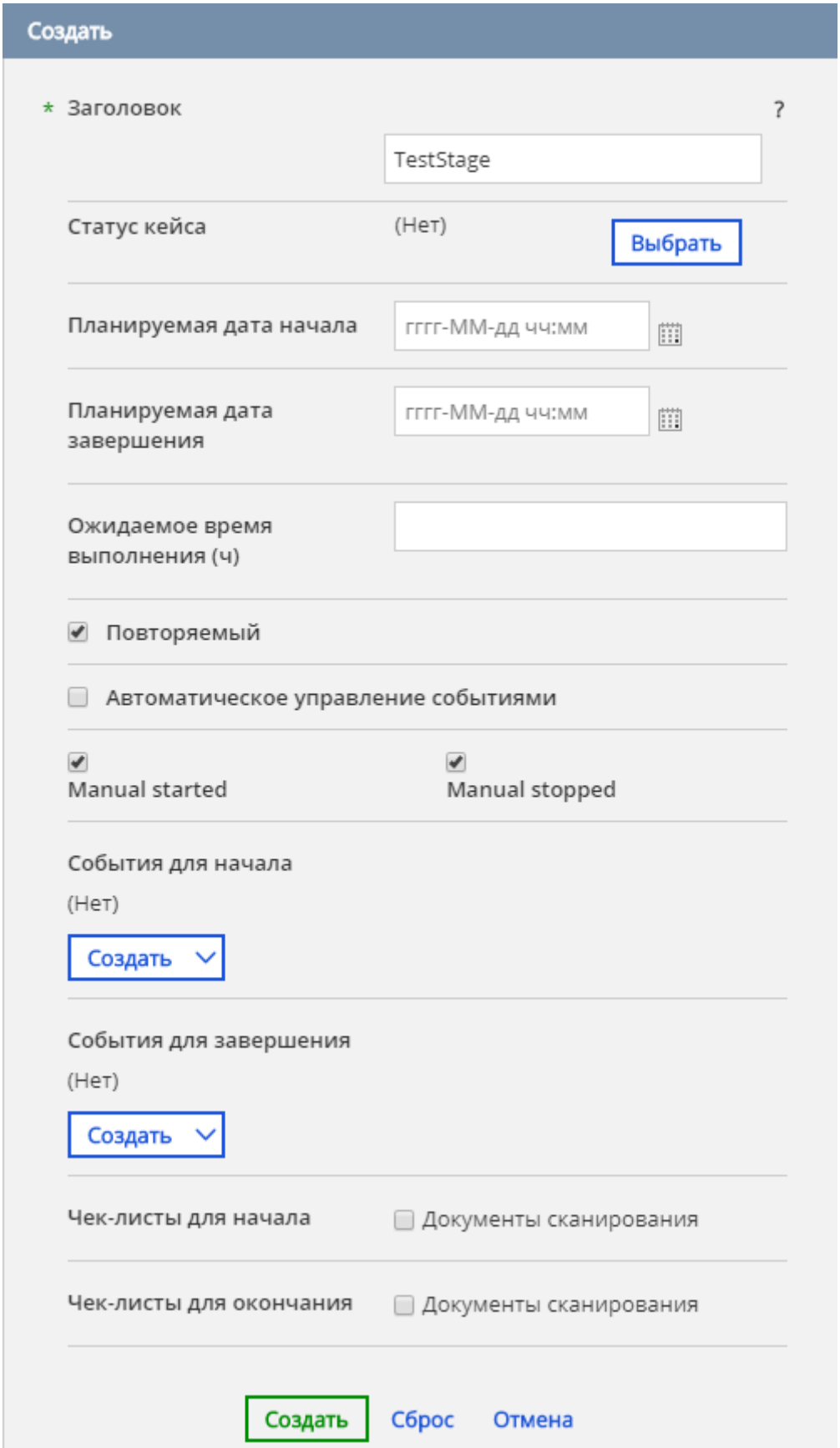


Рисунок 21 – Создание нового этапа

После создания этапа он появляется в кардлете «Конструктор процесса кейса» (рисунок 22).

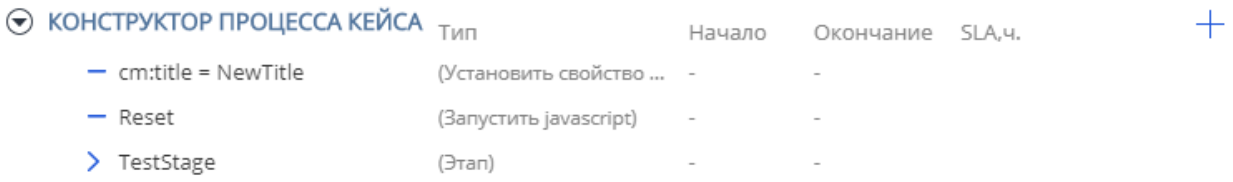


Рисунок 22 – Новый этап «TestStage»

Синяя стрелка вправо означает, что сейчас этап «свернут» и его содержимое не показывается. Если мы кликнем левой кнопкой мыши по этапу, то стрелка укажет вниз и этап «раскроется» (рисунок 23).

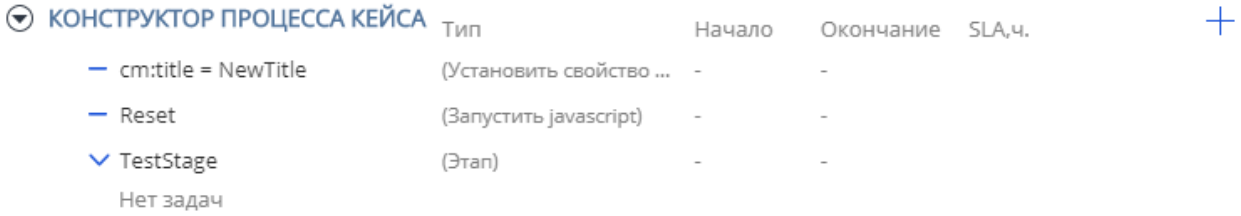


Рисунок 23 – «Раскрытый» этап

Как видим, сейчас этап не содержит задач. Попробуем их создать. Для этого следует найти кнопку «+» напротив имени этапа (рисунок 24). После нажатия на неё мы увидим список аналогичный уже знакомой нам кнопке «+» в правом верхнем углу кардлета «Конструктор процесса кейса». Отличие только в том, что новые активности будут создаваться внутри этапа.

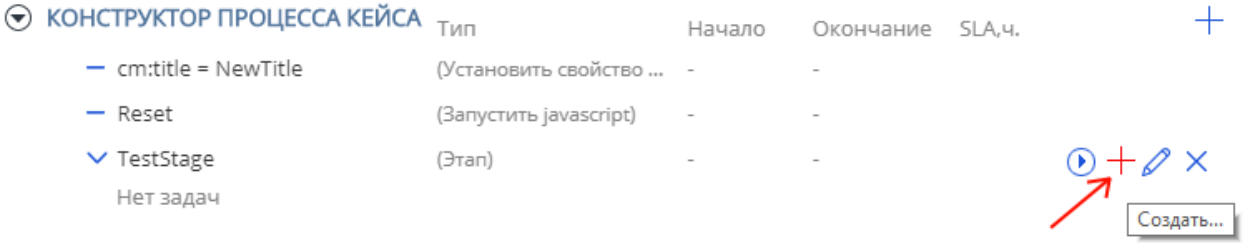


Рисунок 24 – Кнопка «Создать активность в этапе»

Создаем новое действие по уже знакомому нам примеру из раздела 3.1 «Установить свойство документа» и получаем следующий результат:

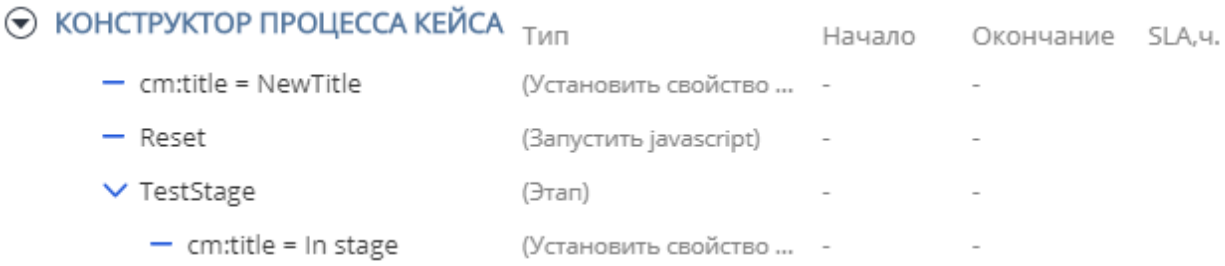


Рисунок 25 – Действие в этапе

Если при создании действия был установлен флаг «Автоматическое управление событиями», то действие запустится при запуске этапа. Ручное управление событиями мы рассмотри позднее.

3.3. Задача

Задача – это активность с участием человека. В основе лежат простые BPMN процессы, которые начинаются при старте задачи и по их завершении задача автоматически завершается. Виды задач различаются BPMN процессом, который стоит за ними и в зависимости от целей выбирается нужный. Если требуется простая задача с полем для комментария и двумя кнопками «Согласовать» и «Отказать», то можно использовать «Согласование».

Реакция на события: запуск задачи (событие для начала.

Примечания: Если незавершенная задача перезапускается, то процесс, который был запущен отменяется и запускается новый.

Чек-листы: Если в документе (кейсе) есть чек-листы, то можно выбрать необходимые требования для начала или завершения. Если указанные требования не выполнены, то задача не может перейти в другое состояние.

Виды задач:

- Согласование

Простое согласование, которое можно использовать для множества случаев, когда от пользователя требуется только комментарий по задаче и результат «Согласовано», «Отказать». Названия кнопок можно менять, но это требует, чтобы в *.properties файлах были нужные записи.

Свойства: Заголовок, Планируемая дата начала, Планируемая дата завершения, Приоритет, Согласующие лица, Без рассмотрения инициатором, Можно согласовать с замечаниями, Прервать при первом отказе, Права согласующих, Префикс локализации вариантов согласования.

Примечания: Возможность менять названия кнопок требует проверки

- Поручение

Свойства: Заголовок, Планируемая дата начала, Планируемая дата завершения, Приоритет, Исполнители, Права исполнителя, Проверять результаты, Проверяющий, Права проверяющего.

- Пользовательская задача

Очень гибкая задача на основе которой можно строить сложные процессы.

Свойства:

- *Роли исполнителей* - роли, на которые будет назначена задача. Перед стартом процесса из всех выбранных ролей выбираются представители и на них назначаются параллельные задачи.
- *Результаты с обязательным комментарием* - перечень outcome'ов задачи (через запятую), при переходе по которым из задачи потребуется

заполнить поле "Комментарий" (иначе пользователь получит предупреждение).

- *Синхронизация Роли->Бизнес-процесс* - параметр, определяющий синхронизацию бизнес-процесса по изменению в роли (если в роли изменяются назначенные лица, то в задаче/задачах они меняются аналогично).
- *Синхронизация Бизнес-процесс->Роли* - параметр, определяющий синхронизацию роли по изменению в бизнес-процессе (например, если задача была переназначена на человека, не участвующего в роли, то он включается в роль).
- *Варианты результатов* - параметр, определяющий список доступных кнопок. Параметр обязательный если тип задачи: **wfcp:performTask**.

Формат:

первый вариант#alf#**второй вариант**#alf#**третий вариант**

Результат:

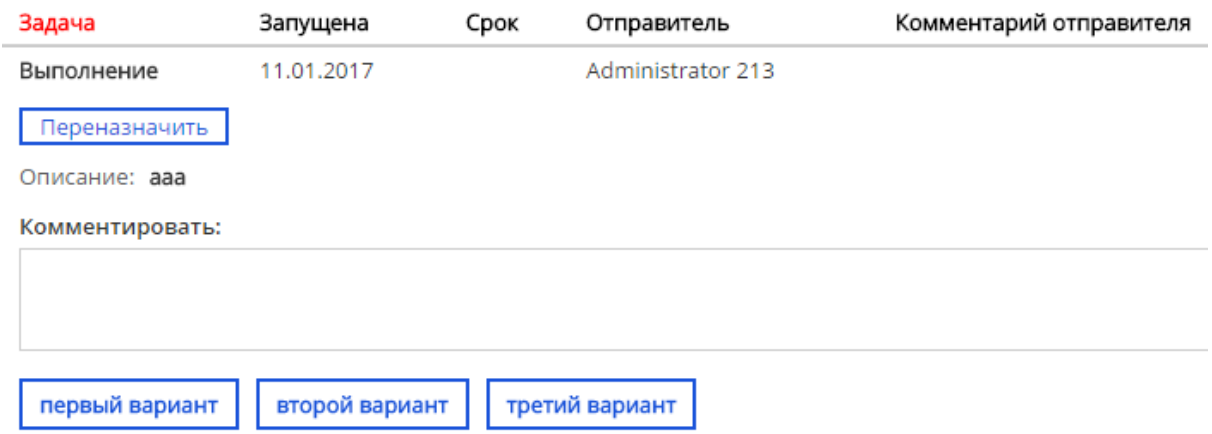


Рисунок 26 – Форма пользовательской задачи

Если нужно некоторое внутреннее значение + локализацию (текст на кнопке), то это реализуется следующим образом:

first|первый вариант#alf#second|второй вариант

Данный параметр можно удобно использовать для ограничения количества кнопок, которые уже заданы через constraints. То есть если у нас есть задача “Согласование” у которой варианты завершения “Confirm, Rework, Reject” то записав в этом поле **“Confirm#alf#Reject”** мы тем самым уберем кнопку “Rework” (если есть локализация для constraints, то она никуда не пропадет при такой записи).

- *Тип задачи* - параметр, определяющий тип задачи. В выпадающем списке доступен тип wfcp:performTask и все его наследники. Для произвольного типа задачи мы можем описать любое количество

свойств/ассоциаций, сделать нестандартную форму и описать варианты завершения задачи.

Пример описания нового типа задачи с тремя вариантами завершения:

```
<type name="uniwf:confirmTask">
  <parent>wfcp:performTask</parent>
  <overrides>
      <property name="wfcp:performOutcome">
          <constraints>
              <constraint name="uniwf:confirmOutcomeConstraint" type="LIST">
                  <parameter name="allowedValues">
                      <list>
      <!--Здесь мы просто указали список возможных вариантов завершения-->
                          <value>Confirm</value>
                          <value>Rework</value>
                          <value>Reject</value>
                      </list>
                  </parameter>
              </constraint>
          </constraints>
      </property>
  </overrides>
</type>
```

И описание формы для этого типа задачи:

```
<config evaluator="task-type" condition="uniwf:confirmTask">
  <forms>
      <form id="inline">
          <edit-form template="/ru/citeck/components/form/workflow/case-
perform/perform-task-inline.ftl" />
      </form>
  </forms>
</config>
```

perform-task-inline.ftl - это стандартный шаблон с описанием (название задачи в кейс-менеджменте), полем для комментария и кнопками выбора результата. В этом шаблоне описана логика работы с полем “Варианты результатов”

- *Прервать процесс при результатах* - параметр, определяющий список результатов через запятую, при получении которых следует прервать параллельный процесс. Например, если у нас есть задача с вариантами результатов “Confirm, Rework, Reject”, то для того, чтобы процесс прервался если кто-то из параллельных исполнителей нажал “Reject” или “Rework” потребуется записать следующее:

Rework,Reject

- *Результаты с обязательным комментарием* - параметр, определяющий список результатов через запятую, при выборе которых исполнитель обязан ввести комментарий. Если комментарий будет отсутствовать, то появится соответствующая ошибка. Формат поля такой же, как и у “Прервать процесс при результатах”.

Пример:

Добавим в наш кейс одну задачу «Согласование» внутри этапа TestStage (рисунок 27).

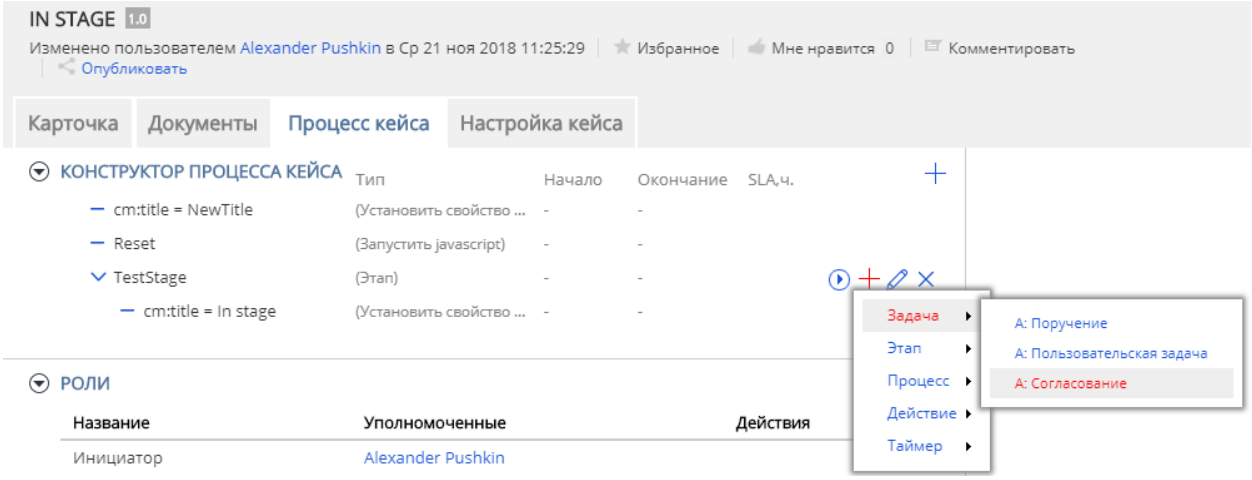


Рисунок 27 – Создание задачи «Согласование»

Заполняем форму как показано на рисунке 28.

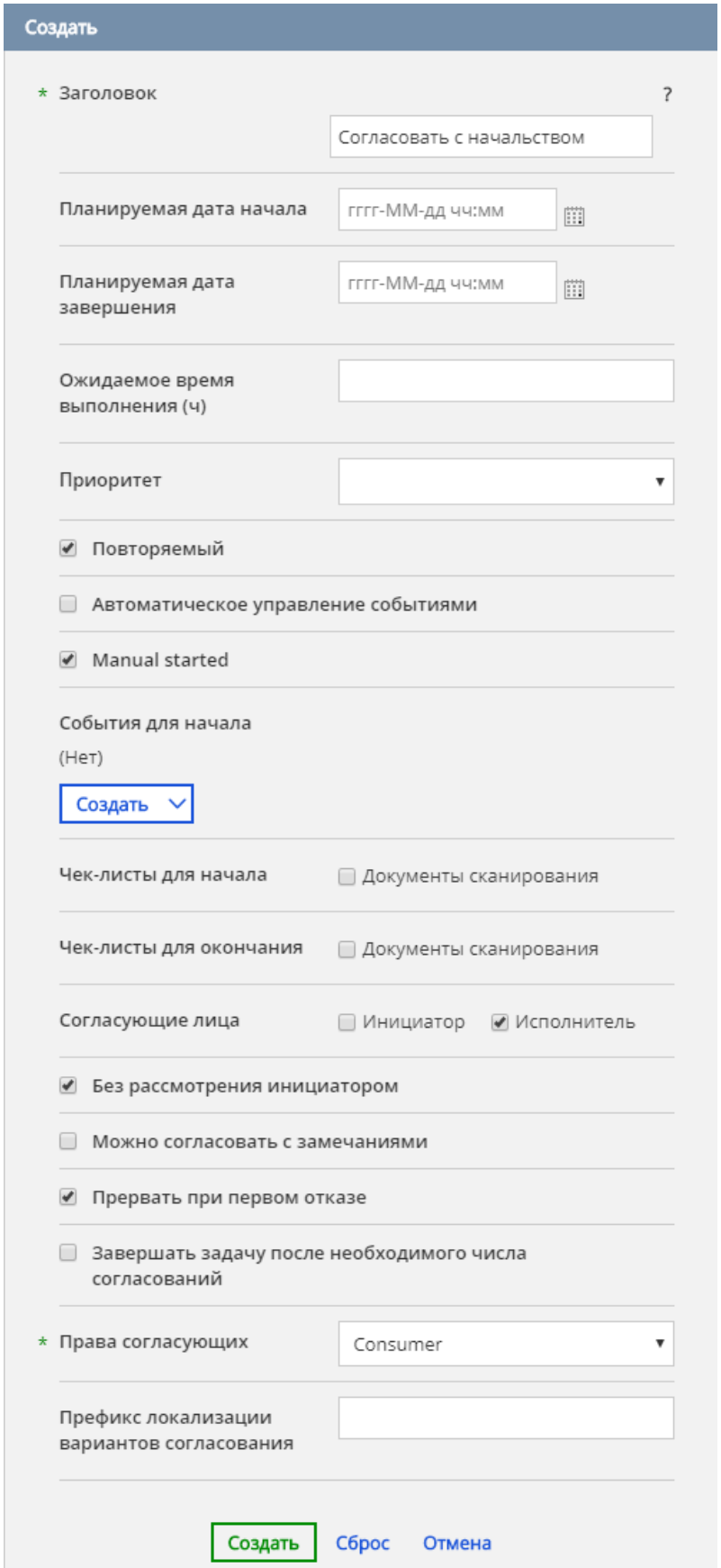


Рисунок 28 – Форма создания задачи «Согласование»

Примечание: Если у вас среди согласующих лиц никого нельзя выбрать, значит в вашем кейсе отсутствуют роли. Для создания задачи требуется как минимум одна роль.

Примечание: Убедитесь, что в выбранной вами роли вы состоите в числе согласующих лиц (или состоите в группе, которая указана в числе согласующих). Это позволит выполнять создаваемые задачи прямо на карточке документа (кейса).

Нажимаем зеленую кнопку «Создать» внизу формы и видим, что наша задача появилась в кардлете «Конструктор процесса кейса» (рисунок 29).

Рисунок 29 – Новая задача

Запускаем задачу так же, как мы это делали с действиями, нажав на кнопку «Начать» напротив задачи, и подтверждаем запуск. Переключаемся на вкладку «Карточка» и если вы оказались среди согласующих (или находитесь в группе, которая указана в роли), то на карточке кейса появится задача, которую можно завершить (рисунок 30).

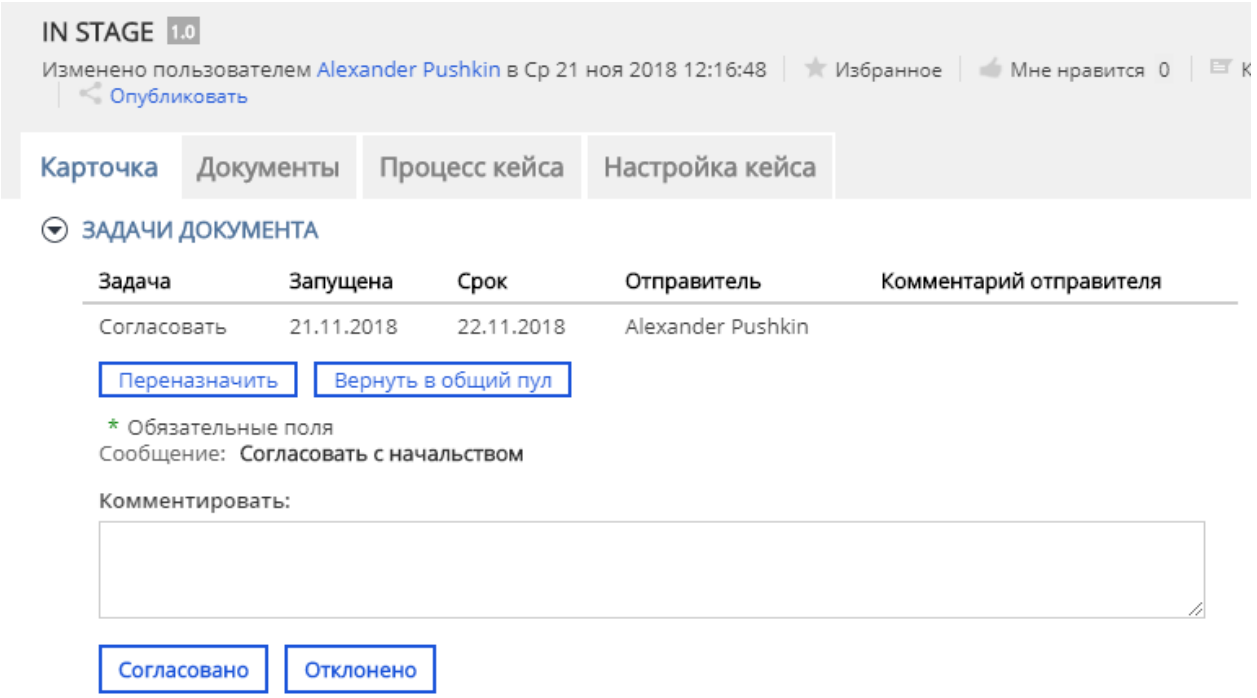


Рисунок 30 – Задача «Согласовать с начальством»

В списках задач на dashboard или на странице «Мои задачи» так же появляется информация о запущенной задаче (рисунки 31 и 32).

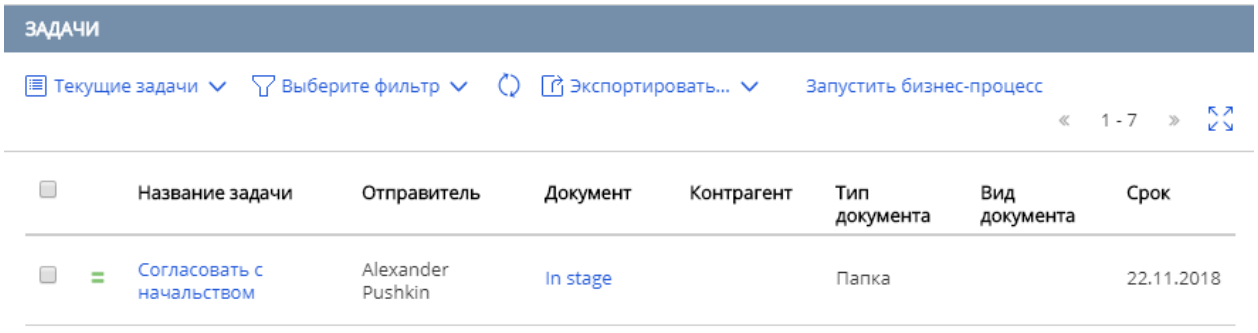


Рисунок 31 – Дашлет «Задачи» на главной странице

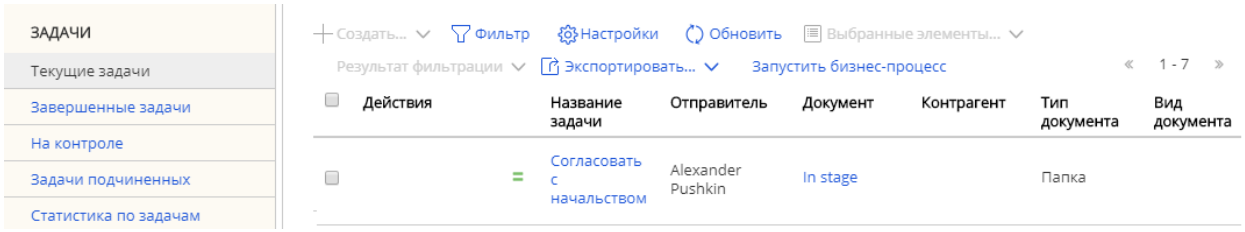


Рисунок 32 – Журнал «Текущие задачи»

После старта задачи можно обратить внимание на то, что у неё установилась дата начала, что свидетельствует о её запуске (рисунок 33). Дата окончания не стоит так как задача еще не завершена.

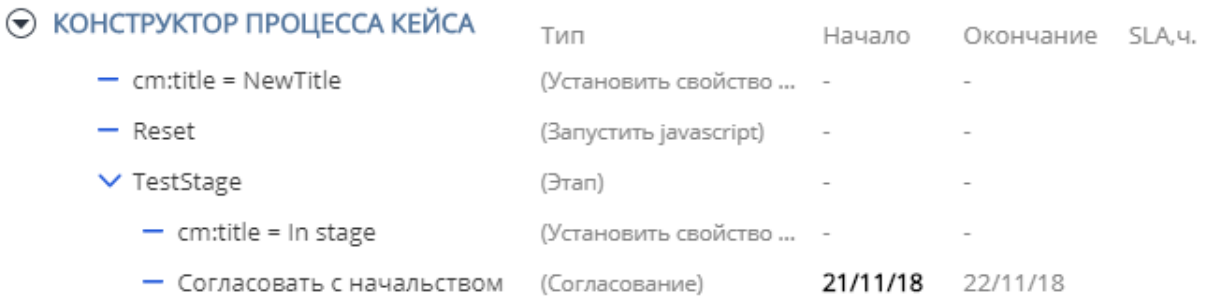


Рисунок 33 – Состояние кардлета «Конструктор процесса кейса» после запуска задачи «Согласовать с начальством»

Попробуем завершить процесс нажав при этом кнопку «Согласовано» на форме, представленной на рисунке 30. После данного действия задача «Согласовать с начальством» автоматически завершается (рисунок 34). Её завершение жестко связано с завершением BPMN процесса, который был запущен и поэтому у задач отсутствует возможность завершения по событию или вручную.

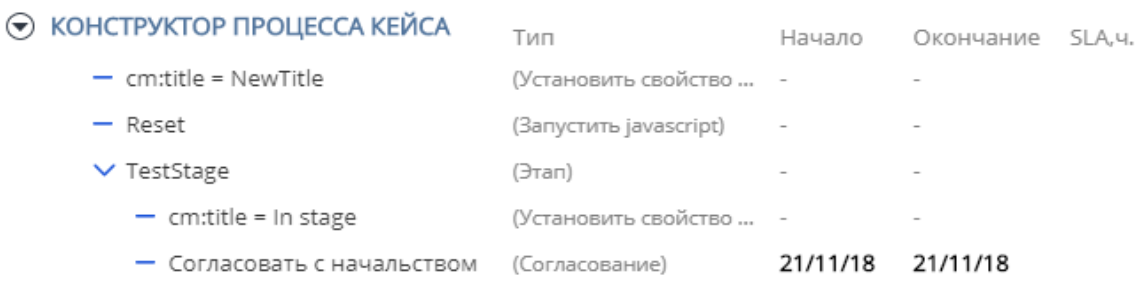


Рисунок 34 – Завершенная задача «Согласовать с начальством»

3.4. Процесс

Процесс – это активность, при старте которой как правило запускается сложный BPMN процесс. Требований к сложности процесса не предъявляется и по механике работы процесс очень похож на задачу.

- Стандартный процесс

Процесс, который позволяет выбрать один из доступных BPMN процессов для одного исполнителя:

  - Процесс печати;
  - Отправить на сканирование;
  - Отправить на подписание;
  - Процесс доработки;
  - Регистрация;
  - Нормоконтроль;
  - Продлить договор;
  - Отправить на подтверждение;
  - Оплатить;
  - Списать в архив;
  - Отправить на утверждение контрагенту.

Свойства: Заголовок, Планируемая дата начала, Планируемая дата завершения, Приоритет, Название процесса, Исполнитель.

4. События

События – гибкий инструмент для связи этапов, задач и действий в единый процесс.

Активности могут запускаться, останавливаться и перезапускаться двумя способами: вручную или по указанному событию. При создании и редактировании активности на форме можно указать, что активность запускается вручную (Manual started) или останавливается вручную (Manual stopped) (рисунок 35). Если мы укажем, что какое-то из этих действий не ручное, то на форме обязательно требуется указать событие, по которому следует его выполнить.

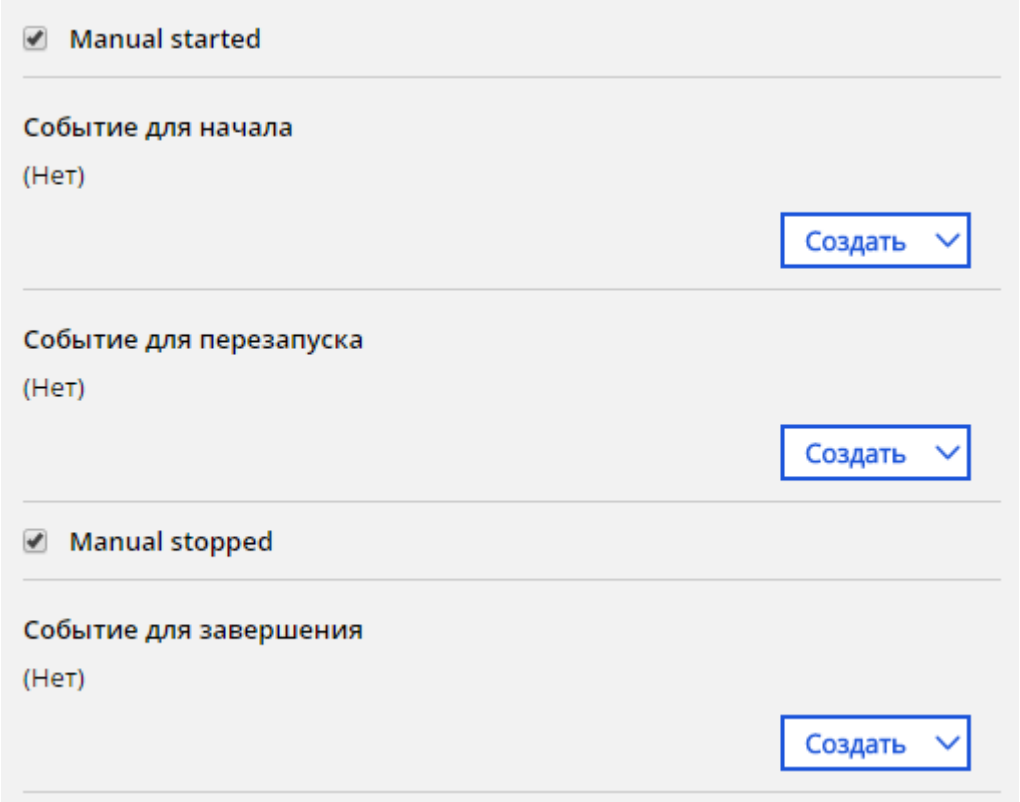


Рисунок 35 – Управление началом, завершением и перезапуском активности

Технически события устроены следующим образом (на примере активностей):

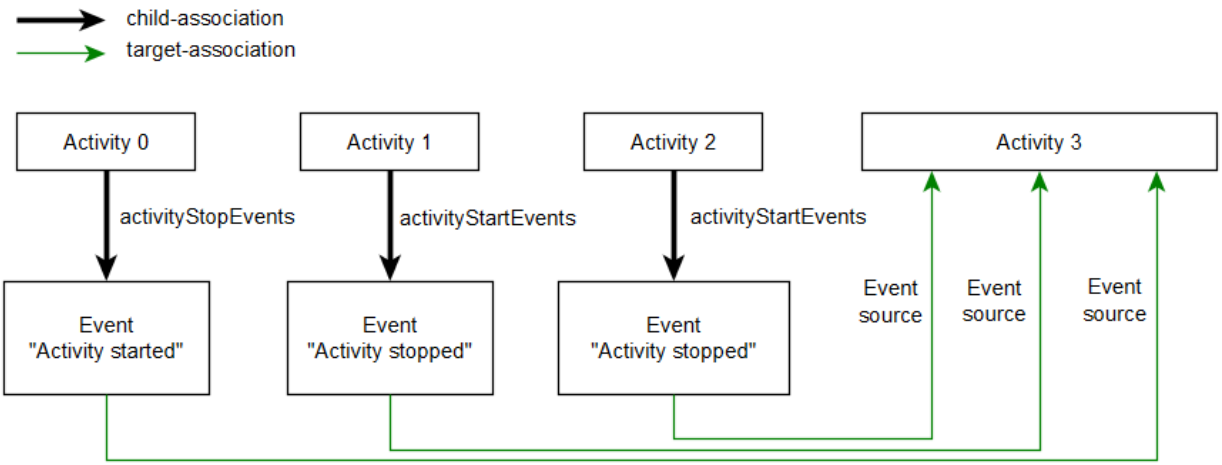


Рисунок 36 – Связь между активностями через события

Активность 0 завершится (activityStopEvents) когда Активность 3 начнется (Event “Activity started”);

Активности 1 и 2 начнутся (activityStartEvent) когда Активность 3 завершится (Event “Activity stopped”).

Жизненный цикл связи активностей через события следующий:

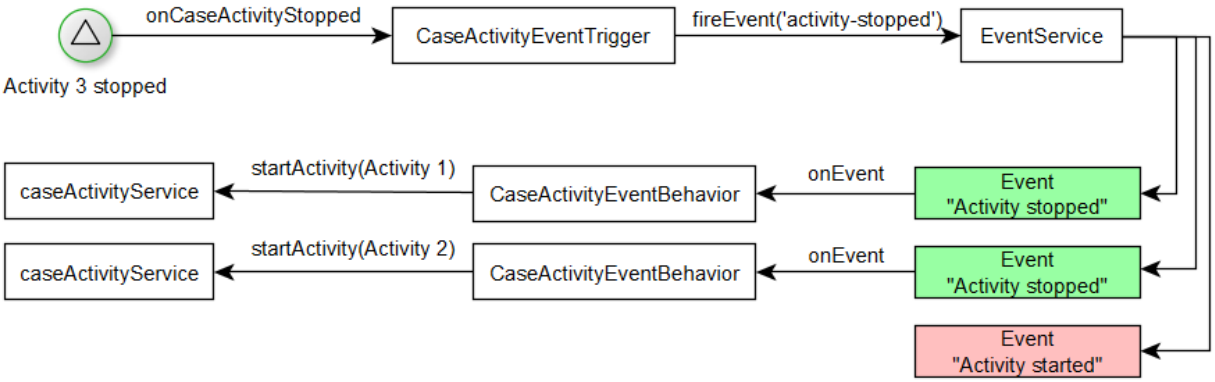


Рисунок 37 – Жизненный цикл связи активностей через события

В данный момент существует два вида реакций на события (в скобках указано название полей на формах редактирования и создания):

- Запустить активность (событие для начала)
- Остановить активность (событие для завершения)

При создании или редактировании активности мы можем указать какие действия приводят к её запуску или завершению.

Виды событий:

- **Завершилась последняя активная подзадача**

Событие, которое возникает, когда у указанного этапа завершается последняя активная подзадача. Это событие удобно использовать для завершения этапа, когда у него не осталось активных подзадач.

Источник события: соседний или текущий этап.

- **Задача началась**

Событие, которое возникает, когда выбранная активность запустилась.

Источник события: соседние активности, текущая активность, родительская активность.

- **Задача закончилась**

Событие, которое возникает, когда выбранная активность завершилась.

Источник события: соседние активности, текущая активность, родительская активность.

- **Действие пользователя**

Событие, которое возникает, когда пользователь нажмет соответствующее действие в списке действий на карточке документа (кейса). Добавление таких событий на любую активность в кейсе приводит к появлению нового действия в колонке «Действия».

Заголовок: Название действия в кардлете «Действия»

Сообщение подтверждения: Сообщение, которое выведется в диалоговом окне для пользователя в момент нажатия действия. Если данное поле не заполнено, то диалога с подтверждением не появится.

Тип дополнительных данных: Тип из модели Alfresco, для которого будет запрошена форма при попытке выполнить действие. Все данные, которые пользователь введет на появившейся форме будут доступны в последующих активностях в пределах одной транзакции через переменную additionalData;

Условия: Условия для отображения действия в колонке «Действия»;

Роли: роли, для которых доступно данное действие.

- **Кейс создан**

Событие, которое возникает, когда создается новый кейс. Это событие можно использовать, например, для запуска процесса после создания кейса или для выполнения некоторых действий для инициализации (пример - выставление начального статуса).

Пример:

Свяжем наши активности в создаваем кейсе через механизм событий. В качестве отправной точки возьмем активности, созданные нами в предыдущих примерах. Они представлены на рисунке 38.

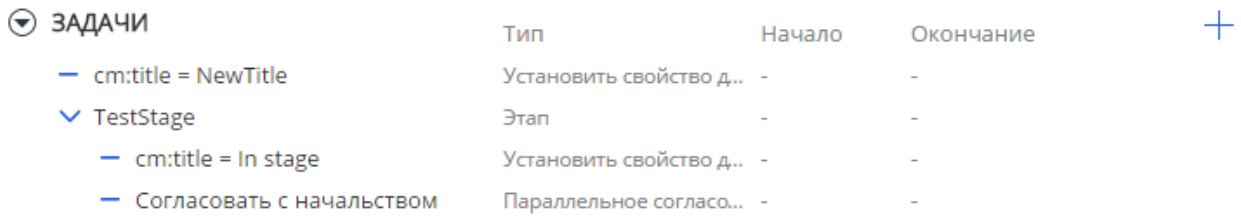


Рисунок 38 – Активности кейса

Отредактируем этап «TestStage» нажав кнопку, отмеченную на рисунке 39.

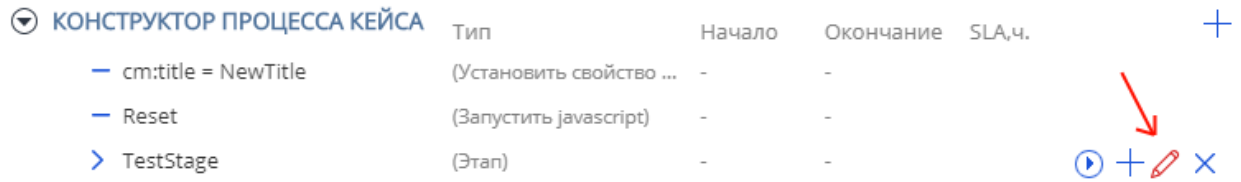
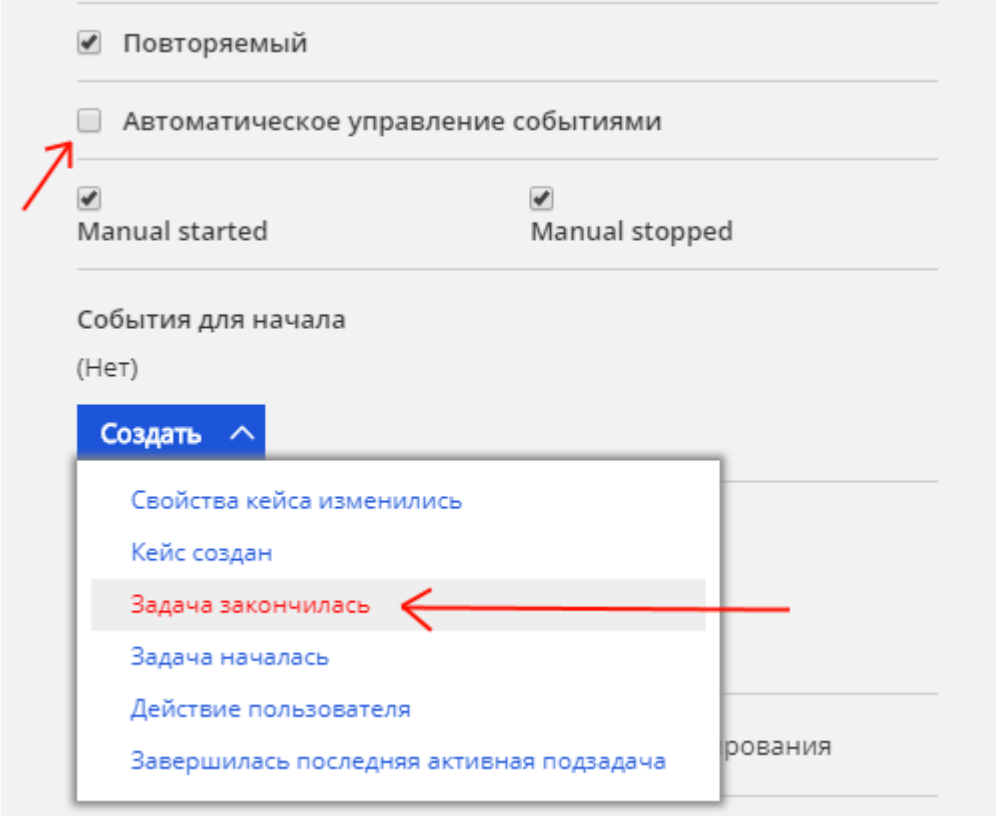


Рисунок 39 – Кнопка «Редактировать активность»

Отключим флаг «Автоматическое управление событиями», удалим сгенерированные события если такие имеются и нажмем кнопку создать в поле «Событие для начала». В выпадающем списке выберем «Задача закончилась» (рисунок 40).



s

Рисунок 40 – Создание события для запуска этапа

В качестве источника события выбираем наше первое действие (рисунок 41). Поле «Условия» оставляем пустым.

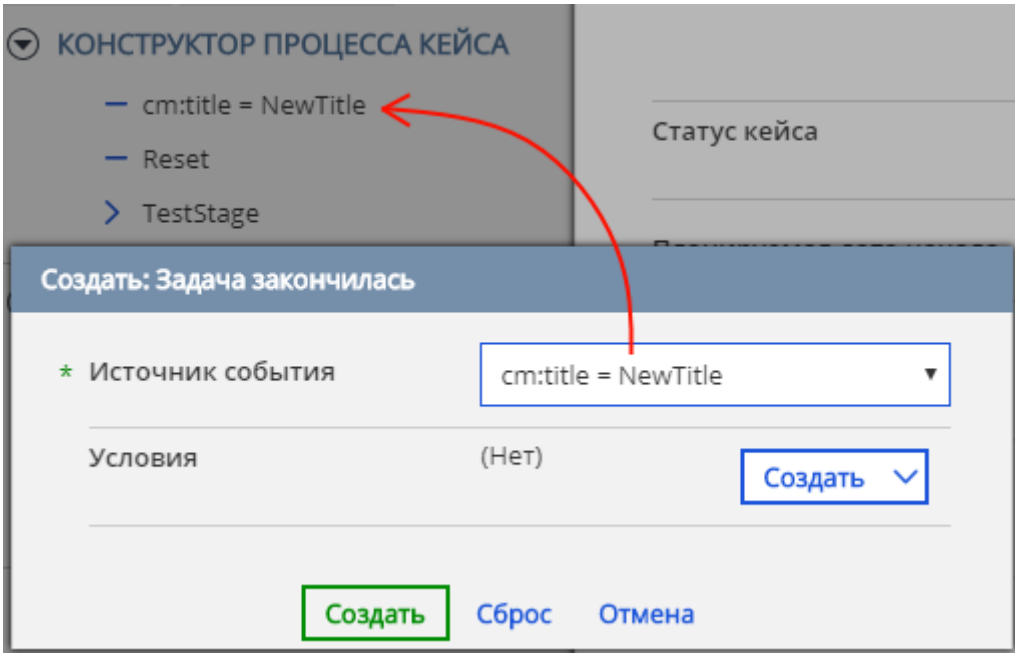


Рисунок 41 – Создание события «Задача закончилась»

Примечание: Если в выпадающем списке источников события отсутствует какая-то активность из кейса, то следует обновить веб страницу и попробовать снова. Если активности по-прежнему нет, то вероятно это свидетельствует о том,

что она недоступна в текущем контексте. Например – в качестве источника события «Задача закончилась» могут служить только соседние активности и родитель.

Нажимаем кнопку создать и создаем событие для завершения нашего этапа. В качестве типа события выберем «Завершилась последняя активная подзадача» (рисунок 42).

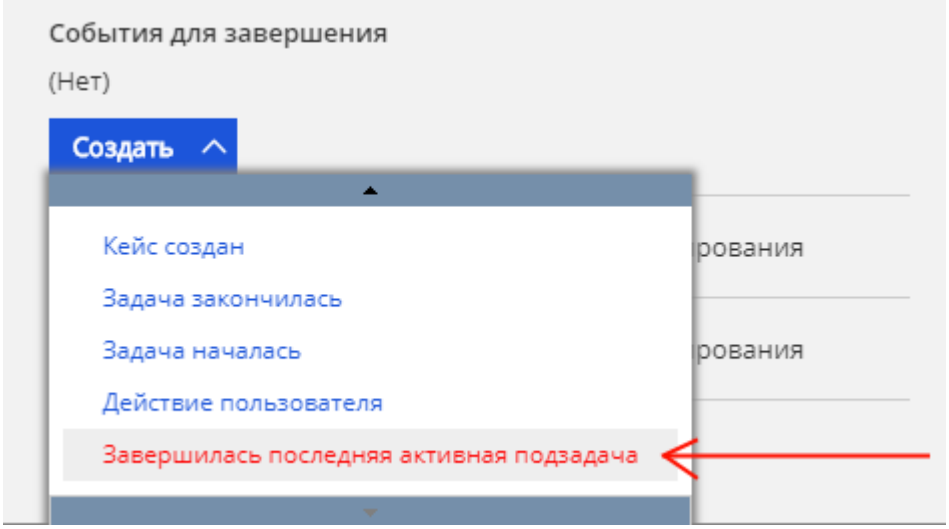


Рисунок 42 – Событие для завершения этапа

В качестве источника события выберем текущий этап (рисунок 43).

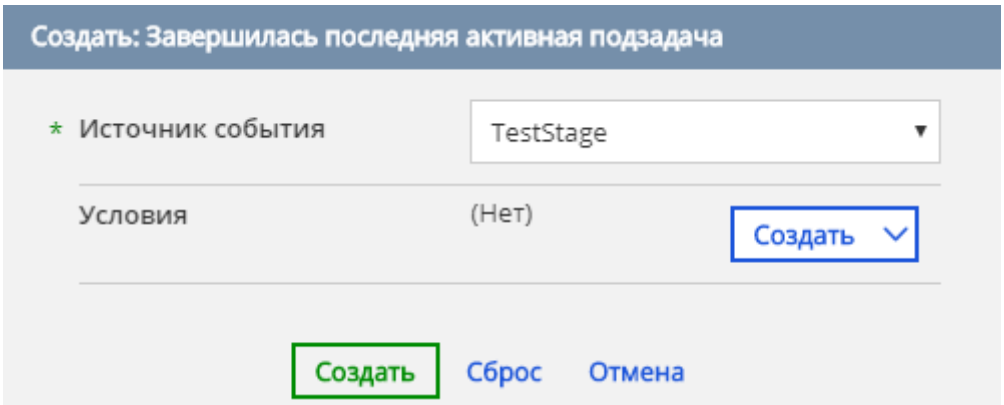


Рисунок 43 – Создание события для завершения

Нажимаем «Создать» на форме создания события и сохраняем изменения в этапе. Теперь попробуем запустить нашу первую активность - действие (cm:title = NewTitle) и видим следующий результат:

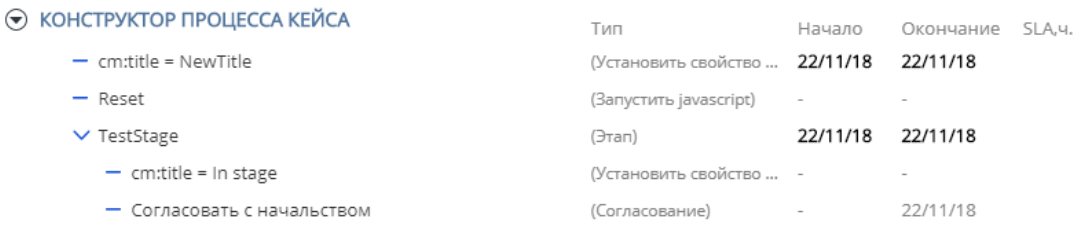


Рисунок 44 – Результат запуска действия вне этапа

Как видим, этап запустился и завершился автоматически после завершения запущенного действия.

Сбрасываем состояние кейса и следующим шагом добавим событие для начала у активностей внутри этапа по уже известной нам схеме (рисунки 45, 46 и 47). Тип события – «Задача началась», источник события – этап «TestStage».

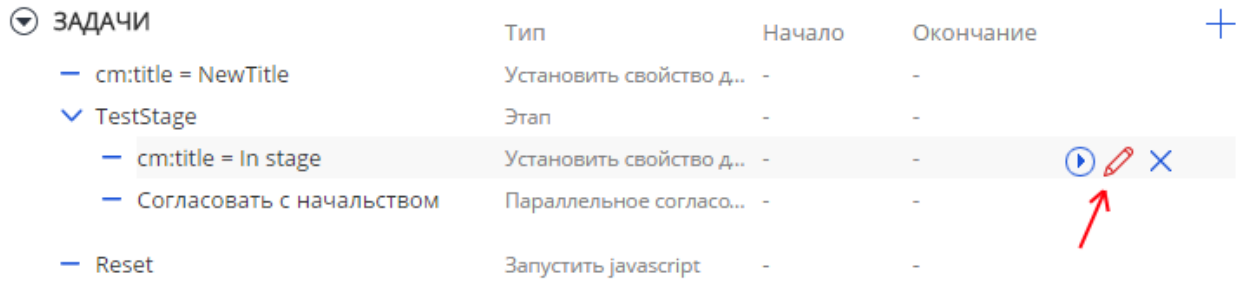


Рисунок 45 – Редактирование действия внутри этапа

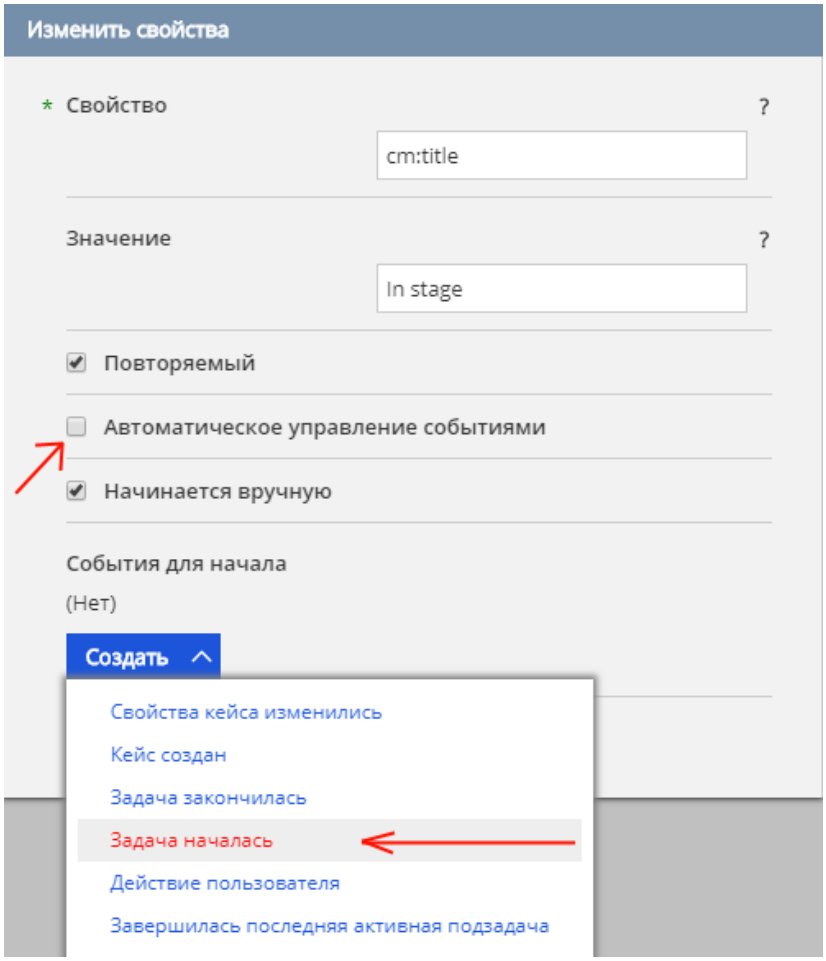


Рисунок 46 – Создание события «Задача началась»

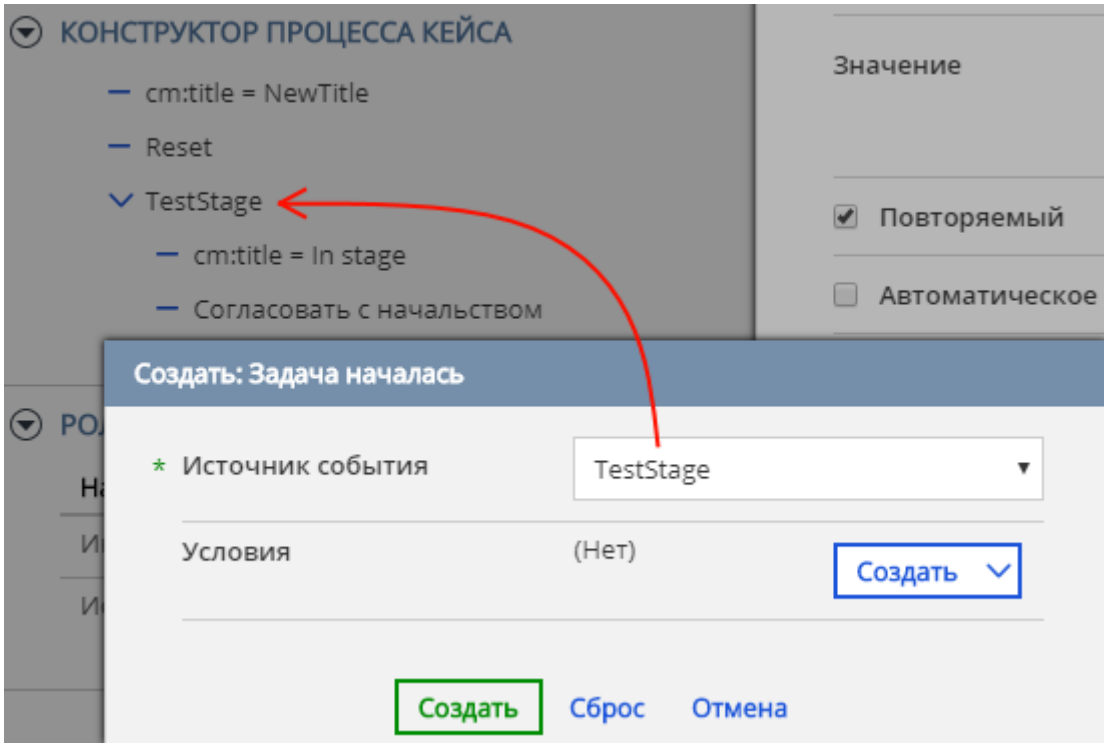


Рисунок 47 – В качестве источника события выбираем родительский этап

Проделываем аналогичные действия для задачи «Согласовать с начальством».

Опробуем созданный процесс. Для этого запускаем первое действие вне этапа (**cm:title = NewTitle**) и перезагрузив страницу видим следующий результат:

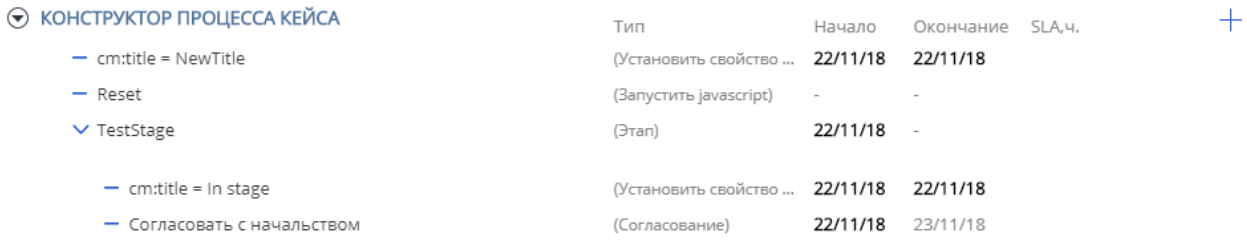


Рисунок 48 – Результат на вкладке «Процесс кейса»

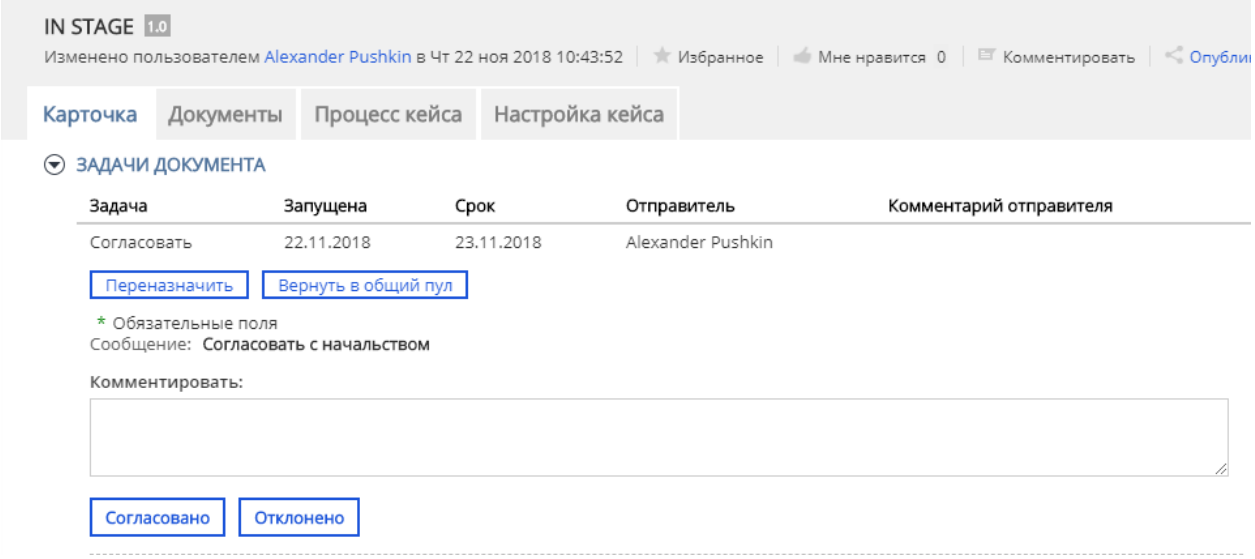


Рисунок 49 – Результат на вкладке «Карточка»

Примечание: если у вас после запуска не появился кардлет «Задачи документа», то это значит, что вы не состоите в роли, выбранной при создании задачи «Согласовать с начальством».

Нажимаем «Согласовано» на форме запущенного процесса и после обновления страницы активности кейса приобретают вид, представленный на рисунке 50.

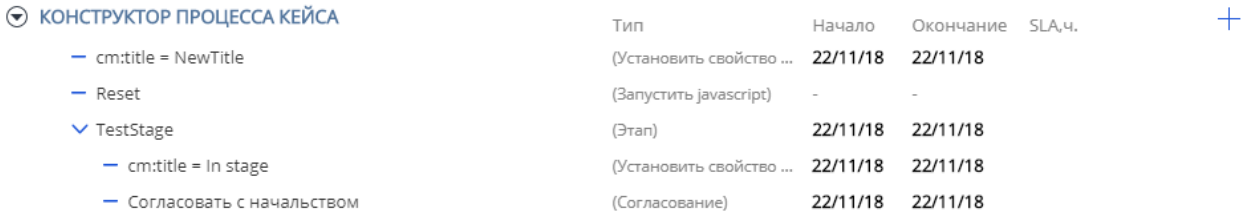


Рисунок 50 – Завершенный процесс

Как видим, все активности, которые мы связывали между собой завершили свою работу и процесс можно считать завершенным.

Добавим возможность старта процесса через колонку действий. Для этого нам потребуется событие «Действие пользователя». Заходим в редактирование действия «**cm:title = NewTitle**» и добавляем событие для старта с типом «Действие пользователя». Устанавливаем произвольные заголовок, сообщение подтверждения и роли, для которых это действие будет показано (если роль не выбрана, то действие доступно для всех) (рисунок 51).

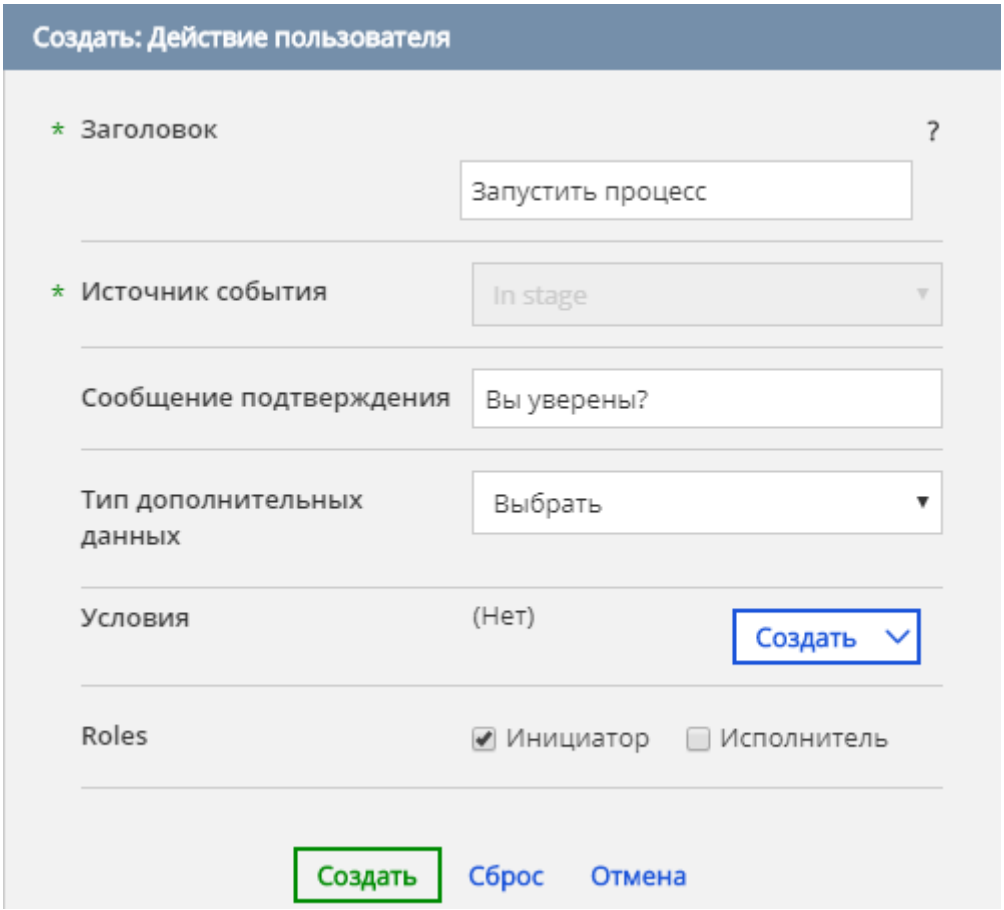


Рисунок 51 – Создание события «Действие пользователя»

После создания нового действия переходим на вкладку «Карточка», обновляем страницу и видим новую запись в колонке «Действия» (рисунок 52).

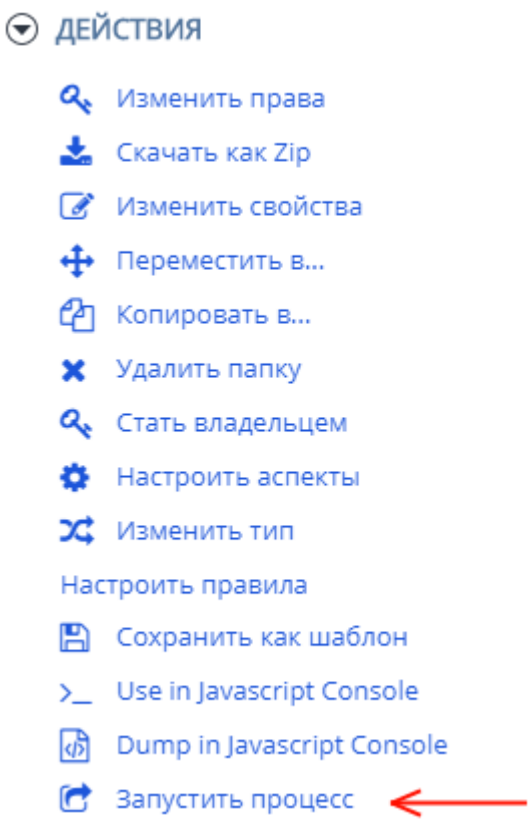


Рисунок 52 – Новое действие

Нажимаем «Запустить процесс», подтверждаем действие и видим, что наш процесс запустился, но действие по-прежнему доступно для выбора. Чтобы скрыть действие, когда процесс уже начат, можно добавить javascript условие для действия пользователя со следующим содержимым:

event.parent.properties['lc:state'] == 'Not started'

Таким образом, после того как связанная с событием активность сменит свой статус, действие пользователя исчезнет из колонки «Действия».

Примечание: сразу после добавления условия видимость действия может не измениться даже если условие возвращает false. Данная особенность связана с кэшированием списка действий на сервере. Можно изменить любой параметр кейса чтобы кэш сбросился.

Условия для событий подробно рассмотрены в следующем разделе.

5. Условия событий

Условия событий – инструмент с помощью которого организуется ветвление процесса. При создании или редактировании любого события можно указать условия, которые должны выполниться для срабатывания события (рисунок 53). Если условий несколько, то они объединяются логическим оператором «И».

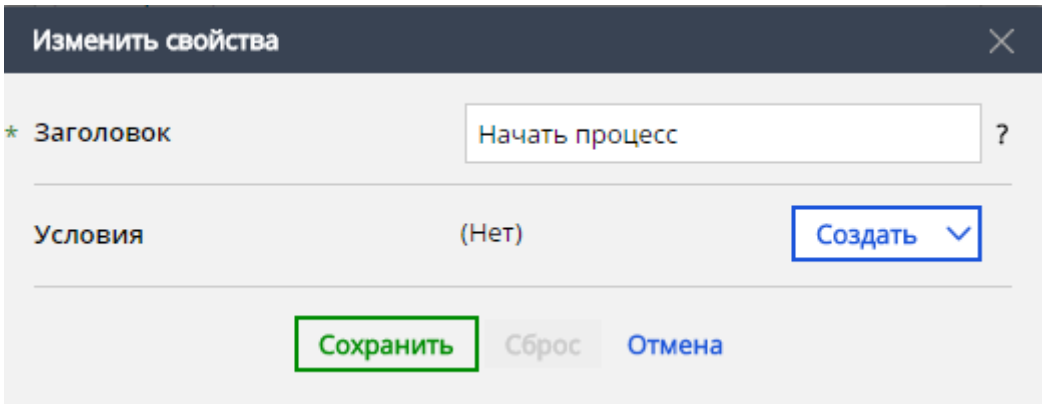


Рисунок 53 – Форма редактирования события

Виды условий:

- **Сравнить свойство**

Сравнивает заданное свойство документа (кейса) с указанным значением. Можно указать разные условия соответствия свойства и значения (Операция) – равно, содержит, начинается с, оканчивается на, больше чем, больше или равно, меньше чем, меньше или равно.

Свойства: Свойство, Операция, Значение

- **Вычислить javascript**

Вычисление произвольного javascript-кода, который должен вернуть true или false. В контексте есть доступ к объекту process, в котором находятся все переменные только что завершенного процесса. Таким образом можно на основе их значений можно принять решение о срабатывании события.

В контексте так же есть объект event, по которому можно получить источник события или активность, которая реагирует на событие.

Свойства: javascript-выражение

Пример:

Для примера использования условий создадим два этапа рядом с TestStage и назовем их «Действия после положительного ответа» и «Действия после отрицательного ответа» (рисунок 54). При создании следует убрать флаг «Автоматическое управление событиями».

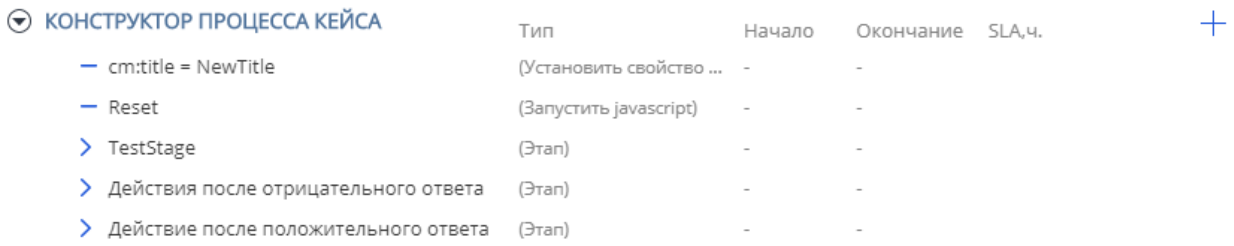


Рисунок 54 – Внешний вид кардлета «Конструктор процесса кейса» после добавления этапов

Для новых этапов добавим «Событие для старта» с типом «Задача завершилась» и источником события TestStage по аналогии с тем как мы это уже делали ранее.

В качестве условия ветвления возьмем результат согласования в задаче «Согласовать с начальством».

Результатом параллельного согласования могут быть значения:

- Confirmed – Согласовано;
- ConfirmedWithComment – Согласовано с комментарием;
- Reject – Отклонено.

Добавим условия для событий старта новых этапов, по которым один этап будет запускаться после успешного согласования, а другой – при отказе. Для этого зайдем в редактирование событий для начала (рисунок 55) и создадим javascript условие (рисунок 56).

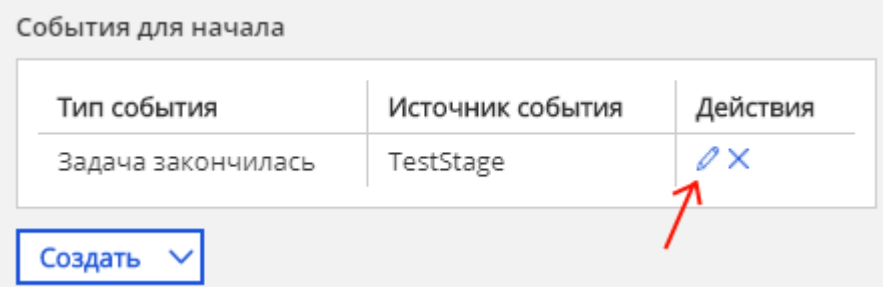


Рисунок 55 – Редактирование события

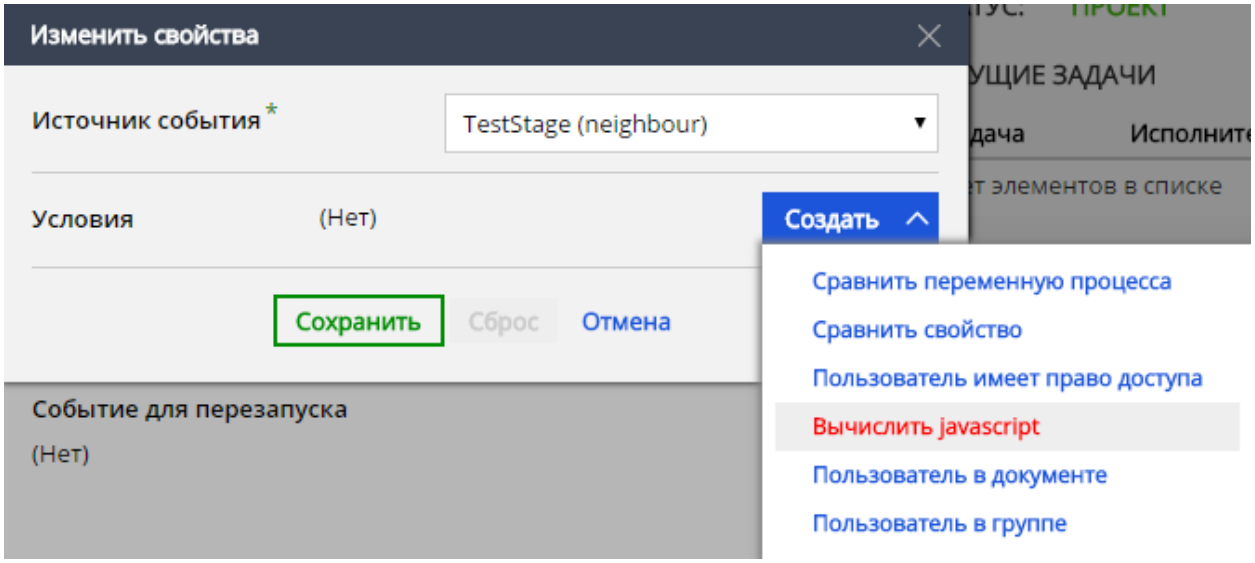


Рисунок 56 – Создание условия для старта

Результат завершения задачи «Параллельное согласование» будет записан в переменной outcome, а все переменные процесса записаны в объекте **process**.

Для этапа «Действия после положительного ответа» в условии события старта записываем следующее javascript выражение:

process.outcome=='Confirmed' ||
process.outcome=='ConfirmedWithComment'

А для этапа «Действия после отрицательного ответа» проверим выход процесса на «отказ»:

process.outcome=='Reject'

Сохраняем изменения и запускаем первое действие вне этапа (**cm:title = NewTitle**). Как и раньше, это должно привести к запуску задачи «Согласовать с начальством» (можно так же включить ручной запуск этапа TestStage и запустить его вручную). Нажимаем кнопку «Согласовано» на вкладке «Карточка» и после перезагрузки страницы наш процесс приобретает следующий вид:

КОНСТРУКТОР ПРОЦЕССА КЕЙСА

	Тип	Начало	Окончание	SLA,ч.
cm:title = NewTitle	(Установить свойство ...	22/11/18	22/11/18	
Reset	(Запустить javascript)	-	-	
TestStage	(Этап)	22/11/18	22/11/18	
Действия после отрицательного ответа	(Этап)	-	-	
Действие после положительного ответа	(Этап)	22/11/18	-	

Нет задач

Рисунок 57 – Нажата кнопка «Согласовать»

Сбросим состояние активностей в кейсе и попробуем еще раз, но уже нажав кнопку «Отклонено». Результат представлен на рисунке 58.

КОНСТРУКТОР ПРОЦЕССА КЕЙСА

	Тип	Начало	Окончание	SLA,ч.
cm:title = NewTitle	(Установить свойство ...	22/11/18	22/11/18	
Reset	(Запустить javascript)	-	-	
TestStage	(Этап)	22/11/18	22/11/18	
Действия после отрицательного ответа	(Этап)	22/11/18	-	
Нет задач				
Действие после положительного ответа	(Этап)	-	-	

Рисунок 58 – Нажата кнопка «Отклонено»

Как видим, благодаря условиям запускается только один этап в зависимости от решения согласующего.

6. Чек-листы

Чек-листы – это механизм, который позволяет предъявить требования к кейсу на заданном этапе процесса. Например, при завершении задачи «Проведение кредитного комитета» можно потребовать документ «Протокол проведения кредитного комитета» и пока он не будет приложен к кейсу задачу нельзя будет завершить.

Чек-листы создаются глобально для всей системы, а в кейсе происходит выбор из этих листов. Создание и редактирование чек-листов производится через журнал чек-листов. Чтобы в него перейти, следует открыть меню и нажать пункт «Системные журналы» (рисунок 59).

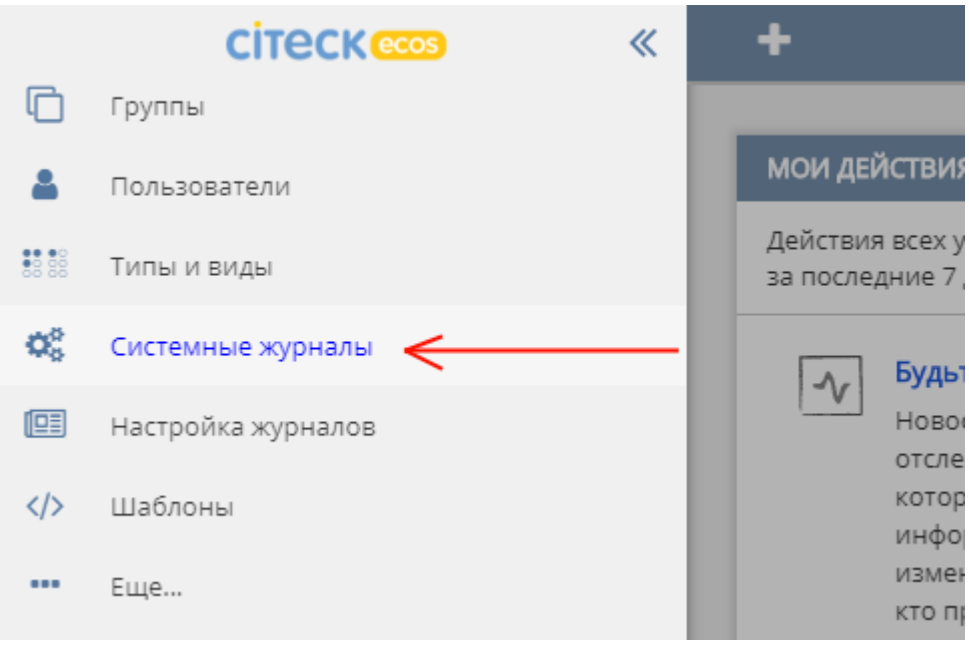


Рисунок 59 – Системные журналы

Если такого пункта меню нет, то можно перейти в системные журналы по прямой ссылке:

<address>:<port>/share/page/journals2/list/system.

Далее переходим в журнал «Чек-листы кейсов»:

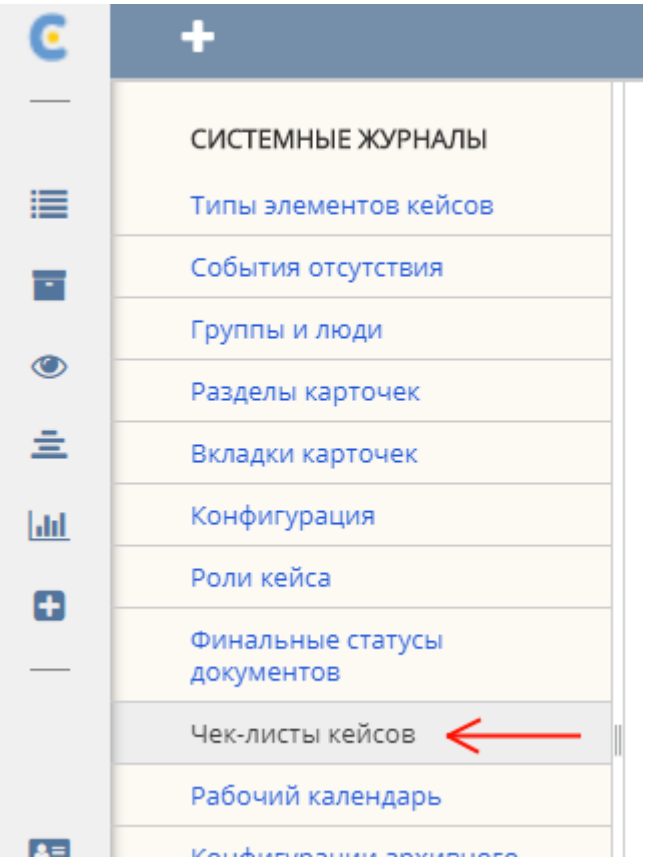


Рисунок 60 – Выбор журнала «Чек-листы кейсов»

По умолчанию доступно три чек-листа: «Документы сканирования», «Документы контрагента» и «Прочее». Создадим новый Чек-лист. Для этого нажмем кнопку «Создать» и выберем пункт «Чек-лист»:

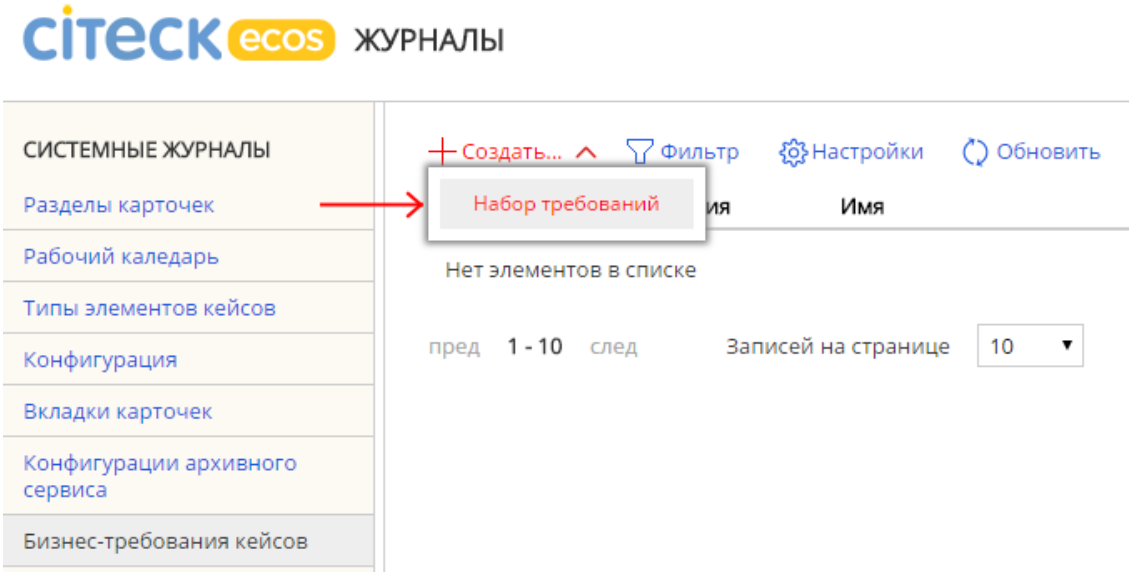


Рисунок 61 – Создание нового чек-листа

Форма создания чек-листа представлена на рисунке 62.

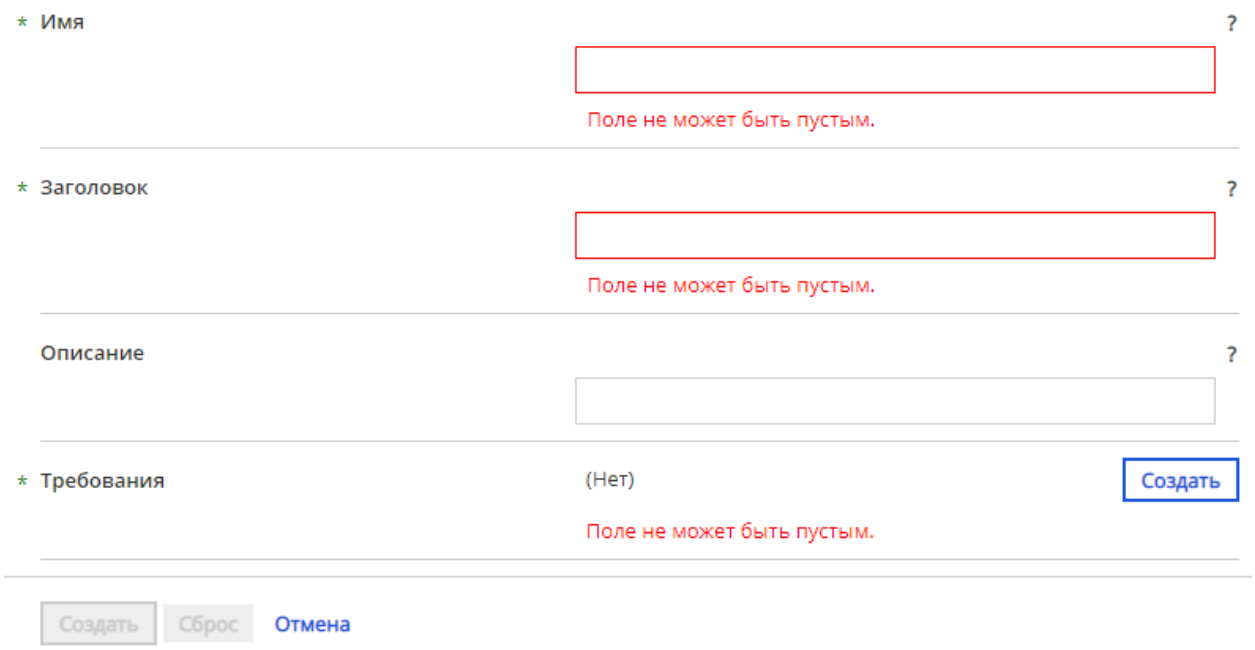


Рисунок 62 – Создание чек-листа

Добавим новые требования. Для этого нажимаем «Создать» в поле «Требования» и видим следующую форму:

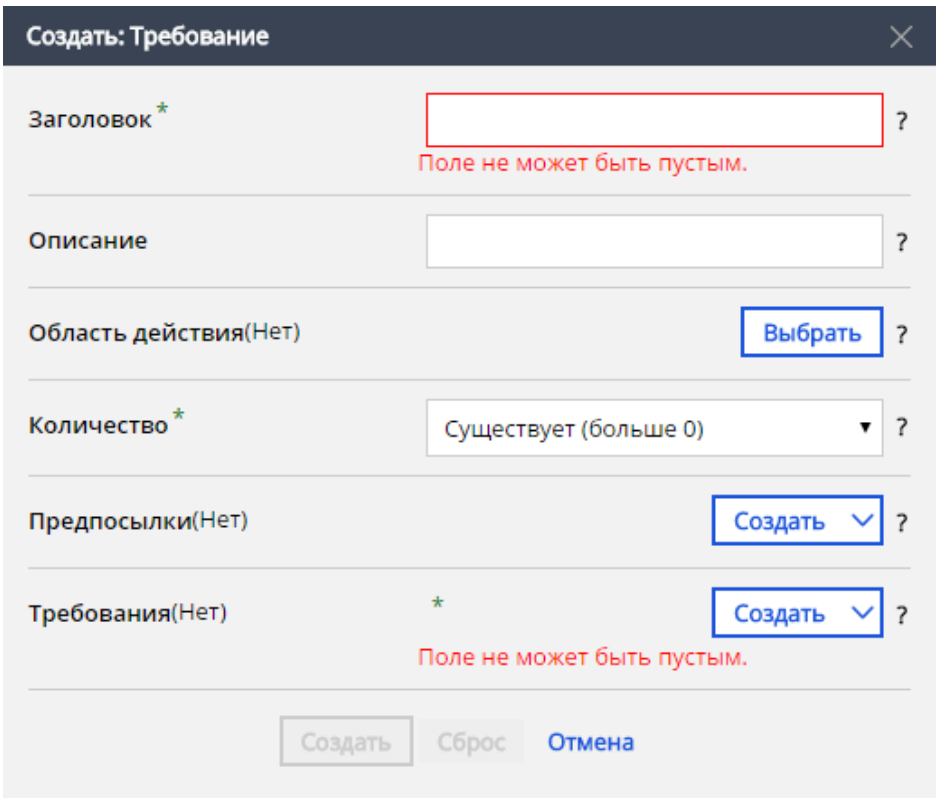


Рисунок 63 – Форма создания требования

Поля «Заголовок» и «Описание» могут быть заполнены произвольным образом.

Поле «Область действия» определяет соответственно область действия требований.

Доступные области действия:

- Типы элементов кейса
- Категории
- Тэги
- Дополнительные файлы
- Документы
- События
- Задачи
- Роли
- Связанные документы
- Подкейсы

Выбираем область действия – «Документы».

Поле «Количество» определяет, как будет проверяться требование. Например, если у нас требование к документу, то мы можем указать в этом поле:

Существует – документ должен быть в кейсе в единственном или множественных экземплярах (количество >= 1);

Единичное – может быть в кейсе в единственном экземпляре (количество 0 или 1);

- Ровно 0 – документа не должно быть в кейсе;
- Ровно 1 – документ должен быть в кейсе в единственном экземпляре;
- Необязательный – требования в любом случае считаются выполненными.

В поле «Предпосылки» добавляются требования, которые определяют следует или нет проверять требования, описанные в поле «Требования». В полях «Предпосылки» и «Требования» доступен одинаковый набор создаваемых видов требований.

Виды требований:

- Требование к типу
- Требуется тип подкейса
- Условие
- JavaScript-предикат
- Выполнен набор требований
- Требование
- Механизм проверки требований следующий:

Для каждого элемента в области действия: Вычисляем предпосылки. Если они все возвращают true или их просто нет, то переходим к проверке требований. Если хотя бы одна из предпосылок вернула false, то родительское требование считается выполненным и требования из поля «Требования» не проверяются. Таким образом вычисляется количество элементов, подходящих требованиям и

это количество сравнивается с полем «Количество». Если количество подходящее, то родительское требование считается выполненным, иначе – нет.

Проверку предпосылок и требований можно записать следующим образом:

p1 & p2 & p3 || r1 & r2 & r3

где p1..3 – предпосылки;

r1..3 – требования.

Создадим новое требование к типу:

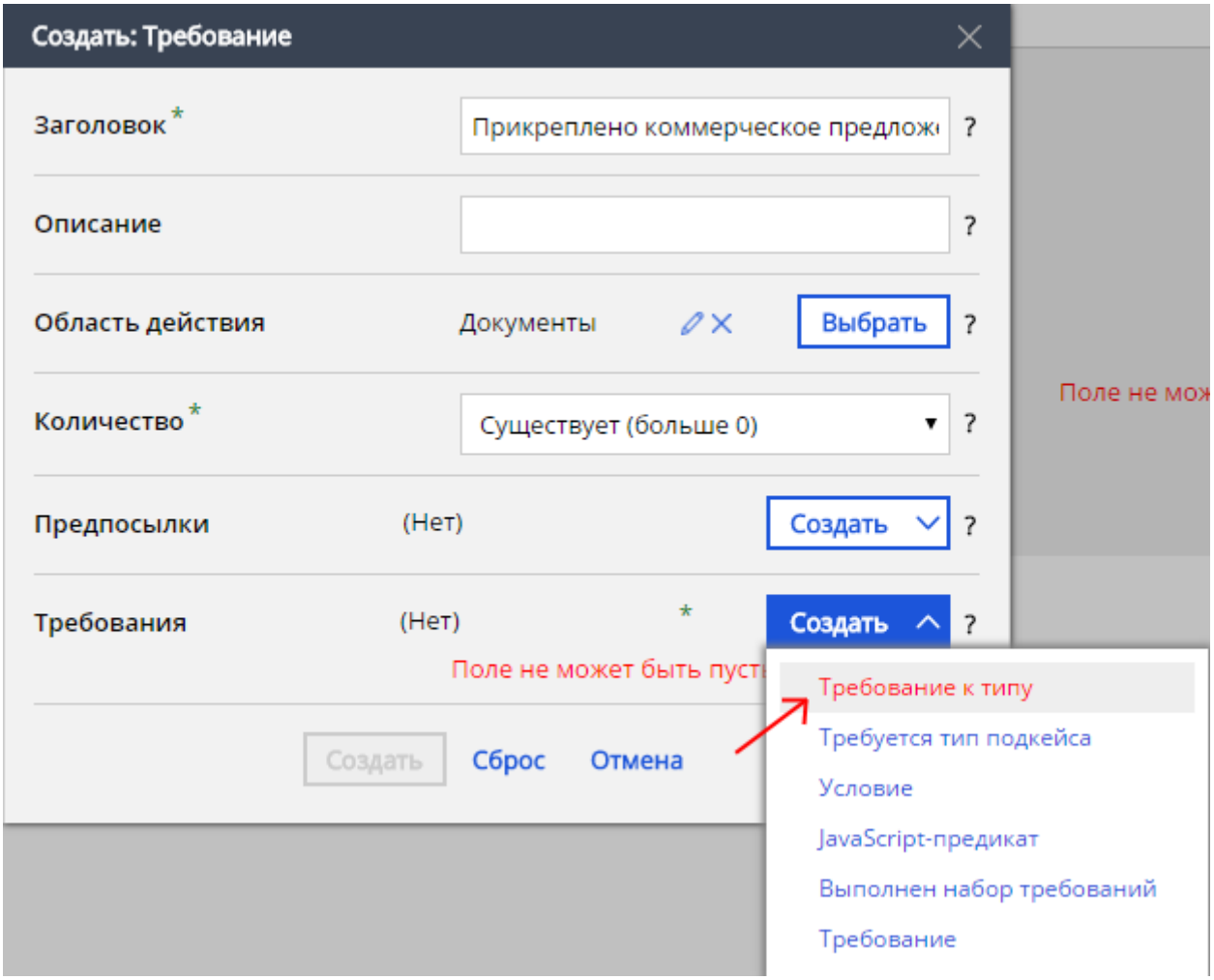


Рисунок 64 – Создание требования к типу

Требуемый тип выберем «Документ», требуемый вид – «Коммерческое предложение»:

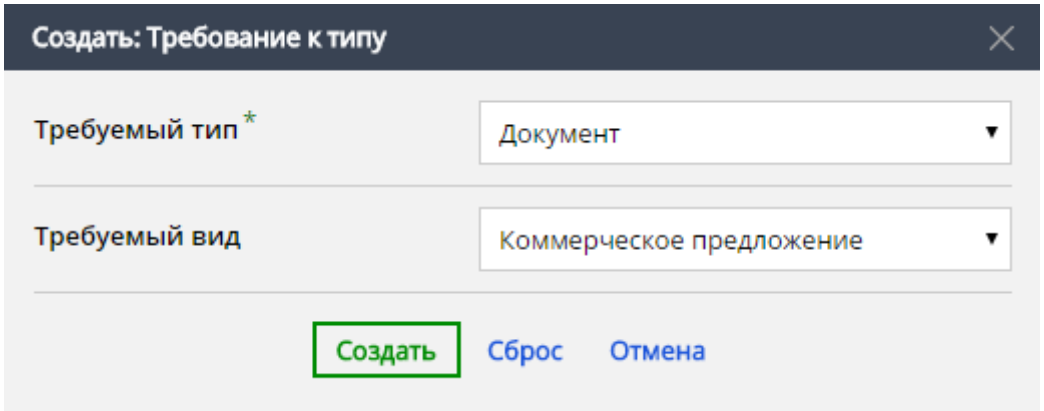


Рисунок 65 – Создание требования к типу

Нажимаем кнопку создать, и наша форма создания принимает вид, представленный на рисунке 66:

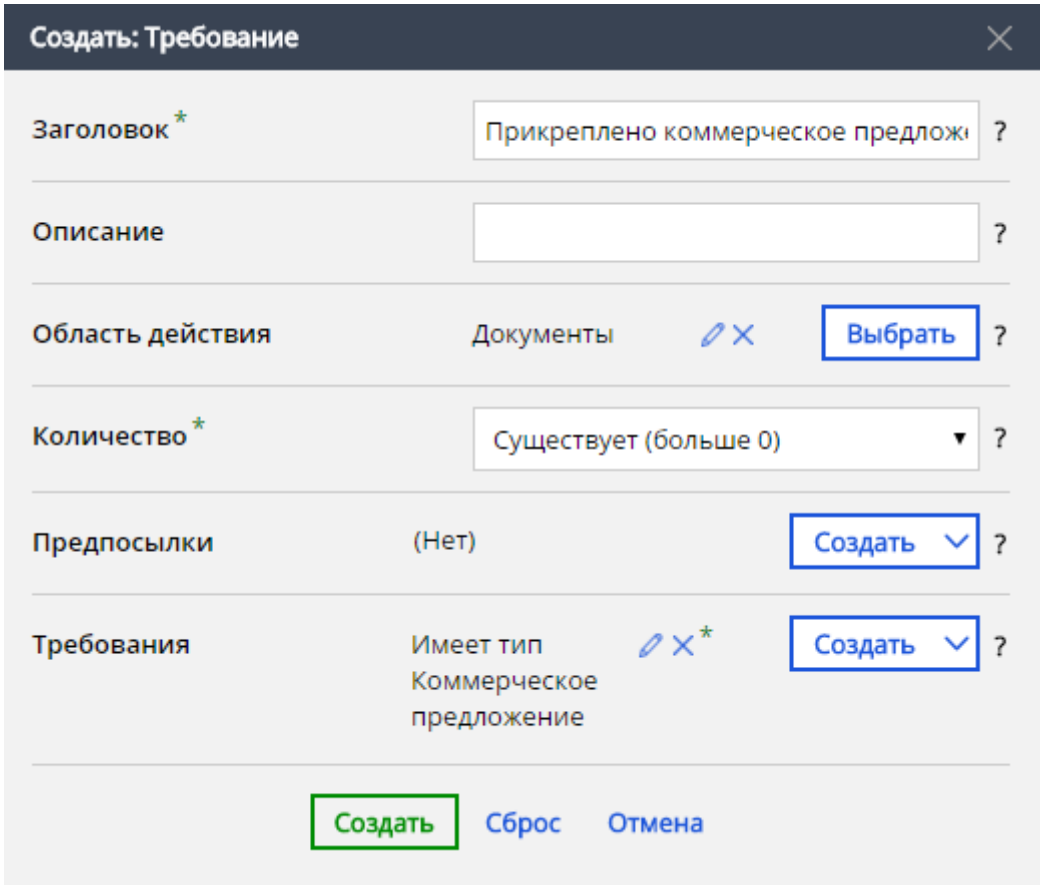


Рисунок 66 – Форма создания требования после заполнения полей

Нажимаем создать и заканчиваем создания нашего набора требований так же нажатием кнопки «Создать».

Переходим на карточку кейса и открываем вкладку «Настройка кейса» (рисунок 67).

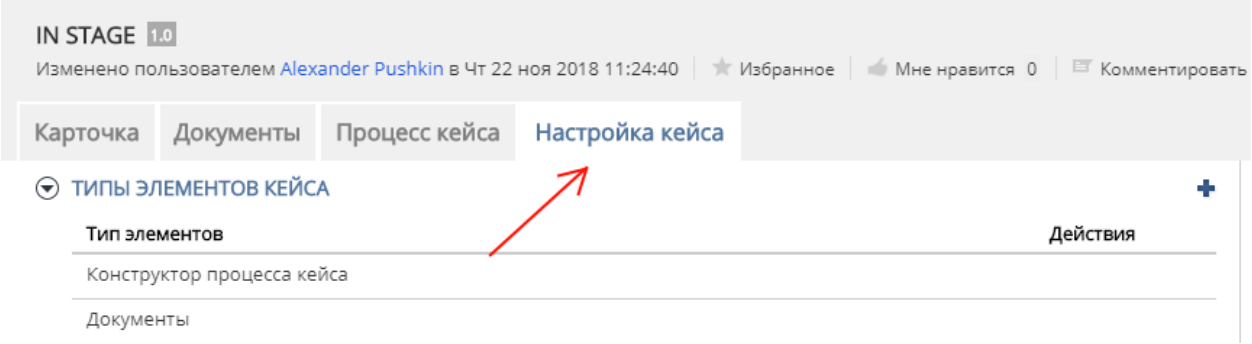


Рисунок 67 – Вкладка «Настройка кейса»

Далее нажимаем кнопку «Добавить чек-лист» (рисунок 68).

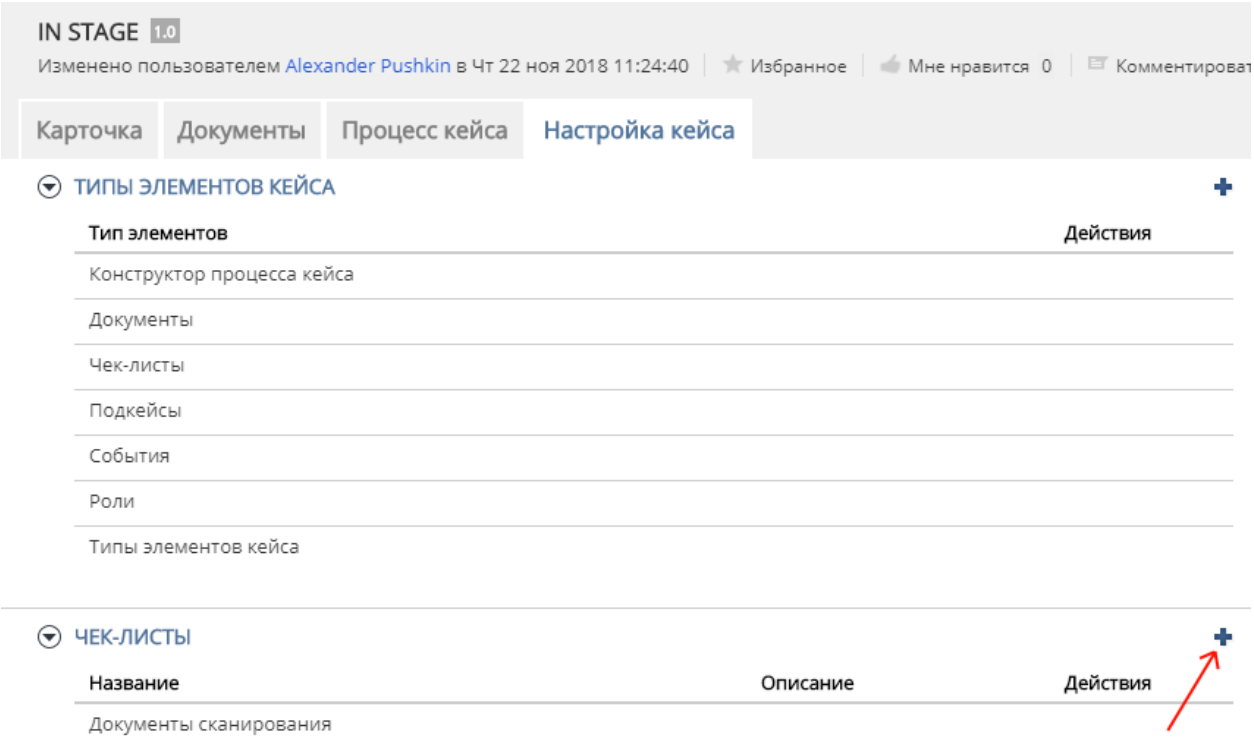


Рисунок 68 – Кнопка «Добавить чек-лист»

В открывшемся окне видим наш чек-лист, который мы создали ранее (Рисунок 69). Нажимаем кнопку «+» рядом с чек-листом и подтверждаем выбор нажатием на кнопку «ОК».

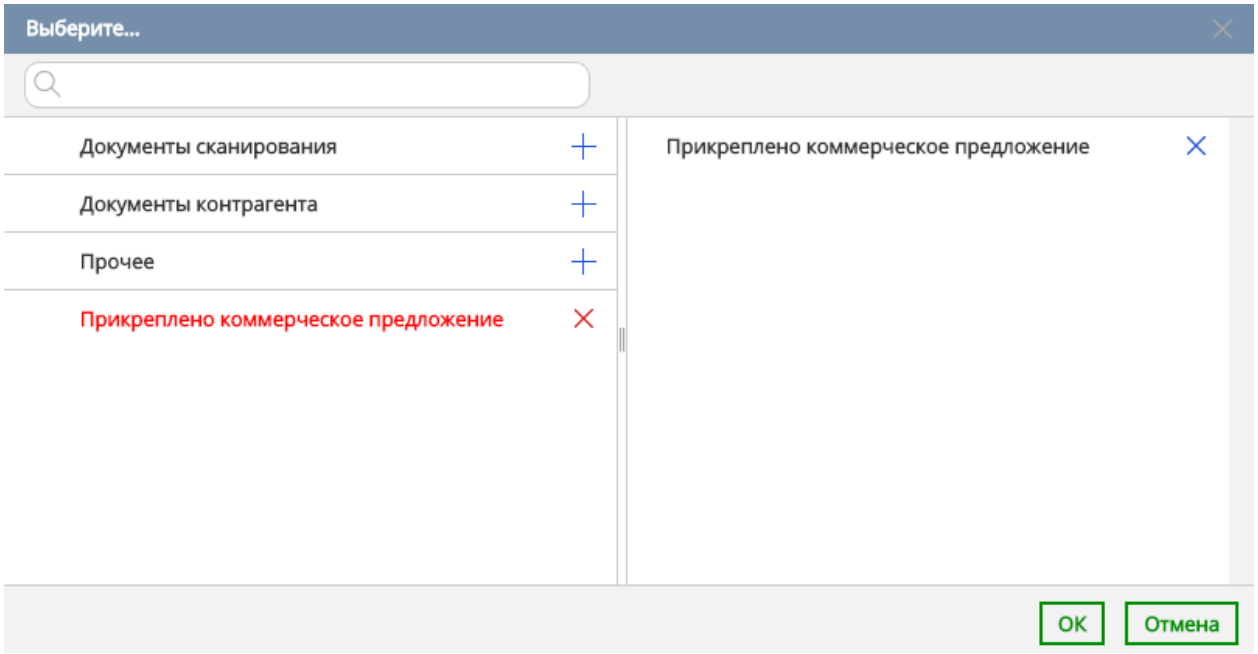


Рисунок 69 – Выбор чек-листов

Примечание: Если в списке ничего нет, то следует ввести «» в поле поиска и нажать Enter.*

После этого в кардлете «Чек-листы» должен появиться наш чек-лист (рисунок 70).

Рисунок 70 – Добавленный чек-лист

Возвращаемся на вкладку «Карточка», обновляем страницу и в правом столбце в кардлете «Выполнение чек-листов» видим следующее:

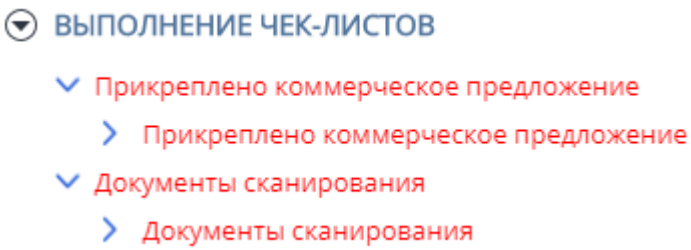


Рисунок 71 – Выполнение чек-листов

Как видим, наше требование горит красным, что свидетельствует о том, что оно не выполнено.

Теперь мы можем выбрать добавленное требование как обязательное для старта или завершения задачи, или этапа. Заходим в редактирование этапа TestStage и видим, что появилась возможность выбора нашего чек-листа:

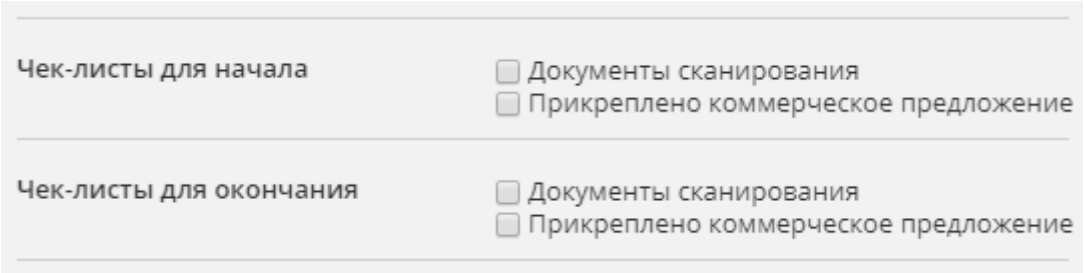


Рисунок 72 – Выбор чек-листов

Установим галочку напротив нашего требования в поле «Чек-листы для начала» и в поле «Manual started», чтобы проверить результат. Сохраняем изменения.

Пробуем запустить этап «TestStage» и видим следующее:

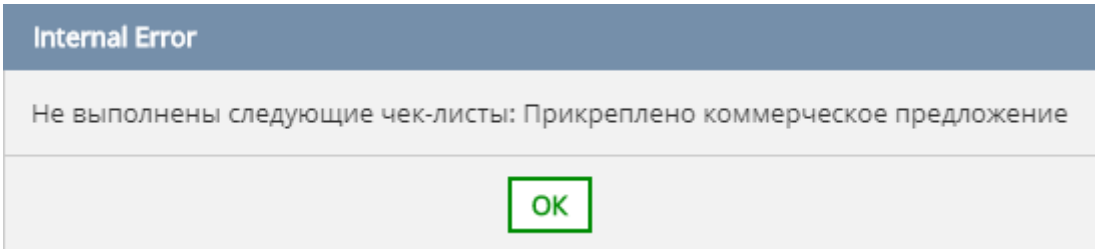


Рисунок 73 – Ошибка «Не выполнены чек-листы»

Этап не изменил своего состояния и до тех пор, пока требования не будут выполнены его запуск заблокирован.

Теперь попробуем запустить первую задачу в нашем процессе – «**cm:title = NewTitle**». После примеров из предыдущих разделов этап TestStage должен стартовать после завершения этой задачи. Запускаем и видим точно такую же ошибку. Если мы посмотрим на процесс, то увидим, что ни одна из активностей не изменила своего состояния.

В данном случае произошло следующее:

Действие «**cm:title = NewTitle**» запустилось, затем завершилось, далее сработало событие для старта TestStage, проверились чек-листы и так как они не выполнились, то вся цепочка изменений отменилась. То есть все активности вернулись в состояние до нажатия кнопки старта. Таким образом, какой бы длинной не была цепочка вызовов – если на одном из её этапов чек-листы не выполнены, то все изменения отменяются.

Попробуем выполнить требования. Для этого перейдем на вкладку «Документы» (рисунок 74).

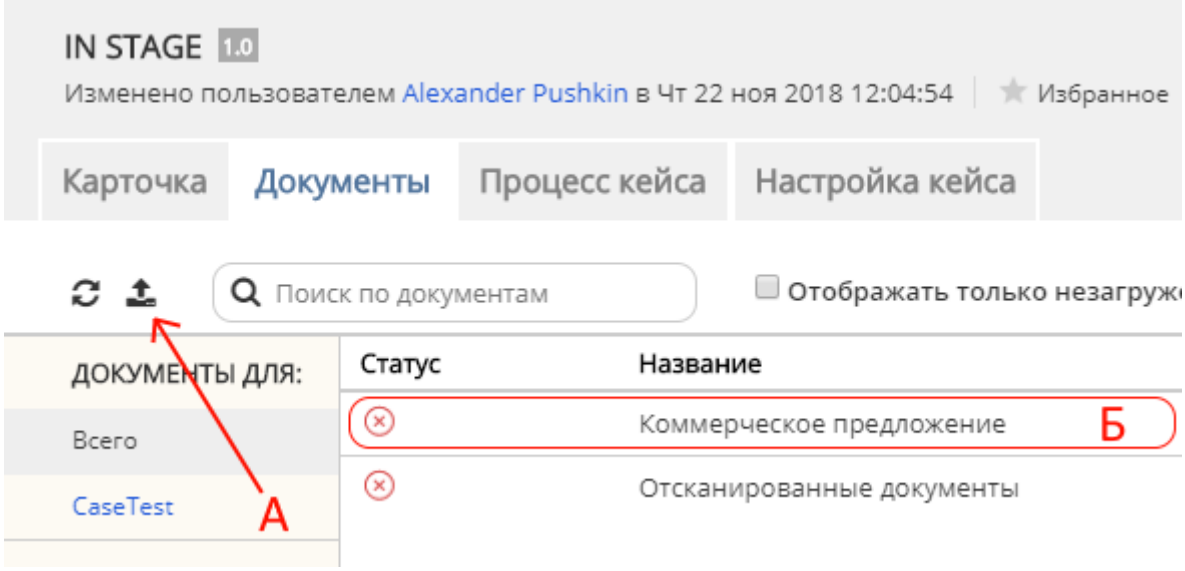


Рисунок 74 – Вкладка «Документы»

Здесь мы видим, что появился документ, который мы добавили в требовании.

Загрузить недостающий документ можно двумя способами:

- А – кнопка, при нажатии на которую открывается диалоговое окно с возможностью выбрать кейс, тип и вид загружаемого документа;
- Б – можно перенести любой файл с компьютера на нужный документ, и он автоматически загрузится (drag & drop). Этот вариант предпочтительнее ввиду его простоты.

После загрузки мы видим следующее:

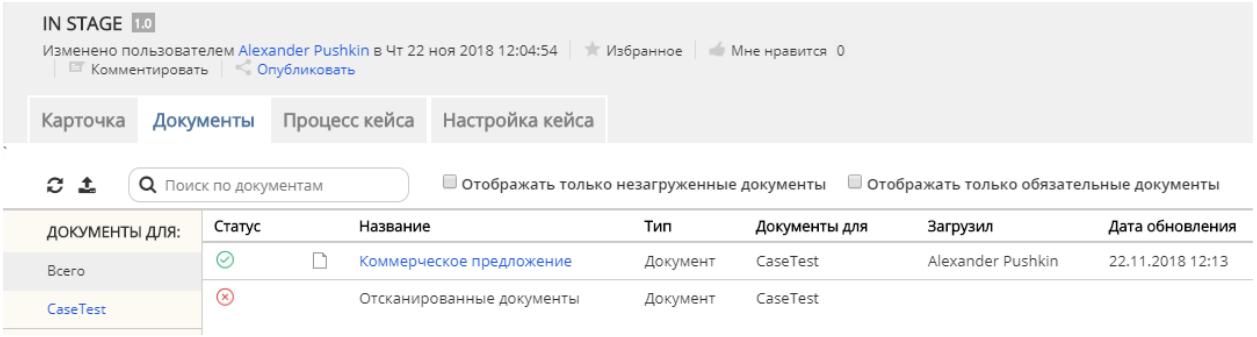


Рисунок 75 – Результат загрузки документа

Возвращаемся на вкладку «Карточка», обновляем страницу и видим, что в кардлете «Выполнение чек-листов» наш чек-лист поменял цвет на зеленый (рисунок 76).

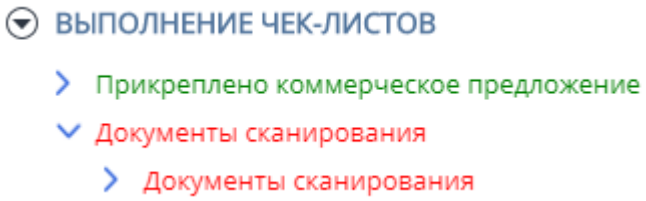


Рисунок 76 – Соответствие кейса требованиям после загрузки документа

Пробуем запустить этап TestStage и видим, что этап успешно запустился так как требования для его старта были выполнены.

7. Шаблон кейса

Создаваемый кейс мы можем сохранить как шаблон, и он будет применяться к каждому новому кейсу. В шаблоне сохраняются чек-листы, активности и роли.

Чтобы сохранить кейс как шаблон следует нажать кнопку «Сохранить как шаблон» (рисунок 77).

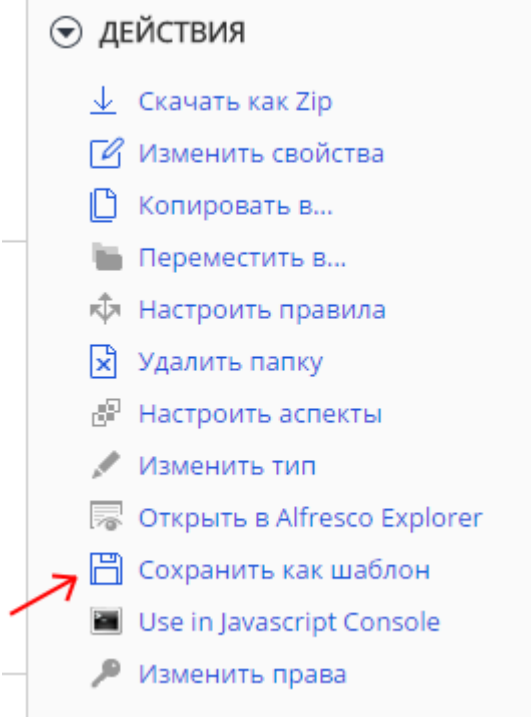


Рисунок 77 – Действие «Сохранить как шаблон»

Теперь ваш процесс будет переноситься на все новые кейсы.

*Примечание: шаблоны применяются только к кейсам (есть аспект «**icase:case**»), то есть если вы сохраните создаваемый нами ранее кейс (тип – **cm:folder**) как шаблон, то он не будет просто применяться ко всем новым папкам. Но если вы добавите аспект «**icase:case**» к другой папке, то к ней применится шаблон. Если же у вас в модели описан тип у которого в **mandatory-aspects** стоит **icase:case**, то все работает без дополнительных действий.*

Для редактирования шаблона кейса необходимо удалить старый шаблон, изменить какой-нибудь кейс и сохранить его как шаблон.

Для удаления, скачивания шаблонов в виде xml и загрузки новых удобно использовать журнал шаблонов:

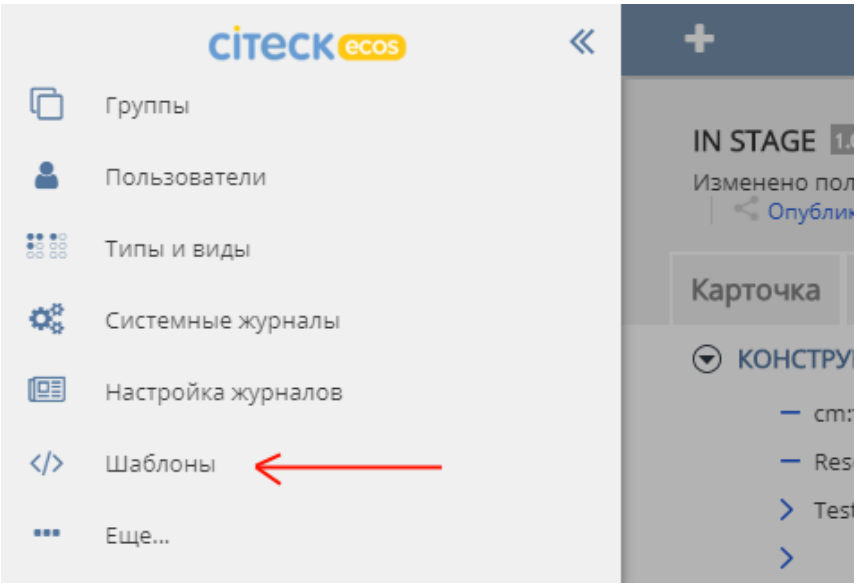


Рисунок 78 – Журнал шаблонов

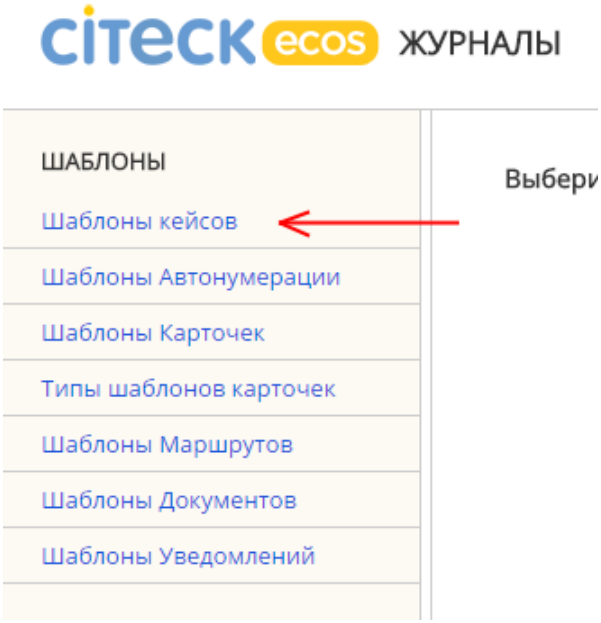


Рисунок 79 – Журнал шаблонов кейсов

8. Продвинутые темы

8.1. Расширение списка доступных процессов кейса на базе Activiti

Если функционала стандартных видов задач недостаточно, то можно добавить новую задачу на основе любого BPMN процесса. Для этого необходимо выполнить как минимум шесть этапов:

1. Создание BPMN процесса;
2. Создание моделей типов для задач в BPMN процессе;
3. Создание модели типа задачи кейс-менеджмента;
4. Создание форм для типов задач в BPMN процессе;
5. Создание формы для типа задачи кейс-менеджмента;
6. Добавление маппинга параметров из задачи кейс-менеджмента в BPMN процесс;

BPMN процесс

С помощью Activiti Designer создаем простой процесс с одним входом, одним выходом и одной задачей.

Примечание: Процесс может быть сколь угодно сложным, но рекомендуется создавать небольшие процессы, из которых на уровне кейс-менеджмента можно сделать сложные.

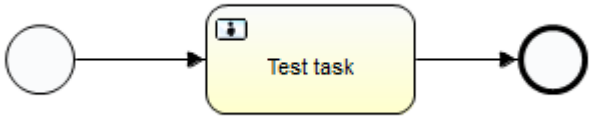


Рисунок 80 – Простой BPMN процесс

Кликаем в пустое место на поле и в свойствах настраиваем процесс как показано на рисунке 81.

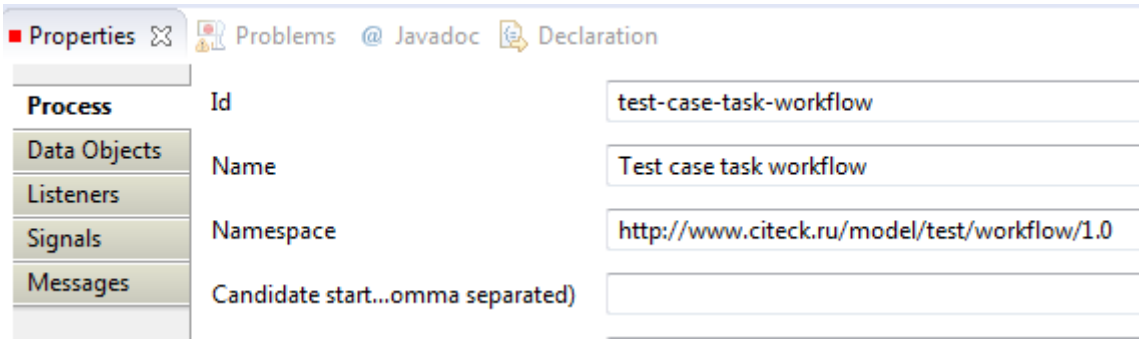


Рисунок 81 – Настройки BPMN процесса

Затем выделяем задачу «Test task» и в свойствах на вкладке «Main config» в поле «Form key» записываем **twf:testBPMNTask**.

Далее необходимо выбрать исполнителя задачи. Чтобы была возможность выбрать как группу, так и человека в Citeck ECOS предусмотрен класс AssignTaskToAuthority. Чтобы им воспользоваться переходим в свойствах задачи на вкладку Linsteners, нажимаем справа кнопку New, выбираем

Event: create,

Type: Java class,

Class: ru.citeck.ecos.workflow.listeners.AssignTaskToAuthority

Field name: authority

Expression: ${twf_performer}

Если возникают трудности с выбором класса, то можно открыть процесс в текстовом редакторе и прописать вручную:

```
<userTask id="TestTask" name="Test task" activiti:formKey="twf:testBPMNTask">
    <extensionElements>
        <activiti:taskListener event="create"
              class="ru.citeck.ecos.workflow.listeners.AssignTaskToAuthority">
            <activiti:field name="authority">
                <activiti:expression>
                    <![CDATA[${twf_performer}]]>
                </activiti:expression>
            </activiti:field>
        </activiti:taskListener>
    </extensionElements>
</userTask>
```

Модели

Создадим модель для нашей BPMN задачи. В общем виде модель типа выглядит следующим образом:

```
<type name="Название_типа">
    <parent>Родитель</parent>
    <properties>
        <!-- Свойства типа -->
    </properties>
    <associations>
        <!-- Ассоциации типа -->
    </associations>
    <overrides>
        <!-- Переопределения свойств родительского типа -->
    </overrides>
</type>
```

Названием типа для BPMN задачи мы выберем **twf:testBPMNTask** (такое же как и при создании BPMN процесса). Для пользовательских задач в BPMN процессах родительским типом как правило является **bpm:activitiOutcomeTask**. В свойствах нашего типа должен находиться как минимум один элемент – свойство для хранения результатов завершения задачи. Назовем это свойство **twf:completeOutcome**. Поскольку вариантов завершения задачи у нас не может быть бесконечное количество, то нам необходимо наложить ограничение на

возможные значения этого свойства (список возможных вариантов). В результате получаем следующее:

```
<property name="twf:completeOutcome">
    <type>d:text</type>
    <default>Complete</default>
    <constraints>
        <constraint name="twf:completeOutcomeOptions" type="LIST">
            <parameter name="allowedValues">
                <list>
                    <value>Complete</value>
                    <value>Reject</value>
                    <value>Other</value>
                </list>
            </parameter>
        </constraint>
    </constraints>
</property>
```

Для имитации некоторых данных, которые необходимо заполнить в ходе нашей задачи, добавим дополнительно два поля – строковое и булево:

```
<property name="twf:someTestField">
    <title>Some text field</title>
    <type>d:text</type>
</property>
<property name="twf:someBooleanField">
    <title>Some Boolean field</title>
    <type>d:boolean</type>
</property>
```

В ассоциациях нам необходим исполнитель задачи с типом **cm:authority** (от этого типа наследуются **cm:person и cm:authorityContainer**):

```
<association name="twf:performer">
    <source>
        <mandatory>false</mandatory>
        <many>true</many>
    </source>
    <target>
        <class>cm:authority</class>
        <mandatory>false</mandatory>
        <many>false</many>
    </target>
</association>
```

В разделе **overrides** нам нужно переопределить свойство **bpm:outcomePropertyName**, чтобы указать название свойства, в котором будут храниться результаты завершения задачи (здесь необходимо указывать полное имя свойства, где префикс **twf** расписывается в **http://www.citeck.ru/model/test/workflow/1.0**):

```
<property name="bpm:outcomePropertyName">

<default>{http://www.citeck.ru/model/test/workflow/1.0}completeOutcome</default>
</property>
```

В результате получаем следующий тип для нашей задачи в BPMN процессе:

```
<type name="twf:testBPMNTask">
    <title>Test BPMN Task</title>
    <parent>bpm:activitiOutcomeTask</parent>
    <properties>
        <property name="twf:someTestField">
```

```
            <title>Some text field</title>
            <type>d:text</type>
        </property>
        <property name="twf:someBooleanField">
            <title>Some Boolean field</title>
            <type>d:boolean</type>
        </property>
        <property name="twf:completeOutcome">
            <type>d:text</type>
            <default>Complete</default>
            <constraints>
                <constraint name="twf:completeOutcomeOptions" type="LIST">
                    <parameter name="allowedValues">
                        <list>
                            <value>Complete</value>
                            <value>Reject</value>
                            <value>Other</value>
                        </list>
                    </parameter>
                </constraint>
            </constraints>
        </property>
    </properties>
    <associations>
        <association name="twf:performer">
            <source>
                <mandatory>false</mandatory>
                <many>true</many>
            </source>
            <target>
                <class>cm:authority</class>
                <mandatory>false</mandatory>
                <many>false</many>
            </target>
        </association>
    </associations>
    <overrides>
        <property name="bpm:outcomePropertyName">
<default>{http://www.citeck.ru/model/test/workflow/1.0}completeOutcome</default>
        </property>
    </overrides>
</type>
```

Для задачи кейс менеджмента модель описывается намного проще. Если исполнитель у нас один, то нам достаточно унаследовать наш тип от **icaseTask:simpleTask** и переопределить свойство, в котором хранится название BPMN процесса для запуска (**icaseTask:workflowDefinitionName**):

```
<type name="tct:testCaseTask">
    <title>Test case task</title>
    <parent>icaseTask:simpleTask</parent>
    <overrides>
        <property name="icaseTask:workflowDefinitionName">
            <default>activiti$test-case-task-workflow</default>
        </property>
    </overrides>
</type>
```

Создаем файл testModel.xml и записываем туда наши типы и дополнительную информацию:

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="twf:testCaseTaskModel"
       xmlns="http://www.alfresco.org/model/dictionary/1.0">
```

```
  <imports>
    <import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d" />
    <import uri="http://www.alfresco.org/model/content/1.0" prefix="cm" />
    <import uri="http://www.citeck.ru/model/icaseTask/1.0" prefix="icaseTask"/>
    <import uri="http://www.alfresco.org/model/bpm/1.0" prefix="bpm"/>
  </imports>

  <namespaces>
    <namespace uri="http://www.citeck.ru/model/test/workflow/1.0" prefix="twf" />
    <namespace uri="http://www.citeck.ru/model/test/case-task/1.0" prefix="tct" />
  </namespaces>

  <types>
      <type name="twf:testBPMNTask">
          <!-- Размещаем описание задачи BPMN процесса -->
      </type>

      <type name="tct:testCaseTask">
          <!-- Размещаем описание задачи кейс-менеджмента -->
      </type>
  </types>

</model>
```

Bootstrap

Для того, чтобы наш новый процесс и новые модели загрузились в систему, необходимо воспользоваться возможностями bootstrap. Для этого создаем файл test-context.xml со следующим содержимым:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'
'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>

  <bean parent="workflowDeployer">
    <property name="workflowDefinitions">
      <list>
        <props>
          <prop key="engineId">activiti</prop>
          <prop key="location">alfresco/module/idocs-repo/workflow/test-case-
task-workflow/test-case-task-workflow.bpmn</prop>
           <prop key="mimetype">text/xml</prop>
           <prop key="redeploy">true</prop>
         </props>
       </list>
     </property>
     <property name="models">
       <list>
         <value>alfresco/module/idocs-repo/model/testModel.xml</value>
       </list>
     </property>
  </bean>
</beans>
```

Пути к модели и BPMN процессу должны соответствовать тем файлам, которые вы создавали.

Mapping

Для передачи параметров из задачи кейса в BPMN процесс необходимо создать mapping параметров. Для этого добавим дополнительный bean в наш context файл:

```
<bean id="test.caseTaskAttributesMappingRegistrar"

class="ru.citeck.ecos.icase.activity.CaseTaskAttributesMappingRegistrar"
      init-method="init">
  <property name="caseTaskBehavior" ref="caseTaskBehavior" />
  <property name="attributesMappingByWorkflow">
    <map>
      <entry key="activiti$test-case-task-workflow">
        <map>
          <entry key="icaseTask:performer" value="twf:performer" />
        </map>
      </entry>
    </map>
  </property>
</bean>
```

Здесь мы передаем исполнителя, который выбран в задаче кейса в ассоциацию twf:performer.

Чтобы context файл загрузился при старте системы, необходимо дополнительно добавить его в module-context.xml, который располагается по следующему пути:

{repo-module}/config/alfresco/module/{repo-module}/module-context.xml

```
<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'

'http://www.springframework.org/dtd/spring-beans.dtd'>


<beans>

    <import resource="classpath:alfresco/module/idocs-repo/context/test-context.xml" />

</beans>
```

Формы

Для создания формы BPMN задачи создадим файл testForms.xml по следующему пути:

{share-module}/config/alfresco/site-config/form/testForms.xml

В данном случае обязательным условием расположения является только папка «**site-config**» поскольку из неё автоматически загружаются любая конфигурация share.

Содержимое файла будет следующим:

```
<alfresco-config>

    <config evaluator="task-type" condition="twf:testBPMNTask">
        <forms>
            <form id="inline">
                <field-visibility>
                    <show id="twf:someTestField" />
                    <show id="twf:someBooleanField" />
                    <show id="bpm:comment" />
                    <show id="twf:completeOutcome"/>
                </field-visibility>
                <appearance>
                    <field id="twf:someTestField" />
                    <field id="twf:someBooleanField" />
                    <field id="bpm:comment">
                        <control
template="/org/alfresco/components/form/controls/textarea.ftl" />
                    </field>
                    <field id="twf:completeOutcome">
                        <control
template="/org/alfresco/components/form/controls/workflow/activiti-transitions.ftl" />
                    </field>
                </appearance>
            </form>
        </forms>
    </config>

</alfresco-config>
```

Здесь следует обратить внимание на id формы «**inline**». Это необходимо для того, чтобы форма отображалась на карточке кейса для исполнителя задачи.

Для создания формы задачи кейс-менеджмента создадим файл test-case-task-views.xml по следующему пути:

{repo-module}/config/alfresco/module/{repo-module}/views/test-case-task-views.xml

Содержимое файла будет следующим:

```
<?xml version="1.0" encoding="UTF-8"?>
<views xmlns="http://www.citeck.ru/ecos/views/1.0">
  <imports>
    <import uri="http://www.citeck.ru/model/icaseTask/1.0" prefix="icaseTask"/>
    <import uri="http://www.citeck.ru/model/test/case-task/1.0" prefix="tct" />
  </imports>

  <view class="tct:testCaseTask">
    <view class="icaseTask:task"/>
    <field assoc="icaseTask:performer"/>
  </view>
</views>
```

Запускаем систему, открываем кейс, нажимаем «+» в кардлете «Конструктор процесса кейса» и в подменю Task наблюдаем нашу новую задачу (рисунок 82).

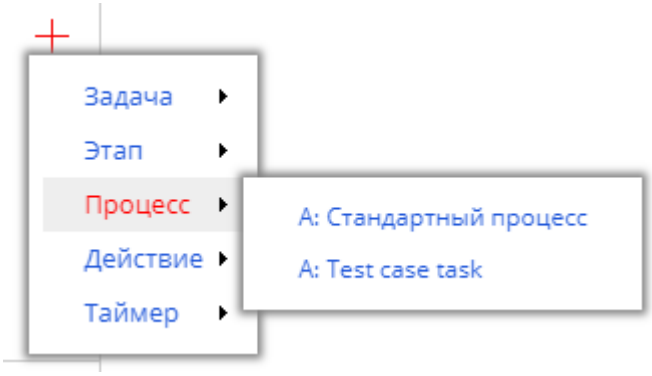


Рисунок 82 – Создание новой задачи кейса

Открываем форму для создания этой задачи. На рисунке 83 изображена открытая форма создания и помечены части формы в соответствии с **test-case-task-views.xml.** В верхней части формы отображаются поля, которые предопределены для **icaseTask:task.** Внизу находится дополнительное поле **icaseTask:performer**, которое необходимо для выбора исполнителя задачи.

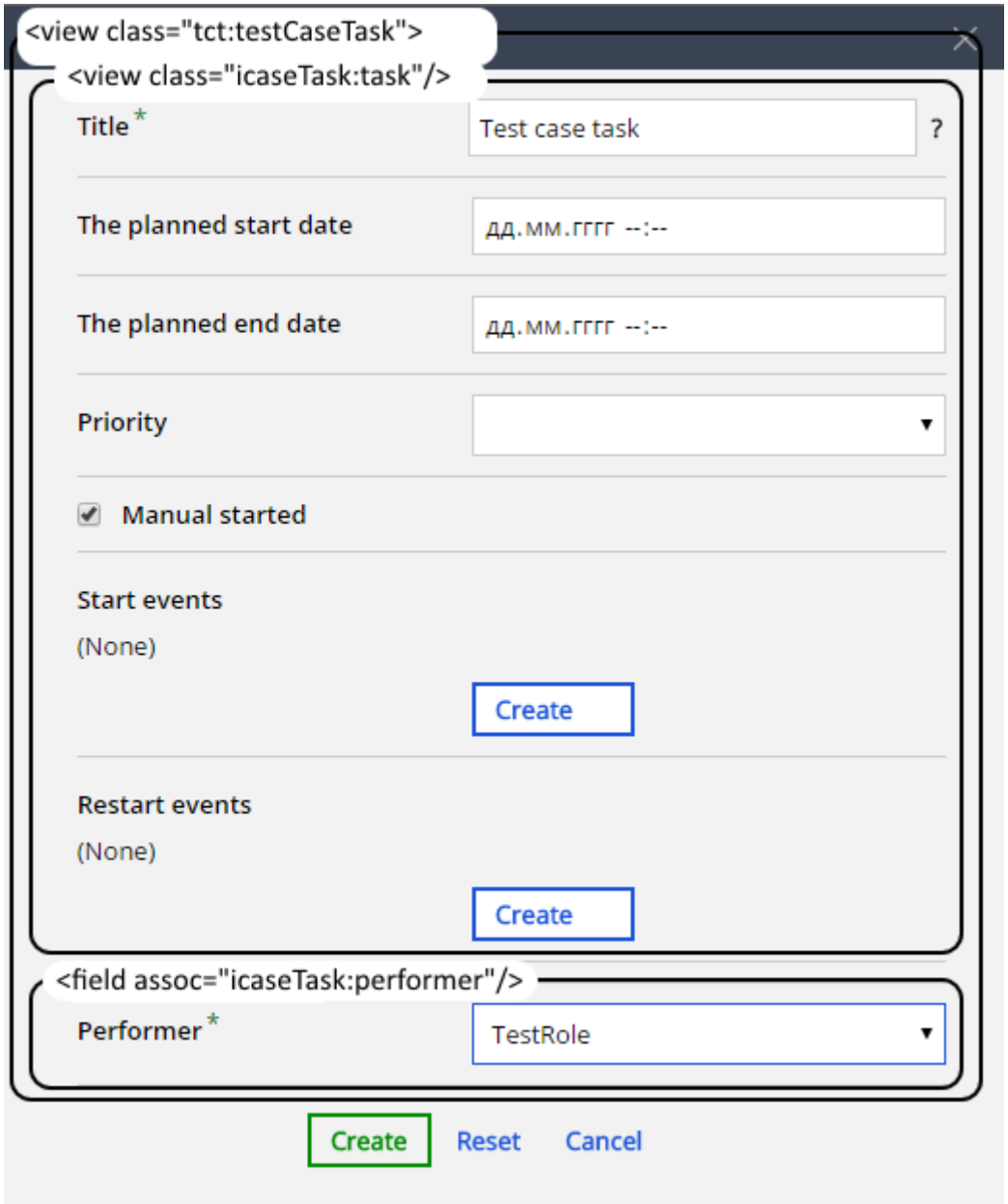


Рисунок 83 – Форма для создания задачи

Завершаем создание новой задачи нажатием кнопки «Create» и запускаем её. В кардлете «Document Tasks» видим форму, которую мы описывали в testForms.xml (рисунок 84).

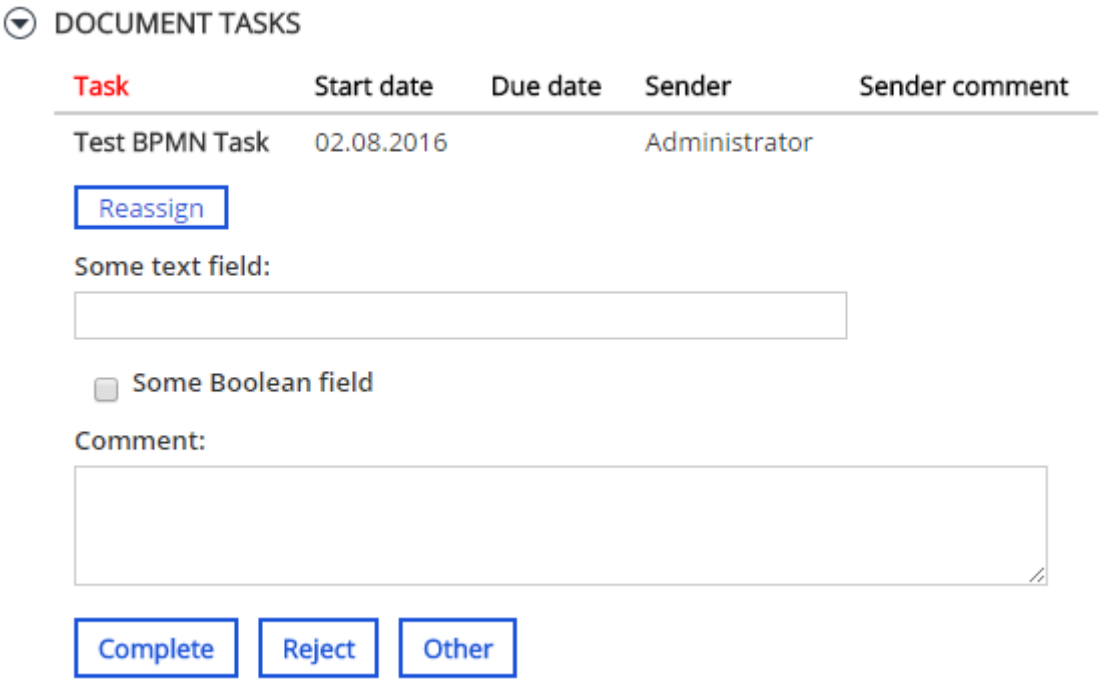


Рисунок 84 – Форма BPMN задачи на карточке документа

На этом этапе можно считать создание новой задачи для кейс-менеджмента завершенным.

citeck

УПРАВЛЯЙТЕ БИЗНЕС-ПРОЦЕССАМИ ЭФФЕКТИВНО

С КОМПАНИЕЙ CITECK

INFO@CITECK.RU
8 800 500-85-36