ООО «АйТи-Альянс»
http://www.it-alnc.ru/
ИНН 7718755096 / КПП 772501001
ОГРН 1097746108745

Инструкция по эксплуатации модуля ITA DESIGNER программного обеспечения ITA::FORMS («Дизайнер форм и бизнес-процессов»))

Москва, 2025

IT-ALLIANCE

Оглавление

Модуль **ITA Designer** предназначен для быстрой настройки бизнес-процессов и экранных форм пользовательских интерфейсов. Модуль предоставляет следующие возможности:

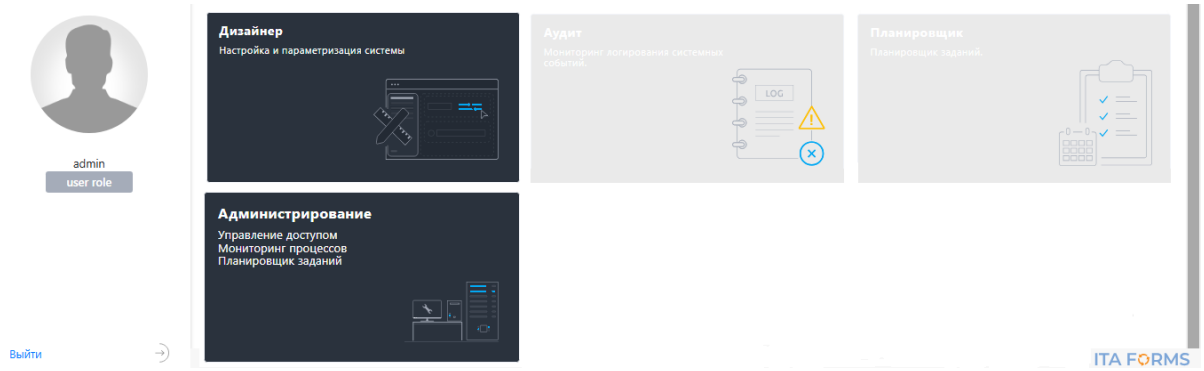
- Удобное конструирование экранных форм и пользовательских задач для процессов в drag-and-drop редакторе
- Быстрой и эффективной интеграции процессов с ИТ системами организации
- Настройка процессов в нотации BPMN в удобном редакторе

1. Начало работы

1.1. Вход в приложение

Для входа в Приложение необходимо перейти по ссылке, предоставленной пользователям.

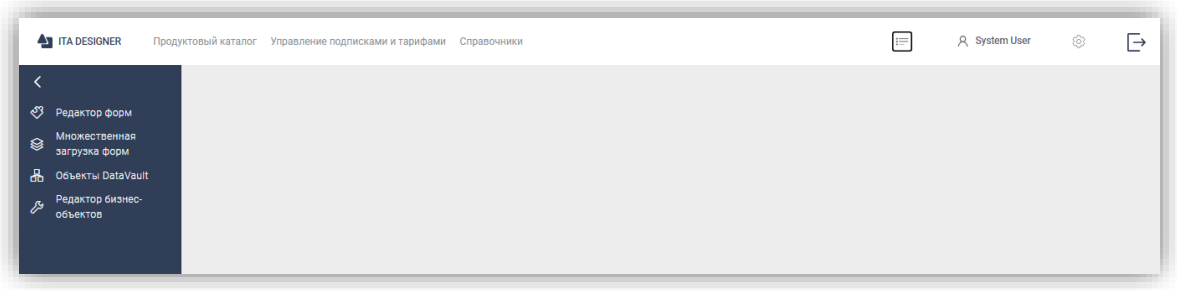
Далее необходимо пройти авторизацию. При успешном прохождении на экране отобразится основное окно системы, откуда возможен вход в различные модули.



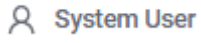
Далее необходимо перейти в модуль «Дизайнер».

1.2. Рабочая область, область навигации, разделы приложения

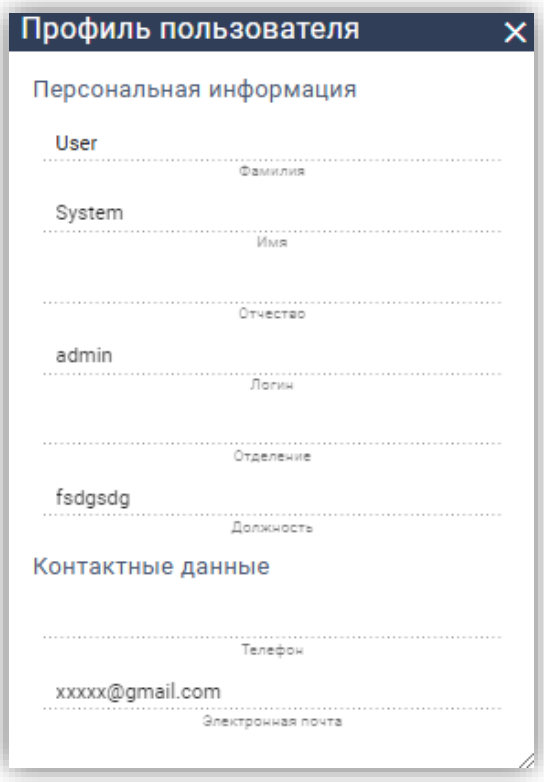
При входе в Приложение открывается Рабочая область, слева - Область навигации:



В Области навигации Меню с названиями разделов Приложения раскрывается и скрывается по стрелке  . Начать работу в каждом разделе можно из иконок разделов, не раскрывая меню.

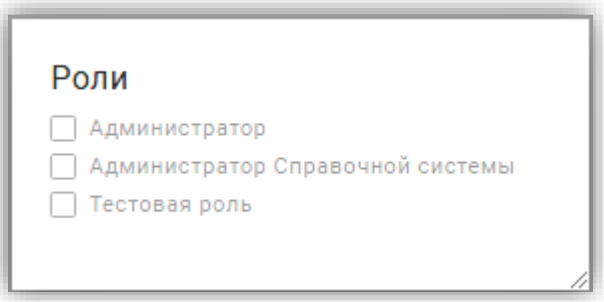
Кнопка 

служит для отображения информации о пользователе, который подключился к системе. При клике на иконку  открывается общая информация о пользователе:

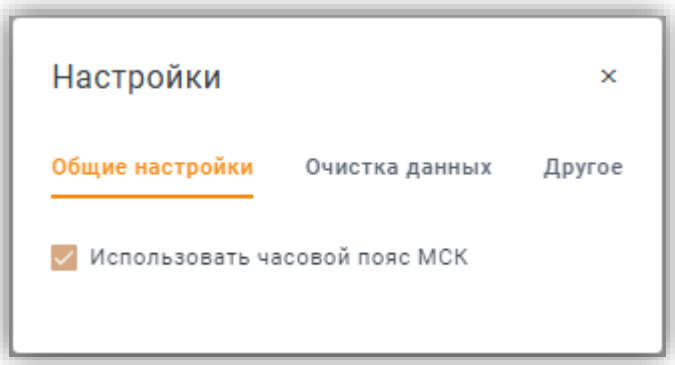


При клике на имя пользователя - 

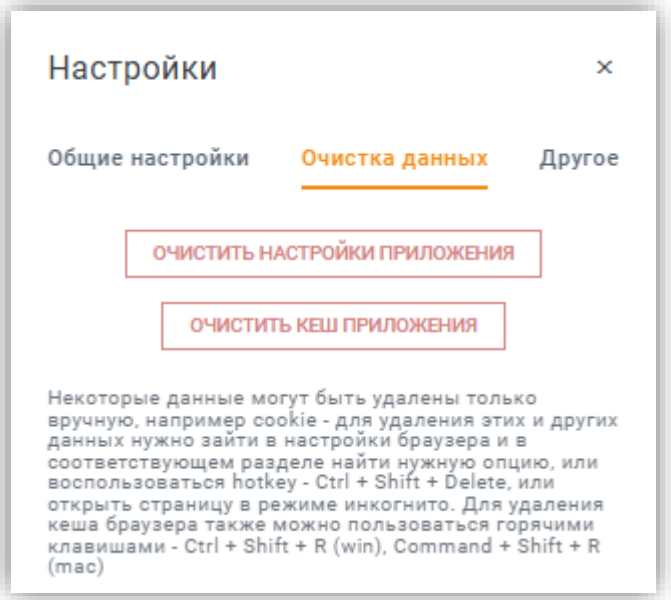
- открывается окно с ролями, которые есть у подключенного пользователя:



При клике на иконку  открывается меню настроек приложения:



В закладке «Общие настройки» можно установить использование часового пояса МСК для приложения, даже если рабочее место пользователя находится в другом часовом поясе.
В закладке «Очистка данных» можно сбросить настройки приложения и кеш приложения.



Кнопка ОЧИСТИТЬ НАСТРОЙКИ ПРИЛОЖЕНИЯ нужная для того, чтобы сбросить пользовательские настройки таблиц, реестров, наборы фильтров до дефолтных (то есть таких, которые были сделаны в редакторе при создании формы); удалить настройки редактора, а также общие настройки приложения.

Кнопка ОЧИСТИТЬ КЕШ ПРИЛОЖЕНИЯ нужна для того, чтобы удалить закешированные справочники, шаблоны, списки HLS объектов, структуры фильтров, списки операций, списки аргументов операций, список классов. Важно: это не кеш браузера!

Кнопка  вернет пользователя в основное окно системы со списком модулей.

Разделы приложения:

- Старт процессов – в данном разделе пользователь может вручную запустить доступные бизнес – процессы;
- Редактор бизнес – процессов – в данном разделе пользователь может создавать и редактировать схемы бизнес – процессов, а также загружать готовые bpmn-схемы процессов;
- Редактор форм – в данном разделе пользователь может создавать экранные формы разной сложности;
- Множественная загрузка форм – в данном разделе пользователь может выгружать формы и схемы бизнес – процессов в виде файлов, для последующей загрузки их на других стендах;
- Объекты DataVault – список бизнес-объектов системы в виде модели Data Vault (хабы, сателлиты и связи-линки);
- Редактор бизнес – объектов

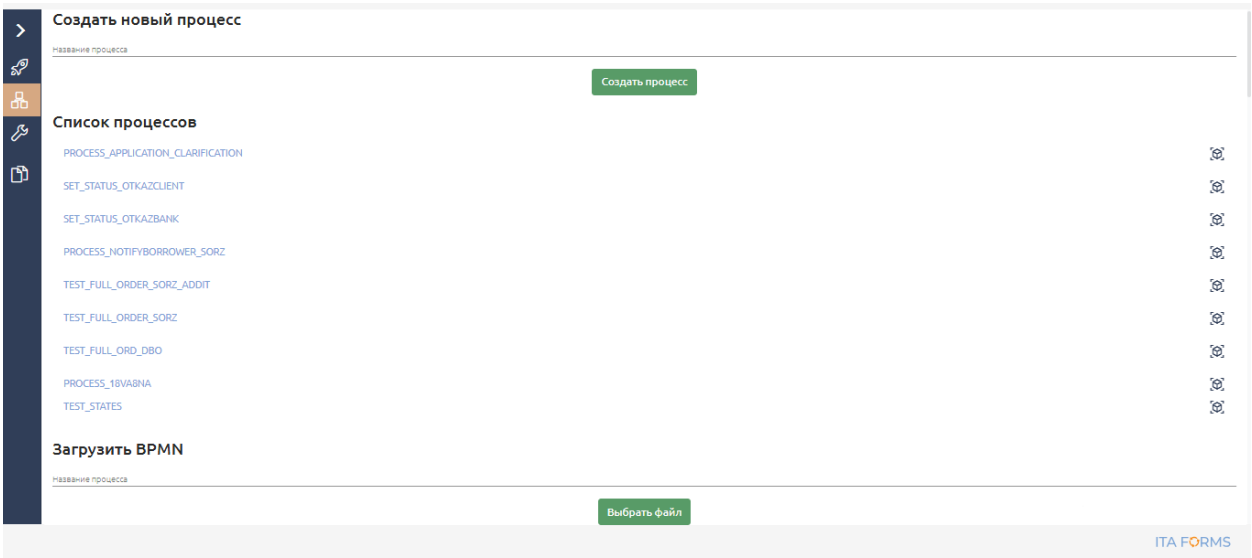
2. Редактор бизнес – процессов

В редакторе бизнес – процессов пользователь с помощью графического интерфейса может настроить схему бизнес – процесса в нотации BPMN, настроить свойства процесса и задач, необходимые для технического исполнения, выложить созданную схему на сервер выполнения бизнес-процессов на основе Camunda.

Также редактор процессов позволяет связывать экранные формы, которые конструируются в редакторе форм, с user-task-ами бизнес – процесса.

Доступ к Редактору бизнес-процессов выполняется из меню «Редактор процессов» приложения, пользователь нажимает на кнопку  в левом меню.

При переходе в редактор процессов пользователь попадает в список доступных бизнес-процессов системы:



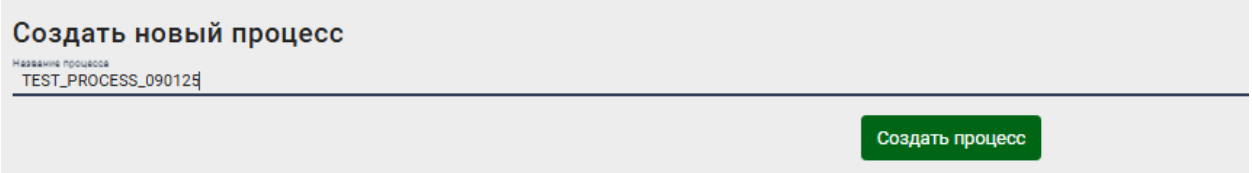
Пользователь может:

- Создать новый процесс – по кнопке «Создать процесс» в верхней части экрана;
- Редактировать процесс из списка – по клику на имя процесса в списке;

- Загрузить заранее подготовленную BPMN-схему процесса – по кнопке «Выбрать файл» в нижней части экрана.

2.1. Создание нового бизнес-процесса

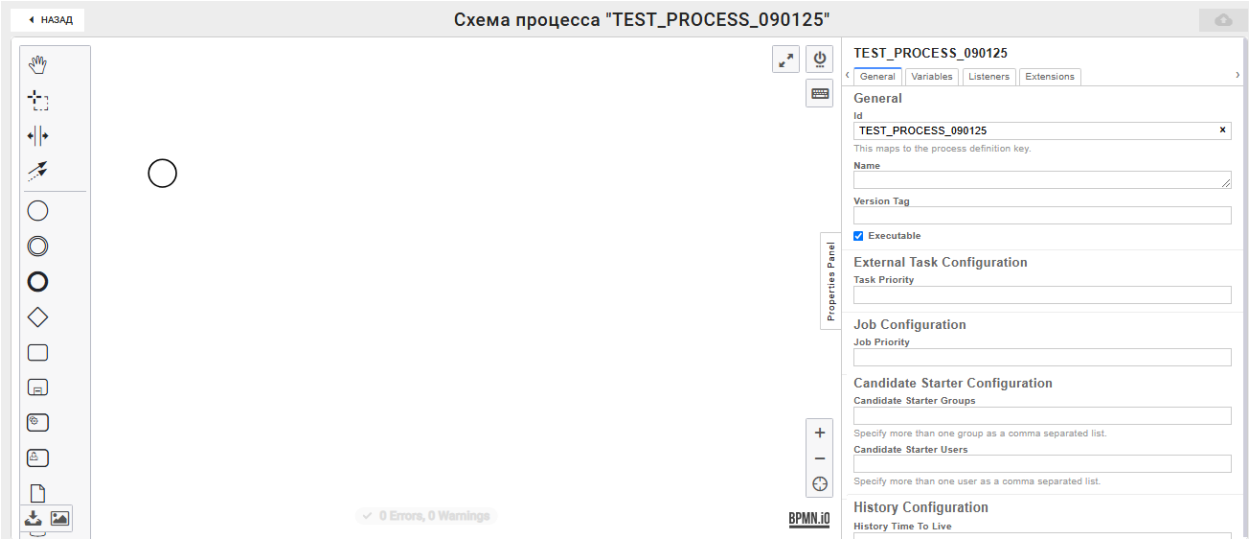
Для создания нового бизнес-процесса пользователь сначала задает имя процесса в области создания нового процесса:



При этом кнопка «Создать процесс» становится активной, и пользователь может нажать на нее для продолжения работы.

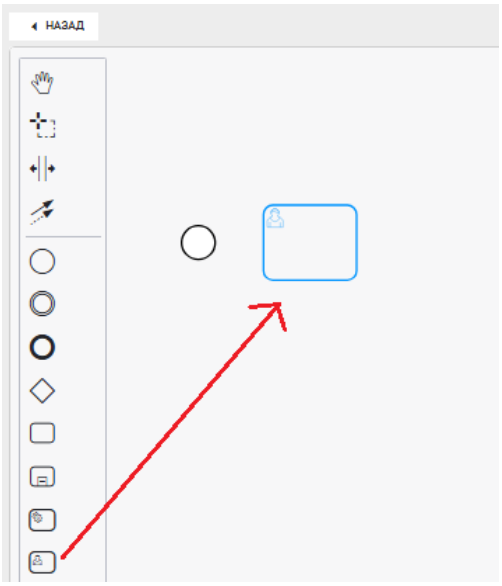
После нажатия на кнопку «Создать процесс» пользователю открывается графический редактор.

Основное рабочее окно редактора процессов выглядит следующим образом:



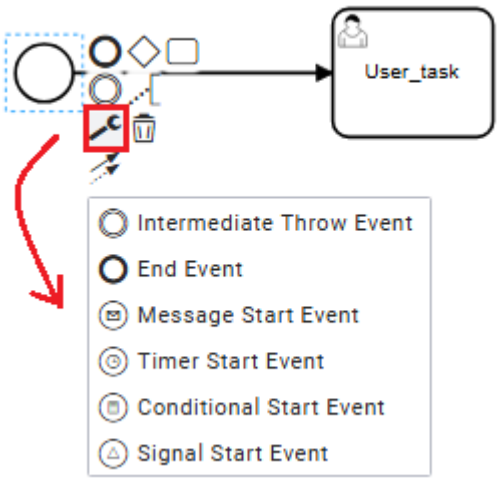
Слева расположена панель базовых элементов:

Пользователь может мышкой перетащить какой-либо элемент из панели в область диаграммы (центральная часть экрана):



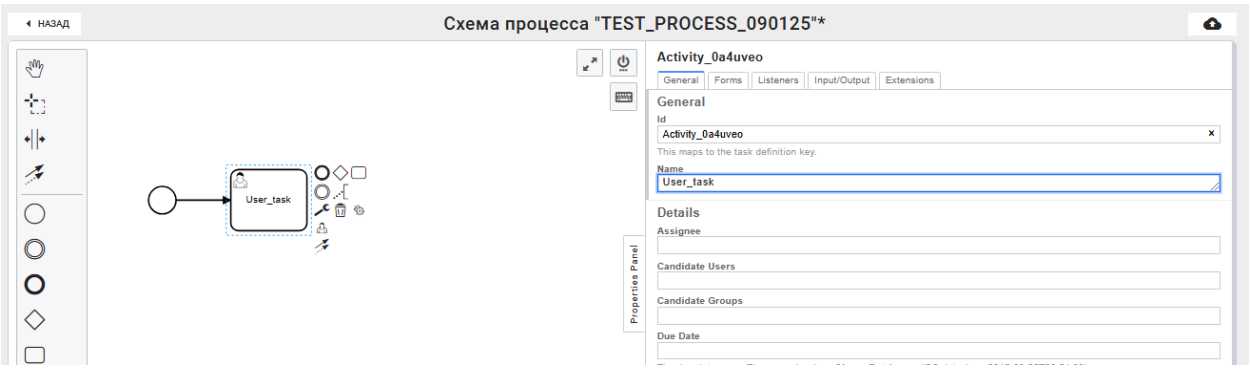
И далее настроить этот элемент.

Из базового элемента можно получить расширенный элемент путем преобразования его в диаграмме. Для этого пользователю необходимо встать мышкой на элемент схемы и нажать на кнопку инструментов . При этом откроется меню с выбором расширений для базового элемента:



Из этого же контекстного набора элементов пользователь может выбрать следующие элементы бизнес-процесса, вместо того, чтобы перетягивать их из палитры и затем связывать.

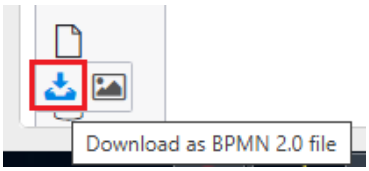
Панель свойств расположена справа от области диаграммы. В этом разделе производится настройка свойств как самого процесса BPMN, так и его элементов.



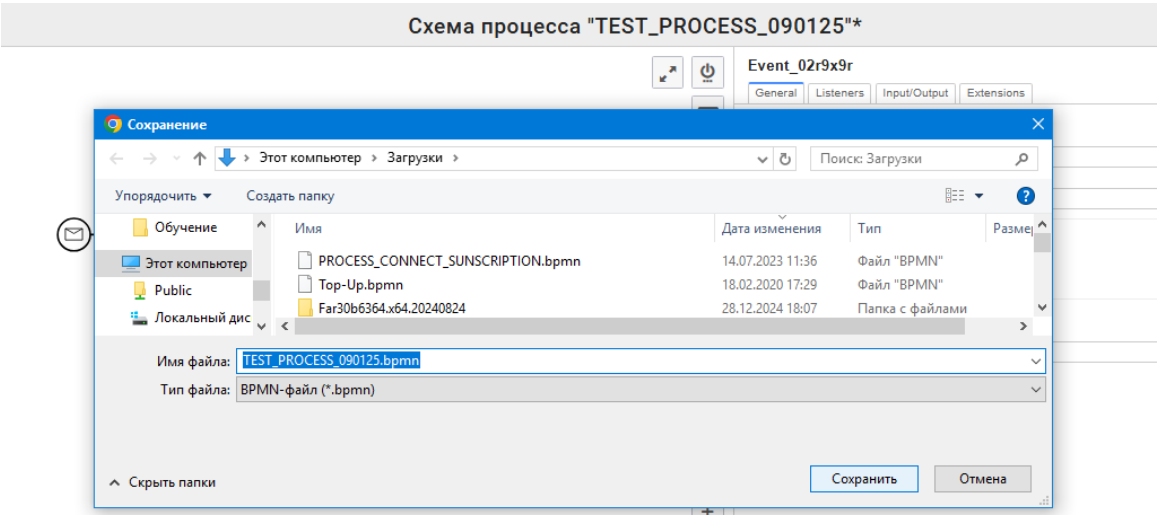
После того как схема процесса создана и настроена, ее можно:

- Сохранить локально в виде файла .bpmn
- Выгрузить на сервер выполнения бизнес-прооцессов

Чтобы локально сохранить схему в виде файла .bpmn , пользователю нужно нажать на кнопку выгрузки в файл в нижней части панели элементов:



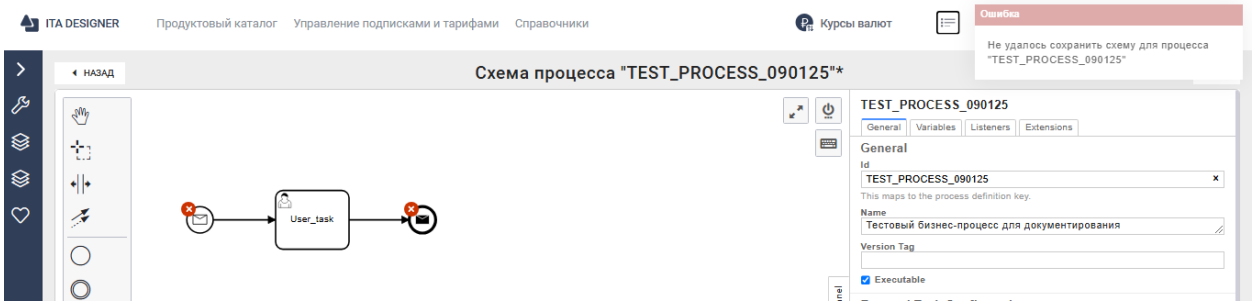
Пользователю будет предложен выбор пути сохранения:



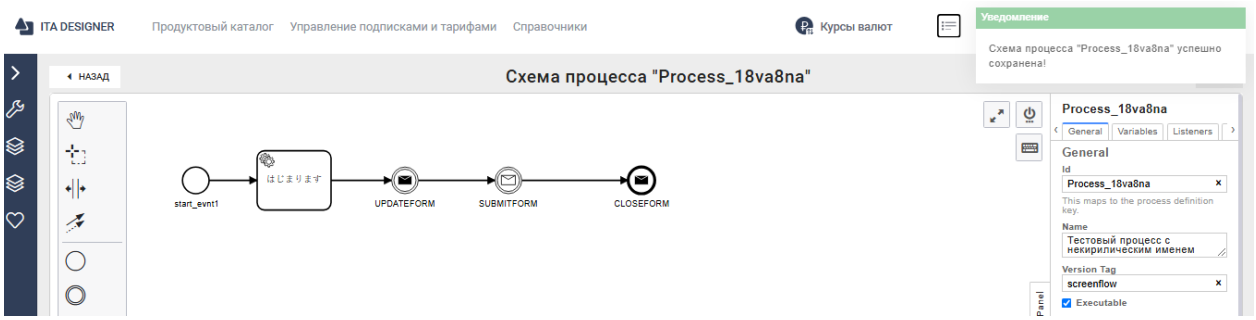
Сохраненный файл .bpmn далее можно загрузить на другом стенде.

Чтобы выгрузить созданную схему на сервер Camunda (эта процедура называется деплоем), пользователю необходимо нажать на кнопку деплоя  , которая расположена в правом верхнем углу экрана редактора.

При выполнении деплоя процесса запускается его валидация. Если обнаружены ошибки в процессе, то процесс не будет задеплоен на сервер, и пользователю будет выдана ошибка



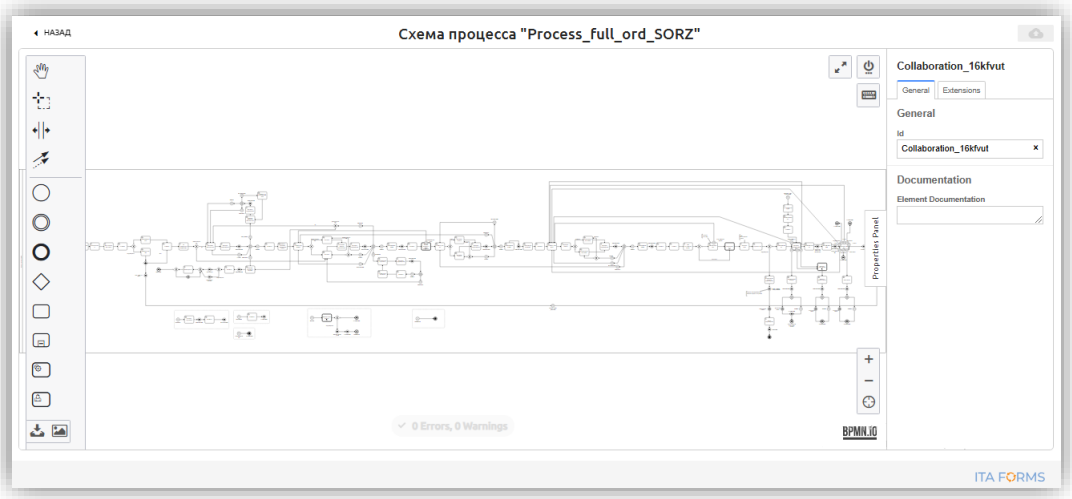
При процесс валиден, то он успешно задеплоится, и пользователю будет выдано сообщение об успешном сохранении схемы:



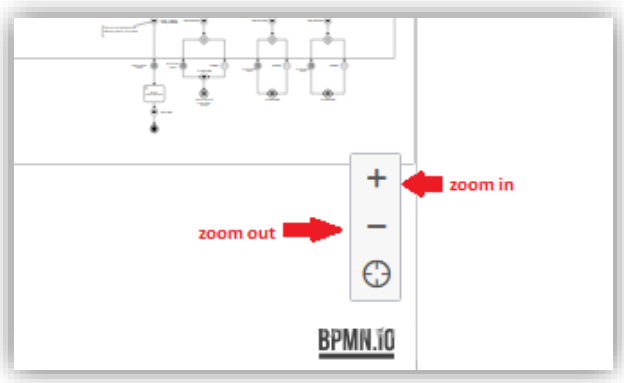
2.2. Редактирование бизнес-процесса

Чтобы отредактировать какой-либо из существующих в системе бизнес-процессов, пользователю необходимо в списке существующих бизнес-процессов кликнуть мышкой по наименованию процесса:

Откроется редактор процессов со схемой выбранного бизнес-процесса:

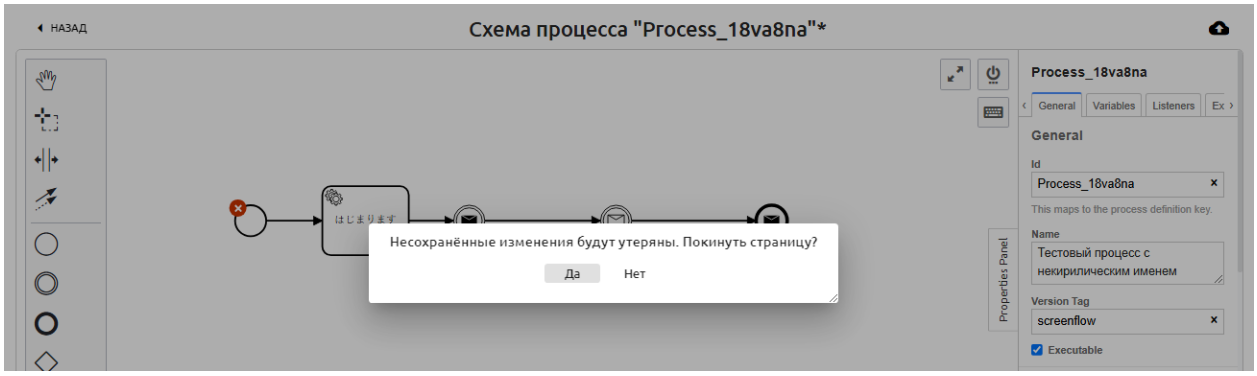


Реальные рабочие процессы могут очень большими по размеру и включать в себя большое количество элементов. Для работы с большими схемами процессов можно увеличить или уменьшить отдельный участок схемы, для этого в редакторе есть кнопки зума, расположенные в области диаграммы процесса:



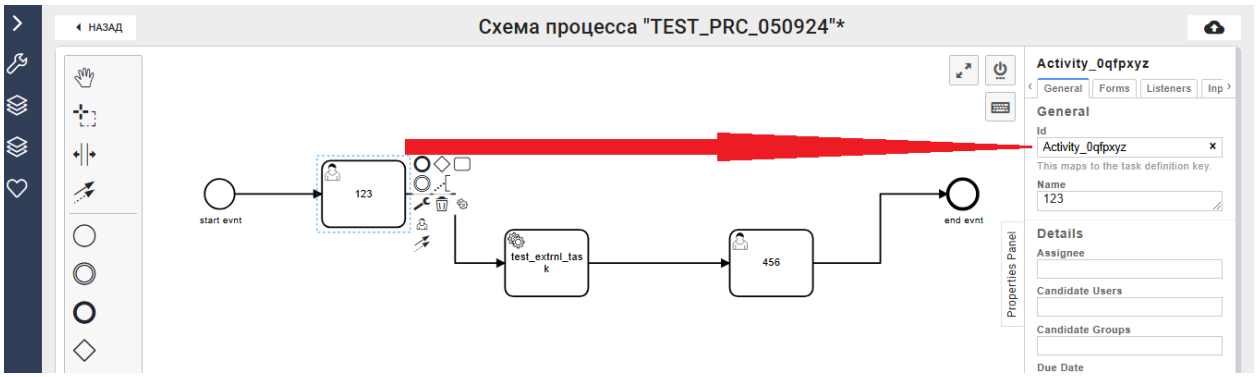
После редактирования схемы пользователь может вернуться назад к списку процессов. Для этого пользователю нужно нажать на кнопку ◂ НАЗАД в верхней части редактора

процессов. При этом если изменения не были сохранены, по пользователю будет выдано следующее предупреждение:

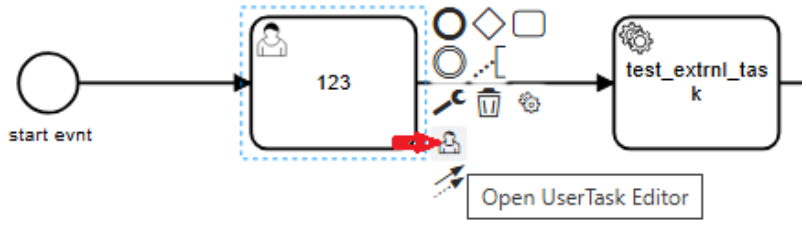


2.3. Привязка формы, создаваемой в редакторе форм, к задаче бизнес-процесса

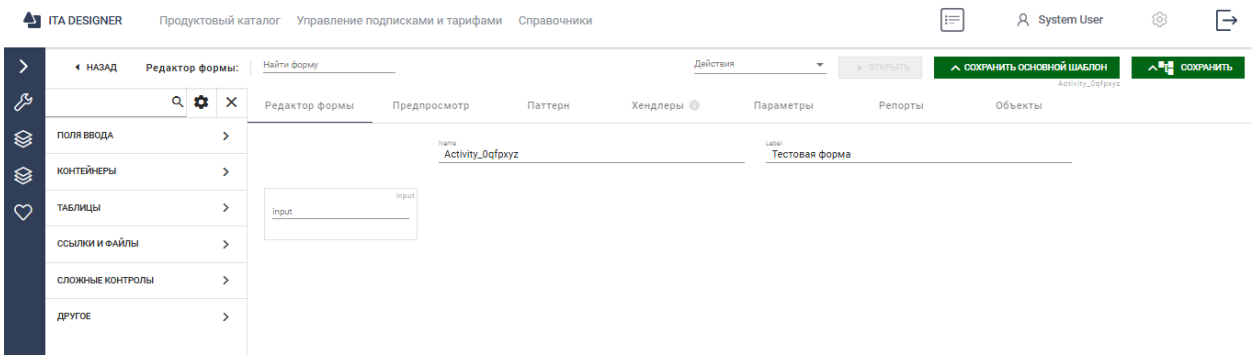
Для привязки формы, которую сделали с помощью редактора форм, к User task бизнес-процесса нужно в основных свойствах задачи в Id указать код Name нужной формы.



У User Task-а есть специальная кнопка , с помощью которой можно открыть привязанную форму в редакторе.



Форма открывается на редактирование



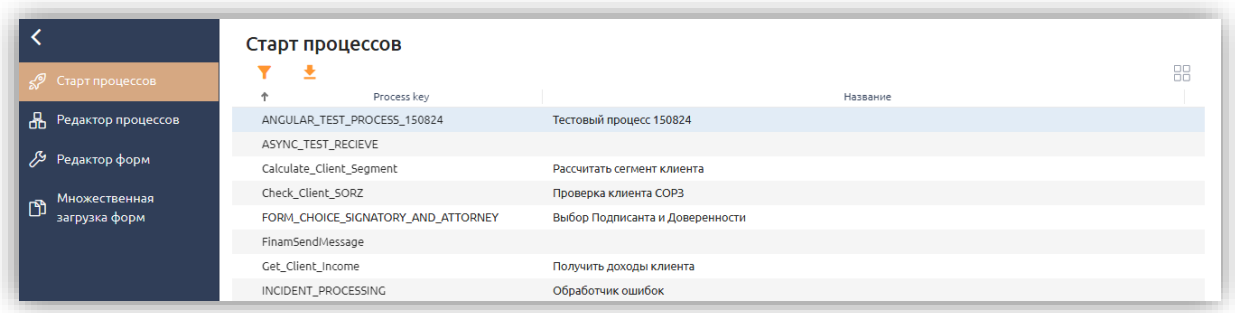
Если же формы еще не существует, то таким же образом с помощью кнопки пользователь может создать форму в редакторе процессов к задаче User Task бизнес-процесса.

3. Старт бизнес – процессов

После того, как пользователь настроил и задеплоил схему бизнес – процесса, при необходимости привязал созданные в редакторе форм экранные формы к пользовательским задачам, такой процесс можно запустить на исполнение.

Процесс может быть запущен как вручную, так и автоматически, если он встроен в какой-нибудь другой процесс.

Ручной запуск происходит из меню процессов. Пользователь выбирает нужный процесс и кликает на него.



При запуске бизнес – процесса для него автоматически генерируется уникальный идентификатор businesskey, далее по этому идентификатору можно отслеживать выполнение.

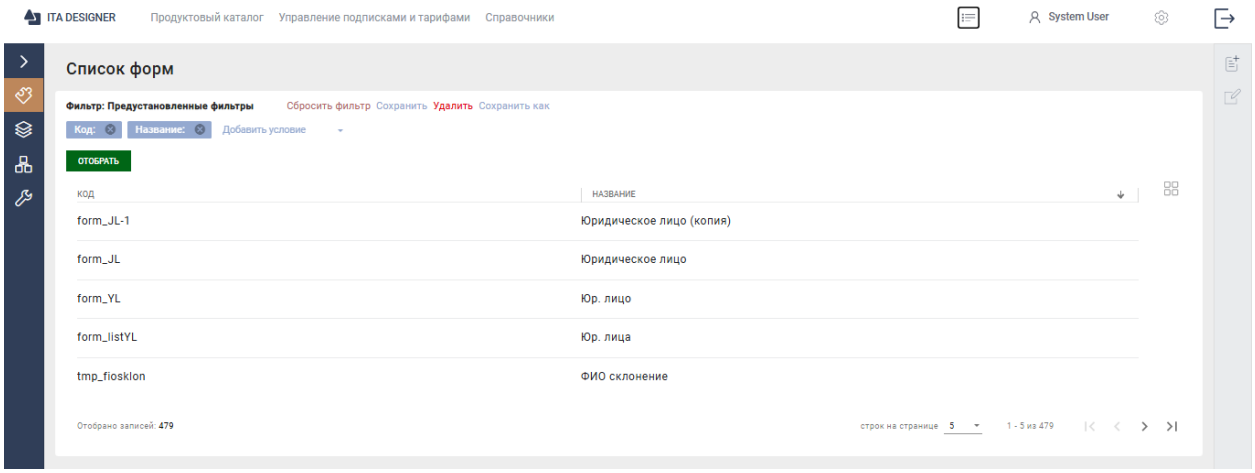
4. Редактор форм

В редакторе форм пользователь может создавать экранные формы, которые в дальнейшем можно вызывать как в бизнес – процессах – в пользовательских задачах User tasks, так и вне процессов. Drag-and-drop интерфейс редактора позволяет перетаскивать элементы дизайна на веб-странице и изменять их положение или порядок.

Форма – совокупность полей на экране. У формы есть параметры и входящие в неё поля. Форму можно поместить в другую форму, как поле. Формы также используются как

шаблоны (template) для форматирования строк таблиц и других полей с множественными значениями.

В редактор форм пользователь попадает, нажав на кнопку  в левом меню. Пользователю открывается реестр форм:

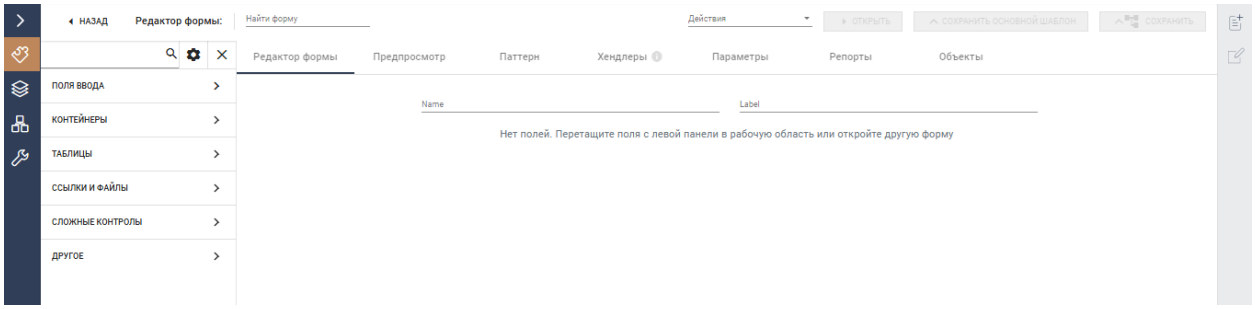


В реестре форм отображаются все существующие в системе формы. Пользователь может найти нужную форму с помощью фильтров.

Для создания новой формы пользователю нужно нажать на кнопку  в правой панели. Откроется основное рабочее окно редактора форм. Для редактирования существующей

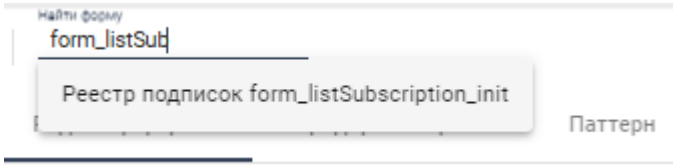
формы пользователю необходимо нажать на кнопку  в правой панели, либо дважды кликнуть на найденную форму в реестре форм.

Основное рабочее окно редактора форм выглядит следующим образом:

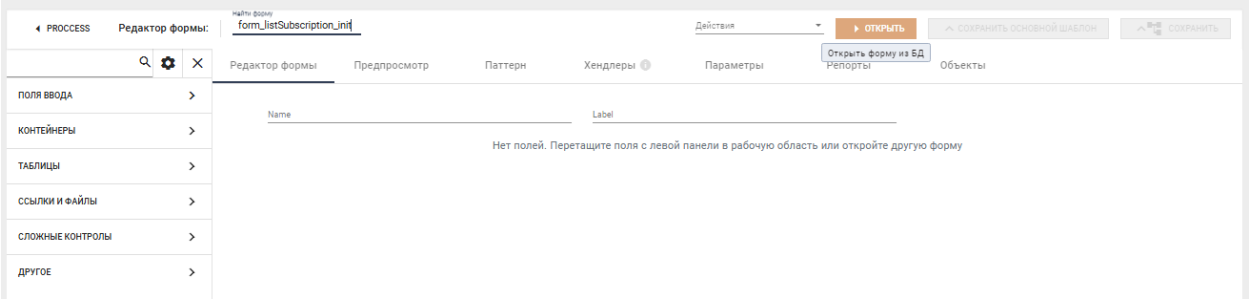


В верхней части рабочего окна редактора располагается поисковая строка:

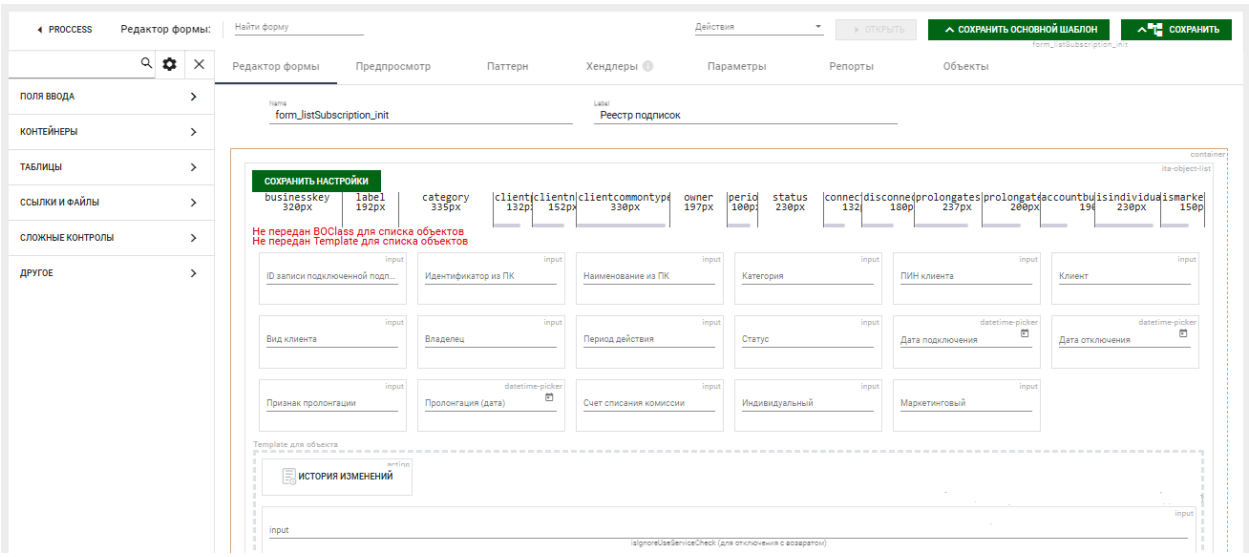
В поисковой строке можно найти форму из существующих, задав название формы (label) или код формы (Name). При этом если такая форма существует, то она отобразится в выпадающем списке:



При выборе формы из списка активизируется кнопка «Открыть», при нажатии на которую выбранная форма откроется на редактирование.



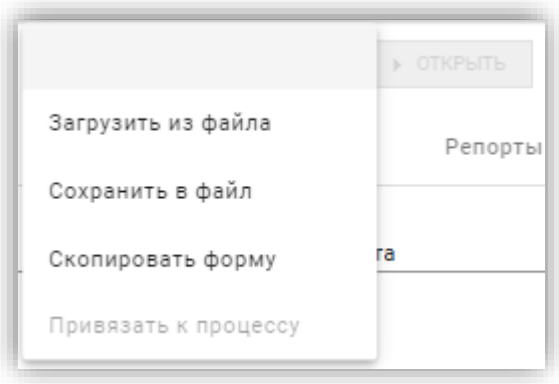
Открытая форма, которую можно редактировать:



Рядом с кнопкой открытия находятся две кнопки сохранения формы: «Сохранить основной шаблон» и «Сохранить». Их функциональность описана в разделе «Сохранение формы».

Рядом с функциональными кнопками открытия и сохранения находится меню действий:

При разворачивании меню открывается список доступных действий с формой:



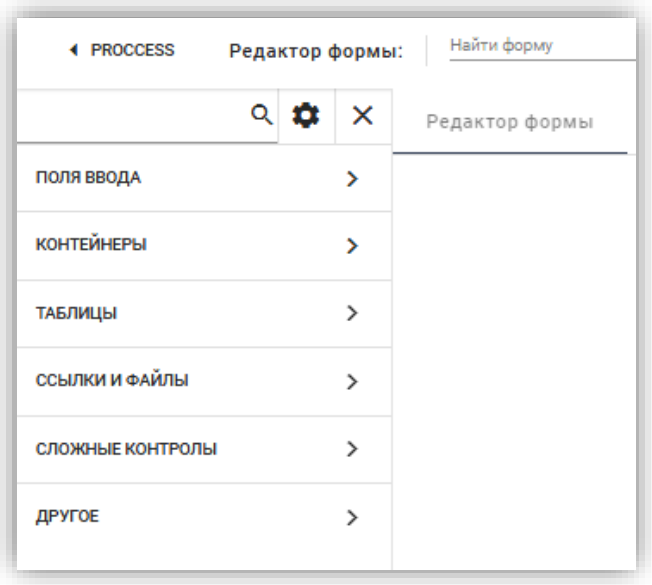
Доступны следующие действия с формой:

- **Загрузить из файла** – позволяет загрузить в форму сохраненный ранее в виде файла .txt / .json шаблон
- **Сохранить в файл** – позволяет сохранить открытую текущую форму в виде файла txt / .json
- **Скопировать форму** – позволяет скопировать форму со всеми вложенными формами (кроме форм, открываемых хендлерами). Получившаяся копия не сохранена в БД, поэтому её нужно сохранить чтобы не потерять изменения.
- **Привязать к процессу** – позволяет связать форму с пользовательской задачей в некотором бизнес – процессе.

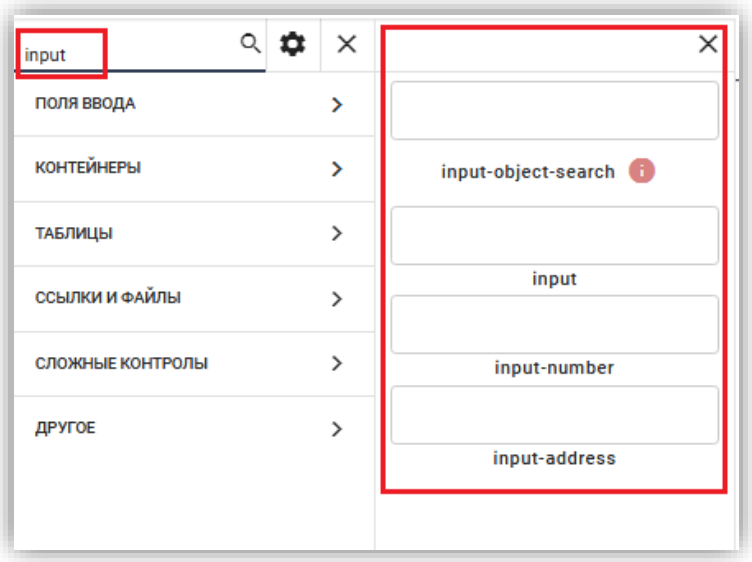
Под функциональными кнопками и меню действий располагается панель с закладками редактора:

В закладках находятся рабочие инструменты редактора.

В левой части рабочей области редактора располагается меню контролов (компонент редактора, с помощью которых реализуется дизайн формы).



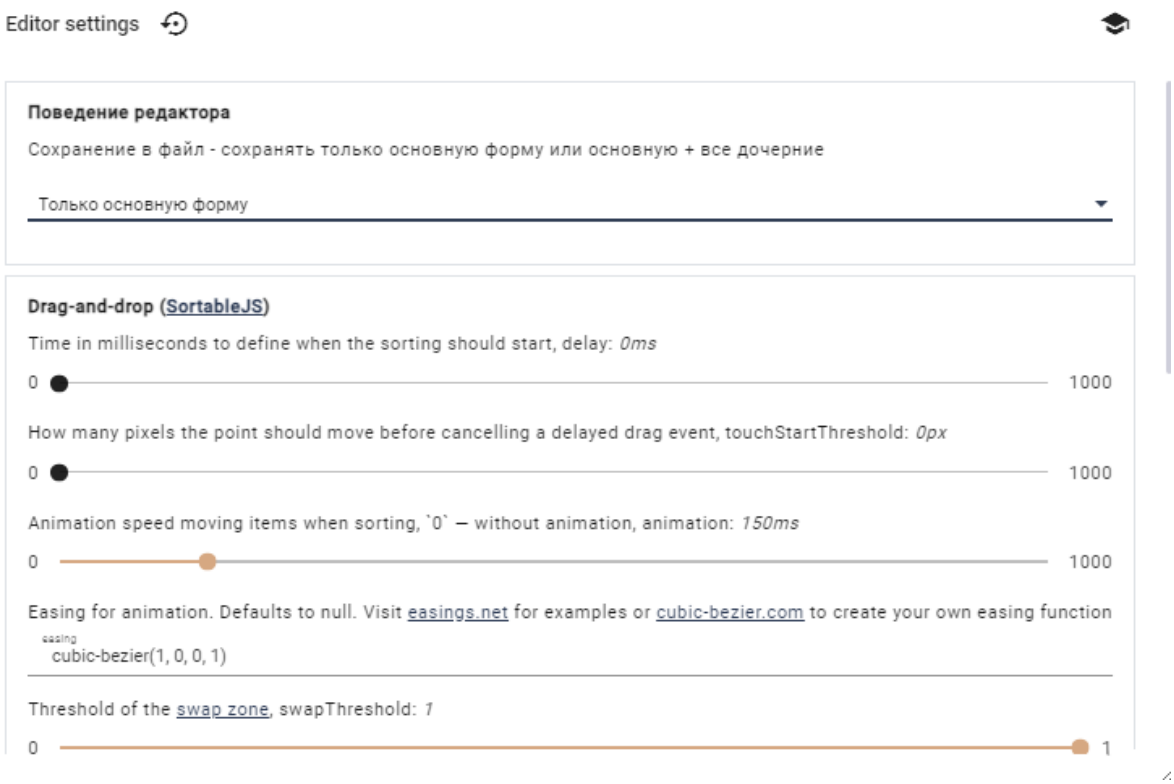
Поисковая строка  позволяет найти какой-либо контрол по наименованию:



Найденный контрол можно перетащить мышкой на рабочую область в закладке «Редактор». Таким образом контролы добавляются на форму.

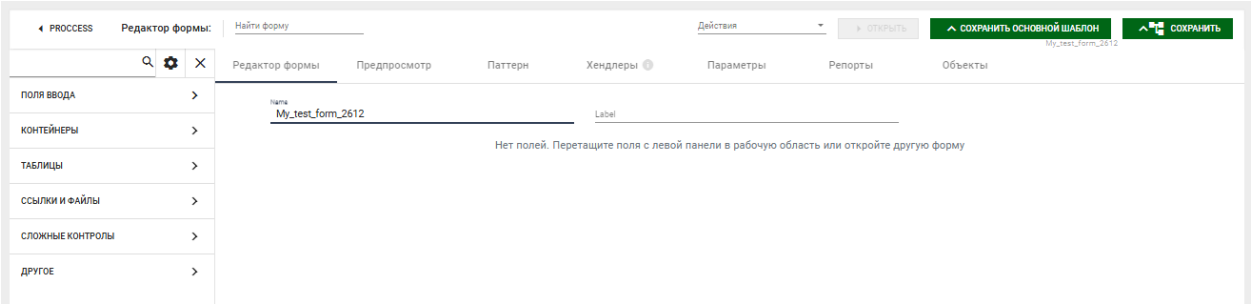
Свернуть меню контролов можно кнопкой ✕ .

По кнопке ⚙ открывается окно настроек редактора:

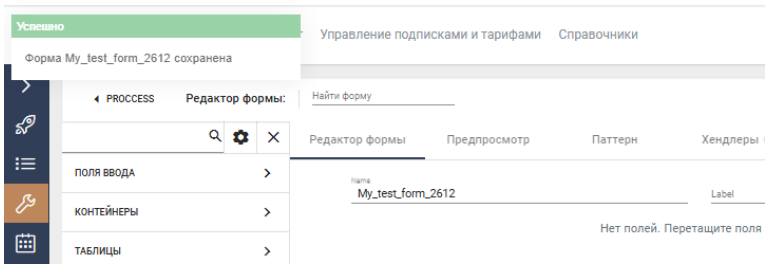


5. Создание и редактирование формы

При нажатии на кнопку  в реестре форм пользователю открывается новое окно редактора форм. Для создания новой формы нужно задать уникальный код формы в поле Name, поле Label можно заполнять или не заполнять, далее нажать на кнопку «Сохранить основной шаблон».



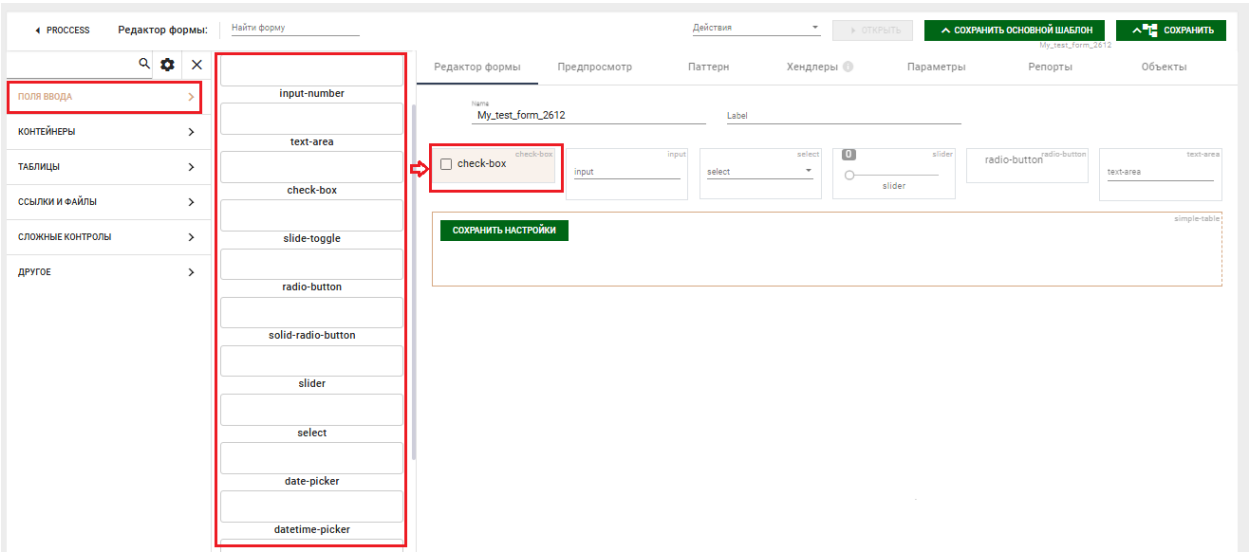
При этом пользователю на экран выводится уведомление об успешном сохранении формы в базе данных:



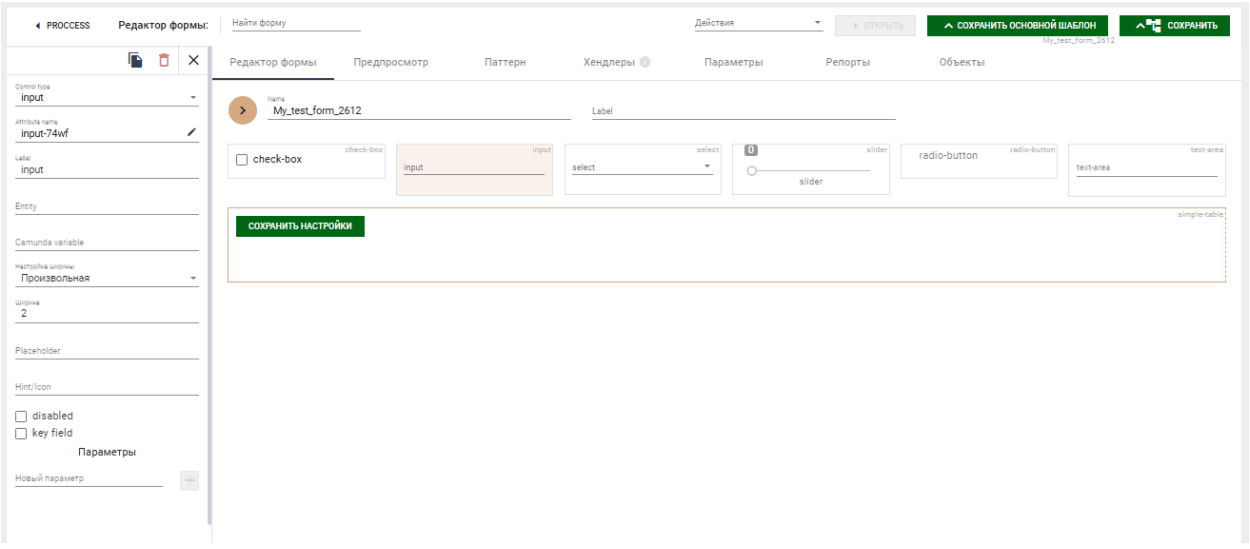
Таким образом будет создана новая пустая форма, на которую можно добавлять различные элементы дизайна.

Редактирование формы

Для добавления новых элементов дизайна (также называются контролами) на форму нужно перетянуть нужный элемент (контрол) из списка элементов на экран формы.



Добавляется новый элемент, для которого необходимо задать атрибуты. Для задания атрибутов нужно встать на элемент мышкой, при этом слева откроется панель управления атрибутами поля формы.



Обязательными атрибутами, без указания которых контрол на форме не сохранится, являются:

- **Attribute name** – идентификатор поля формы; здесь указывается *путь к данным внутри HLS-данных* в формате hub.{link[x].hub.}sat.attr. Путь должен однозначно соответствовать единственному значению в наборе данных, полученному из источника данных. Пример доступа к данным в простом поле input: hub_MPShowcase.link_MPVersion_MPShowcase[0].hub_MPVersion.sat_MPVersion_Main.Description

- **Ширина (в связке с Настройка ширины)** – ширина отображения поля на странице;
- **Control type** – тип контрола;

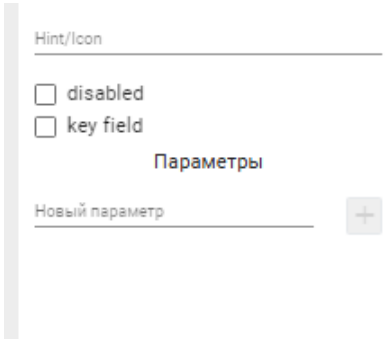
Также, у контролов (кроме сложных кастомных) есть обязательный скрытый атрибут **fieldType**, который определяет, для каких данных предназначен контрол – для простых, содержащих значение одного атрибута данных, или для множественных объектов данных. У этого атрибута два значения:

- **Array_type** – для отображения таблиц, аккордеонов и тп., то есть таких, которые содержат множественные значения данных (например, поле типа Array - таблица, содержимое поля - значения в строках таблицы). Помимо параметров для отрисовки на экране также имеют еще и привязанную к ним форму-шаблон для форматирования строк данных. В свою очередь, формы строк данных также могут содержать поля типа Array со своими шаблонами и т.д.
- **String_type** – для полей ввода, текстов, и тп., то есть таких, которые содержат значение одного атрибута данных. Простые поля не могут иметь внутри себя других полей или виджетов.

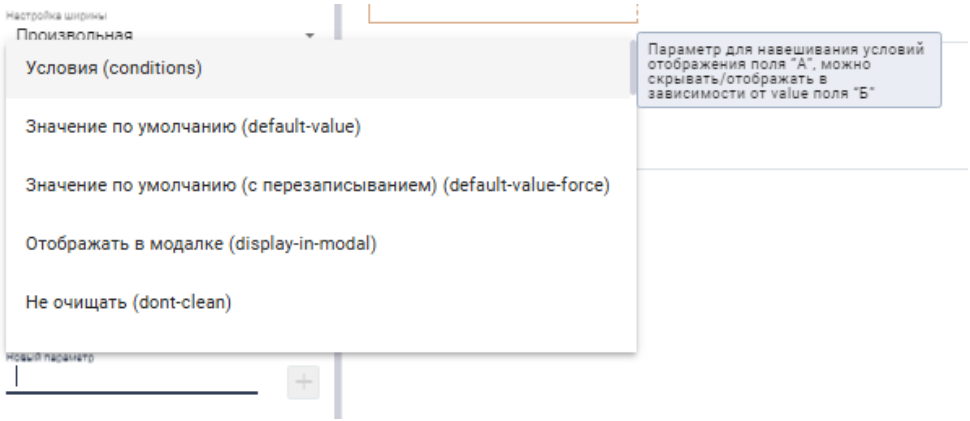
Другие атрибуты могут стать обязательными в сочетании с какими-то определенными полями. Так, например, при создании таблицы необходимо заполнять и поле Template name.

- **Label** – отображаемое название поля формы (то, что будет отображаться на экране в качестве подписи к полю);
- **Переменная** – переменная Camunda, связанная с полем формы, которая передается в бизнес-процесс;
- **Placeholder** – текстовая подсказка-заполнитель, для полей вида input, text-area;
- **Test formulae** – для контрола text позволяет задать формулу преобразования значения (можно указать значение другого поля в соответствии с правилами построения выражений);
- **Color** – для кнопок action, позволяет выбрать цвет и рамку кнопок из списка;
- **Hint/icon** - позволяет назначать иконки некоторым полям, например, кнопкам; требуется указать код иконки; поддерживаются как коды из Angular Material Icons, так и кастомные из собственной библиотеки;
- **Items** – опции, позволяет задавать константные значения для контролов, предполагающих выбор из нескольких значений;
- **Disabled** – позволяет сделать поле неактивным / нередактируемым;
- **key field** – признак ключевого поля;
- **Template name** – указывается имя шаблона, который вызывается в экранной форме (для таблиц, массивов, аккордеонов, некоторых кастомных контролов);
- **Title** – здесь задается формула для заголовка элемента аккордеона;
- **Description** – здесь задается формула для описания элемента аккордеона;

К каждому контролу можно задать определенные параметры из списка параметров, который находится под основными атрибутами.

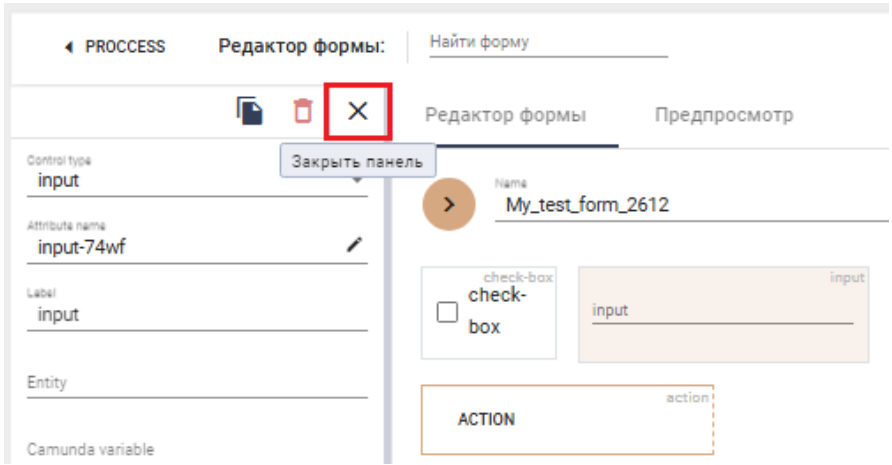


Для открытия списка доступных для контрола параметров нужно кликнуть мышью в поле «Новый параметр»:



Возможные параметры контрола определяются его типом. Какому контролу какие параметры доступны – см. в приложении.

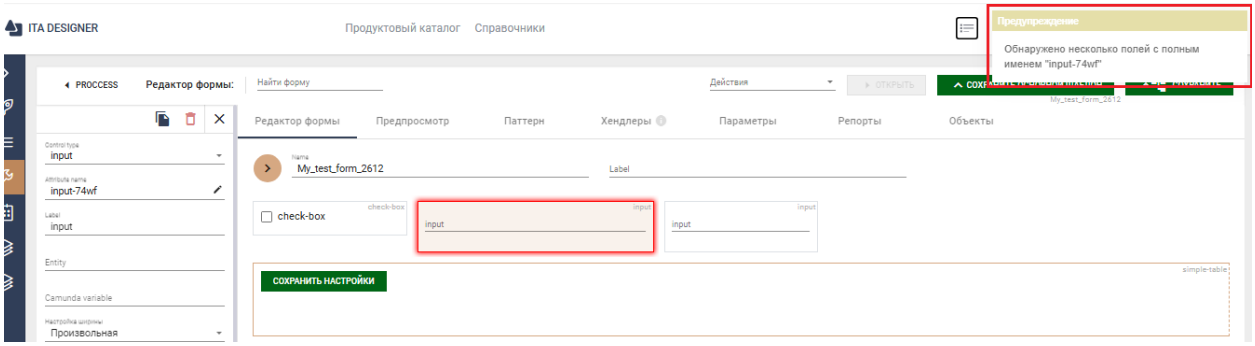
Для возврата к списку контролов необходимо закрыть панель управления атрибутами поля. Это можно сделать с помощью кнопки «X» панели управления атрибутами:



Для удаления контрола с формы нужно нажать на кнопку « » в панели управления атрибутами поля.

Кнопка « » позволяет скопировать контрол и вставить эту копию на форму.

Важно: Attribute_name у этой копии при этом назначается новый, поскольку не может быть двух контролов на форме с одинаковыми attribute_name, это уникальное значение. Если попытаться сохранить форму с контролами с двумя одинаковыми attribute_name, то редактор не даст этого сделать, пользователю будет выдано следующее сообщение о дублировании:



При этом также будет подсвечен дублирующий контрол.

5.1. Список параметров

Условия (conditions) - Параметр для добавления условий отображения поля “А”, можно скрывать/отображать в зависимости от значения поля “Б”

У кондишенов есть свои параметры:

Действие – что именно будет делать условие condition. Возможны следующие действия:

- Phone – контрол будет валидироваться как номер телефона
- Email – контрол будет валидироваться как email
- default-value – контролу будет назначаться значение по умолчанию
- equal – валидатор, сравнивающий значение поля с эталонным: поле валидно если оно равно введённому выражению
- not-equal – валидатор, сравнивающий значение поля с эталонным: поле валидно если оно не равно введённому выражению
- color – устанавливает определенный цвет текста
- hide – скрыть контрол
- show – показать контрол
- require – сделать контрол обязательным для заполнения
- disable – запретить редактирование значения контрола
- reset – сбросить значение

AttrName управляещего поля – контрол на форме, по которому рассчитывается условие. Можно задать формулу в соответствии с правилами построения выражений: {{some_control}}

Условие – арифметический оператор. Поддерживаются следующие арифметические операторы:

- « = »
  « != »
- « > »
- « >= »
- « < »
- « <= »

Значение управляющего поля – значение для условия. Формат значения ["string1", "bah", true, false...]. Возможно задание формулы в соответствии с правилами построения выражений: {{some_control}}

Если нужно задать какое-то сложное условие, зависящее от нескольких контролов, то можно добавить ещё один управляющий филд в условие conditions, это делается с помощью кнопки «зеленый +». При этом во втором филде появляется параметр «Логический оператор» - это «И» / «ИЛИ».

Также, для контрола можно добавлять несколько условий на разные действия (то есть например, можно дизейблить при одних условиях, и скрывать / показывать при других). Добавляется новое условие кнопкой «синий +»

Выбор опции из списка по его порядковому номеру (default-option) - Значение по умолчанию, которое можно установить из редактора. Принимает в себя число от 0. 0 - значение по умолчанию. Применяется в контролах, где предполагается выбор значения: select, radio-button

Значение по умолчанию (default-value) - Значение по умолчанию, которое можно установить из редактора. Выражения для этого параметра поддерживают JSONPath. Для контрола типа можно в качестве значения написать NOW, это подразумевает текущую дату. Важно: параметр не переписывает значение контрола (value).

Значение по умолчанию (с перезаписыванием) (default-value-force) - Значение по умолчанию, которое можно установить из редактора. Выражения для этого параметра поддерживают JSONPath. Для контрола типа date- можно в качестве значения написать NOW, это подразумевает текущую дату. Важно: параметр переписывает значение контрола (value).

Отображать в модалке (display-in-modal) - Отображать контрол в модалке. На данный момент возможность поднять форму элемента в модалке есть только у таблицы. Значение true / false

display-in-subrow – отображать в подстроке таблицы. Специальный табличный параметр для различных склеек таблиц

Не очищать (dont-clean) - Не очищает поле при срабатывании clean-form. Значение true / false

Не сохранять (dont-save) – не сохранять значение в контекст. Значение true / false

Полностью скрыть контрол (hide) - Полностью скрыть контрол в форме, чтобы не отображался. Значение true / false

Скрыть контрол визуально (invisible) - Визуально скрывает контрол, но сохраняет его функциональность. Значение true / false

isExpanded – параметр, регулирующий изначальное состояние контейнера: если true, то контейнер развёрнут; если false или отсутствует, то свёрнут. Пока что используется только для expansion-panel и accordion

Равно значению (equal) – валидатор, сравнивающий значение поля с эталонным: поле валидно если оно равно введённому выражению. Можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}. Поддерживаются специальные ключи, которые дают доступ к данным из других источников.

Валидатор email (email) - Валидатор, проверяющий является ли введенное пользователем значение корректным адресом электронной почты. Значение true / false

Валидатор ИНН ФЛ (inn_fl) - Валидация поля по формату ИНН ФЛ. Значение true / false

Валидатор СНИЛС (snils) - Валидация поля по формату СНИЛС. Значение true / false

Валидатор КПП (kpp) – Валидация поля по формату КПП. Значение true / false

Валидатор телефона (phone) - Валидатор, проверяющий является ли введенное пользователем значение корректным номером телефона. Значение true / false

list-key - Название атрибута, содержащего ключ

list-path - Название атрибута, содержащего путь к списку значений

list-sort - Список групп условий, по которым будет производиться сортировка набора данных. Каждая группа условий применяется в соответствии с приоритетом в порядке

возрастания. Т.е. сначала выводятся записи, удовлетворяющие группе условий 1, затем 2 и т. д. Записи, не удовлетворяющие ни одной группе условий, выводятся последними.

list-sublabel - Выражение, по которому будет формироваться дополнительное название (надпись сверху более бледным шрифтом) для каждого элемента набора данных

list-value - Название атрибута, содержащего значение, которое будет отображаться пользователю

Цвет фона (background) - Цвет фона контейнера; принимает строку формата HEX -- #RRGGBB, или RGB/RGBA -- rgba(255,212,221, 0.5), или HSL -- hsl(0deg 0% 98.04%)

Граница (border) - Добавляет границу контролу. По умолчанию граница устанавливается сразу с 4 сторон, цвет #dee2e6

border // инициирующий границу класс

Для изменения цвета используются классы:

- border-primary //черный
- border-secondary //салатовый
- border-success //тёмно-зелёный
- border-danger //красный
- border-warning //жёлтый
- border-info //бирюзовый
- border-light //светло-серый
- border-dark //темно-серый
- border-white ///белый

Чтобы задать скругление границы, используются следующие классы:

- rounded //со всех четырёх сторон
- rounded-top //в двух соотвествующих углах
- rounded-right //в двух соотвествующих углах
- rounded-bottom //в двух соотвествующих углах
- rounded-left //в двух соотвествующих углах
- rounded-circle // border-radius: 50%
- rounded-0 //без закругления

Также есть правила для указания отдельно ширины и цвета границы, задаются следующим образом:

- border-сторона(right/top/left/bottom)-width-ширина_в_писелях(1/2/3/4/5); например: border-left-width-4
- border-сторона(right/top/left/bottom) –цвет (primary/warning/secondary/green/success/orange); например: border-left-orange

Чтобы убрать границу, нужно передать пустую строку

Color – цвет текста, принимает строку формата HEX -- #RRGGBB, или RGB/RGBA -- rgba(255,212,221, 0.5), или наименование цвета – white, green, black

Размер шрифта (font-size) – размер шрифта контролов (работает, например, у text), можно использовать с container (будет применён ко всем доступным контролам внутри).

Возможные значения: t-1, t-2, t-3, t-4, t-5, h-1, h-2, h-3, h-4, h-5, small. t-1 — самый большой, t-5 — наименьший. h-X имеют те же размеры что и t-X, но дополнительно более жирные и с бóльшим межстрочным интервалом. small -- 90% от дефолтного размера.

Выравнивание по горизонтали (horizontal-align) – выравнивание содержимого в контейнере (container) по горизонтальной оси. Может принимать следующие значения:

- left - Выравнивание элементов по левому краю;
- center - Выравнивание элементов по центру;
- right - Выравнивание элементов по правому краю;
- space-between - Равномерно распределяет все элементы по ширине flex-блока. Первый элемент вначале, последний в конце;
- space-around - Равномерно распределяет все элементы по ширине flex-блока. Все элементы имеют полноразмерное пространство с обоих концов;
- space-evenly - Равномерно распределяет все элементы по ширине flex-блока. Все элементы имеют равное пространство вокруг;
- stretch - Равномерно распределяет все элементы по ширине flex-блока. Все элементы имеют "авто-размер", чтобы соответствовать контейнеру.

Отступ снаружи (margin) - Внешние отступы. Параметр достаточно гибкий, можно указать направление и размер отступа для каждой из 4-ых сторон. Общая формула — m{сторона}-{размер}, m{tblrxy}-{012345}

- t - от 'top', отступ сверху
- b - от 'bottom', отступ снизу
- l - от 'left', отступ слева
- r - от 'right', отступ справа
- x - отступ по оси X, устанавливает отступ одновременно справа и слева
- y - отступ по оси Y, устанавливает отступ одновременно снизу и сверху

пропуск этого параметра (например, m-3) установит отступ сразу с четырёх сторон

- 0 - установит внешний отступ в 0
- 1 - размер отступа 16px * .25
- 2 - размер отступа 16px * .5
- 3 - размер отступа 16px
- 4 - размер отступа 16px * 1.5
- 5 - размер отступа 16px * 3

Передать можно до четырёх (столько максимум сторон у контейнера) 'кусочков' через пробел (БЕЗ ЗАПЯТОЙ), например, mt-1 mr-2 mb-3 ml-4

Отступ внутри (padding) - Внутренние отступы для контейнера. Параметр достаточно гибкий, можно указать направление и размер отступа для каждой из 4-ых сторон.

Общая формула — p{сторона}-{размер}, p{tblrxy}-{0123456}

- t - от 'top', отступ сверху
- b - от 'bottom', отступ снизу
- l - от 'left', отступ слева
- r - от 'right', отступ справа

- x - отступ по оси X, устанавливает отступ одновременно справа и слева
- y - отступ по оси X, устанавливает отступ одновременно снизу и сверху

пропуск этого параметра (например, p-3) установит отступ сразу с четырёх сторон

- 0 - установит внутренний отступ в 0
- 1 - размер отступа 16px * .25
- 2 - размер отступа 16px * .5
- 3 - размер отступа 16px
- 4 - размер отступа 16px * 1.5
- 5 - размер отступа 16px * 3
- 6 - размер отступа 16px * 5

Передать можно до четырёх (столько максимум сторон у контейнера) 'кусочков' через пробел (БЕЗ ЗАПЯТОЙ), например, pt-1 pr-2 pb-3 pl-4

Тень (shadow) – тень у контейнера (container). Возможные значения имеют вид s-X, где X = 0, 1, 2, 3 s-0 — нет тени, s-1, s-2, s-3 — тень разной интенсивности, по возрастанию

Выравнивание текста (text-align) - Параметр выравнивания теста для контролов (работает не у всех; можно, например, использовать с text) и container (применяется сразу на все вложенные контролы)

- left - Выравнивание элементов по левому краю
- right - Выравнивание элементов по правому краю
- center - Выравнивание текста по центру

Выравнивание контрола по вертикали (vertical-align-self) - Выравнивание контрола по вертикали. Может принимать следующие значения:

- top (по умолчанию),
  center,
  bottom

Маска (mask) - Маска для input. Поддерживает формат imask. Синтаксис imask:

- 0 - любая цифра
- a - любая буква
- - любой символ

другие символы подразумеваются как фиксированные части маски

- [] - опциональная часть для значения
- {} - в фигурных скобках фиксированное значение

Пример: по маске +{7}(000)000-00-00 значение 73141434324 будем отображаться как +7(314)143-43-24,

Также, реализована поддержка обычных регулярных выражений типа /[а-я]*/i, (не забывать про якоря ^ и $).

Max – валидатор, ограничивающий максимальное значение введённого пользователем числа. Может работать как для числовых контролов, так и для контролов типа date-picker, поддержка даты в формате YYYY-MM-DD или абстрактного выражения NOW +(-) 12321, где NOW -- текущая дата, число после оператора +- задано в днях. Можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}. Поддерживаются специальные ключи, которые дают доступ к данным из других источников.

Максимальная длина (max-length) - Валидатор, ограничивающий максимальную длину введённого пользователем значения. Можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}. Поддерживаются специальные ключи, которые дают доступ к данным из других источников.

Min – валидатор, ограничивающий минимальное значение введённого пользователем числа. Может работать как для числовых контролов, так и для контролов типа date-picker, поддержка даты в формате YYYY-MM-DD или абстрактного выражения NOW +(-) 12321, где NOW -- текущая дата, число после оператора +- задано в днях. Можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}. Поддерживаются специальные ключи, которые дают доступ к данным из других источников.

Минимальная длина (min-length) - Валидатор, ограничивающий минимальную длину введённого пользователем значения. Можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}. Поддерживаются специальные ключи, которые дают доступ к данным из других источников.

Не равно значению (not-equal) - Валидатор сравнивающий значение поля с эталонным: поле валидно если оно не равно введённому выражению. Можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}. Поддерживаются специальные ключи, которые дают доступ к данным из других источников.

on-demand – отображать элементы array-field'ов в режиме "По требованию". При включении параметра облегчает вычисления и таким образом позволяет отображать большие объемы данных без лишней нагрузки на UI. Компонент аккордеона (по умолчанию on_demand = true) при включении не рендерит свёрнутые формы

Упрощённый вид (plain) - Отображаем значение контрола как обычный текст, без "обёртки" тултипа / лейбла / рамочки / пр.

Точность (precision) - Количество знаков после запятой. Можно указать в том числе 0

Обязательное поле (required) - Обязательное поле. Значение true / false

Должно быть true (required-true) – Поле должно обязательно иметь значение true (применяется для check-box, slide-toggle)

Роль (role) - Роль поля. Задаёт роль поля, которая имеет определенное значение для некоторых типов контролов, например ita-object-list требует наличия двух полей с ролями id и label. На данный момент существуют роли:

- ID – поле, содержащее идентификатор объекта. Применяется в компоненте ita-object-list;

- Лейбл – поле, содержащее лейбл объекта. Применяется в компоненте ita-object-list;
- Дополнительный ID – поле, содержащее дополнительный (вторичный) ID объекта. Используется компонентом ita-object-list. Нужен для тех случаев, когда надо отобразить пользователю дополнительный (вторичный) ID объекта.

Количество строк (rows) - Высота поля в строках (пока только для text-area)

Дата без времени (show-date-without-time) - Показывать дату без времени. Значение true / false. Устаревший параметр, практически не используется.

Показывать секунды (show-seconds) – Показывать у даты datetime-picker время с секундами (по умолчанию показывается без секунд). Значение true / false.

Выражение (source-value-modifier) - Выражение для расчета финального значения. Можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}. Поддерживаются специальные ключи, которые дают доступ к данным из других источников (описание модификаторов см далее).

Тултип (tooltip) – отображение подсказки для контрола.

5.2. Выражения и модификаторы

Для более гибкого управления отображением данных на форме используются выражения. В выражении можно указать значение другого поля в соответствии с правилами построения выражений: {{some_control}}, где some_control – это attribute_name другого контрола на форме.

В выражениях поддерживаются специальные ключи, которые дают доступ к данным из других источников, например:

- SELECTED_OBJECT - идентификатор объекта, выбранного в таблице/списке;
- user.* - данные о текущем пользователе. Доступны такие поля: id, name, firstName, middleName, surName, login, office, office_id, position, phone, roles, token, email. Например "{{ user.office_id }}"
- ROOT_ID – id корневого хаба

Также в выражениях можно использовать различные модификаторы, которые преобразуют вывод значения поля. Модификаторы указываются через символ "|" после имени поля, например вот так указывается модификатор number: {{some_control | number}}. При значении some_control = '1234567' вывод будет такой: '1 234 567.00'.

5.2.1. Список модификаторов

Условия:

is-not-empty - проверяет является ли значение не пустым - если да, то вернётся 1, иначе - 0

Если field1 не существует, то {{ field1 | is-not-empty }} вернёт 0; если field2 === null, то {{ field2 | is-not-empty }} вернёт 0; если field3 === "", то {{ field3 | is-not-empty }} вернёт 0; если field4 === "some-value", то {{ field4 | is-not-empty }} вернёт 1

is-set - проверяет существует ли значение у указанного контрола - если да, то вернётся 1, иначе - 0

Если field1 не существует, то {{ field1 | is-set }} вернёт 0; если field2 === null, то {{ field2 | is-set }} вернёт 0; если field3 === "some-value", то {{ field3 | is-set }} вернёт 1

if - простейший условный оператор: принимает на вход 3 значения - условие, значение 1 и значение 2. Если условие истинно/существует, то возвращает значение 1; иначе - возвращает значение 2. this - полученное модификатором значение.

Если value === 3, то {{ value | if:2:"value is equal to two!":"value is not equal to two..." }} вернёт "value is not equal to two..."

or - возвращает STATEMENT если значение falsy: Если выполняется над нулевым значением (null, undefined, '') или над выражением с ошибкой (типа "x + undefined = z", по которому видно что из-за ошибки одно значение не удалось получить), то возвращает STATEMENT

Если field1 не существует, то {{ field1 | or:'error' }} вернёт "error"; если field2 === null, то {{ field2 | or:'error' }} вернёт "error"; если field3 === "some-value", то {{ field3 | or:'error' }} вернёт "some-value"

Пример:

{{ link_Product_Client.hub_Client.link_Product_Client.hub_Product.BusinessKey | or: "{{ link_Product_Client_Family.hub_Client.link_Product_Client.hub_Product.BusinessKey }}" }}

здесь нужно было взять либо код владельца подписки, либо код участника подписки

switch - комплексный условный оператор: принимает на вход список кейсов в виде пар условие-значение, включая кейс default, представляющий ситуацию когда ни одно из перечисленных условий не выполнилось

Выражение {{ x | switch:1:"uno":2:"dos":default:"extraviado" }} вернёт "uno" если x === 1; "dos" если x === 2; в остальных случаях - "extraviado"

Даты:

date - преобразовывает дату в человекопонятный вид. Если передать true, то вернёт дату без времени

Если my_date === "2014-02-27T10:00:00", то {{ my_date | date }} вернёт "27.02.2014 10:00", а {{ my_date | date:true }} вернёт "27.02.2014"

Пример:

{{ date-now | date:true |
switch:({{link_Product_Client.hub_Product.sat_Product_Main.ConnectDate |
date:true}}):"1":({{link_Product_Client.hub_Product.sat_Product_Main.ProlongateDate |
date:true }}):"1":default:"0" }}

тут нужно было сравнить два поля с текущей датой, и если равны, то присвоить значение 1 (полю в котором такую модификацию вызываем), иначе 0

addPeriod - добавляет период к дате по коду

Если my_date === "2014-02-27T10:00:00", то {{ my_date | addPeriod:12M | date }} вернёт "2015-02-27T10:00:00"

moment_get - получает для даты количество единиц, указанных в unit - годы, месяцы, часы и тд.

Например, если date === 2023-04-21T13:51:27.677, то выражение {{ date | moment_get:year }} вернёт 2023, а {{ date | moment_get:month }} - 4.

Полный список доступных единиц - year (синонимы years, y), month (синонимы months, M), week (синонимы weeks, w), day (синонимы days, d), hour (синонимы hours, h), minute (синонимы minutes, m), second (синонимы seconds, s), millisecond (синонимы milliseconds, ms), quarter (синонимы quarters, Q), isoWeek (синонимы isoWeeks, W), date (синонимы dates, D), weekYear (синонимы weekYears, gg), isoWeekYear (синонимы isoWeekYears, GG), dayOfYear (синонимы dayOfYears, DDD), weekday (синонимы weekdays, e), isoWeekday (синонимы isoWeekdays, E). Описание всех единиц на сайте momentjs - https://momentjs.com/docs/#/get-set/

moment_add/moment_subtract - добавить к дате/отнять от даты указанное количество единиц. Например, если my_date === "2023-04-21T13:51:27.677", то {{ my_date | moment_add:2:days | date}} вернёт "23.04.2023 13:51"

Строки:

displayPeriod - функция для отображения периода в человекопонятном виде, принимает на вход код периода вида 20D, 3M, 12M

Если period_code === '12M', то выражение {{ period_code | displayPeriod }} вернёт "12 мес."

trim - удаляет пробелы в начале и конце строки

Если value === " x ", то {{ value | trim }} вернёт "x"

split - разбивает строку по разделителю. В качестве разделителя можно указать RegExp

Если my_string === "строка, с, запятыми", то {{ my_string | split:',' }} вернёт ["строка", " с", " запятыми"], а {{ my_string | split:'\s?,\s?' }} вернёт ["строка", "с", "запятыми"]

replace - ищет соответствия и заменяет их. Для поиска соответствия используется регулярное выражение. Например, если hub_Client.BusinessKey === "aclm14110", то выражение {{ hub_Client.BusinessKey | replace:1:2 }} вернёт 'aclm24220', а выражение {{ hub_Client.BusinessKey | replace:\\d+:2 }} вернёт 'aclm2'. Регулярное выражение создаётся сразу с ключом g - global; спецсимволы должны быть экранированы

includes - возвращает true/false если множества A и B пересекаются, то есть все или несколько элементов совпадают. Например, если a = '123', b = '234' и c = '987', то {{ a | split:"" | includes:({{ b | split:"" }}) }} вернёт true, а {{ a | split:"" | includes:({{ c | split:"" }}) }} вернёт false

number - приводит строку представляющую число к стандартному формату отображения чисел. По умолчанию количество знаков после запятой - 2. Также принимает на вход количество знаков после запятой

Если x === "12345.14159127369", то {{ x | number }} вернёт "12 345.14", а {{ x | number:0 }} вернёт "12 345"

to-number - приводит строку к числу. Умеет парсить float

Если x === "3.14", то {{ x | to-number }} вернёт 3, а {{ x | to-number:true }} вернёт 3.14

Списки/массивы/таблицы:

selection - возвращает выбранные в таблице строки. На данный момент актуально только для object-list и multiple-table-select

length - возвращает количество элементов списка/массива. Также можно применить и к строке

Если в таблице my_table 17 элементов, то {{ my_table | length }} вернёт "17"

sumBy - суммирует все строки списка/массива по конкретному полю

for-each - выполняет выражение для каждого элемента массива. this - текущий элемент

Если array === ['1', '2', '3'], то {{ array | for-each:(={{this}}=) }} вернёт ['_-=1=-_', '_-=2=-_', '_-=3=-_']

H.BusinessKey in ({{ list-of-business-keys | trim | split:'\s?,\s?' | for-each:('{{this}}') | join:', ' }})

get-item - получает из списка/массива элемент по индексу

Пример: {{ table | get-item:0 }} вернёт первую строку

pick - собирает все значения конкретного поля из каждого элемента списка/массива и возвращает в виде списка примитивных значений

Если в строке таблицы table есть поле id, то {{ table | pick:id }} вернёт [{значение поля id из первой строки}, {значение поля id из второй строки}, ...]

map - собирает нужные значения из каждого элемента списка/массива и возвращает в виде списка объектов. Нужен для того чтобы отобрать из списка/массива конкретные поля. this - текущий элемент списка

Пример: {{ table | map:id:hub_Client.id:label:hub_Client.Name }} вернёт

```
[
  {
     id: {значение поля hub_Client.id из первой строки},
     label: {значение поля hub_Client.Name из первой строки },
  },
  {
     id: {значение поля hub_Client.id из второй строки},
     label: {значение поля hub_Client.Name из второй строки },
  }
  ...
]
```

find - ищет элемент, соответствующий критерию, то есть возвращает элемент списка для которого переданное выражение-условие вернуло true. this - текущий элемент списка

Пример: если в таблице clients перечислены клиенты, для каждого из которых существует таблица счетов accounts, то например с помощью find можно проверить есть ли такой клиент у которого нету счетов - {{ clients | find:(this | get-property:accounts | length | if:0:true:false) | is-set }} вернёт 1 если в таблице clients есть клиенты без счетов

filter - ищет элементы, соответствующие критерию, то есть возвращает элементы списка для которых переданное выражение-условие вернуло true. Аналогичен модификатору find, только возвращает множество элементов, а не один элемент. this - текущий элемент списка

Пример: если pin === 1000008, то {{ pin | split:'' | filter:(this | if:'0':true:false) | length }} вернёт 5. Пример довольно примитивный - берём строку, делим на символы, получаем список символов, и фильтруем по критерию "если элемент равен '0', то true, иначе false", .

В живых формах filter конечно можно использовать для поисков элементов в любых массивах, в том числе таблицах, реестрах, аккордеонах и тд.

Пример:

{{ users-with-accounts | filter:(this | get-property:link_Account_Client | length | if:0:true:false) | pick:pin | join:', ' }}

{{hub_Product.link_Invitation_Product | filter:(this | get-property:hub_Invitation.sat_Invitation_Main.StatusName | if:"Отправлено":true:false) | length }}

или

{{hub_Product.link_Invitation_Product | filter:(this | get-property:hub_Invitation.sat_Invitation_Main.StatusName | switch:"Отправлено":true:"Отозвано":true:default:false) | length }}

Другой пример из фильтра для справочников, в хендлере «Наполнение селекта из справочников» (в фильтре "Условия фильтрации", там используется фильтрация на фронте):

{{ refKey | split:'' | filter:(this | if:'M':true:false) | length }} > 0

join - объединяет элементы массива в строку, между элементами будет помещен указанный пользователем разделитель.

Если array === [1, 2, 3], то {{ array | join:'|' }} вернёт "1|2|3"

Объекты:

get-property - возвращает значение запрашиваемого атрибута объекта. Чаще всего используется в связке с get-item: get-item отбирает нужный объект из списка, а get-property отбирает у него нужный атрибут.

Если в первом элементе массива arr есть свойство id со значением 1234, то {{ arr | get-item:0 | get-property:id }} вернёт "1234".

Специальные:

get-file - возвращает объект файла, выбранный в контроле file. Файл имеет такие поля:

- name - имя файла
- size - размер, в байтах
- type - MIME-тип файла
- content - содержимое файла в виде строки

Чтобы получить, например, содержимое файла нужно написать такое выражение: {{ my-file | get-file | get-property:content }}

get-policy-decision - возвращает true/false если объект доступен/не доступен пользователю. Объекты разбиты на группы - можно запросить разрешение объекта из той или иной группы, но т.к. пока что такой функционал нужен был только для того чтобы определить доступна ли конкретная форма, то и доступная группа сейчас только одна - forms. Например, если значение контрола test === 'form_JL', то выражение {{ test | get-policy-decision:forms }} вернёт true если форма form_JL доступна пользователю

get-status - возвращает один из статусов формы. Существующие статусы:

- changed
- unchanged
- valid
- invalid

Например, если текущая форма валидна и не изменена пользователем, то {{ this | get-status:valid }} и {{ this | get-status:changed }} вернут true и false.

Самостоятельные:

get-context - возвращает текущий контекст формы: FL или UL. Пример: выражение {{ get-context }} вернёт текущий контекст, если он вообще у формы задан.

5.3. Сохранение изменений формы

После того, как на форму добавили все нужные элементы, выполнили все необходимые настройки, ее нужно сохранить.
Для сохранения в редакторе есть две кнопки:

- Сохранить основной шаблон 

- Сохранить 

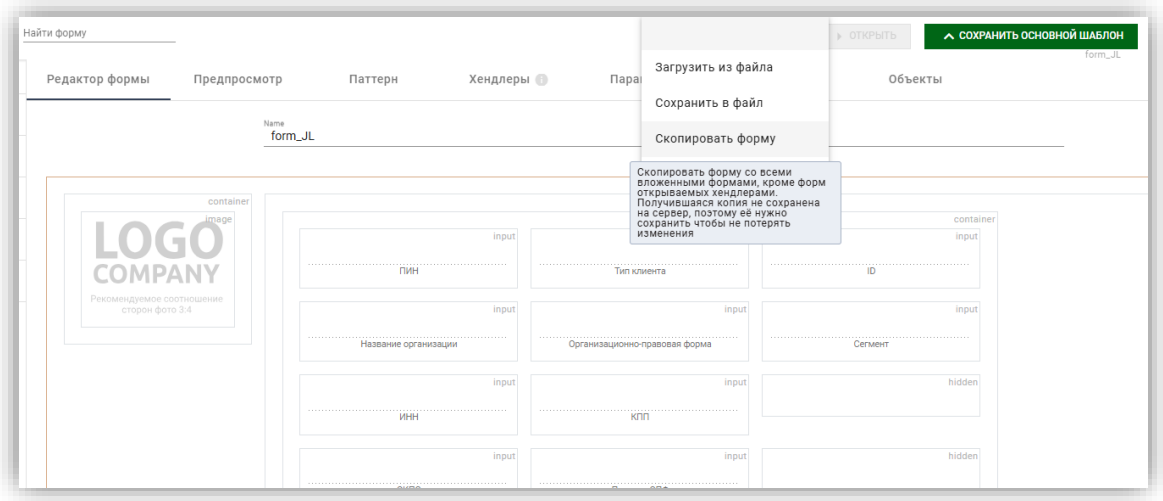
Кнопка «Сохранить основной шаблон» выполнит сохранение основной формы, при этом не будет сохранять изменения в шаблонах таблиц, виджетов и других контролов, использующих внутри себя формы-шаблоны.

Кнопка «Сохранить» сохранит всю цепочку вложенных форм.

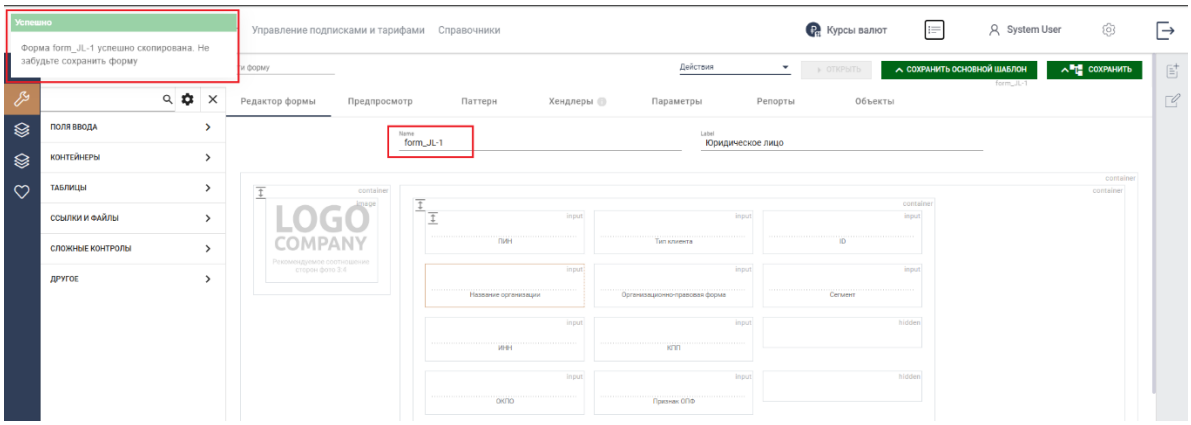
Важно: при сохранении цепочки вложенных форм необходимо иметь ввиду, что пользователь сохранит то, что в данный момент открыто у него в редакторе. И если при этом другой пользователь редактировал форму, которая так или иначе фигурирует где-то в цепочке у первого, то первый пользователь своим сохранением перетрет все изменения другого пользователя.

5.4. Копирование формы

Можно создать копию существующей формы. Для этого в открытой в редакторе форме нужно выбрать действие «Скопировать форму».

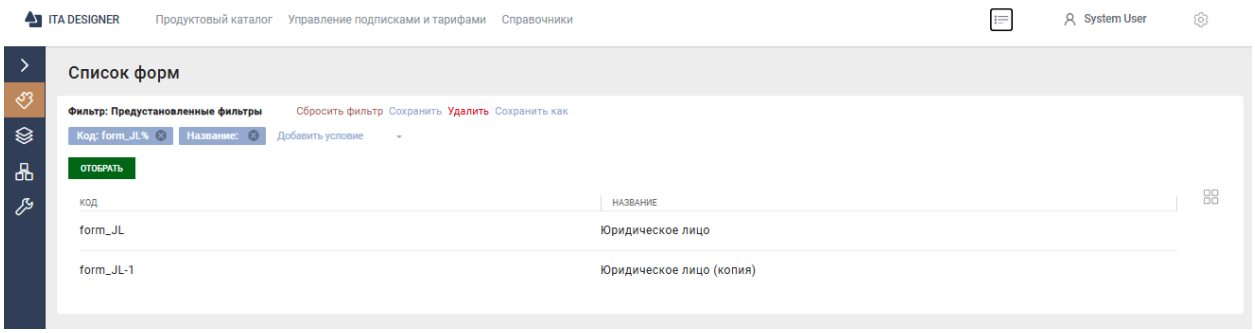


Форма будет скопирована с новым кодом Name, пользователю при этом будет выведено уведомление об успешном копировании:



Новую форму необходимо сохранить кнопкой «Сохранить» - будет сохранена вся скопированная цепочка форм, то есть основная форма, которую открыли, и вложенные формы: шаблоны таблиц, виджетов и т.п.

Новую форму можно найти в списке форм:



Для поиска в фильтрах можно задавать маску поиска через %, например в вышеуказанном скрине при задании фильтра Код = form_JL% будут найдены две формы: оригинальная и скопированная с нее.

5.5. Описание контролов

Все контролы в редакторе форм условно сгруппированы в несколько разделов:

- Поля ввода
- Контейнеры
- Таблицы
- Ссылки и файлы
- Сложные контролы
- Другое

5.5.1. Поля ввода

Здесь собраны простые поля, предназначенные как для просмотра, так и для ввода / редактирования данных

Input

Простое текстовое поле для ввода и отображения данных; текст отображается одной строкой

Input-number

Поле для ввода числовых значений, значение форматируется с разделителями разрядов и дробной части

Text-area

Текстовое поле для ввода и отображения данных; позволяет ввести длинный текст, который отображается в несколько строк

Check-box

Чек-бокс. Используется для различных признаков, подразумевающих значение «Да» или «Нет»

Slide-toggle

Переключатель, по смыслу это чек-бокс

Radio-button

Для различных признаков, вариантов.

Варианты можно указать с помощью опций в поле Items, тогда если значение поля соответствует одному из вариантов в Items, то автоматически будет устанавливаться в этом варианте.

Solid-radio-button

Как Radio-button, но внешний вид - панель

Slider

Позволяет выбирать значение из диапазона с определенным шагом, перемещая ползунок

Select

Выбор значения из списка (список выпадающий). Список можно задать:

- С помощью опций в поле «Items»,
- С помощью хендлера - получить данные из каких-либо справочников (описание хендлеров см. далее)
- С помощью парамеров listPath, listKey, listValue – выбрать массив из контекста или из переменной

Date-picker

Для отображения и задания даты без времени

Datetime-picker

Для отображения и задания даты со временем

Date-picker-range

Для задания диапазона дат

5.5.2. Контейнеры

Здесь собраны контролы, подразумевающие группировку полей на форме.

Accordion

Аккордеон. Позволяет отображать вложенные массивы.

Array

Массив. Позволяет отображать на форме множественные значения данных. ()

Container

Элемент дизайна; предназначен для красивой группировки полей на форме.

Expantion-panel

Элемент дизайна; также предназначен для красивой группировки полей на форме, при этом группу можно свернуть или развернуть.

Modal

Контейнер для полей, которые будут отображены в модальном окне

Используется в связке с хендлером "Вывод модального окна" - чтобы открыть модальное окно нужно создать этот хендлер, указать соответствующий тип модального окна и указать нужное модальное окно. Далее этот хендлер можно выполнить в любом удобном месте, например - привязать к кнопке.

Для закрытия модального окна в ней нужно создать кнопки с соответствующими ролями - "Кнопка согласия" и "Кнопка отказа". Клик вне области модального окна также закроет её с отрицательным решением.

Важно: также существует старый упрощенный способ отображения модального окна - у контрола типа "modal" в поле "Target message" указывается целевой код, далее этот код указывается в поле "message" нужной кнопки, по нажатию на эту кнопку поднимается модальное окно. Этот способ можно считать устаревшим, поэтому он больше не поддерживается, и он не гарантирует корректную работу модального окна.

Tab

Элемент tab-панели – одна из закладок горизонтального меню.

Tablist

Tab-панель – это горизонтальное меню закладок.

5.5.3. Таблицы

В меню Таблицы собраны различные виды таблиц, для которых источником данных является паттерн формы.

Simple-table

Простая таблица с данными.

Multiple-table-select

Для множественного выбора данных; данные отображаются в виде таблицы (такая таблица с чек-боксом).

Object-table

Сложная таблица, в которой можно провалиться в содержимое строки. У такой таблицы есть собственное меню действий; детальное содержимое строки открывается поверх таблицы с возможностью возврата назад.

Address-table

Специальная таблица для отображения адресов клиентов. Это кастомный контрол, разработанный для целей заказчика.

5.5.4. Ссылки и файлы

Link

Позволяет отображать значение в виде ссылки с открытием другого окна по клику.

File

Контрол, позволяющий пользователю выбрать файл. Доступ к выбранному файлу осуществляется с помощью модификатора get-file: {{ my-file | get-file }}

Файл имеет такие поля:

- name - имя файла
- size - размер, в байтах
- type - MIME-тип файла
- content - содержимое файла в виде строки

Поэтому чтобы получить, например, содержимое файла нужно написать такое выражение: {{ my-file | get-file | get-property:content }}

Важно: максимальный допустимый размер файла - 50Мб. Если пользователь выберет файл больше, то форма не сможет получить доступ к содержимому файла.

5.5.5. Сложные контролы

Ita-object-list

Контрол, предназначенный для реализации реестровых таблиц. Отбирает объекты по заданным критериям. Фильтрацию для отбора можно либо настроить в интерфейсе, либо в редакторе с помощью параметра "ita-filter".

Значениями фильтров могут быть выражения. Если не заданы параметры объекта (object-template, object-boclass), то открытие формы объекта будет невозможно.

В шаблоне строки таблицы отбора должно присутствовать поле с параметром role: id - значение этого поля будет использовано в качестве идентификатора объекта при двойном клике по строке.

Для того чтобы по двойному клику по строке форму объекта можно было открыть, нужно добавить соответствующие параметры - object-template и object-boclass.

Параметр "simple" скрывает фильтры и кнопку отбора, таким образом отбор происходит автоматически при показе формы, а фильтрация возможна только с помощью преднастроенных фильтров.

Параметр "target" регулирует способ открытия формы объекта: при значении "_modal" форма будет открыта в модалке, при других - как самостоятельная форма.

Ita-object-history-list

Таблица истории изменений - самостоятельный реестр, которому для получения данных необходимо указать имя сателлита и идентификатор объекта. Под объектом подразумевается родительский для нужного хаб. В этом реестре отображаются состояния объекта после каждого изменения, изменённые поля подсвечены цветом #B5E61D

Таблица истории изменений не имеет отдельного шаблона - шаблон строится прямо на лету по данным от ITAObjects. По этой причине изменение шаблона в редакторе отключено.

Важно: для получения данных необходимо чтобы у родительской формы был установлен контекст.

Ita-object-search

Позволяет пользователю отобрать объект по заданным критериям. Фильтрацию для отбора можно либо настроить в интерфейсе, либо в редакторе с помощью параметра "ita-filter".

Значениями фильтров могут быть выражения. Если не заданы параметры объекта (object-template, object-boclass), то открытие формы объекта будет невозможно

В шаблоне строки таблицы отбора должны присутствовать поле с параметром role: id и поле с параметром role: label - значения этих полей будут использованы в качестве идентификатора и имени объекта при выборе объекта соответственно

Ita-object-tree

Представляет собой иерархический список элементов. Обязательным параметром является object-tree-levels, который представляет собой набор параметров для каждого уровня дерева.

На данный момент существует 2 вида узлов:

- Статические узлы - узлы, которые не обращаются к операциям и, соответственно, не получают данные извне. Имеют простую структуру key + value;
- Динамические узлы - узлы, которые при клике обращаются к операции с id выбранного узла. Каждый узел имеет полный набор данных, пришедших от операции.

Важно: дерево настроено только на hls'ый формат данных с обязательной передачей id из поля code.

Widget

Блок с динамическим контентом для отображения данных объекта: по заданным типу (object-boclass) и идентификатору хаба (object-bocode) загружаются данные объекта и привязываются к полям шаблона (object-template). При изменении связанных объектов виджет самостоятельно обновляет данные шаблона в соответствии с выбранной политикой обновления.

Ita-mass-upload

Кастомный контрол, разрабатывался для реализации функционала Массовых операций.

5.5.6. Другое

Action

Кнопки, предназначено для запуска операций или выполнения специальных действий. Является триггером в хендлерах.

Dual-select

Для множественного выбора данных; позволяет перемещать данные из одного окошка в другое.

Chips-select

Для множественного выбора данных; позволяет выбирать множество данных, которые будут отображаться отдельными «чипами».

Divider

Задает линию-разделитель, может быть вертикальным и горизонтальным.

Hidden

Скрытое поле, может использоваться как в разметке, так и для задания скрытых значений для вычислений.

Image

Позволяет добавлять картинку в форму.

Input-address

Поле для задания адреса строкой (в некоторых проектах взаимодействие со справочником ФИАС),

Multiple-select

Для множественного выбора данных; данные выбираются из выпадающего списка и отображаются строкой через запятую.

Stepper

Элемент дизайна, позволяет задавать текущий шаг процесса; может быть горизонтальным и вертикальным.

Text

Простой текст; можно задать в поле «Text formulae» как константу, так и более сложное выражение с помощью модификаторов. По умолчанию отображается значение атрибута attribute_name.

Dynamic-field

Для задания динамической таблицы или динамических полей. Сложный кастомный контрол, реализован для отображения таблиц тарифов.

Json-table

Сложный кастомный контрол, реализован для функционала настройки уведомлений.

6. Закладка «Хендлеры»

Хендлеры - это типовые процедуры, которые можно использовать в форме для получения данных или выполнения определенных действий. Выполняются они при наступлении событий, перечисленных в списке триггеров; в качестве триггера можно использовать одно из фиксированных событий или имя поля. Если триггер - имя поля, то хендлер

выполнится при изменении пользователем этого поля (при условии, что значение поля вообще установлено).

Для вызова хендлера необходимо выбрать вид хендлера из списка возможных, задать триггер (также из списка), параметры хендлера, при необходимости – условия выполнения хендлера.

Фиксированные события триггера:

- **Сборка формы** – событие сборки формы, это самое раннее событие, происходящее в процедуре обработки формы, до любого другого;
- **Инициализация формы** – событие инициализации формы, выполняется после сборки формы и подхватывает значения, заданные контролам в форме;
- **Возврат к форме** – событие возврата к форме;
- **Нажатие кнопки «Назад»** - нажали на кнопку Назад

Контролы, которые могут выступать в качестве триггера:

- Action
- Select
- Dynamic-field
- (Другие изменяемые контролы)

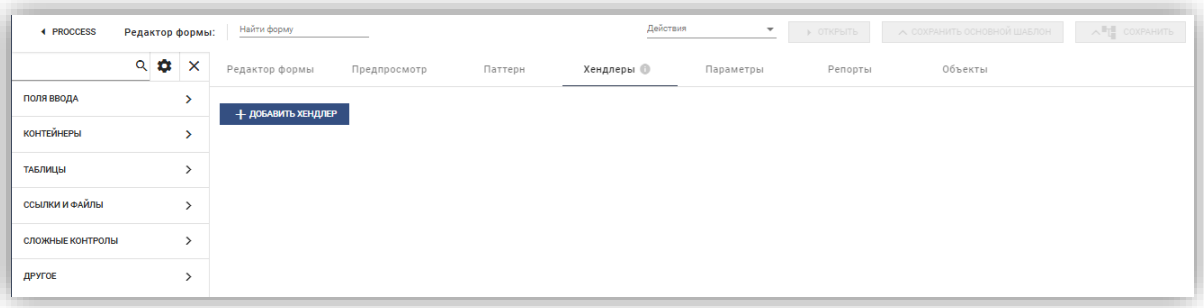
Параметр «Имя хендлера» для всех видов хендлеров опциональный, его нужно задавать тогда, когда данный хендлер нужно вызвать из другого хендлера.

Важно: любой из параметров хендлера может быть выражением.

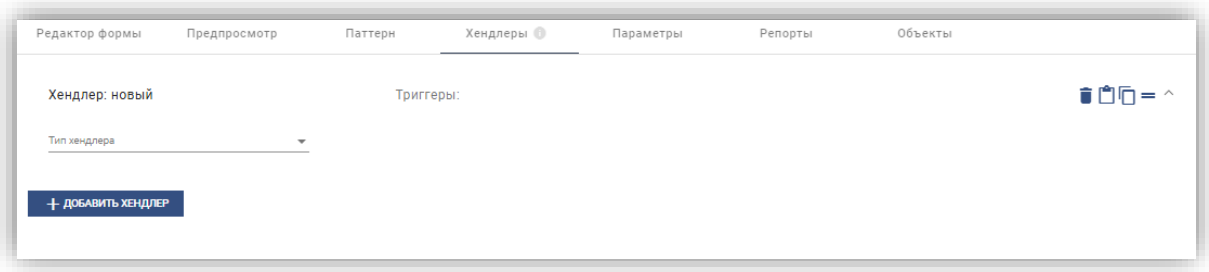
Условия выполнения – это дополнительные условия, при выполнении которых хендлер будет выполняться. Этот параметр позволяет более гибко управлять запуском хендлеров.

6.1. Добавление хендлеров на форму

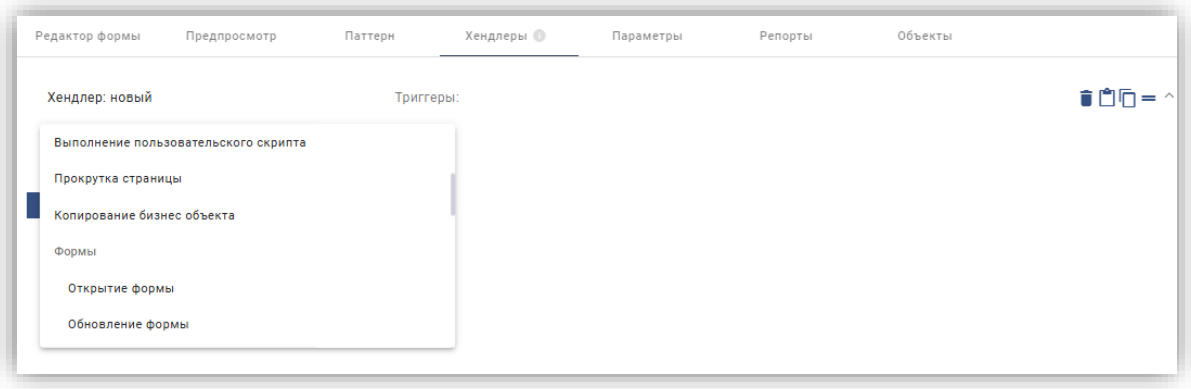
Для создания хендлера в форме пользователь переходит в закладку «Хендлеры» редактора форм. Открывается экран создания хендлера.



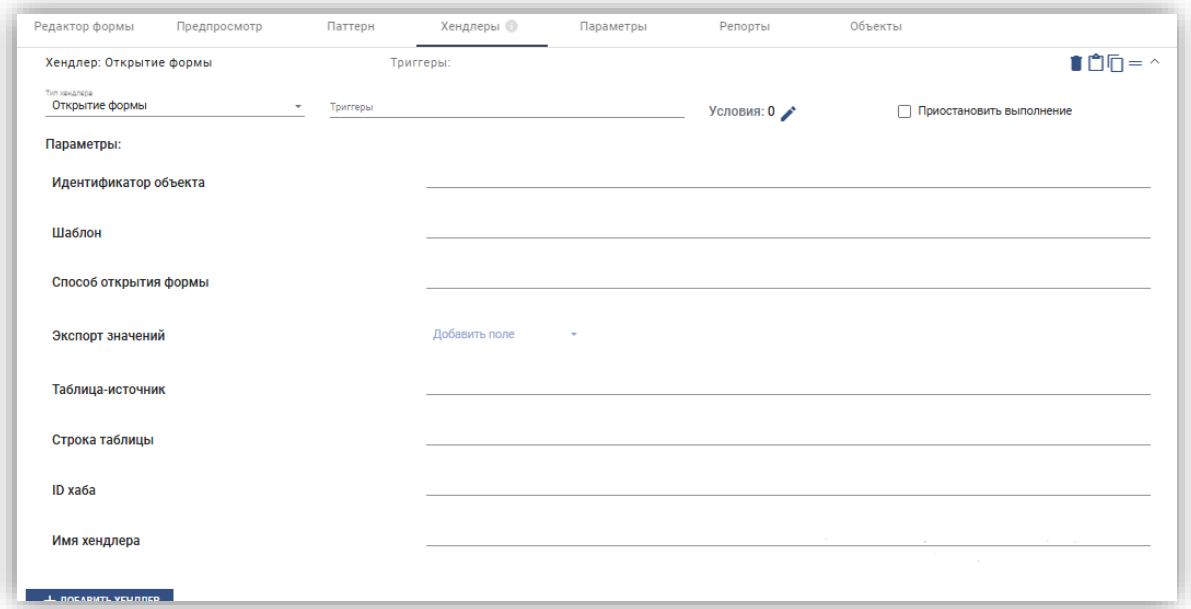
Пользователю доступна кнопка «Добавить хендлер». При нажатии на эту кнопку пользователю становятся доступны параметры задания хендлера.

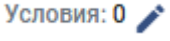


В первую очередь необходимо выбрать тип хендлера.

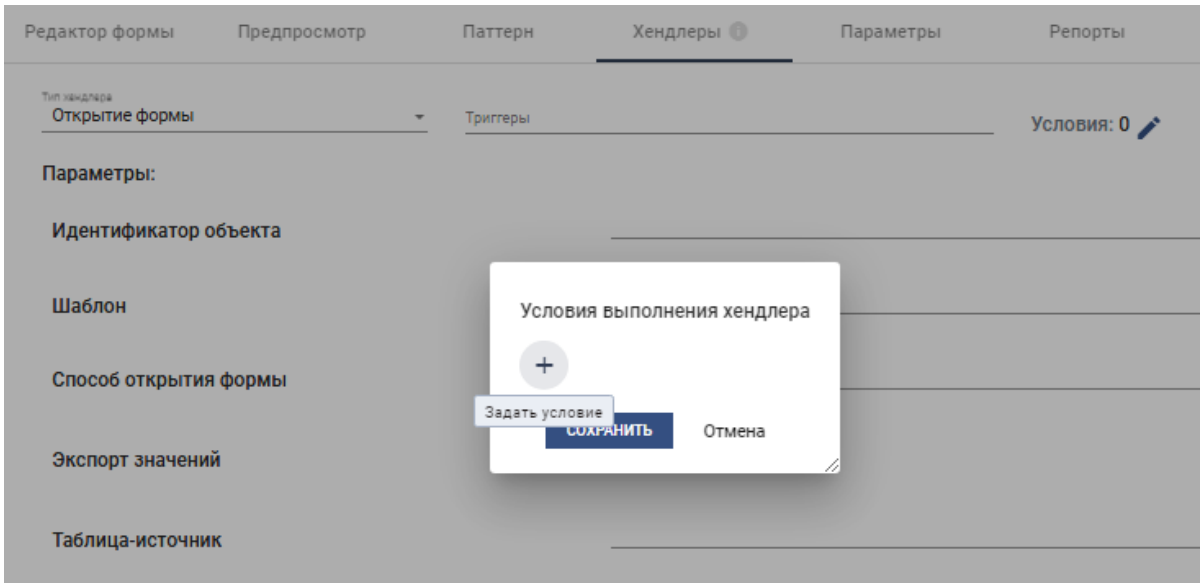


В зависимости от типа хендлера откроется список допустимых триггеров и параметров.

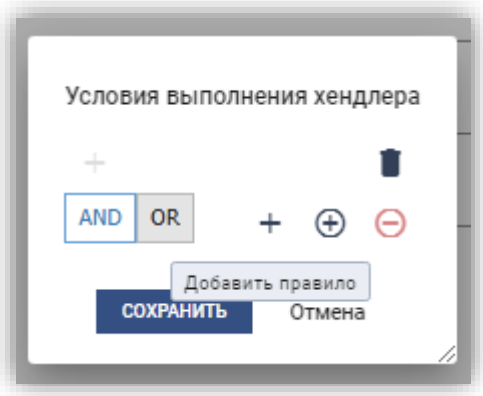


Для задания условий выполнения хендлера пользователя нужно нажать на символ  в 

. Откроется специально окно для описания условий.

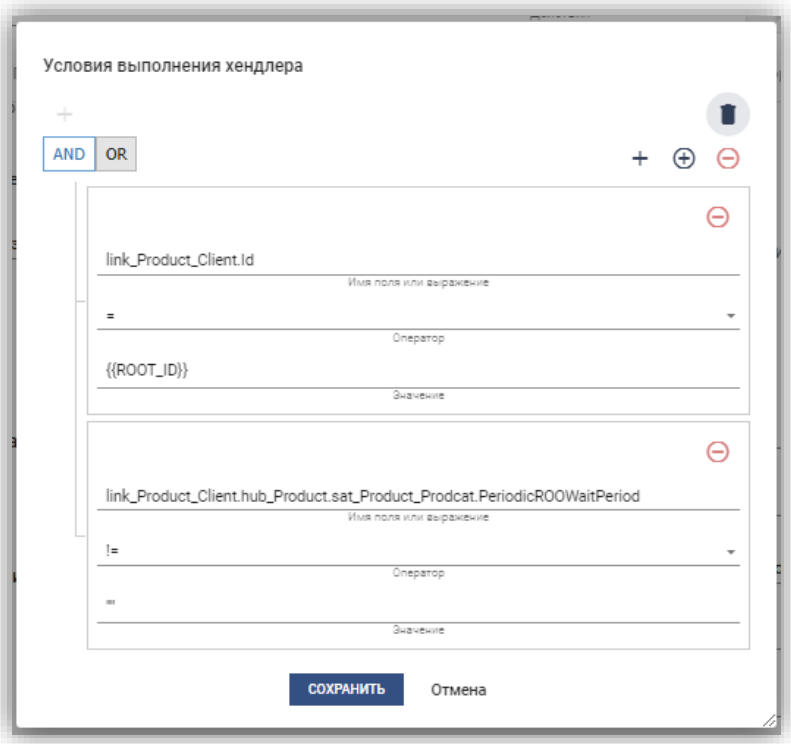


По кнопке «+» разворачивается панель с кнопками для задания правил. Правил может быть несколько, они объединяются с помощью логических операторов И / ИЛИ (переключатель AND / OR).

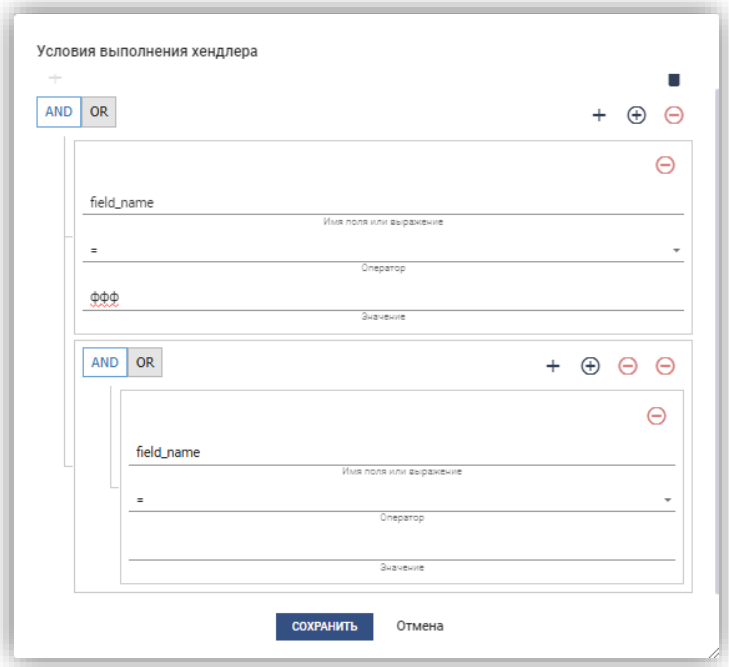


Кнопка + добавит новое правило. В правиле нужно задать:

- **Имя поля (или выражение)** – Attribute_name существующего контрола из формы, либо выражение (по правилам задания выражений с использованием модификаторов) с использованием attribute_name контролов из формы,
- **Оператор** – логический оператор, допустимые операторы: = / != / > / >= / < / <= ,
- **Значение** - поле поддерживает ввод выражений, чисел, строковых и специальных значений: 2 - число; "2" - строка; null, undefined - соответствующие объекты.



Кнопка ⊕ позволяет добавить внутри правила дополнительные вложенные правила, которые также объединяются с помощью логических операторов AND / OR внутри этого правила.

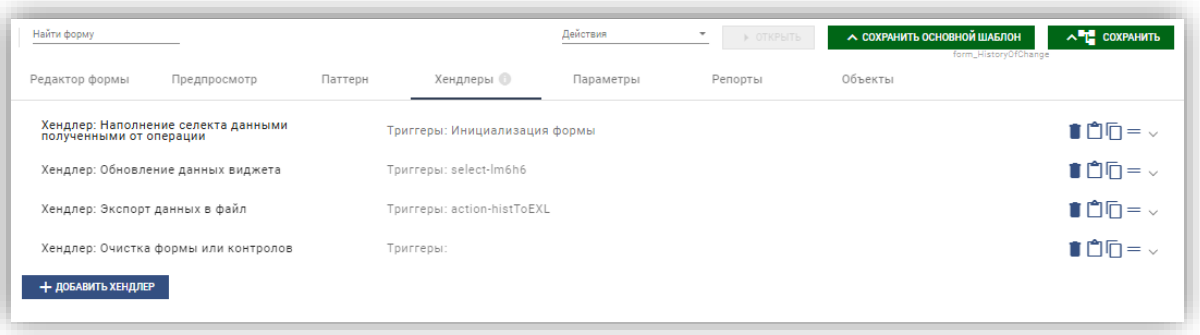


Кнопка ⊖ удаляет правила или вложенные правила.

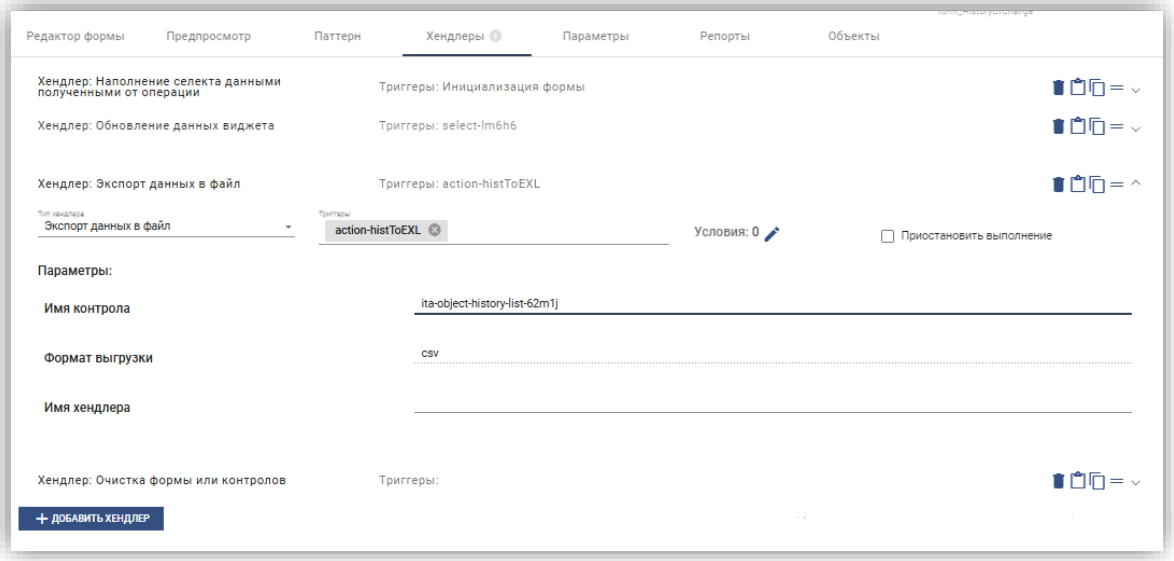
Кнопка удаляет условие целиком, то есть всю цепочку правил и вложенных правил сразу.

После добавления хендлера и заполнения всех необходимых параметров необходимо сохранить форму с помощью кнопки «Сохранить основной шаблон»

Все последовательно добавленные хендлеры располагаются на экране в виде аккоредона.

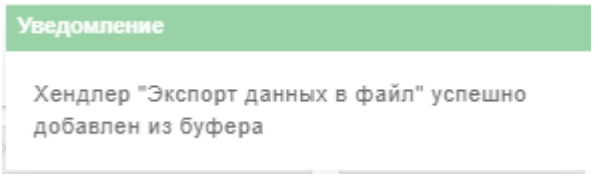


Кнопкой пользователь может развернуть хендлер и посмотреть/отредактировать его настройки



Кнопка удаляет хендлер с формы. После удаления форму необходимо сохранить кнопкой «Сохранить основной шаблон».

Кнопки («Скопировать хендлер в буфер») и («Создать копию хендлера») нужны для копирования хендлеров. Но кнопка создает копию хендлера в текущей форме. Это может быть нужно для создания нескольких хендлеров одинакового типа, но для разных контролов. А кнопка копирует хендлер (у которого эта кнопка была нажата) в буфер для вставки в форму, при этом форма может быть как текущая, так и другая, открытая в другом окне браузера. Вставка такого хендлера из буфера выполняется с помощью сочетания клавиш «CTRL» + «V». После этого пользователю будет выведено сообщение:



Кнопкой ═ («Переместить хендлер») пользователь может менять порядок хендлеров на форме. Для этого пользователь захватывает мышкой кнопку перемещения и передвигает хендлер на экране вверх или вниз.

6.2. Виды хендлеров

Хендлеры, как и контролы, также сгруппированы в несколько разделов.

6.2.1. Хендлеры вне групп

Вывод сообщения – выводит на экран сообщение. Если не задан текст сообщения, то сообщение выведено не будет, а в консоль будет выведено соответствующее предупреждение

Параметры:

- Тип – тип сообщения, поддерживается 5 типов сообщений: success, error, warning, info, show
- Текст – текст сообщения
- Заголовок – заголовок окошка, в котором выводится текст сообщения

Вывод модального окна - отображает на экране модальное окно, которое в зависимости от указанного типа ждёт от пользователя того или иного действия. От действия пользователя зависит дальнейшая обработка: если этот хендлер используется перед выполнением операции и пользователь изъявил отказ, то целевой хендлер выполнение операции выполнен не будет. Специальный тип custom позволяет использовать собственный шаблон для модального окна, для чего нужно предварительно создать контейнер modal с требуемым наполнением. В качестве заголовка модального окна используется заголовок, указанный в хендлере или заголовок указанного контрола modal. Соответственно, если нет ни того, ни другого, то у модального окна не будет заголовка. Для закрытия модального окна в ней нужно создать кнопки с соответствующими ролями - Кнопка согласия и Кнопка отказа. Клик вне области модального окна также закроет её с отрицательным решением.

Параметры:

- Тип – тип модального окна, поддерживается 3 типа: alert, confirm и custom
  - Alert – информационное окно с одной кнопкой «ОК»;
  - Confirm – окно с текстом (вопроса) и двумя кнопками «Да» и «Нет»;
  - Custom – модальное окно, которое для отображения использует контрол типа modal.
- Заголовок – заголовок окна
- Текст – текст, который будет отображаться в модальном окне (для alert и confirm)
- Модальное окно – (для custom) имя контрола modal из формы

Модификация и сохранение данных в форму для дальнейшего использования другими контролами – используется в связке с выполнением пользовательского скрипта

Параметры:

- Имя создаваемого контрола, куда будут сложены данные
- Модификация входных значений

Выполнение пользовательского скрипта – позволяет выполнять скрипты, написанные пользователями, для какого-либо преобразования данных, которое невозможно реализовать простыми средствами редактора

Параметры:

- Скрипт – окно редактирования пользовательского скрипта. Поддерживается JavaScript.

Прокрутка страницы – прокручивает страницу до указанного поля

Параметры:

- Целевое поле – имя поля, до которого будет прокручена страница. После прокрутки это поле будет видно пользователю
- Выравнивание по вертикали – тип выравнивания, возможно 4 типа:
  - Элемент будет расположен в центре (center)
  - Элемент будет расположен в конце страницы, в самом низу (end)
  - Докрутить до ближайшего края элемента (nearest)
  - Элемент будет расположен в начале страницы, в самом верху (start)

6.2.2. Группа «Формы»

В данном разделе собраны хендлеры, предназначенные для взаимодействия между формами

Открытие формы – открывает форму объекта по заданным параметрам. Работает с контролом action и пока не работает с object- list и link.

Важно: также существует старый упрощенный способ открытия формы - у контрола типа action нужно указать набор соответствующих параметров, после чего по нажатию на эту кнопку будет открываться форма. Этот способ можно считать устаревшим, он более не поддерживается и не гарантирует корректное открытие формы. Тем не менее пока поддерживается в link.

Параметры:

- Идентификатор объекта – идентификатор объекта, который будет использован для загрузки данных в форму. Поддерживает выражения. Если не задан, то шаблон не будет обогащен данными объекта.
- Шаблон – имя шаблона, который будет использован в качестве новой формы.
- Способ открытия формы – на данный момент существует три способа открытия формы:
  - В качестве самостоятельной формы (по умолчанию, также можно задать как _form)
  - В модальном окне (задается через _modal)
  - В контейнере (указывается имя контейнера, созданного в данной форме; контейнеры появятся в выпадающем списке)

- Экспорт значений – в открываемую форму можно перенести значения из текущей формы, для этого надо добавить нужные поля в список для экспорта и назначить им имя в целевой форме. Например, если выбрать для экспорта поле test-field и назначить ему имя imported-test-field, то при открытии целевой формы в ней появится скрытое поле imported-test- field с тем значением, которое было у поля test-field в предыдущей форме. Если при импорте поля окажется что такое поле в форме уже есть, то его значение будет перезаписано, а в консоль будет выведено оповещение.
- Таблица-источник - для того чтобы открыть форму объекта из таблицы, нужно указать целевую таблицу. Это нужно только для того чтобы создать линк для нового объекта - если в таблице нет линка для открываемого объекта, то он будет создан, и его можно будет впоследствии сохранить стандартными средствами.
- Строка таблицы - если указана таблица-источник, то также нужно указать какую именно строку открываем. В качестве значения может быть только выражение. Если строка не найдена или не указана, то при открытии формы будет создан линк, который можно будет сохранить стандартными средствами.
- ID хаба - для того чтобы создать линк, нужно знать идентификаторы двух хабов, но в общем случае их нельзя узнать сразу, так как один из хабов, возможно, будет собран позже. Поэтому нужен способ указать переменный ID хаба. Как раз для этого и нужен этот параметр - указывается выражение, а при сохранении второй хаб нашего линка получит тот ID который получился в результате обработки выражения.
- Имя хендлера - устанавливается для того, чтобы можно было сослаться на конкретный хендлер. Опционально.

Обновление формы – обновляет форму. На данный момент варианта всего 2 можно обновить эту форму, ту в которой создан хендлер, или можно обновить текущую форму в цепочке форм.

Параметры:

- Целевая форма – два типа целевой формы:
  - Эта форма (_self)
  - Текущая форма в цепочке открытых форм (_current)

Переход на форму / страницу – совершает переход: либо на одну из страниц, перечисленных в меню; либо на одну из предыдущих форм в текущей цепочке форм; либо закрывает активные модальные окна; либо по конкретному URL, достроенному после текущего base. Если указан переход на предыдущую форму, но указанная форма не найдена, то переход осуществлён не будет, а в консоль будет выведено соответствующее сообщение.

Параметры:

- Тип перехода – поддерживается три типа перехода:
  - Предыдущая форма (previous-form) – переход к одной из предыдущих форм из цепочки форм;
  - Закрытие модального окна (close-modal) – при указании данного типа 1) не обязательно заполнять target и 2) закрыты будут все активные модальные окна
  - Навигация (url) – переход по заданному адресу

- Экспорт значений - в открываемую форму можно перенести значения из текущей формы, для этого надо добавить нужные поля в список для экспорта и назначить им имя в целевой форме. Например, если выбрать для экспорта поле test-field и назначить ему имя imported-test-field, то при открытии целевой формы в ней появится скрытое поле imported-test- field с тем значением, которое было у поля test-field в предыдущей форме. Если при импорте поля окажется что такое поле в форме уже есть, то его значение будет перезаписано, а в консоль будет выведено оповещение.
- URL строка (с поддержкой выражений) – можно задать путь до конкретной формы, например: 'audit-ui/audit/operations/{{Client_Id}}'
- Целевая форма/страница – для типа перехода url значение '_blank' будет открывать ссылку в новой вкладке, пустое - в текущей.

6.2.3. Группа «Операции»

В данном разделе собраны хендлеры, запускающие какие-либо операции.

Выполнение операции – выполняет указанную в параметрах операцию.

Параметры:

- Класс – класс BOCLass объекта из модели данных ita-objects. В выпадающем списке отображаются все доступные классы
- Операция – доступные для выбранного класса объекта операции.
- Идентификатор объекта – идентификатор объекта, над которым выполняется операция.
- Аргументы для операции – набор входных аргументов для операции. В качестве значения любому аргументу можно указать выражение, по которому при выполнении операции будет рассчитано значение аргумента. Если при расчёте значения аргумента произойдет ошибка (например, в случае отсутствия поля, использованного в выражении), то это поле будет исключено из аргументов операции.
  Также существует возможность сформировать аргументы списочного типа из данных таблиц или списков. Для этого пока нету удобного интерфейса, поэтому придётся в качестве элементов списочных аргументов указывать специальные выражения. Например, вот как может выглядеть выражение, которое отберёт у каждого объекта таблицы test-table свойство (поле) id и сформирует список строк:
  {{ test-table | pick:id }}
  А вот так выглядит выражение, которое сформирует список аргументов на основе выбранных в таблице объектов:
  {{ test-table | selection | pick:id }}
  Модификатор selection применим к любой таблице у которой есть функционал выбора объектов - "multiple-table-select", "ita-object-list" и тд
- Перед выполнением – перед выполнение операции можно выполнить те или иные действия, представленные набором хендлеров. Ошибка или отрицательный результат при обработке этих хендлеров остановит цепочку выполнения: ни последующие хендлеры, ни сама операция не будут выполнены.
- Критерии вывода на успех/ошибку – после выполнения операции, её ответ не всегда представляет собой положительный результат. Иногда он может быть вполне успешным, но говорящим о том, что на процессе есть ошибка и, как

пример, должен не давать возможность пропустить процесс далее. Этот параметр служит как раз регулятором того, на какой случай мы должны наткнуться «успех / ошибка / другое» в случае выполнения условия. Пример условия {{ response | get-property: clientStateList | find:(this | get- property: error)}} - говорит о том, что в ответе операции есть массив clientStateList, который ищет поле еггог в любом из его элементов. Если такое поле находится, то ивент бросается в либо успех, либо ошибку, либо другое

- После выполнения – после выполнения операции, в зависимости от результата, можно выполнить те или иные действия, представленные набором хендлеров. Для этого нужно задать хендлерам имена и потом указать их в поле требуемого результата «успех», «ошибка» или другого. Например, можно вывести сообщение об успешном выполнении операции и вернуться на заданную форму: для этого нужно создать два соответствующих хендлера и указать их последовательно в данном параметре.
  Всем хендлерам, выполняемым в рамках этой операции, ответ операции доступен по ключу response. Например, если в ответе есть поле message, то оно доступно по ключу response.message.

READER – операция чтения данных и сборки формы. Для этого хендлера не нужны триггеры, но он выполняется при сборке формы. Если на форме в разделе «Паттерн» не задан паттерн, то будет использован паттерн текущего шаблона.

Параметры:

- Класс – класс BOCLass объекта из модели данных ita-objects. В выпадающем списке отображаются все доступные для чтения классы
- Операция – список доступных операций для выбранного класса. Нужно выбирать операцию, которая возвращает в ответе json в структуре HLS
- Идентификатор объекта – идентификатор объекта над которым выполняется операция. Если не задан, то будет использован идентификатор, с которым была вызвана форма. Например, в случае с виджетом идентификатор указывается в параметре object-bocode самого поля widget, и при сборке формы виджета READER получит этот идентификатор
- Аргументы для операции – набор входных аргументов для операции. В качестве значения любому аргументу можно указать выражение, по которому при выполнении операции будет рассчитано значение аргумента. Если при расчёте значения аргумента произойдет ошибка (например, в случае отсутствия поля, использованного в выражении), то это поле будет исключено из аргументов операции. Поддержка выражений – такая же, как и в хендлере Выполнение операций
- После выполнения – функционал действий в зависимости от результата выполнения взят из хендлера Выполнения операций.

SAVER – Операция сохранения данных и сборки формы. В случае, если параметры не заполнены, будут использованы значения по умолчанию.

Параметры:

- Класс – по умолчанию Project, но можно также выбрать класс из доступных BOClass модели

- Операция – по умолчанию DVSaver, но также можно выбрать операцию
- Идентификатор – идентификатор объекта, над которым выполняется сохранение

6.2.4. Группа «Справочники»

В данном разделе собраны хендлеры для работы со справочниками.

Важно: под справочниками подразумеваются справочники модуля ITA Info – это составная часть ITA Forms.

Наполнение селекта из справочника – загружает для указанного поля опции из справочников. Допустимые типы поля для данного хендлера: select, multiple-select, dual-select, radio-button, solid-radio-button, chips-select, stepper

Параметры:

- Справочник – код справочника из справочной системы ITA Info. Можно вписать выражение, по которому будет рассчитан код справочника.
- Лейбл – имя атрибута справочной записи, значение которого будет использовано в качестве лейбла, отображаемого на форме. Выбор в выпадающем списке происходит из атрибутов справочников в справочной системе ita-info. По умолчанию выбран label
- Ключ – имя атрибута справочной записи, значение которого будет использовано в качестве ключа (сохраняемое и передаваемое значение). Выбор в выпадающем списке происходит из атрибутов справочников в справочной системе ita-info. По умолчанию выбран refKey
- Фильтр – задание условия для фильтра данных из справочника. Можно выбрать атрибут справочника и указать критерий
- Установить значением первый элемент – если установлен, то первая запись справочника будет автоматически выбрана в поле (если нет – то поле будет пустым, пока что-нибудь не выберем)
- Атрибут для сортировки – имя атрибута справочной записи, по которому будет производиться сортировка. По умолчанию выбран атрибут showOrder
- Сортировать по убыванию – по умолчанию, когда данный параметр не установлен, сортировка выполняется по возрастанию. Ее можно изменить, установив данный параметр в «Да».
- Поле – имя поля на форме, для которого будут загружены опции.

Наполнение контрола значением из справочника – загружает в указанное поле значение из справочника. Данный хендлер нужен, чтобы для ключевого значения (например, какого-то кода) найти человекочитаемую расшифровку в справочнике и отобразить на форме. Для данного хендлера допустим тип поля input.

Параметры:

- Справочник – код справочника из справочной системы ita-info. Можно вписать выражение, по которому будет рассчитан код справочника.
- Атрибут для вывода – имя атрибута справочной записи, значение которого будет выведено в целевой контрол. По умолчанию выбран label
- Идентификатор записи – идентификатор записи, то есть на самом деле ее refKey. То есть здесь указывается имя контрола (в {{}}), значение которого будет являться

refKey в заданном справочнике, и по которому будет выполняться поиск атрибута для вывода.
- Целевое поле – имя поля, в которое будет загружено значение

6.2.5. Таблицы

В данном разделе собраны хендлеры для манипуляций с данными в таблицах, которые невозможно выполнить стандартными способами в редакторе.

Добавление объектов – добавляет объекты в таблицу из списка объектов. Используется в multiple-table- select, simple-table, object-table, address-table.

Хендлер принимает на вход имя целевой таблицы и имя формы отбора: при выполнении этого хендлера будет отображена форма отбора, пользователь сможет выбрать объекты и нажать на кнопку отбора, после чего выбранные объекты будут добавлены в таблицу.

На данный момент хендлер не учитывает возможность наличия нескольких action и ita-object-list в форме отбора, так что в качестве кнопки и списка отбора будут использованы первые найденные контролы соответствующих типов.

Параметры:

- Целевая таблица – таблица, в которую будут добавлены объекты. Важно: так как для получения данных реестра используется паттерн, то в шаблоне целевой таблицы должны быть использованы атрибуты именно этого паттерна, а не атрибуты хаба. Иначе нужно указать связь полей паттерна с полями HLS-объекта
- Форма отбора – имя шаблона, который будет использоваться в качестве формы отбора
- Маппинг полей – так как для получения данных используется паттерн, то для успешного создания HS-объекта нужно указать с каким атрибутом HLS-объекта связан атрибут паттерна: pattern -> HLS. Нужно только если поля целевой таблицы непосредственно представляют атрибуты HS-объекта.
- Источник объектов – таблица или массив, из которой будут получены объекты. Специфическая настройка, нужная для тех случаев, когда список объектов полученный из целевого ita-object-list нуждается в пользовательской обработке перед добавлением.

6.2.6. Виджеты

В данном разделе собраны хендлеры для взаимодействия с виджетами.

Обновление данных виджета – обновляет данные виджета. Список объектов (ita-object-list) в том числе является виджетом, поэтому и его можно обновить при желании. Также возможно обновить родительский виджет, в случае применения этой функции будет найден ближайший родитель-виджет и обновлен.

Параметры:

- Виджеты – список доступных полей-виджетов на форме. Можно выбрать из выпадающего списка, или в некоторых случаях написать вручную

- Обновить родительский виджет – при установке в «Да» происходит поиск и обновление ближайшего виджета-родителя.

Импорт данных в файл – выгружает данные контрола в файл. На данный момент хендлер применим только к ita-object-list

Параметры:

- Имя контрола – имя контрола ita-object-list, данные из которого который предполагается импортировать в файл
- Формат выгрузки – на данный момент возможна только выгрузка в csv-файл

Удаление бизнес-объекта – проставляет бизнес-объекту все атрибуты, соответствующие удалённому состоянию. Если аргументы не заданы, то проставляет атрибуты корневому для текущей формы бизнес-объекту. Для фактического удаления после проставления атрибутов требуется сохранить объект.

Параметры:

- Класс – по умолчанию Project
- Операция – по умолчанию DVSaver
- Идентификатор объекта – идентификатор объекта, над которым выполняется операция
- Перед выполнением – перед выполнением операции можно выполнить те или иные действия, представленные набором хендлеров. Ошибка или отрицательный результат при обработке этих хендлеров остановит цепочку выполнений ни последующие хендлеры, ни сама операция выполнены не будут

7. Закладка «Паттерн»

Источник данных

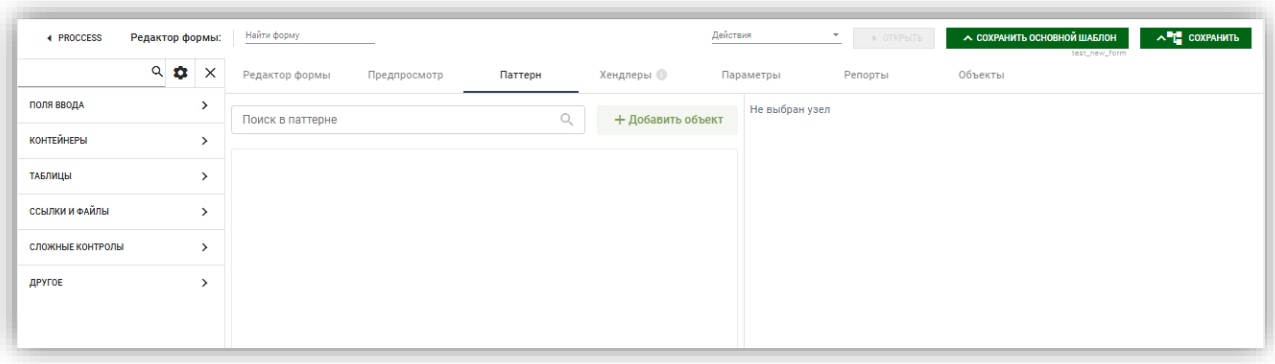
Паттерн – это описание структуры данных в формате HLS (хабов / сателлитов / линков). В такой структуре есть один главный хаб. Для работы паттерна ID этого главного хаба должен быть каким-то образом задан: можно задать константой, а можно передать динамически.

Пример паттерна с указанием hub_id в параметре BOCode:

```
{
 "Pattern": {
      "hub_MPShowcase": {
        "sat_MPShowcase_Main": {},
        "link_MPVersion_MPShowcase": {
          "hub_MPVersion": {
            "sat_MPVersion_Main": {},
            "link_MPField_MPVersion": {
              "hub_MPField": {
                "sat_MPField_Main": {}
              }}}}} ,
   "BOCode": "32b37788-2ff2-4431-9931-eca3c9d7879e"
}
```

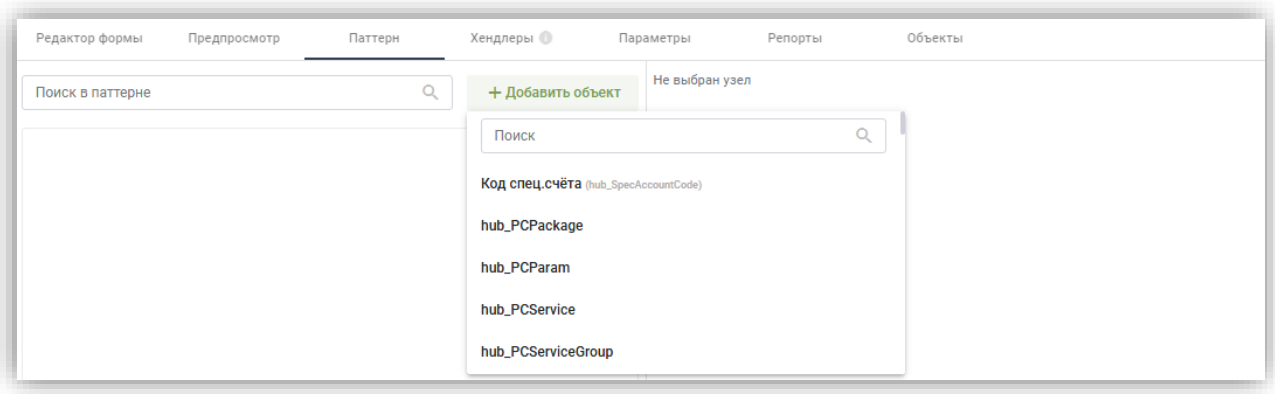
Для задания паттерна формы пользователь переходит в закладку «Паттерн» в редакторе.

Экран задания паттерна выглядит следующим образом:

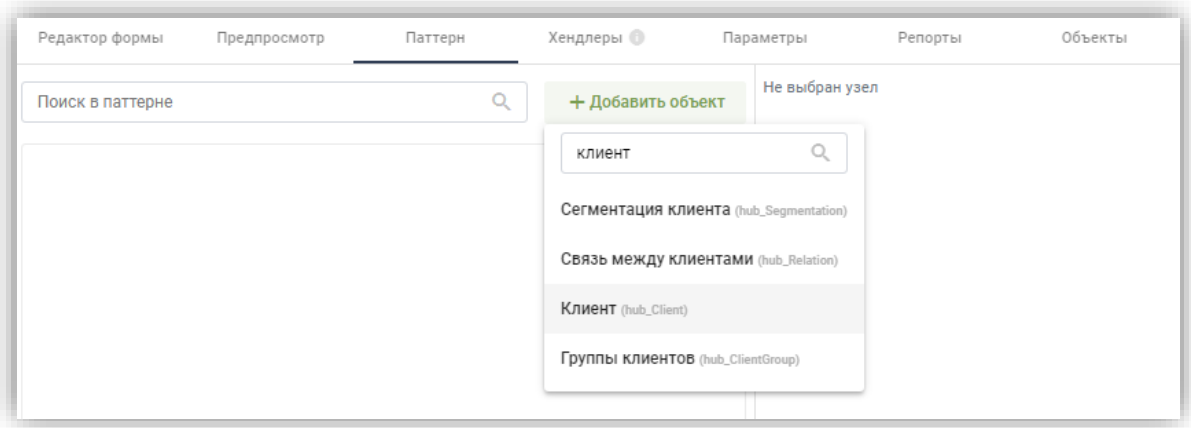


Иерархия бизнес-объектов добавляется путем нажатия на кнопку 

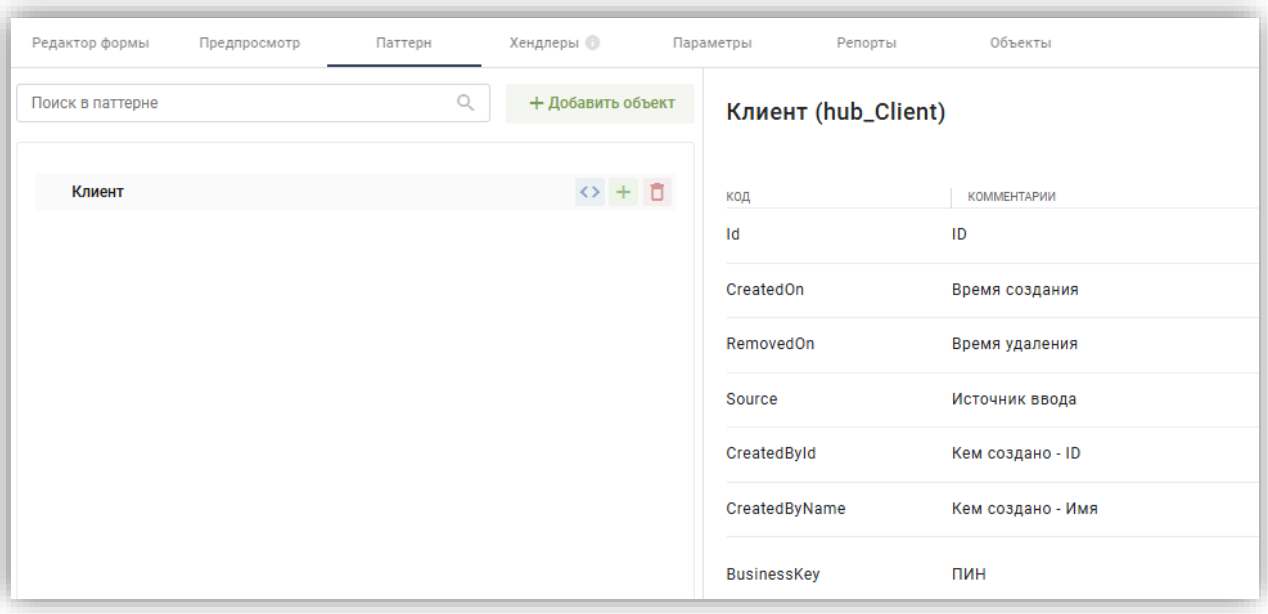
.
При нажатии на кнопку открывается список бизнес – объектов системы:



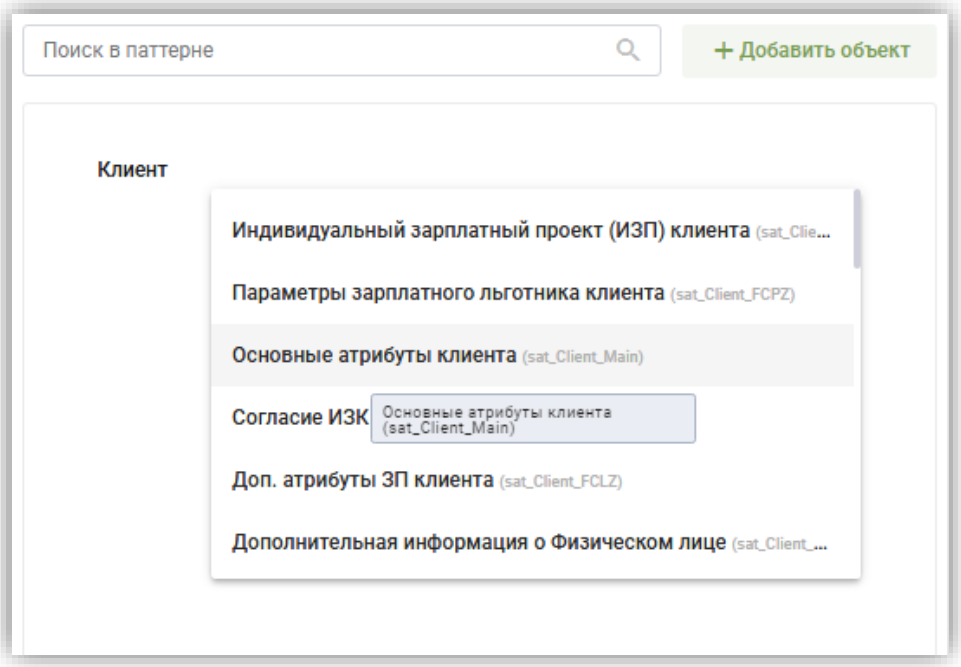
Пользователь может найти нужный бизнес-объект, используя строку поиска в этом списке – задает код хаба бизнес-объекта или название бизнес-объекта:



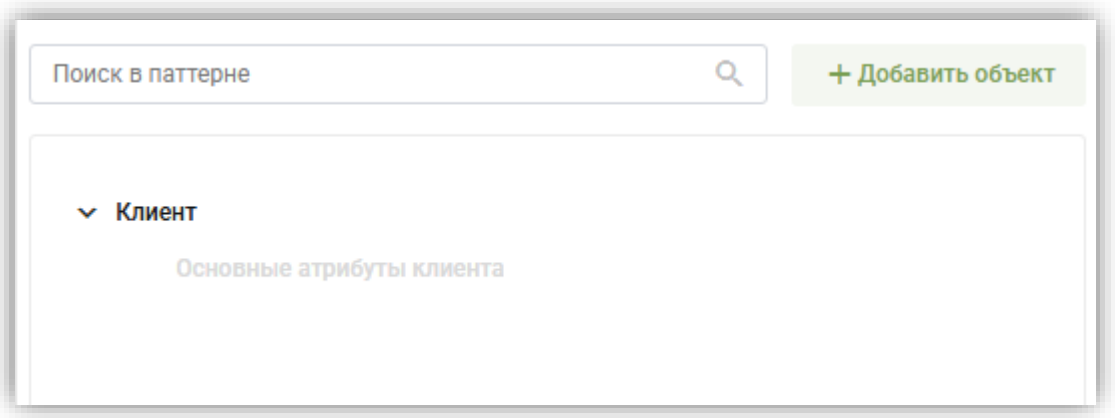
Далее пользователь выбирает бизнес-объект, и он записывается в паттерн как узел дерева патерна:



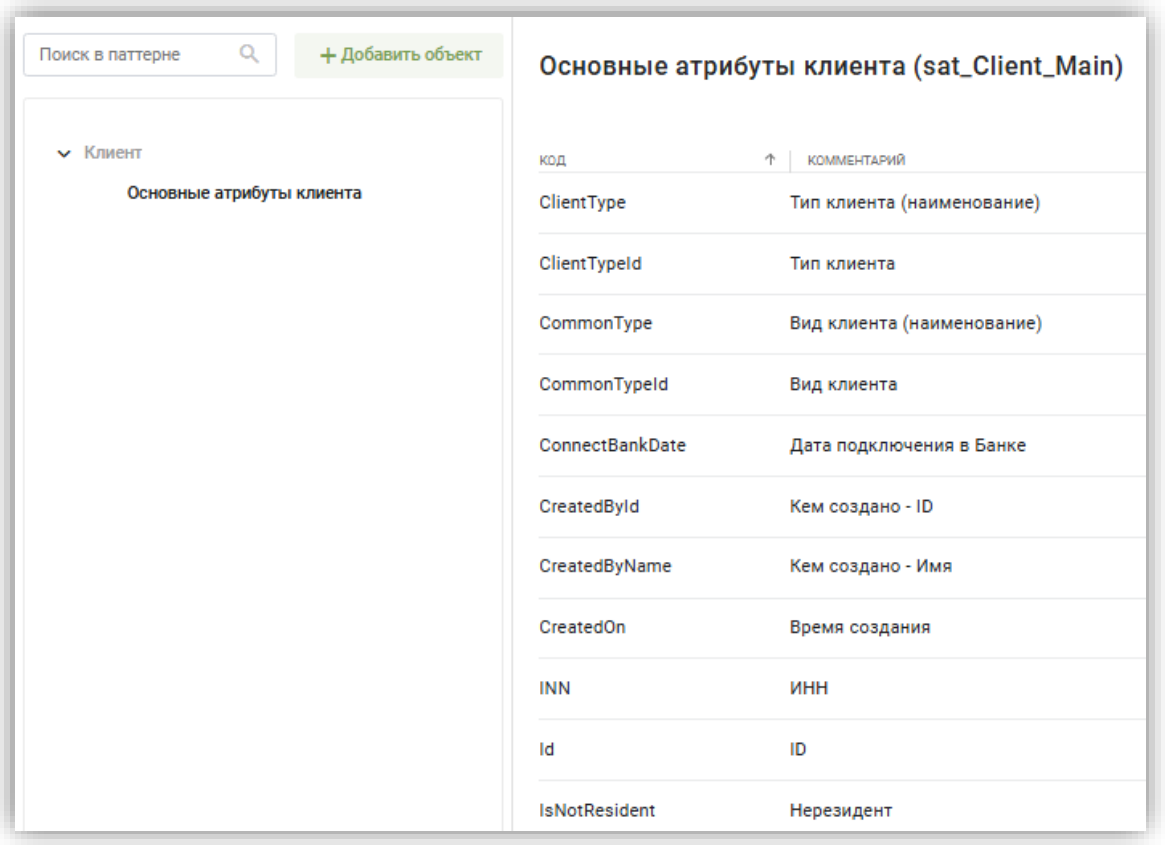
Далее к узлу бизнес-объекту нужно добавить атрибуты бизнес-объекта. Для этого пользователь кликает мышкой по узлу, при этом становятся доступны кнопки добавления / удаления групп атрибутов  . Для добавления атрибутов пользователю нужно нажать на кнопку «+», в результате откроется список групп атрибутов (сателлитов) данного бизнес-объекта:



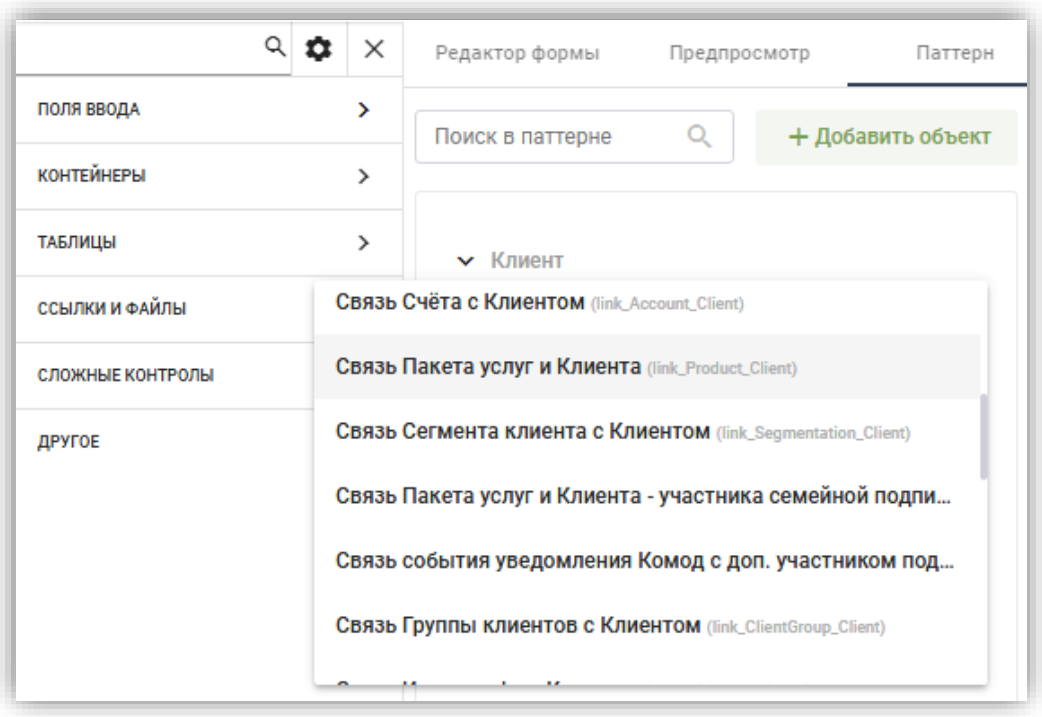
Пользователь выбирает нужный группу атрибутов, и она добавляется в иерархию:



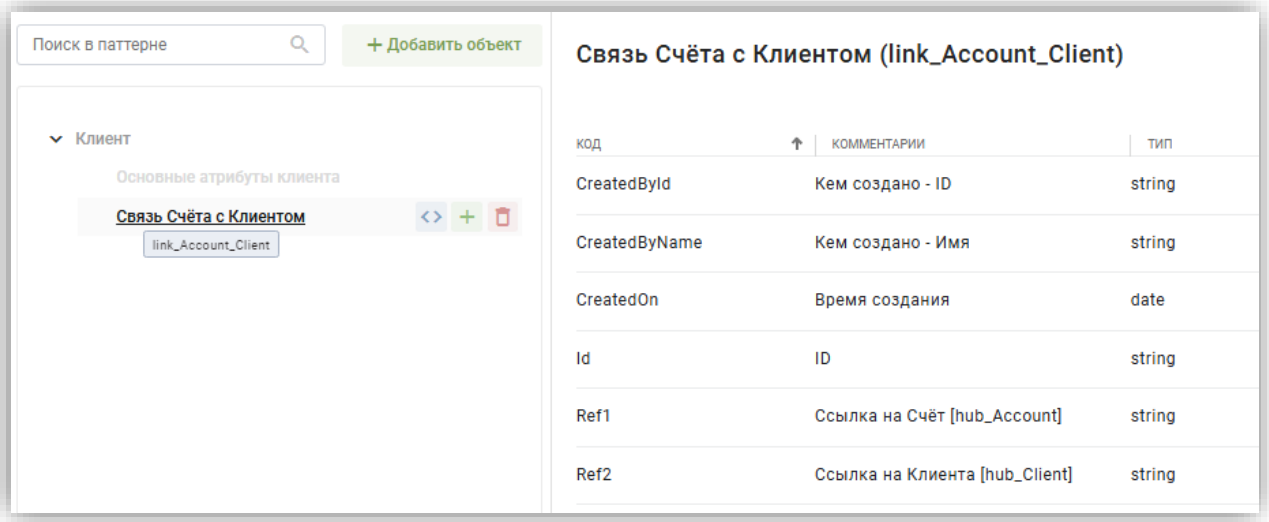
Полное наполнение группы атрибутов можно увидеть, если кликнуть мышкой по группе:



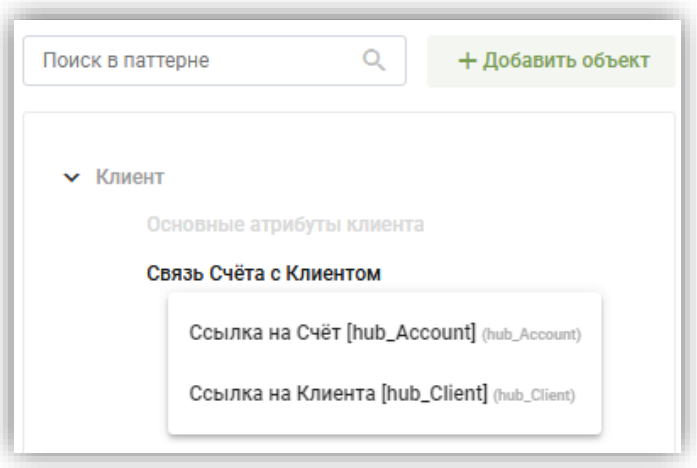
Также кроме атрибутов в паттерн можно добавить связи выбранного бизнес-объекта с другими бизнес-объектами. Для этого пользователю нужно нажать на кнопку «+» у узла бизнес-объекта и выбрать нужную связь (линк):



Выбранная связь – линк добавляется в дерево паттерна:

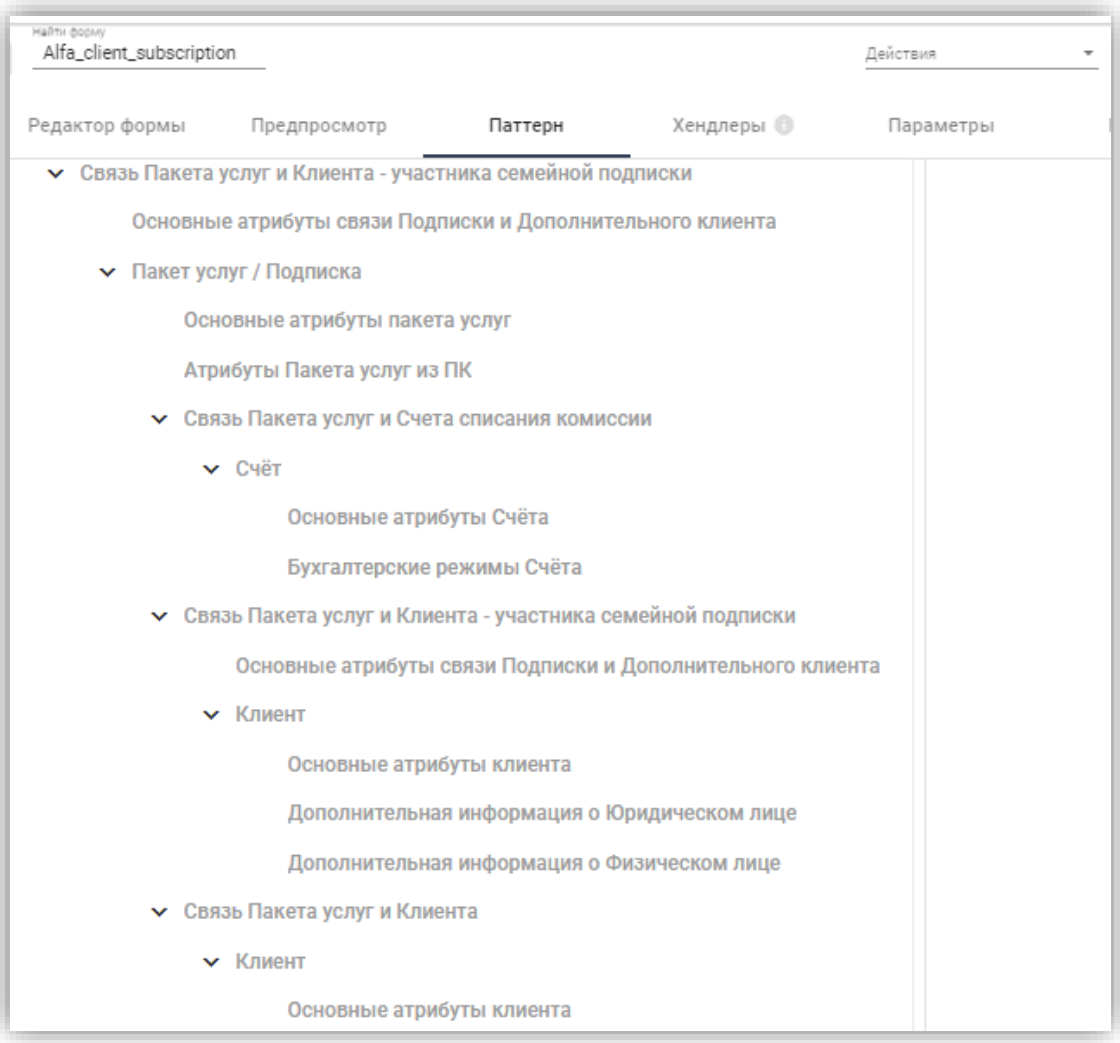


Далее к связи добавляется бизнес – объект, с которым связываем вышестоящий узел – бизнес-объект. Для этого пользователю нужно нажать на кнопку «+» у связи, при этом откроется список связанных через этот линк бизнес-объектов:



Таким образом добавляем все нужные объекты, атрибуты и связи.

Пример итогового паттерна:



После добавления паттерна на форму ее нужно сохранить. Сохранения выполняется путем нажатия на кнопку «Сохранить основной шаблон» редактора.

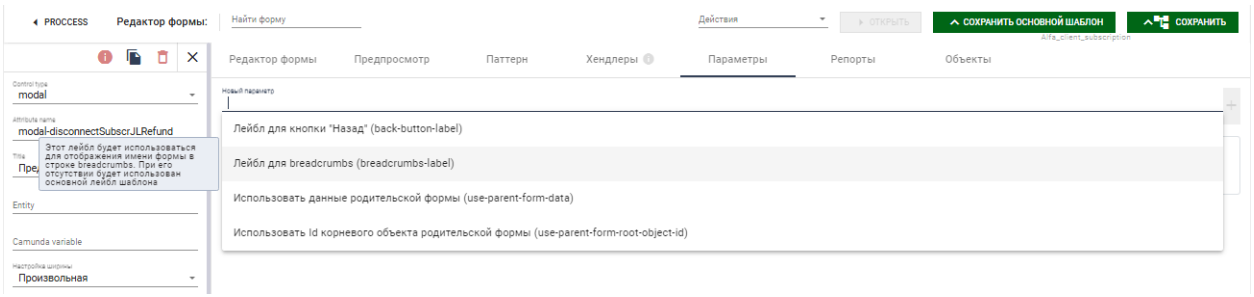
8. Закладка «Параметры»

Глобальные параметры формы. Изначально параметрами формы были имя шаблона формы Name, лейбл шаблона формы Label и имя хаба. В процессе развития имя хаба в качестве параметра стало ненужно.

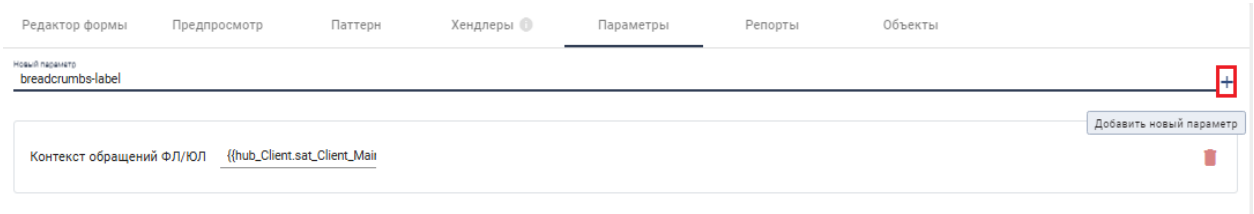
Сейчас в данный раздел добавлены следующие параметры:

- **Лейбл для кнопки «Назад»** - позволяет подменять название стандартной кнопки «Назад» на другое произвольное название.
- **Лейбл для breadcrumbs** – этот лейбл будет использоваться для отображения имени формы в строке breadcrumbs. При его отсутствии будет использован основной лейбл шаблона формы.
- **Контекст обращений ФЛ / ЮЛ** – специальный параметр для запросов в отдельные БД ФЛ / ЮЛ. Все хендлеры, виджеты, реестры добавляют в запрос этот параметр, указывающий на контекст - ФЛ или ЮЛ.
- **Использовать данные родительской формы** – используется, когда данные родительской формы еще не хранятся на сервере.
- **Использовать Id корневого объекта родительской формы** – используется, когда данные родительской формы еще не хранятся на сервере.

Для добавления глобального параметра на форму пользователю нужно выбрать нужный параметр в списке параметров:



Далее нужно нажать на кнопку «+»:



Удаление параметра выполняется нажатием кнопки .

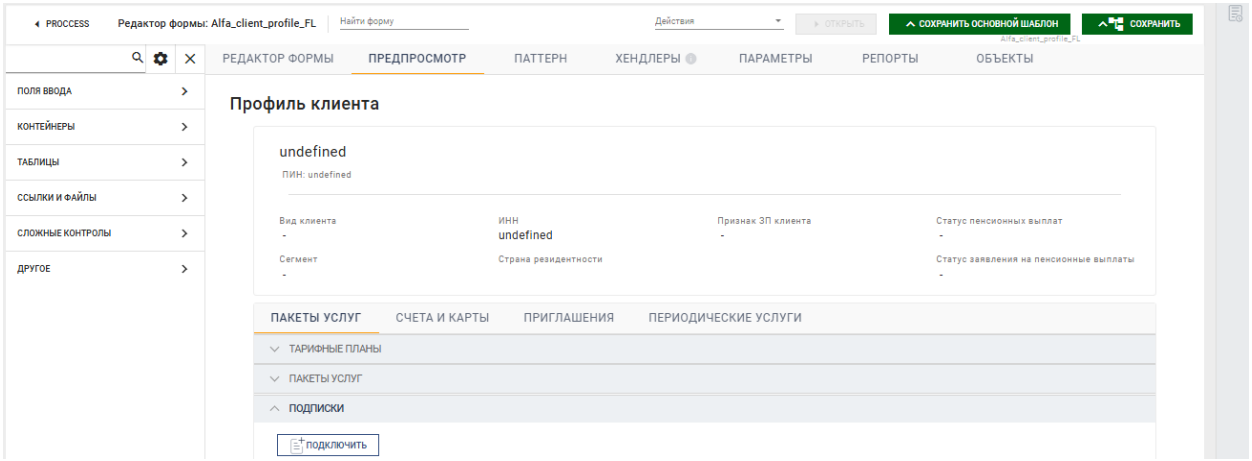
9. Закладка «Объекты»

В данном разделе редактора находится добавление на форму специального контрола object – это объект вертикального меню, которое можно разместить справа на странице формы. В объекте вызываются другие формы.

Важно: это именно правое меню, которое существует в рамках определенной формы. Не путать с левым меню приложений.

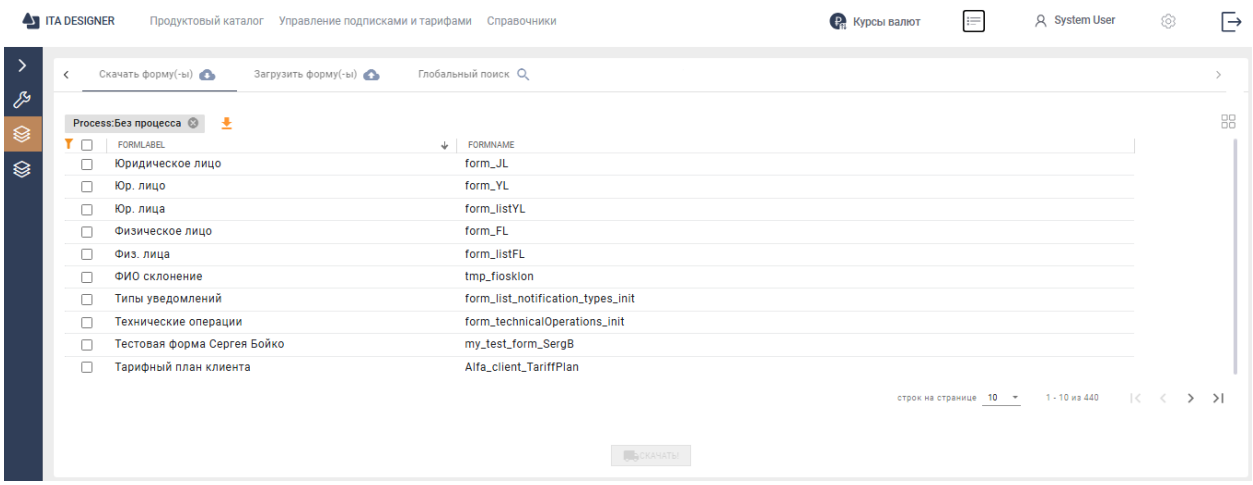
10. Закладка «Предварительный просмотр формы»

Чтобы посмотреть, какой получается конструируемая форма, пользователю не обязательно вызывать форму из приложения или запускать бизнес- процесс. В редакторе доступна функция «Предпросмотр» для предварительного просмотра формы.



11. Множественная загрузка форм

Данный функционал предназначен для ручной загрузки большого количества форм на разные стенды, ручной массовой выгрузки форм и схем бизнес – процессов для последующей загрузки на другой стенд, поиска форм.

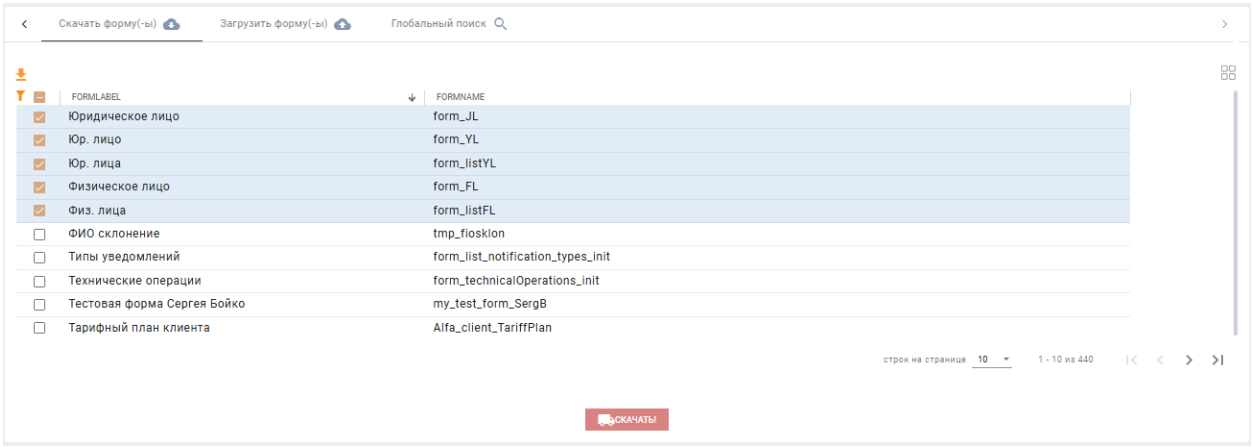


Поддерживаются следующие возможности:

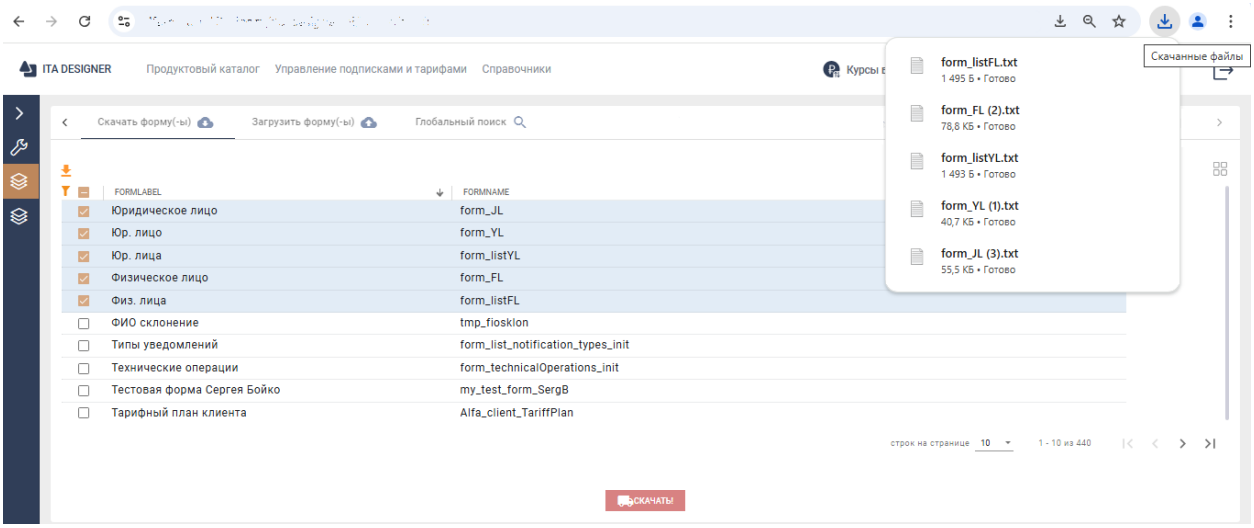
- Глобальный поиск – данная функциональность позволяет осуществлять поиск по всем зарегистрированным формам. Поиск осуществляется по всем атрибутам формы - по имени, лейблу, содержимому полей и т. д. Кликните по имени формы чтобы открыть ее в редакторе; по клику на имя поля форма также откроется в редакторе - поле будет подсвечено.
- Загрузить форму / формы – данная функциональность позволяет загружать форму или несколько форм из файлов .json или .txt

- Скачать форму / формы – данная функциональность позволяет выгружать форму или несколько форм в файлы .json или .txt для последующей загрузки на другом стенде.

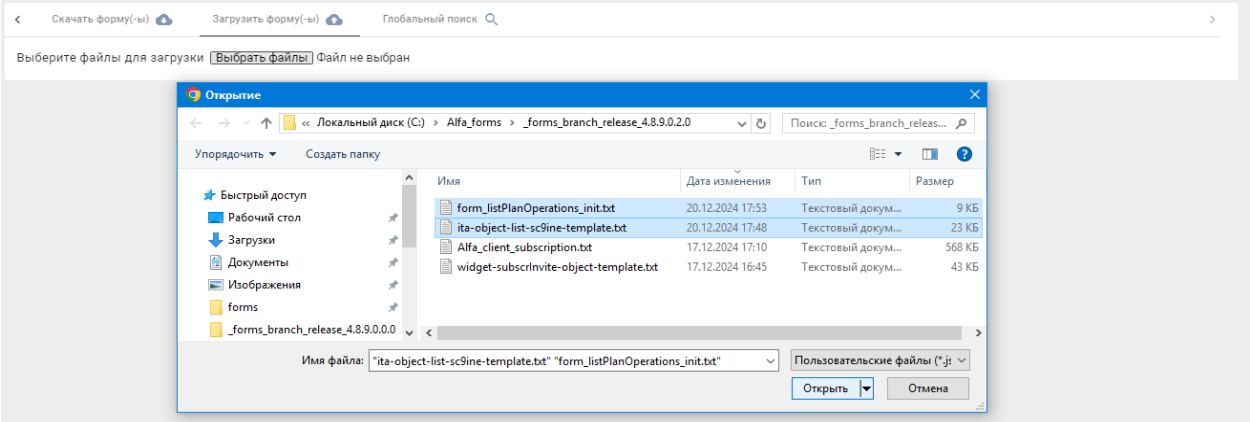
Для скачивания форм пользователю необходимо выбрать нужные формы в закладке «Скачать форму / формы». Можно также выбрать все формы.



Далее необходимо нажать на кнопку «Скачать!». Файлы скачиваются в папку загрузок браузера.

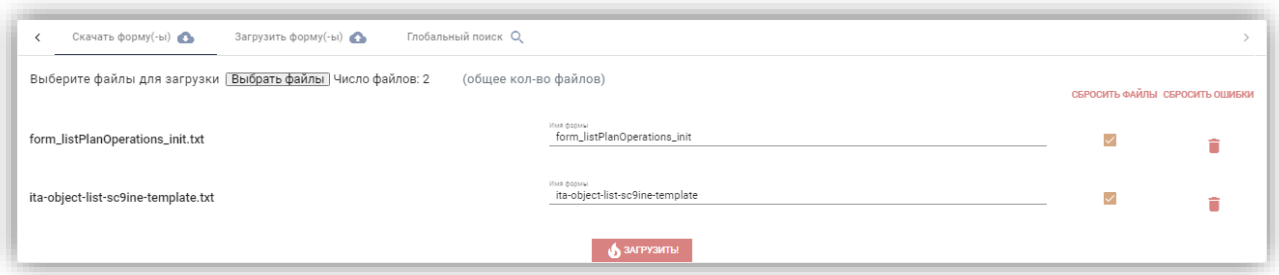


Для массовой загрузки форм на другой стенд пользователь в закладке «Загрузить форму / формы» сначала выбирает файлы для загрузки.



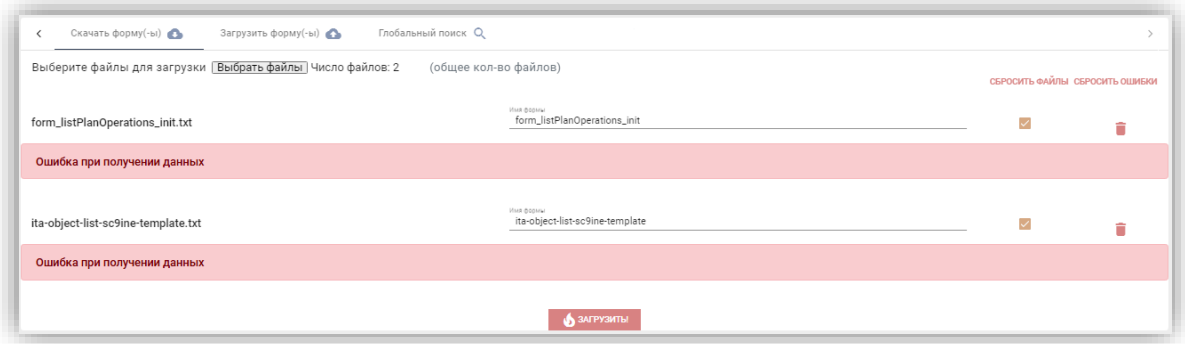
Загружать можно текстовые файлы и файлы json.

В результате пользователь получает на экране список файлов для загрузки.



Далее пользователь нажимает кнопку «Загрузить!», и все выбранные файлы загружаются в систему.

В случае каких-либо ошибок при загрузке пользователю будут выводиться сообщения, и незагруженные строки будут подсвечиваться цветом:



12. Объекты DataVault

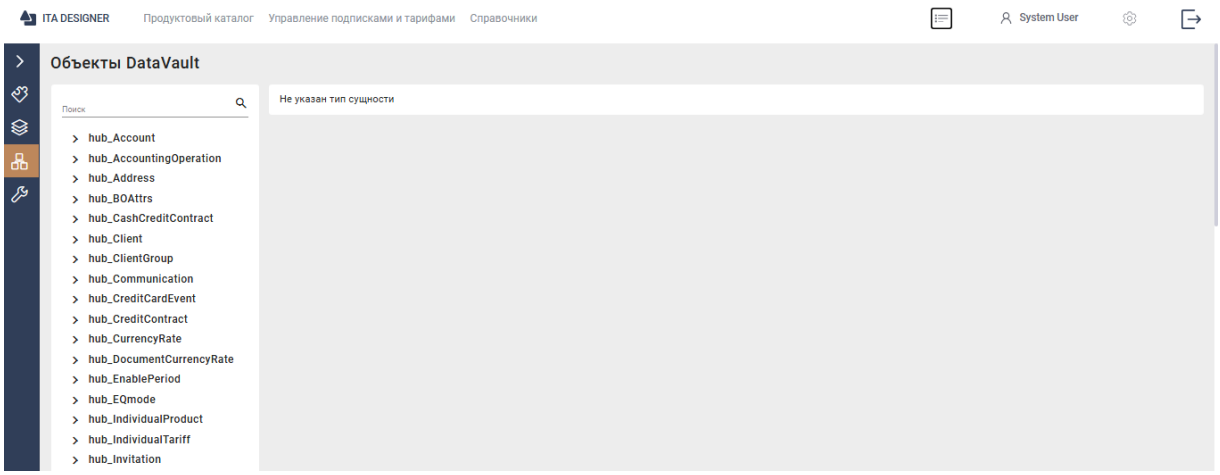
В данном разделе представлен список бизнес-объектов системы в виде модели DataVault.

DataVault — это модель хранения данных, которая представляет собой детализированный, исторически отслеживаемый и уникально связанный набор нормализованных таблиц, поддерживающий одну или несколько функциональных областей бизнеса. DataVault организует данные вокруг бизнес-процессов и функций, а не отдельных предметных

областей. Модель Data Vault содержит временные отметки размещения данных и позволяет проследить изменение хранимой информации во времени.

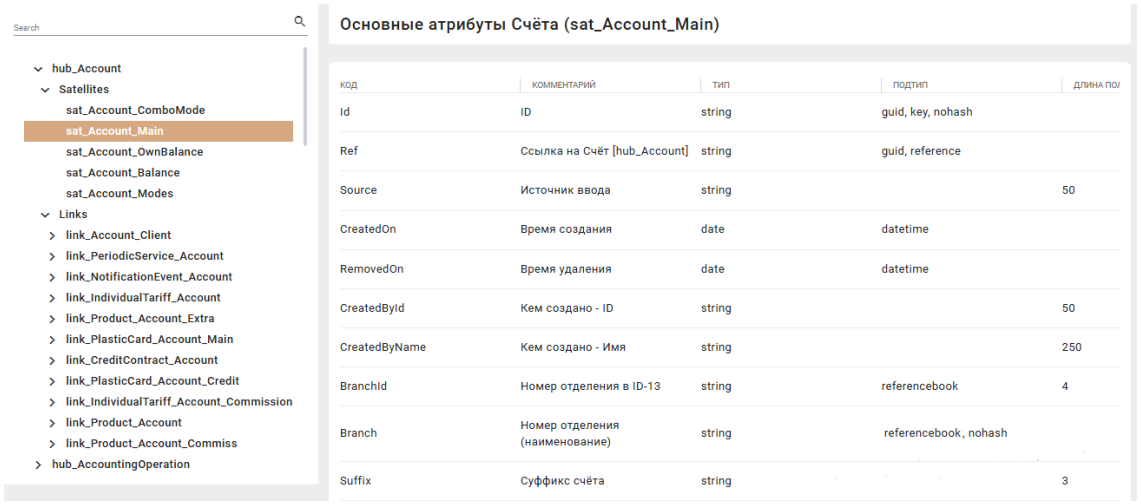
Структура Data Vault включает следующие компоненты:

- **Хаб (Hub)** — единичная бизнес-сущность, например: клиент, счет, пакет услуг. Содержит уникальный бизнес-ключ, дату загрузки и источник записи.
- **Связь (Link)** — связь между сущностями (хабами). Обеспечивает транзакционную интеграцию между ними.
- **Сателлит (Satellite)** — описательные атрибуты сущностей (хабов) и связей. Содержит суррогатный ключ родителя, значения атрибутов, дату начала действия версии, дату загрузки и источник записи.



В основном окне находится реестр бизнес-объектов системы в виде дерева, с поисковой строкой.

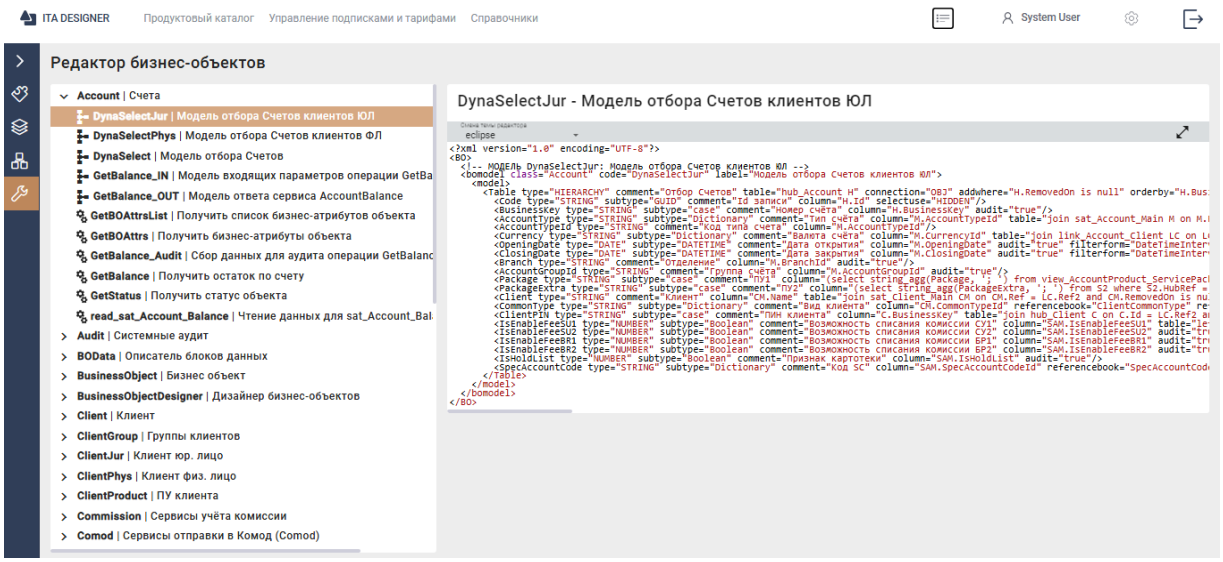
У пользователя есть возможность развернуть каждый узел дерева, провалиться на более низкий уровень и посмотреть структуру бизнес-объекта.



13. Редактор бизнес-объектов

В редакторе бизнес-объектов пользователь может описать модель данных.

В редактор бизнес-объектов пользователь попадает из меню дизайнера «Редактор бизнес-объектов», нажав на кнопку  .



В основном окне, так же, как и в реестре Объектов DataVault, отображается реестр бизнес-объектов в виде дерева. При раскрытии узла пользователю становятся доступны описания модели бизнес-объекта и операции, которые можно выполнять над бизнес-объектом.