

Creatio IDE

Разработка конфигурационных элементов

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

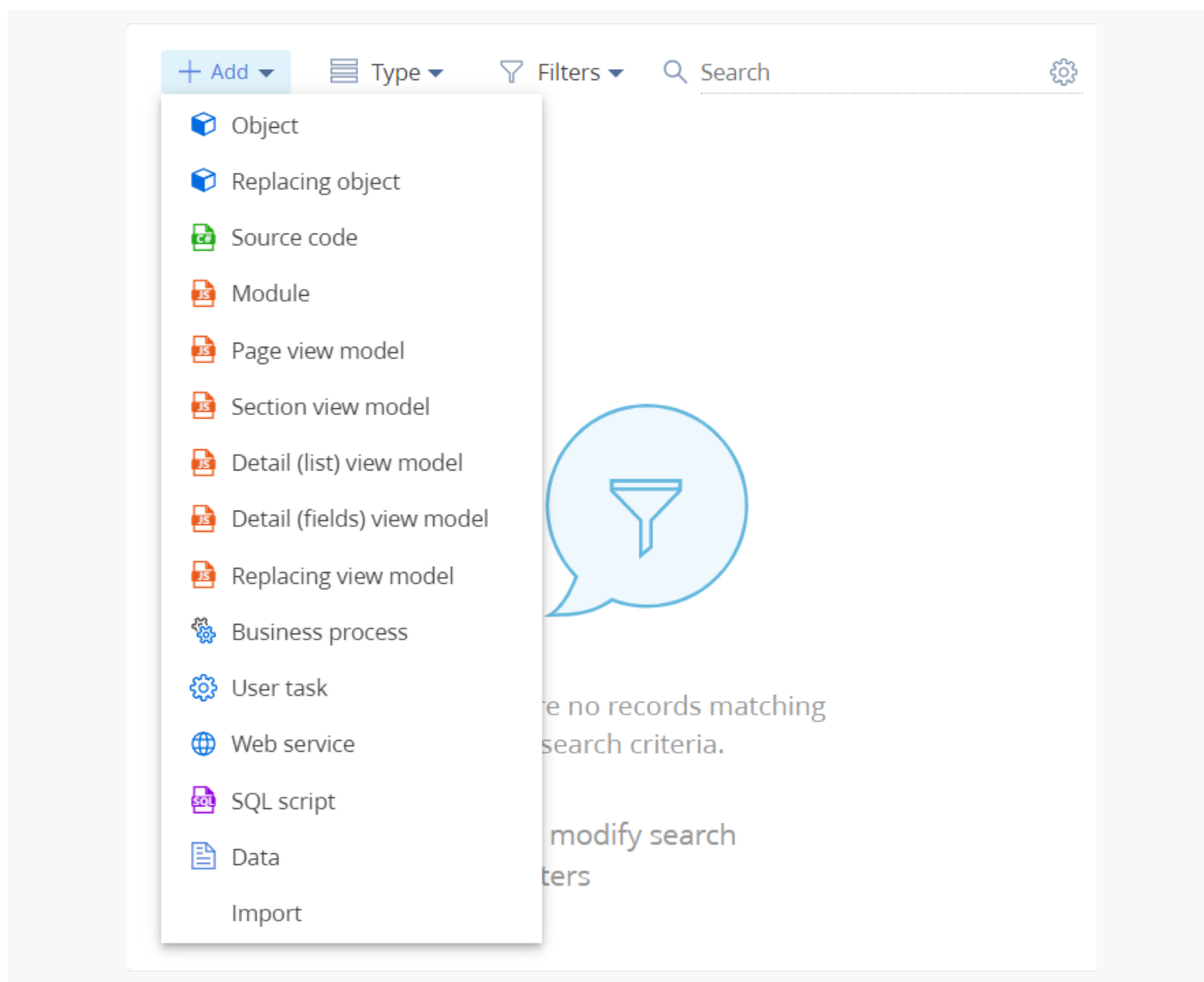
Разработка конфигурационных элементов	4
Клиентский модуль	5
Объект	11
Исходный код	20
Замещение конфигурационных элементов	22
Создать пользовательское действие процесса	23
1. Создать схему действия процесса	24
2. Добавить параметры пользовательского действия	25
3. Добавить логику пользовательского действия	26
4. Выполнить тестирование	27
Результат выполнения примера	34
Добавить действие процесса на вкладку [Элементы процесса]	35

Разработка конфигурационных элементов

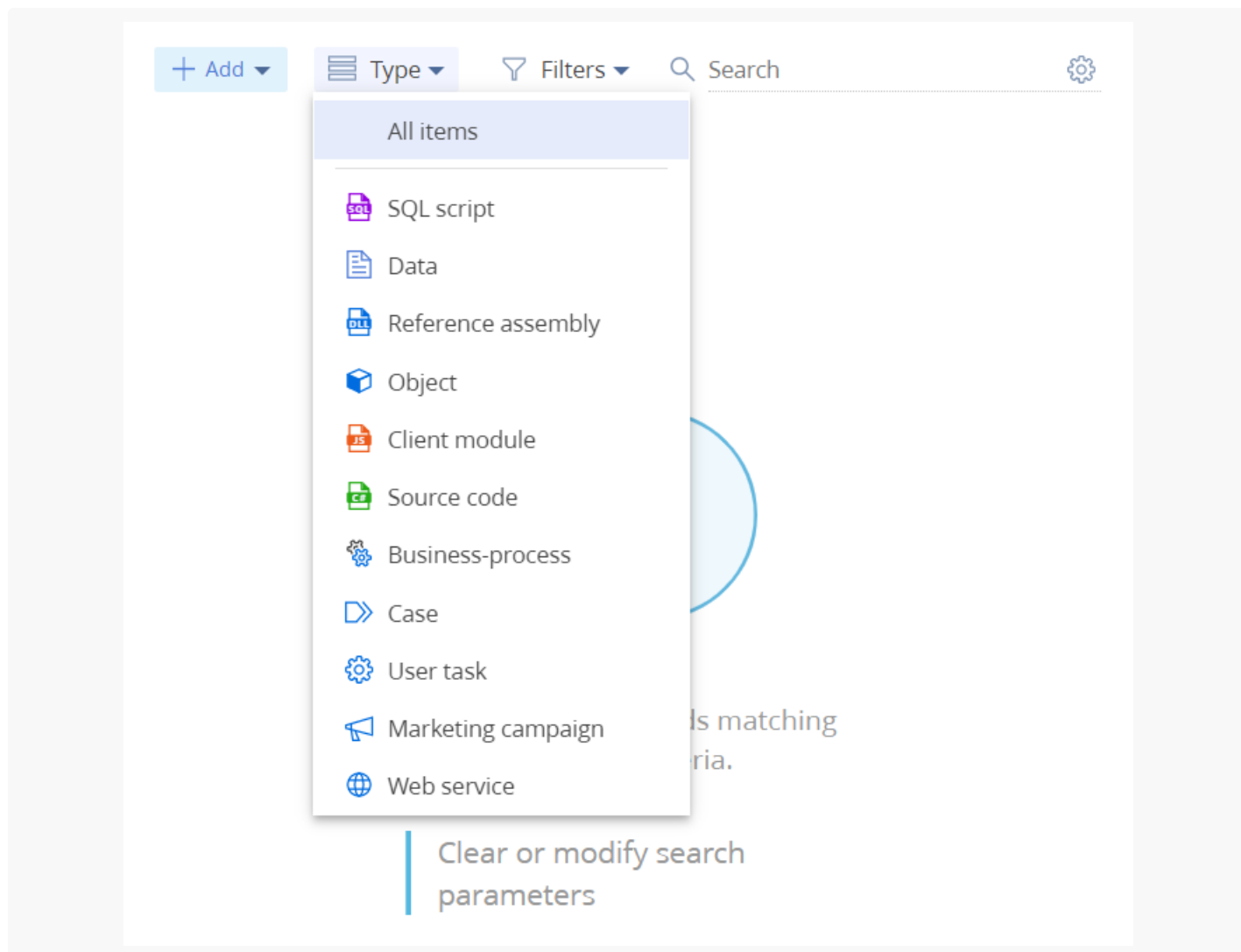
 Сложный

Схема — основа конфигурации Creatio. С точки зрения программной реализации схема — это класс ядра, который наследуется от базового класса `Schema`. Конфигурационные элементы (значения выпадающего списка [*Добавить*] ([*Add*])) панели инструментов реестра раздела [*Конфигурация*] ([*Configuration*])), представлен схемой соответствующего типа (выпадающий список [*Тип*] ([*Type*])).

Виды конфигурационных элементов



Типы схем



Клиентский модуль

Клиентский модуль — это отдельный блок функциональности, который загружается и запускается по требованию. В соответствии с подходом [AMD](#) и несмотря на некоторые функциональные различия, клиентские модули Creatio имеют одинаковую структуру описания.

Клиентские модули используются для front-end разработки (на языке JavaScript) в приложении Creatio.

Виды схем клиентских модулей:

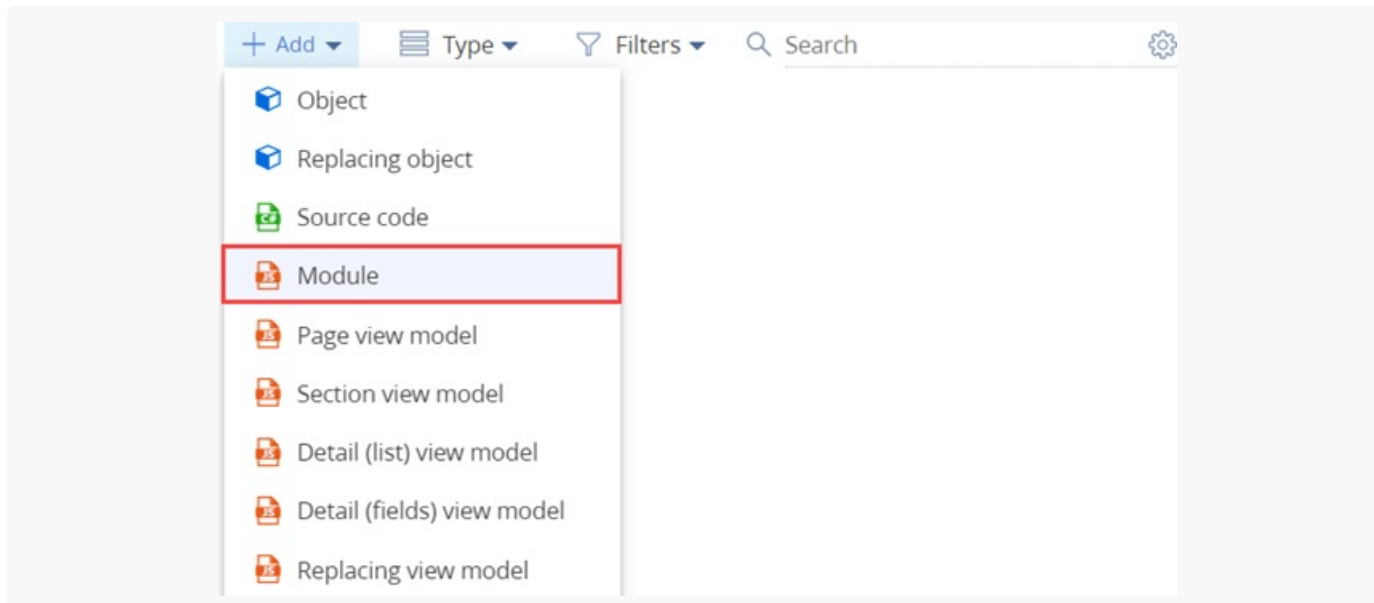
- Невизуальный модуль (схема модуля).
- Визуальный модуль (схема модели представления).
- Модуль расширения и замещающий клиентский модуль (схема замещающей модели представления).

Модули описаны в статье [Виды модулей](#).

Схема модуля

Алгоритм разработки схемы:

1. [Перейдите в раздел \[Конфигурация \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Модуль*] ([*Add*] —> [*Module*]).





3. В дизайнере модуля заполните свойства схемы.



Основные **свойства** схемы:

- [*Код*] ([*Code*]) — название схемы (обязательное свойство). Должно содержать префикс (по умолчанию `usr`), указанный в системной настройке [*Префикс названия объекта*] (код [*SchemaNamePrefix*]), символы латинского алфавита и цифры.
- [*Заголовок*] ([*Title*]) — локализуемый заголовок схемы (обязательное свойство).
- [*Пакет*] ([*Package*]) — пользовательский пакет, в котором создается схема. Заполняется автоматически и недоступно для редактирования.
- [*Описание*] ([*Description*]) — локализуемое описание схемы.

Для применения заданных свойств нажмите [Применить] ([Apply]).

Панель свойств позволяет изменить основные свойства схемы (кнопка ) и задать дополнительные (кнопка ). Дополнительными свойствами являются [Локализуемые строки] ([Localizable strings]), [Сообщения] ([Messages]), [Изображения] ([Images]).

- В дизайнере модуля добавьте исходный код. Название модуля в функции `define()` должно совпадать с названием схемы (свойство [Код] ([Code])).

Если при написании кода допущена ошибка, то слева возле номера строки отображается тип ошибки (ошибка  или предупреждение ). При наведении курсора на тип ошибки отображается всплывающая подсказка с текстовым описанием.

- На панели инструментов дизайнера модуля нажмите [Сохранить] ([Save]).

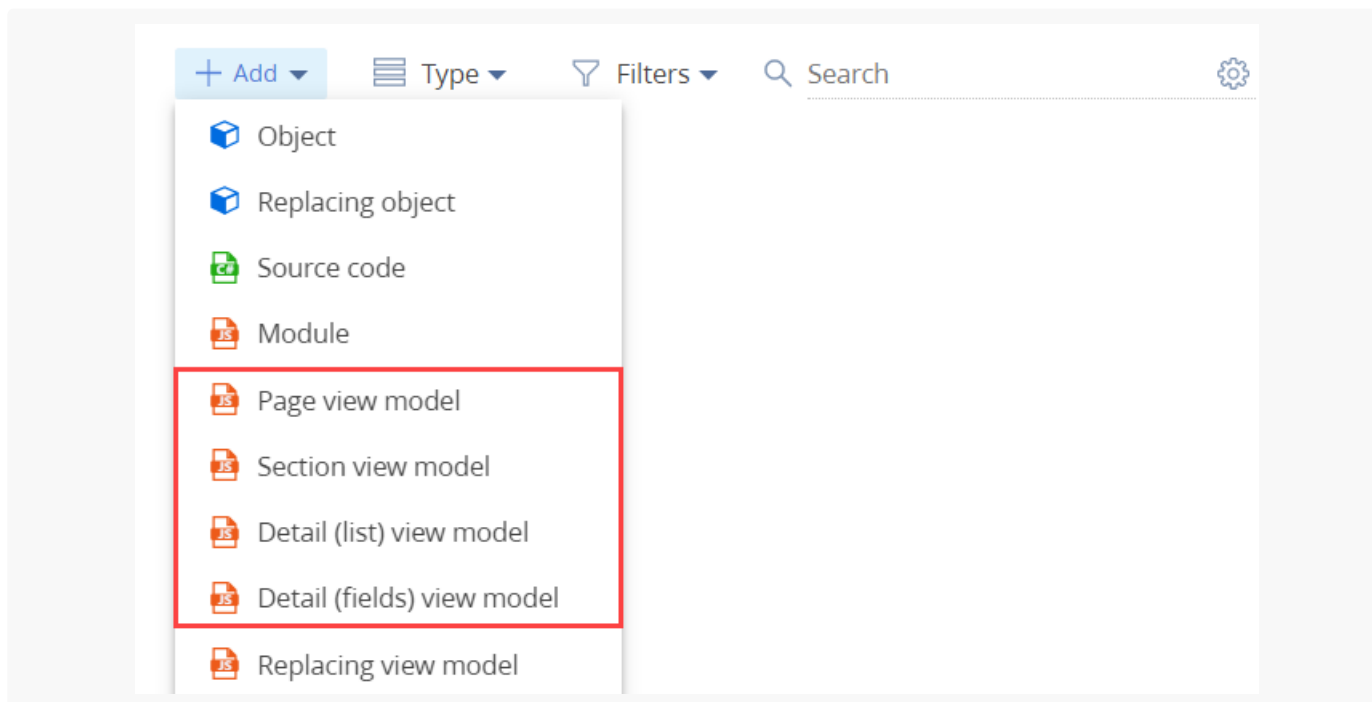
Схема модели представления

Виды схем модели представления:

- Схема страницы записи раздела (пункт [Модель представления страницы] ([Page view model])).
- Схема страницы раздела с реестром и итогами (пункт [Модель представления раздела] ([Section view model])).
- Схема страницы детали с реестром (пункт [Модель представления детали с реестром] ([Detail (list) view model])).
- Схема страницы детали с полями ([Модель представления детали с полями] ([Detail (fields) view model])).

Алгоритм разработки схемы:

1. [Перейдите в раздел \[Конфигурация \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] ([*Add*]) и выберите вид схемы модели представления.



3. В дизайнера модуля заполните свойства схемы.

Основные **свойства** схемы:



- [*Код*] ([*Code*]) — название схемы (обязательное свойство). Должно содержать префикс (по умолчанию `usr`), указанный в системной настройке [*Префикс названия объекта*] (код [`SchemaNamePrefix`]), символы латинского алфавита и цифры.
- [*Заголовок*] ([*Title*]) — локализуемый заголовок схемы (обязательное свойство).
- [*Пакет*] ([*Package*]) — пользовательский пакет, в котором создается схема. Заполняется автоматически и недоступно для редактирования.
- [*Родительский объект*] ([*Parent object*]) — родительский объект для текущего объекта. В выпадающем списке выберите родительский объект, свойства которого необходимо наследовать.
- [*Описание*] ([*Description*]) — локализуемое описание схемы.

The screenshot shows a 'Module' configuration window with the following fields and values:



- Code ***: UsrContactSection
- Title ***: Contacts section
- Parent object ***: Contacts section
- Package**: Custom
- Description**: (empty)

At the bottom right, there are two buttons: 'CANCEL' and 'APPLY'.

Для применения заданных свойств нажмите [Применить] ([Apply]).

Панель свойств позволяет изменить основные свойства схемы (кнопка ) и задать дополнительные (кнопка ). Дополнительными свойствами являются [Локализуемые строки] ([Localizable strings]) и [Изображения] ([Images]).

- В дизайнере модуля добавьте исходный код. Название модуля в функции `define()` должно совпадать с названием схемы (свойство [Код] ([Code])). Схема модели представления обязательно должна быть наследником базовой схемы `BaseModulePageV2`.

Если при написании кода допущена ошибка, то слева возле номера строки отображается тип ошибки (ошибка  или предупреждение ). При наведении курсора на тип ошибки отображается всплывающая подсказка с текстовым описанием.

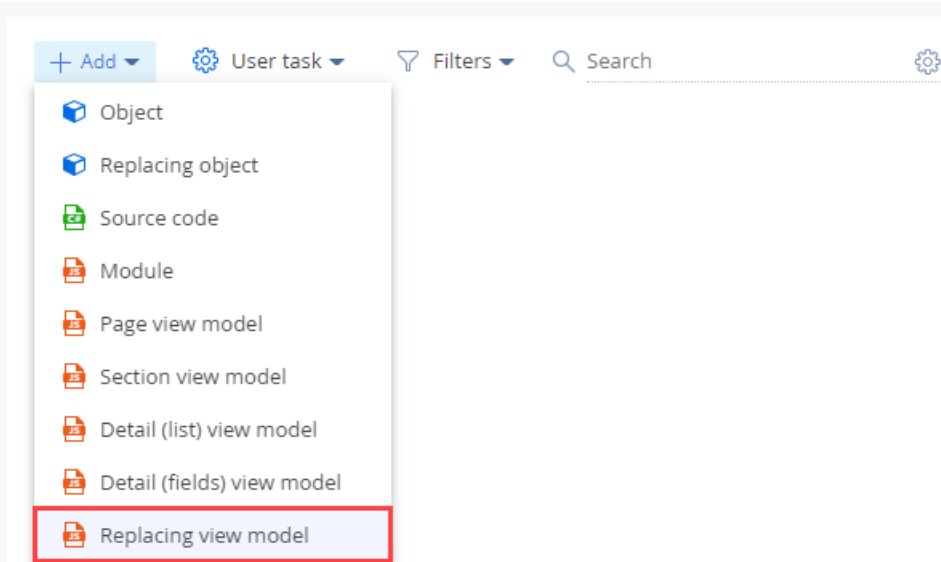
- На панели инструментов дизайнера модуля нажмите [Сохранить] ([Save]).

Схема замещающей модели представления

Схемы **замещающих моделей представления** предназначены для расширения функциональности существующих схем. При этом существующие схемы также могут быть замещающими и принадлежать разным пакетам.

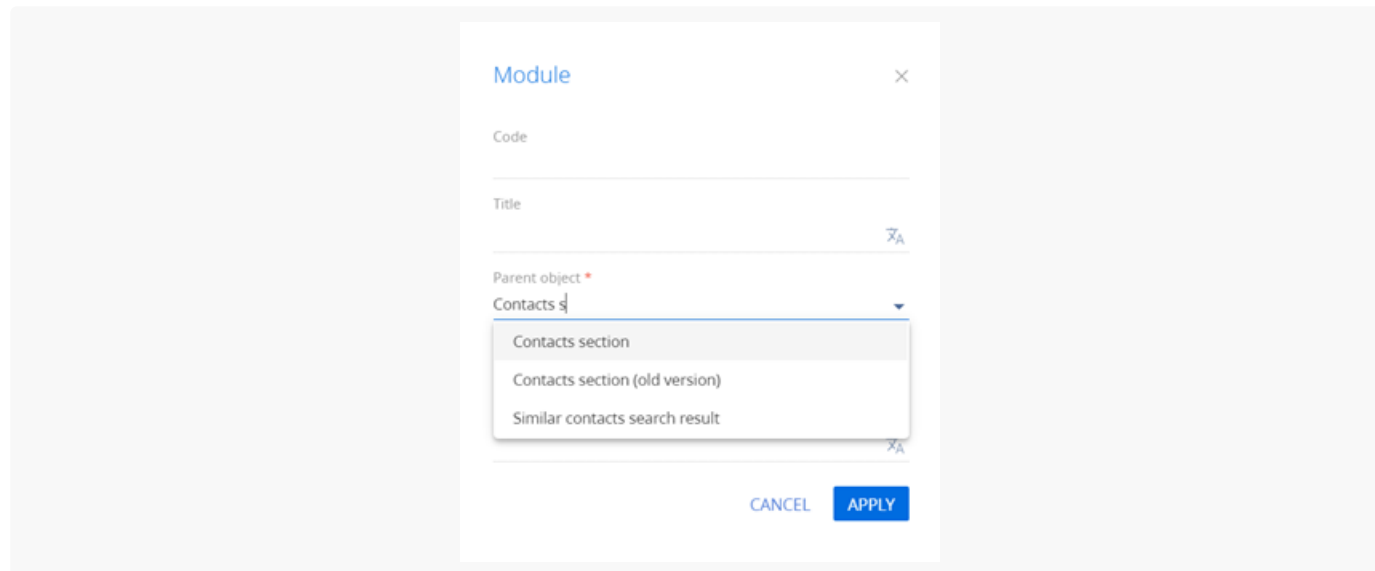
Алгоритм разработки схемы:

- Перейдите в раздел [Конфигурация] ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- Установите [зависимости](#) пакета. В зависимости обязательно добавьте пакет, который содержит замещаемый клиентский модуль.
- На панели инструментов реестра раздела нажмите [Добавить] —> [Замещающая модель представления] ([Add] —> [Replacing view model]).



4. В дизайнере модуля выберите родительский объект.

Чтобы модуль замещал раздел или страницу, в обязательном свойстве [*Родительский объект*] ([*Parent object*]) схемы укажите заголовок той базовой схемы модели представления, которую необходимо заместить. Например, для создания замещающей схемы раздела [*Контакты*] ([*Contacts*]) в качестве родительского объекта необходимо указать схему `ContactSectionV2`. Для этого в поле [*Родительский объект*] ([*Parent object*]) свойств замещающей схемы необходимо начать вводить заголовок "Раздел контакты" ("Contacts section") и выбрать нужное значение из выпадающего списка.





После подтверждения выбранного родительского объекта остальные свойства будут заполнены автоматически.

The screenshot shows a 'Module' configuration window with the following fields:



- Code ***: UsrContactSection
- Title ***: Contacts section
- Parent object ***: Contacts section
- Package**: Custom
- Description**: (empty)

At the bottom right, there are two buttons: 'CANCEL' and 'APPLY'.

Для применения заданных свойств нажмите [Применить] ([Apply]).

Панель свойств позволяет изменить основные свойства схемы (кнопка ) и задать дополнительные (кнопка ). Дополнительными свойствами являются [Локализуемые строки] ([Localizable strings]) и [Изображения] ([Images]).

- В дизайнере модуля реализуйте функциональность, которая отличает замещающий клиентский модуль от замещаемого. Название модуля в функции `define()` должно совпадать с названием схемы (свойство [Код] ([Code])).

Если при написании кода допущена ошибка, то слева возле номера строки отображается тип ошибки (ошибка  или предупреждение ). При наведении курсора на тип ошибки отображается всплывающая подсказка с текстовым описанием.

- На панели инструментов дизайнера модуля нажмите [Сохранить] ([Save]).

Объект

Объектный слой ORM ([Object-relational mapping](#)) в Creatio основан на объектах (`Entity`). **Объект** — это бизнес-сущность, которая на уровне серверного ядра позволяет объявить новый класс ORM-модели. На уровне базы данных создание объекта означает создание записи таблицы с таким же именем, как у созданного объекта, и с таким же набором колонок. То есть в большинстве случаев каждый объект в системе является системным представлением одной физической таблицы в базе данных.

Объект, как элемент конфигурации, представлен схемой, которая реализована соответствующим классом `EntitySchema`. Именно в схеме объекта описывается набор колонок, индексов и методов объекта.

Виды схем объектов:

- Базовые. Недоступны для редактирования, находятся в предустановленных [пакетах](#). Базовые схемы могут замещаться пользовательскими.

- Пользовательские. Создаются при кастомизации, находятся в пользовательских пакетах.

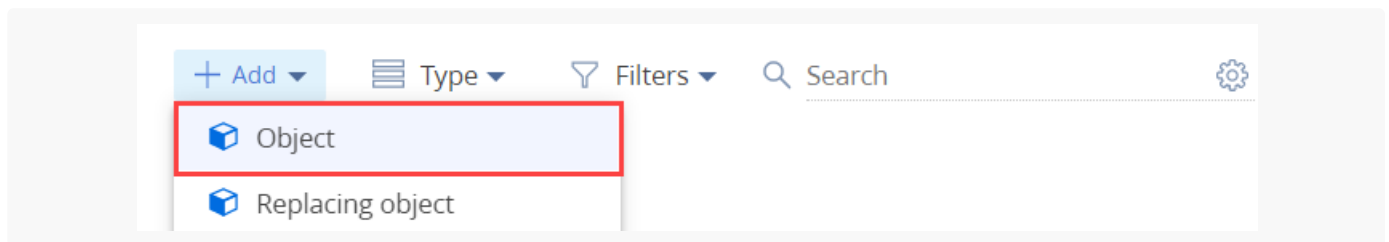
Платформа Creatio не ограничивает количество колонок объекта. Количество колонок в объекте ограничивается максимально допустимым количеством столбцов в таблицах базы данных, которую использует клиент.

Объекты используются для back-end разработки (на языке C#) в приложении Creatio.

Схема объекта

Алгоритм разработки схемы:

1. [Перейдите в раздел \[Конфигурация \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Объект*] ([*Add*] —> [*Object*]).



3. В дизайнере объекта заполните свойства схемы.

Основные **свойства** схемы:

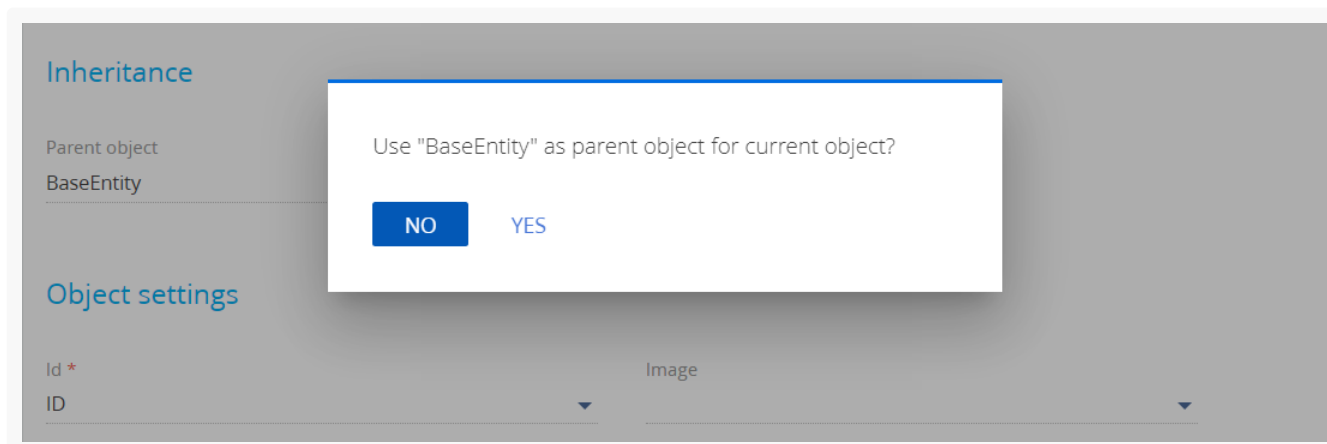
- [*Код*] ([*Code*]) — название схемы (обязательное свойство). Должно содержать префикс (по умолчанию `Usr`), указанный в системной настройке [*Префикс названия объекта*] (код [*SchemaNamePrefix*]), символы латинского алфавита и цифры. Допустимая длина имени объекта — 128 символов. На базах Oracle ниже версии 12.2 не допускаются к использованию объекты с длиной имени более 30 символов.
- [*Заголовок*] ([*Title*]) — локализуемый заголовок схемы (обязательное свойство).

 A screenshot of the 'General' tab in the Creatio object designer. It shows two main sections. The left section has a 'Code*' label and a text input field containing 'UsrEntity'. Below it, there is a 'Package' dropdown menu with 'Custom' selected. The right section has a 'Title*' label and a text input field containing 'Object'. Below it, there is a 'Description' label and a text input field. Both input fields have a small 'X' icon on the right side.

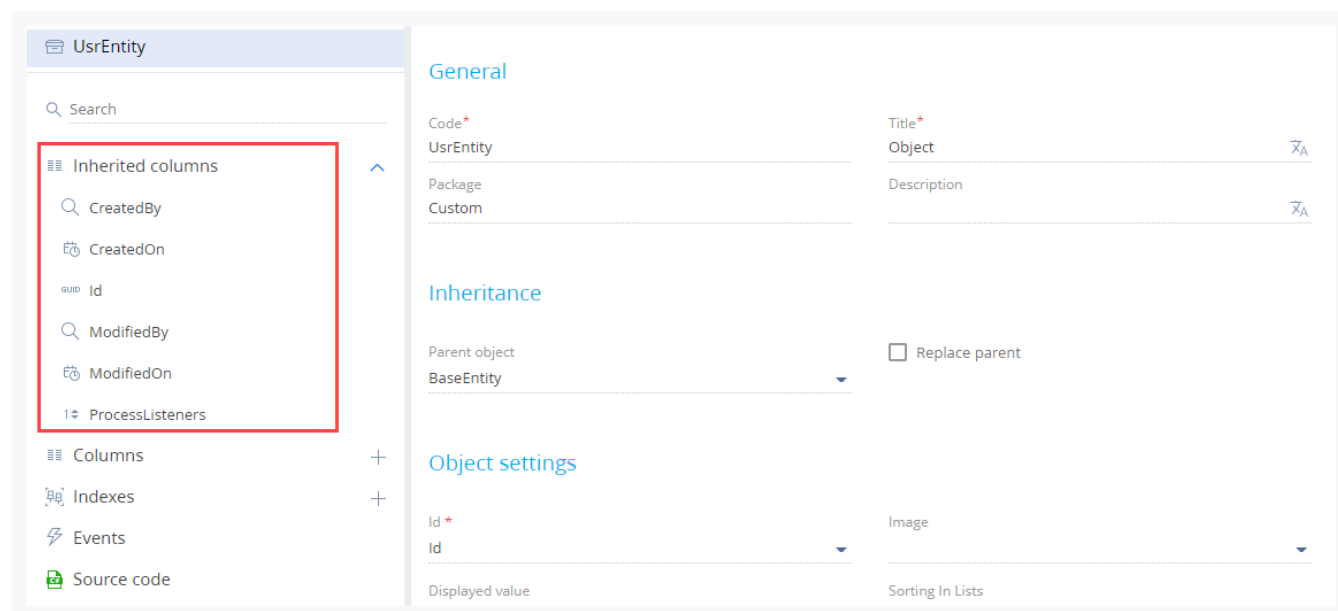
- [*Родительский объект*] ([*Parent object*]) — родительский объект для текущего объекта. Чтобы объект наследовал функциональность базового объекта, в свойстве [*Родительский объект*] ([*Parent object*]) схемы укажите код той базовой схемы объекта, функциональность которой необходимо наследовать. Например, для наследования функциональности базовой схемы `BaseEntity` в поле [*Родительский объект*] ([*Parent object*]) свойств схемы необходимо начать вводить код `BaseEntity` и выбрать нужное значение из выпадающего списка. После подтверждения выбранного родительского объекта к структуре объекта будут добавлены колонки,

унаследованные от базового объекта.

Подтверждение использования родительского объекта



Унаследованные колонки в структуре объекта



- [Идентификатор] ([Id]) — системная колонка, используемая в качестве первичного ключа в таблице базы данных (обязательное свойство). Заполняется автоматически после установки свойства [Родительский объект] ([Parent object]).

Поскольку объект в системе является представлением таблицы в базе данных, то он обязательно должен содержать колонку-идентификатор. Для установки значения свойства [Идентификатор] ([Id]) в качестве родительского объекта укажите один из базовых объектов системы или в выпадающем списке выберите пользовательскую колонку типа [Уникальный идентификатор] ([Unique identifier]). Добавление пользовательской колонки рассмотрено ниже. Если попытаться сохранить схему объекта без идентификатора, то система выдаст предупреждение.

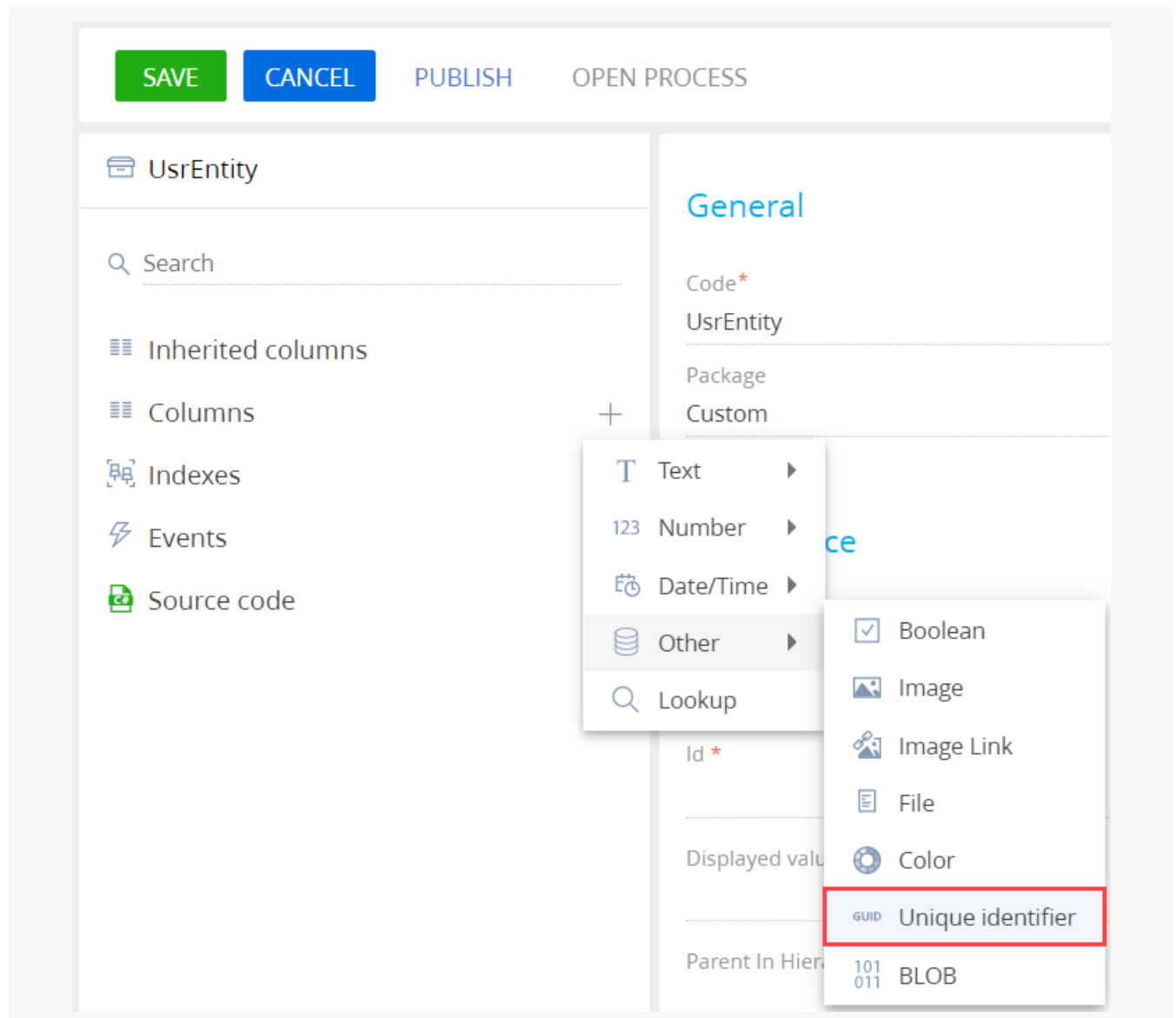
4. Добавьте пользовательскую колонку в объект.

Алгоритм добавления в объект пользовательской колонки:

- В контекстном меню узла [Колонки] ([Columns]) структуры объекта нажмите +.

- b. В выпадающем меню выберите тип колонки и задайте ее свойства.

Для добавления колонки-идентификатора нажмите [*Другие*] —> [*Уникальный идентификатор*] ([*Other*] —> [*Unique identifier*]).



- c. В дизайнера объекта заполните свойства добавляемой колонки.

Основные **свойства** добавляемой колонки:


- [*Код*] ([*Code*]) — название колонки (обязательное свойство). Значение по умолчанию устанавливается дизайнером объекта и может быть изменено.

Важно. При создании колонки объекта, значение свойства [*Код*] ([*Code*]) не должно совпадать со значением аналогичного поля родительского объекта колонки. В другом случае, при попытке опубликовать объект, будет отображено сообщение об ошибке.

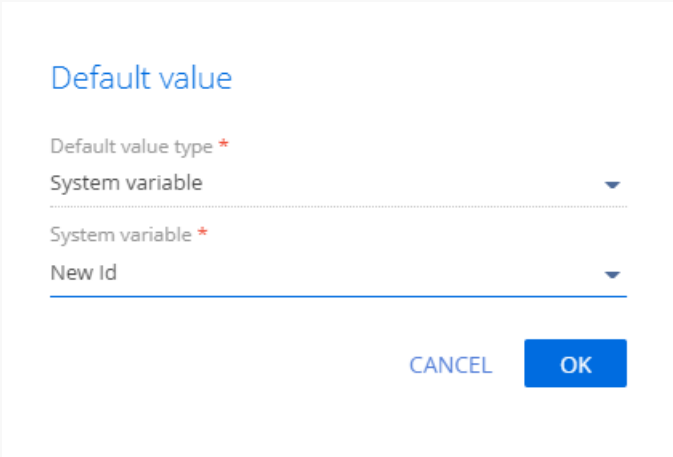
- [*Заголовок*] ([*Title*]) — локализуемый заголовок колонки (обязательное свойство).
- [*Тип данных*] ([*Data type*]) — тип данных, содержащихся в колонке. Значение по умолчанию

устанавливается дизайнером объекта в зависимости от выбранной команды добавления колонки.

- [*Обязательное*] ([*Required*]) — обязательность колонки. Выберите "На уровне приложения" ("Application Level"), поскольку колонка должна обязательно содержать значение.
- [*Значение по умолчанию*] ([*Default value*]) — значение по умолчанию.

Для установки значения по умолчанию нажмите  и заполните **поля**:

- [*Тип значения*] ([*Default value type*]) — выберите "Системная переменная" ("System variable").
- [*Системная переменная*] ([*System variable*]) — выберите "Новый идентификатор" ("New Id"), поскольку идентификаторы должны быть уникальными.



Default value

Default value type *

System variable

System variable *

New Id

CANCEL OK

- [*Режим использования*] ([*Usage mode*]) — выберите "Расширенный" ("Advanced").

General

Code*	Title*
UsrId	Id
Data type	Description
Unique identifier	
Required	Default value
Application Level	(None)

☒ Copy this value when copying records

Data source

Behavior

☐ Indexed ☐ Update change log

Usage mode

Advanced

Режимы использования колонок, реализованные в Creatio IDE:

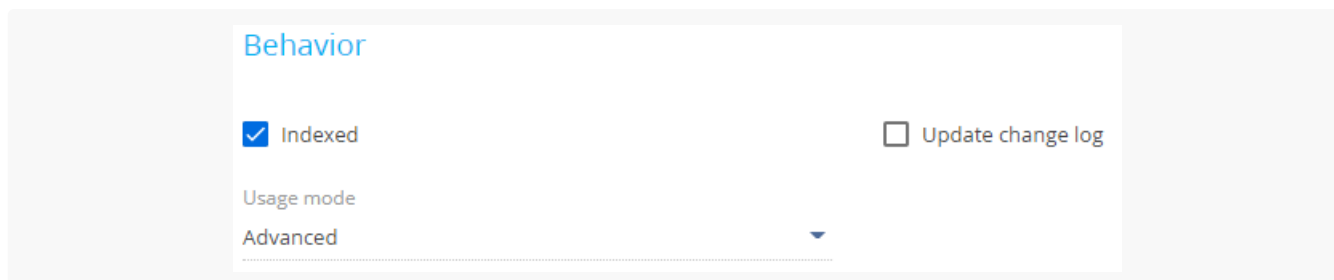
- [*Общие*] ([*General*]) — стандартный режим колонок в приложении.
- [*Расширенный*] ([*Advanced*]) — колонка отображается в конфигурации и доступна для использования в приложении.
- [*Никогда*] ([*None*]) — колонка отображается в конфигурации как системная и недоступна для использования в приложении.

l. На панели инструментов дизайнера объекта нажмите [*Сохранить*] ([*Save*]) для временного сохранения изменений в метаданных.

m. Добавьте индексы в объект.

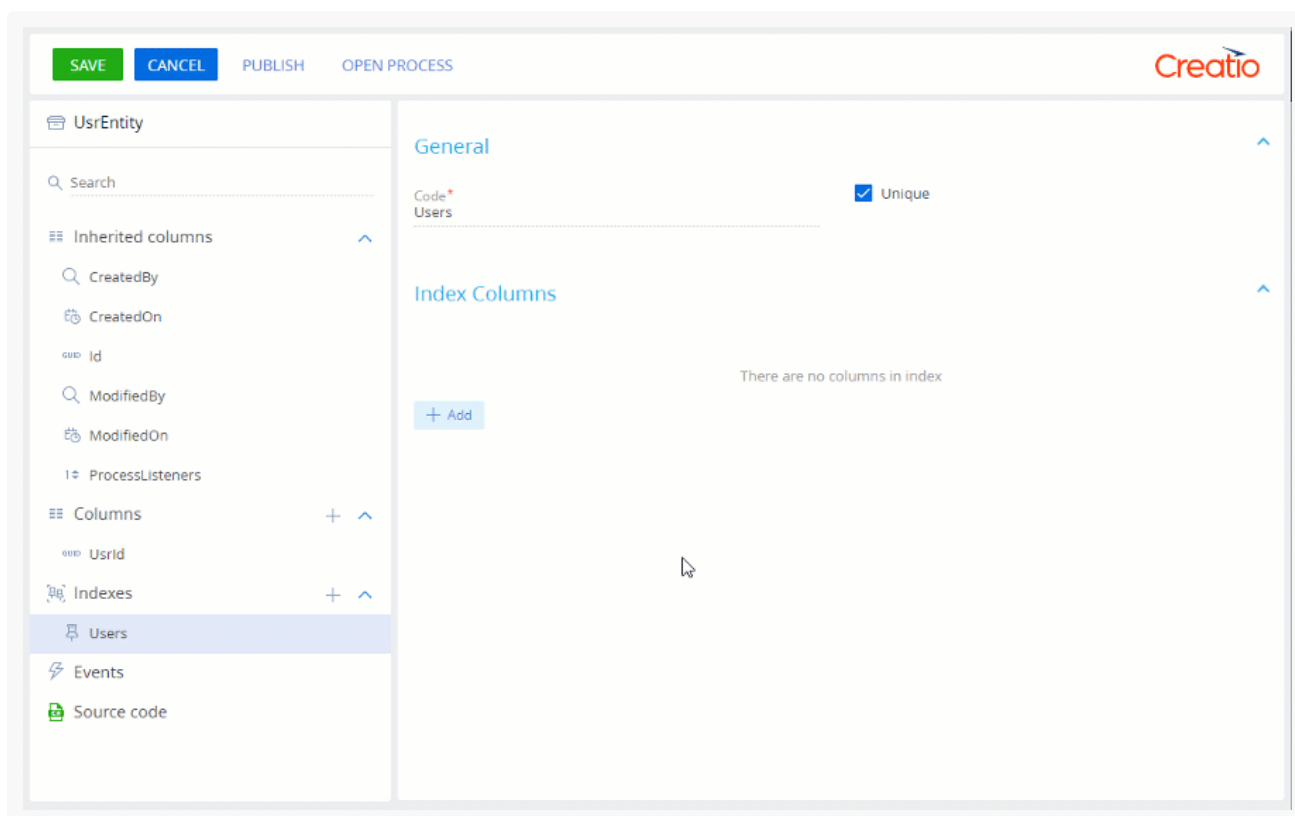
Кроме колонок, в объект могут быть добавлены индексы, которые при публикации объекта будут автоматически созданы в таблице базы данных.

В блоке свойств [*Поведение*] ([*Behavior*]) установите признак [*Индексируемая*] ([*Indexed*]), если необходимо создать индекс по одной колонке. В системе по умолчанию справочные колонки являются индексируемыми.



Алгоритм добавления составного индекса:

- Задайте название индекса. Для этого в контекстном меню элемента [*Индексы*] ([*Indexes*]) нажмите **+** и в поле [*Код*] ([*Code*]) укажите пользовательское название.
- Установите признак [*Уникальный*] ([*Unique*]) если для колонок индекса необходимо реализовать ограничение целостности (исключить возможность вставки повторяющихся комбинаций значений).
- Добавьте необходимые колонки в индекс. Для этого в блоке [*Колонки индекса*] ([*Index Columns*]) нажмите [*Добавить*] ([*Add*]), выберите колонку объекта и укажите направление сортировки.



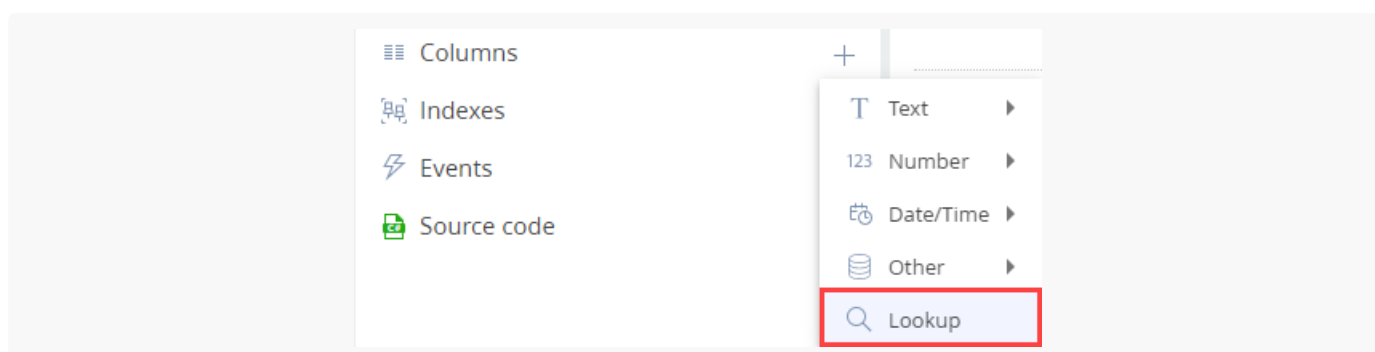
- На панели инструментов дизайнера объекта нажмите [*Сохранить*] ([*Save*]) для временного сохранения изменений в метаданных схемы.
 - На панели инструментов дизайнера объекта нажмите [*Опубликовать*] ([*Publish*]) для окончательного сохранения схемы и создания соответствующей таблицы в базе данных.
- Начиная с версии 7.18.5, кнопка [*Опубликовать*] ([*Publish*]) позволяет генерировать статический

контент и обновлять структуру базы данных. При этом компиляция конфигурации не выполняется. Это позволяет ускорить разработку объектов и замещающих объектов. Компиляция при публикации объекта необходима, если при редактировании встроенного процесса объекта он был сохранен, но не был опубликован в дизайнера процессов. Чтобы компилировать конфигурацию, генерировать статический контент и обновлять структуру базы данных, выберите [*Опубликовать и компилировать*] ([*Publish and compile*]) в выпадающем меню кнопки [*Опубликовать*] ([*Publish*]).

Для объекта можно установить **каскадную связь**. Она настраивается для колонки типа [*Справочник*] ([*Lookup*]).

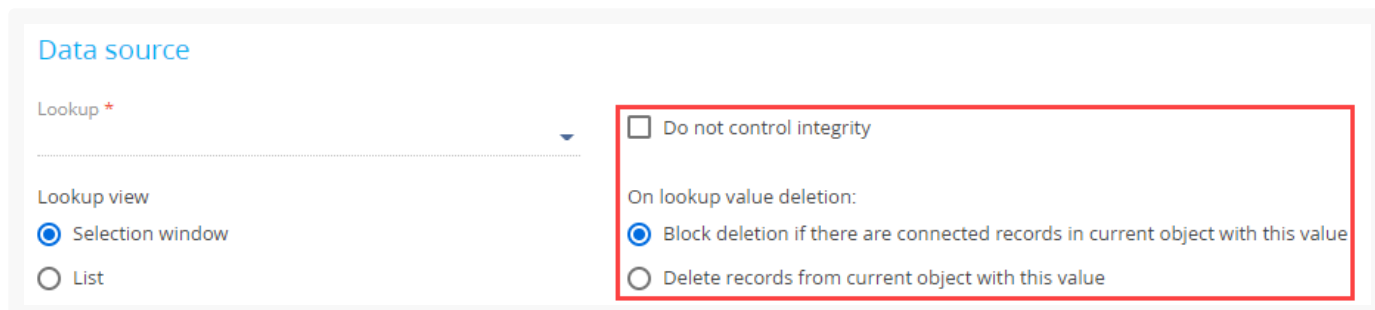
Алгоритм добавления в объект колонки типа [*Справочник*] ([*Lookup*]):

1. В контекстном меню узла [*Колонки*] ([*Columns*]) структуры объекта нажмите **+**.
2. Для добавления колонки типа [*Справочник*] ([*Lookup*]) нажмите [*Справочник*] ([*Lookup*]).



Каскадная связь настраивается в блоке свойств [*Источник данных*] ([*Data source*]) с помощью:

- признака [*Не контролировать целостность*] ([*Do not control integrity*]).
- опций пункта [*При удалении значения справочника*] ([*On lookup value deletion*]).



Рассмотрим каскадную связь на примере объекта [*Контакт*] ([*Contact*]), который связан по справочной колонке [*AccountId*] с объектом [*Контрагент*] ([*Account*]). Для этого в поле [*Выбор объекта*] ([*Lookup*]) выберите [*Account*].

Варианты настройки каскадной связи:

- Если **установлен признак** [*Не контролировать целостность*] ([*Do not control integrity*]), то удаление контрагента будет выполнено. При этом не будут удалены контакты, связанные с текущим контрагентом.
- Если **не установлен признак** [*Не контролировать целостность*] ([*Do not control integrity*]) и **выбрана опция** [*Блокировать удаление, если есть связанные записи в текущем объекте с этим*

значением] ([*Block deletion if there are connected records in current object with this value*]), то удаление контрагента будет заблокировано, если присутствуют контакты, связанные с текущим контрагентом. В этом случае приложение выдаст предупреждающее сообщение. После подтверждения удаление контрагента будет выполнено. При этом не будут удалены контакты, связанные с текущим контрагентом.

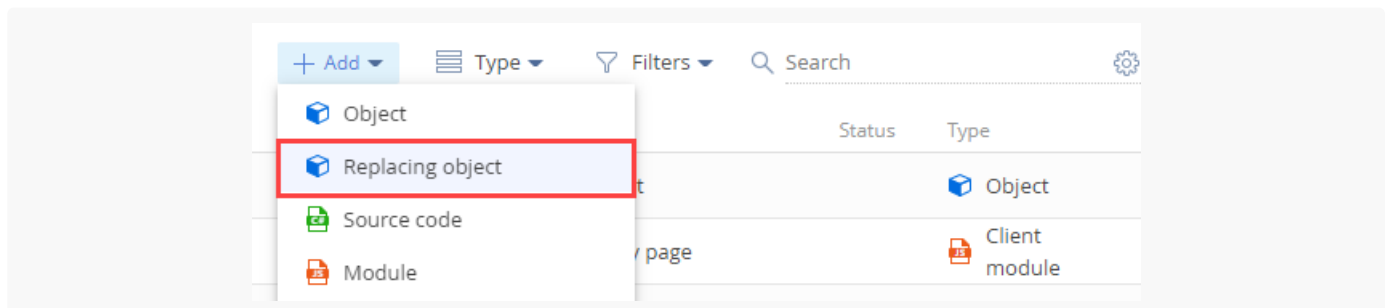
- Если **не установлен признак** [*Не контролировать целостность*] ([*Do not control integrity*]) и **выбрана опция** [*Удалять записи из текущего объекта с этим значением*] ([*Delete records from current object with this value*]), то удаление контрагента будет выполнено вместе с удалением контактов, связанных с текущим контрагентом.

Схема замещающего объекта

Схемы **замещающих объектов** предназначены для расширения функциональности существующих схем. При этом существующие схемы также могут быть замещающими и принадлежать разным пакетам.

Алгоритм разработки схемы:

1. [Перейдите в раздел \[Конфигурация \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. Установите [зависимости](#) пакета. В зависимости обязательно добавьте пакет, который содержит замещаемый объект.
3. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Замещающий объект*] ([*Add*] —> [*Replacing object*]).



4. В дизайнера объекта выберите родительский объект.

Чтобы объект замещал функциональность базового объекта, в обязательном свойстве [*Родительский объект*] ([*Parent object*]) схемы укажите название той базовой схемы объекта, функциональность которой необходимо заместить. Например, для замещения функциональности базовой схемы `BaseEntity` в поле [*Родительский объект*] ([*Parent object*]) свойств схемы необходимо начать вводить код `BaseEntity`) и выбрать нужное значение из выпадающего списка. После подтверждения выбранного родительского объекта остальные свойства будут заполнены автоматически.

General

Code*	Title*
BaseEntity	Base object
Package	Description
Custom	

Inheritance

Parent object*	<input checked="" type="checkbox"/> Replace parent
BaseEntity	

Object settings

Id*	Image
Id	
Displayed value	Sorting In Lists
Parent In Hierarchy	Owner
Change Log Object Name	Permission Object Name
Localization Object Name	

- В дизайнере объекта реализуйте функциональность, которая отличает замещающий объект от замещаемого.
- На панели инструментов дизайнера объекта нажмите [*Сохранить*] ([*Save*]) для временного сохранения изменений в метаданных схемы.
- На панели инструментов дизайнера объекта нажмите [*Опубликовать*] ([*Publish*]) для выполнения изменений на уровне базы данных. Результатом успешной публикации объекта являются созданные (или измененные) физические объекты в базе данных — таблица, столбцы, индексы.

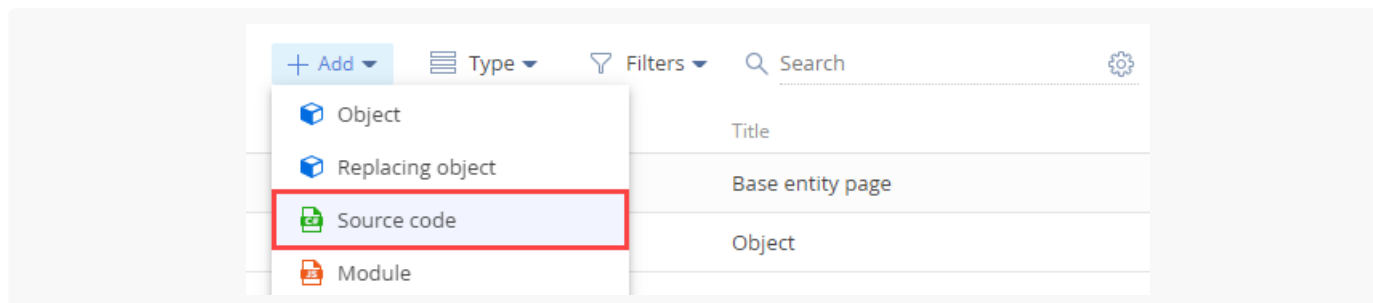
Исходный код

Схема [*Исходный код*] ([*Source code*]) используется для back-end разработки (на языке C#) в приложении Creatio.

Алгоритм разработки схемы:

- [Перейдите в раздел \[*Конфигурация* \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Исходный код*] ([*Add*] —>



[Source code]).

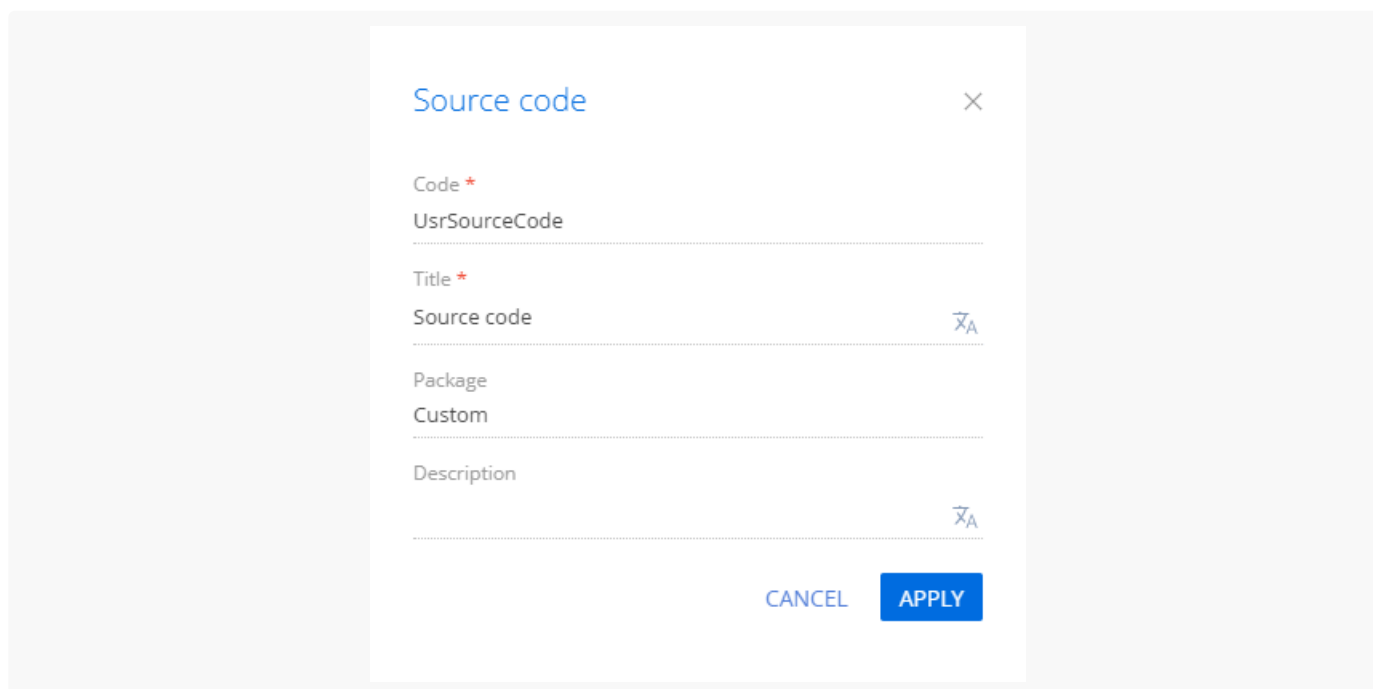


3. В дизайнера исходного кода заполните свойства схемы.

Основные **свойства** схемы:

- [Код] ([Code]) — название схемы (обязательное свойство). Должно содержать префикс (по умолчанию `Usr`), указанный в системной настройке [Префикс названия объекта] (код [`SchemaNamePrefix`]), символы латинского алфавита и цифры.
- [Заголовок] ([Title]) — локализуемый заголовок схемы (обязательное свойство).
- [Пакет] ([Package]) — пользовательский пакет, в котором создается схема. Заполняется автоматически и недоступно для редактирования.
- [Описание] ([Description]) — локализуемое описание схемы.

Панель свойств позволяет изменить основные свойства схемы (кнопка ) и задать дополнительные (кнопка ). Дополнительным свойством является [Локализуемые строки] ([Localizable strings]).



Для применения заданных свойств нажмите [Применить] ([Apply]).

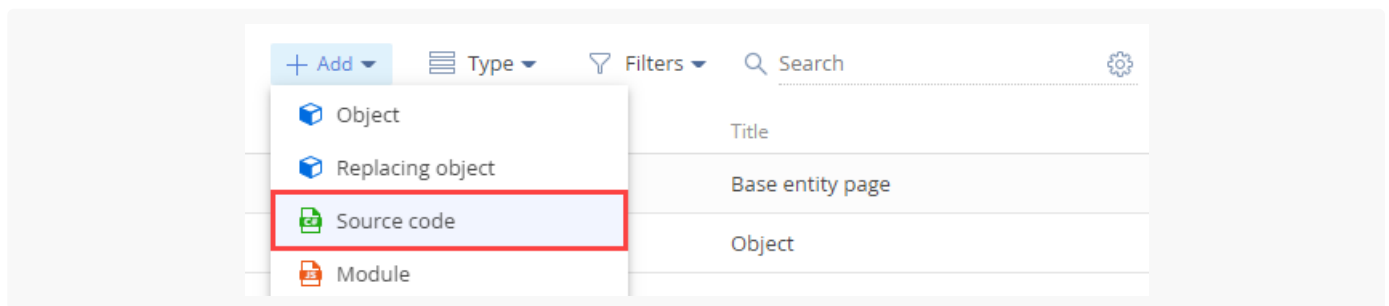
4. В дизайнера исходного кода добавьте исходный код. Название класса, объявленного в исходном коде, должно совпадать с названием схемы (свойство [Код] ([Code])).

5. На панели инструментов дизайнера исходного кода нажмите [Сохранить] ([Save]) для временного сохранения изменений в метаданных схемы.
6. На панели инструментов дизайнера исходного кода нажмите [Опубликовать] ([Publish]) для выполнения изменений на уровне базы данных.

Принцип замещения классов, в частности создание и использование в конфигурации экземпляров замещаемых классов, имеет свои особенности.

Чтобы **создать замещающий класс**:

1. [Перейдите в раздел \[Конфигурация \]](#) ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлен замещающий класс.
2. Установите [зависимости](#) пакета. В зависимости обязательно добавьте пакет, который содержит замещаемый класс.
3. На панели инструментов реестра раздела нажмите [Добавить] —> [Исходный код] ([Add] —> [Source code]).



4. Создайте класс, который является наследником замещаемого класса.
5. Для класса добавьте атрибут `[Override]`. Описание атрибута содержится в статье [Атрибут \[Override\]](#).
6. Реализуйте функциональность, которая отличает замещающий класс от замещаемого. Например, реализуйте свойства и методы, расширяющие функциональность замещаемого класса, перегрузку методов замещаемого класса и т. д.). В замещающем классе для свойств и методов добавьте модификатор `override`. В пользовательском замещаемом классе для свойств и методов, которые необходимо заместить, добавьте модификатор `virtual`. В базовом классе можно заместить только виртуальные или реализовать абстрактные методы. До компиляции все замещающие свойства и методы, которые объявлены без использования ключевого слова `override`, недоступны. Привязка и внедрение зависимостей типов выполняется open-source фреймворком внедрения зависимостей [Ninject](#) только во время выполнения.

Замещение конфигурационных элементов

Разработка на платформе Creatio базируется на основных принципах объектно-ориентированного программирования. В частности, модель расширения Creatio основана на **принципе открытости-закрытости**, при котором основная логика приложения закрыта для изменения напрямую, но открыта для расширения. Это означает, что функциональность должна разрабатываться путем добавления новых сущностей, а не путем изменения существующих.

[Конфигурационные элементы](#), расположенные в предустановленных пакетах, недоступны для изменения на уровне системы. Разработку и модификацию функциональности необходимо выполнять в пользовательских [пакетах](#) с использованием **механизма замещения**. При реализации замещения в

Creatio используются понятия замещающего и замещаемого конфигурационных элементов.

Замещающий конфигурационный элемент — конфигурационный элемент, который замещает другой конфигурационный элемент соответствующего типа.

Замещаемый конфигурационный элемент — конфигурационный элемент, который замещен другим конфигурационным элементом соответствующего типа.

Конфигурационные элементы, которые можно заместить в Creatio:

- **Клиентский модуль, определяющий модель представления.**

Клиентский модуль реализует front-end часть приложения. Для создания замещающего клиентского модуля необходимо использовать схему замещающей модели представления, которая описана в статье [Разработка конфигурационных элементов](#).

- **Объект.**

Объект реализует back-end часть приложения. Для создания замещающего объекта необходимо использовать схему замещающего объекта, которая описана в статье [Разработка конфигурационных элементов](#).

- **Исходный код.**

В качестве замещающего конфигурационного элемента выступает класс. Для создания замещающего класса необходимо использовать схему типа [*Исходный код*] ([*Source code*]), которая описана в статье [Разработка конфигурационных элементов](#).

После реализации замещающего конфигурационного элемента при обращении к замещаемому конфигурационному элементу система будет выполнять логику замещающего конфигурационного элемента.

Замещение одного и того же конфигурационного элемента можно выполнять в нескольких пользовательских пакетах. При этом конечная реализация замещающего конфигурационного элемента в скомпилированной конфигурации определяется [иерархией пакетов](#), содержащих замещающие конфигурационные элементы.

Создать пользовательское действие процесса



Во время работы с бизнес-процессами в Creatio возникает необходимость выполнять однотипные операции. Для этих целей используется элемент [*Выполнить действие процесса*] ([*User Task*]), для которого существует возможность выбрать наиболее подходящий в конкретной ситуации тип действия — [*Пользовательское действие*] ([*User Task*]). Подробнее об элементе [*Выполнить действие процесса*] ([*User Task*]) можно узнать из статьи [Элемент процесса \[*Выполнить действие процесса* \]](#).

По умолчанию в системе доступен набор преднастроенных пользовательских действий, однако, могут возникнуть ситуации, когда для выполнения определенного бизнес-процесса необходимо создать новое пользовательское действие.

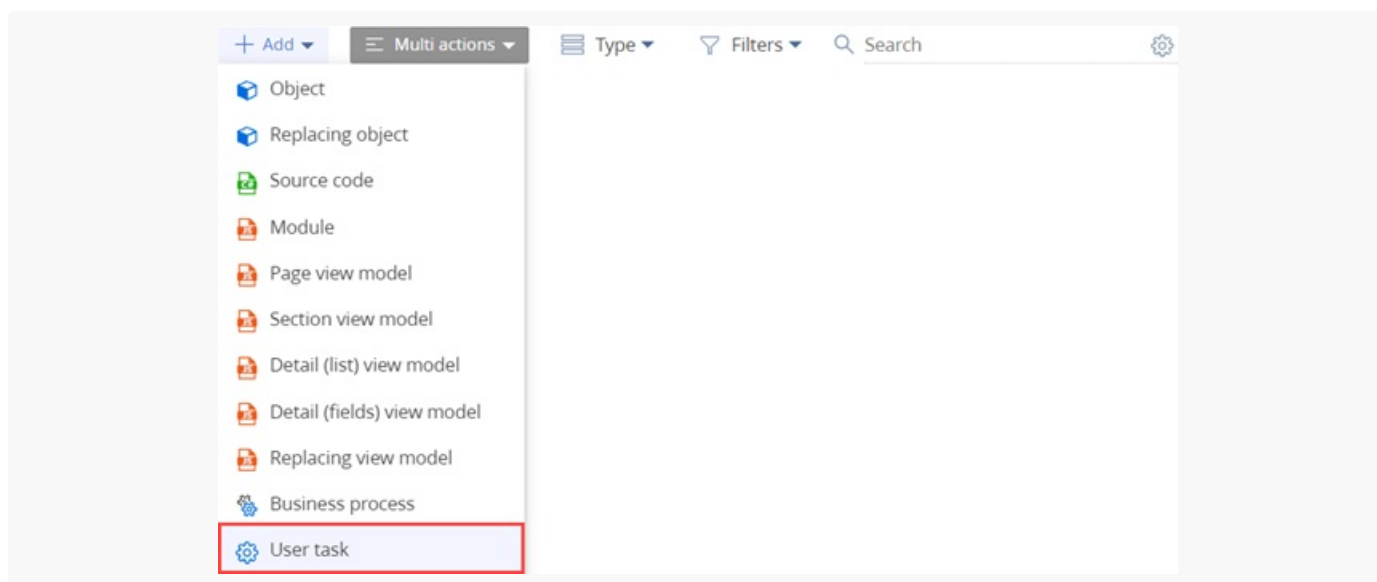
Создать новое пользовательское действие можно с помощью конфигурационной схемы [*Действие процесса*] ([*User Task*]). В простой реализации действие процесса частично повторяет логику элемента процесса [*Задание-сценарий*] ([*Script Task*]). При этом, созданное пользовательское действие можно

многократно использовать в разных процессах. При внесении изменений в пользовательское действие, эти изменения будут применены ко всем процессам, в которых используется текущее пользовательское действие.

Пример. Создать простое пользовательское действие процесса, которое вычисляет сумму двух чисел. Для вычисления суммы используйте два числа, которые задаются в качестве параметров действия.

1. Создать схему действия процесса

1. [Перейдите в раздел \[Конфигурация \]](#) ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [Добавить] —> [Действие процесса] ([Add] —> [User Task]).



3. В дизайнере схем заполните свойства схемы:

- [Код] ([Code]) — "UsrSampleProcessUserTask".
- [Заголовок] ([Title]) — "User Task Sample".

User task

Code *
UsrSampleProcessUserTask

Title *
User Task Sample

Package *
sdkCreatingUserTask

Description

Parameters edit page

Dcm parameters edit page

Color
#839DC3

Small vector image

Large vector image

CANCEL APPLY

Для применения заданных свойств нажмите [Применить] ([Apply]).

2. Добавить параметры пользовательского действия

Назначение параметров пользовательского действия — вернуть результат выполнения действия.

Чтобы **добавить параметры**:

1. В узле [Параметры] ([Parameters]) нажмите кнопку **+**.
2. Заполните **свойства параметра**:
 - [Код] ([Code]) — "FirstNumber".
 - [Заголовок] ([Title]) — "First number".
 - [Тип] ([Type]) — выберите "Целое" ("Integer").
 - Установите признак [Сериализуемый] ([Serializable]).

3. Добавьте второй параметр пользовательского действия:

- [Код] ([Code]) — "SecondNumber".
- [Заголовок] ([Title]) — "Second number".
- [Тип] ([Type]) — выберите "целое" ("Integer").
- Установите признак [Сериализуемый] ([Serializable]).

4. Добавьте третий параметр пользовательского действия:

- [Код] ([Code]) — "SumOfNumbers".
- [Заголовок] ([Title]) — "Sum of numbers".
- [Тип] ([Type]) — выберите "целое" ("Integer").
- Установите признак [Сериализуемый] ([Serializable]).

3. Добавить логику пользовательского действия

Логика работы пользовательского действия задается в методе `InternalExecute()` автогенерируемого исходного кода схемы пользовательского действия.

1. В дизайнера схемы измените метод `InternalExecute()` для реализации необходимой бизнес-логики.

Метод InternalExecute()

```
protected override bool InternalExecute(ProcessExecutingContext context) {
    // Выполнение операций с параметрами действия.
    SumOfNumbers = FirstNumber + SecondNumber;
    // Указывает на успешное выполнение сценария действия.
    return true;
}
```

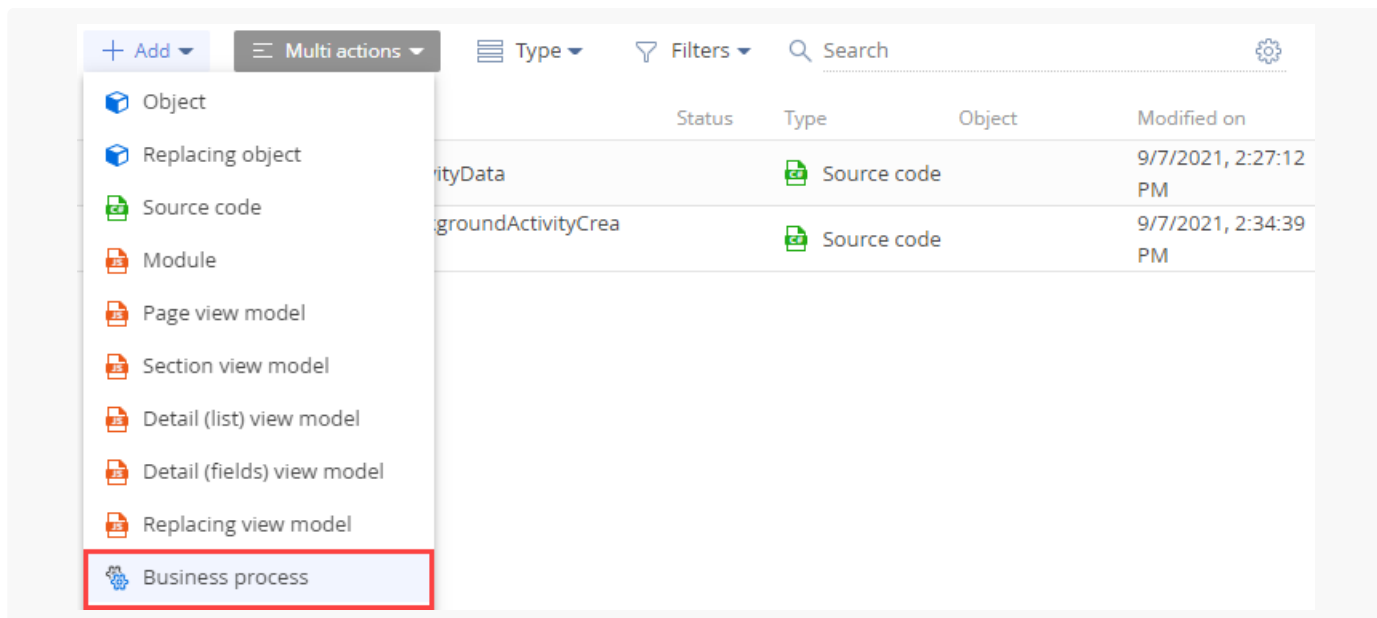
2. На панели инструментов дизайнера нажмите [*Опубликовать*] ([*Publish*]).

После успешной публикации схемы пользовательское действие процесса можно использовать при создании бизнес-процессов.

4. Выполнить тестирование

Для проверки работоспособности созданного действия процесса создайте новый бизнес-процесс. Подробно о том, как создать пользовательский бизнес-процесс, можно узнать из статьи [Дизайнер процессов](#).

1. [Перейдите в раздел \[Конфигурация \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [*Добавить*] —> [*Бизнес процесс*] ([*Add*] —> [*Business process*]).



3. Заполните **свойства процесса**.

- На панели настройки элементов заполните свойство [*Заголовок*] ([*Title*]) — "Testing process".
- На вкладке [*Настройки*] ([*Settings*]) панели настройки элементов заполните свойство [*Имя*] ([*Code*]) — "UsrTestingProcess".

Process

Testing process

SETTINGS PARAMETERS METHODS

Code*

UsrTestingProcess

Version

0

Tag

Business Process

Process description

Package*

sdkCreatingUserTask

Maximum Number of Repetitions

100

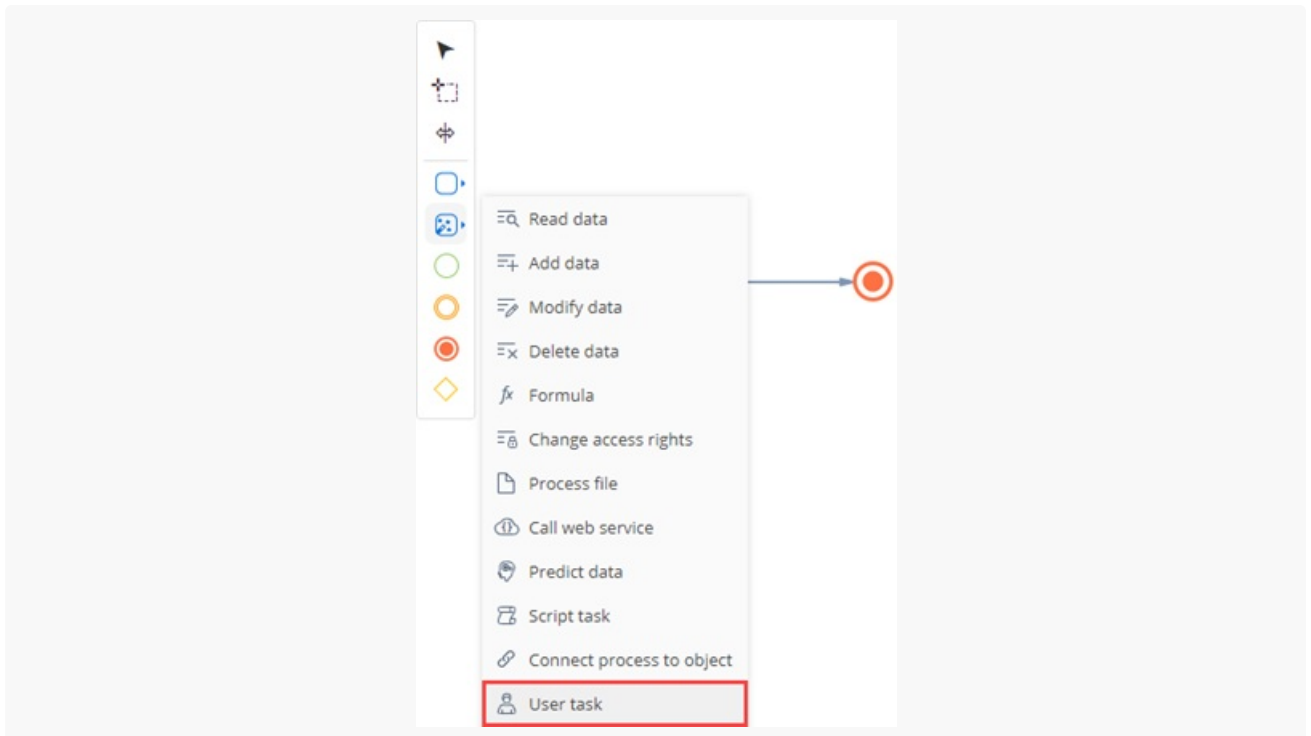
Process instance caption

[#[PropertyValue:Caption]#]

4. Реализуйте бизнес-процесс.

а. Добавьте действие процесса.

- а. В области элементов дизайнера нажмите [*Действия системы*] ([*System actions*]) и разместите элемент [*Выполнить действие процесса*] с в рабочей области дизайнера процессов между начальным событием [*Простое*] ([*Simple*]) и завершающим событием [*Останов*] ([*Terminate*]).

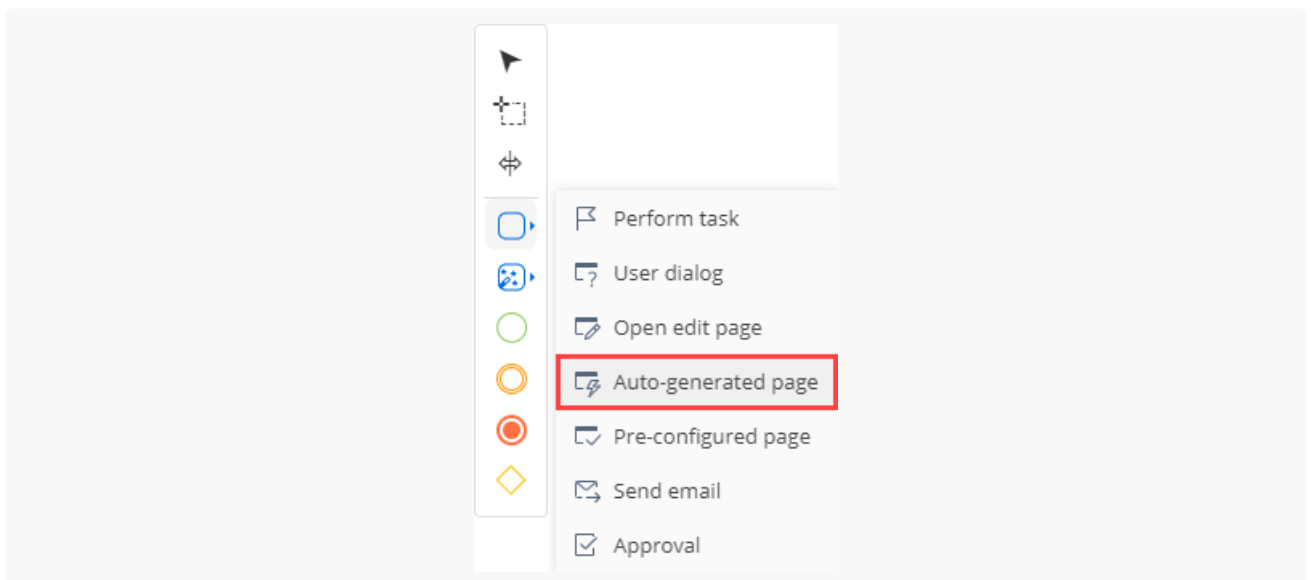


b. Заполните **свойства действия процесса**.

- [*Какое пользовательское действие выполнить?*] ([*Which user task to perform?*]) — выберите "Пример действия процесса" ("User Task Sample").
- Заполните значения параметров действия процесса.
 - [*First Number*] — "12".
 - [*Second Number*] — "23".

b. Добавьте автогенерируемую страницу.


- a. В области элементов дизайнера нажмите [Действия пользователя] ([*User actions*]) и разместите элемент [Автогенерируемая страница] ([*Auto-generated page*]) в рабочей области дизайнера процессов.

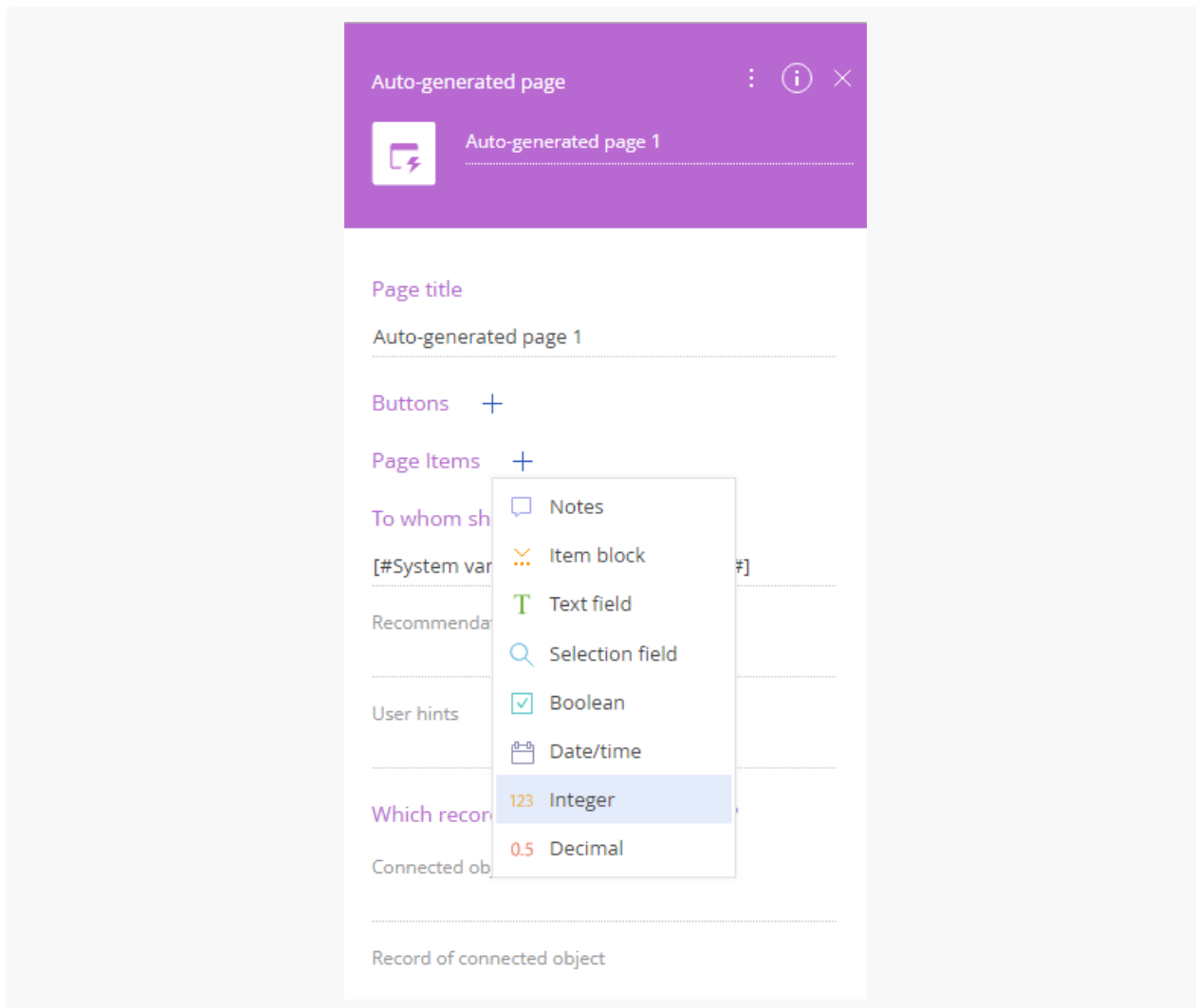


b. Заполните **свойства автогенерируемой страницы**.


- [Заголовок] ([*Title*]) — "Sample page".
- [Название страницы] ([*Page title*]) — "Sample page".

е. Добавьте элемент страницы.

- а. В блоке [*Элементы страницы*] ([*Page Items*]) нажмите кнопку  и выберите "Целое число" ("Integer").



б. Заполните **свойства элемента**.

- [*Заголовок*] ([*Title*]) — "SUM:".
- [*Значение*] ([*Value*]) — нажмите кнопку  и вызовите окно формулы значения.
- На вкладке [*Элементы процесса*] ([*Process elements*]) выберите элемент [*Выполнить действие процесса 1*] ([*User task 1*]).
- Двойным кликом выберите параметр процесса [*Сумма чисел*] ([*Sum of numbers*]) .
В результате отобразится формула, по которой будет вычисляться отображаемое на автогенерируемой странице значение.
- Сохраните формулу.

Диалоговое окно [*Формула*] ([*Formula*])

Formula

SAVE CANCEL [Read more about formula](#)

[#User Task 1.Sum of numbers#]

PROCESS ELEMENTS PROCESS PARAMETERS LOOKUP SYSTEM SETTINGS SYSTEM VARIABLES F>

Search process element Search element parameter

User Task 1 123 First number

Sample page 123 Second number

123 Sum of numbers

Свойства элемента страницы

Title*

SUM:

Code*

PageItem1

☐ Required

Value

[#User Task 1.Sum of numbers#]

SAVE CANCEL

е. Сохраните изменения.

Настройки автогенерируемой страницы представлены на рисунке ниже.

Auto-generated page

Sample page

Page title

Sample page

Buttons +

Page Items +

123 SUM:

[#User Task 1.Sum of numbers#]

To whom should the page be shown?

[#System variable.Current user contact#]

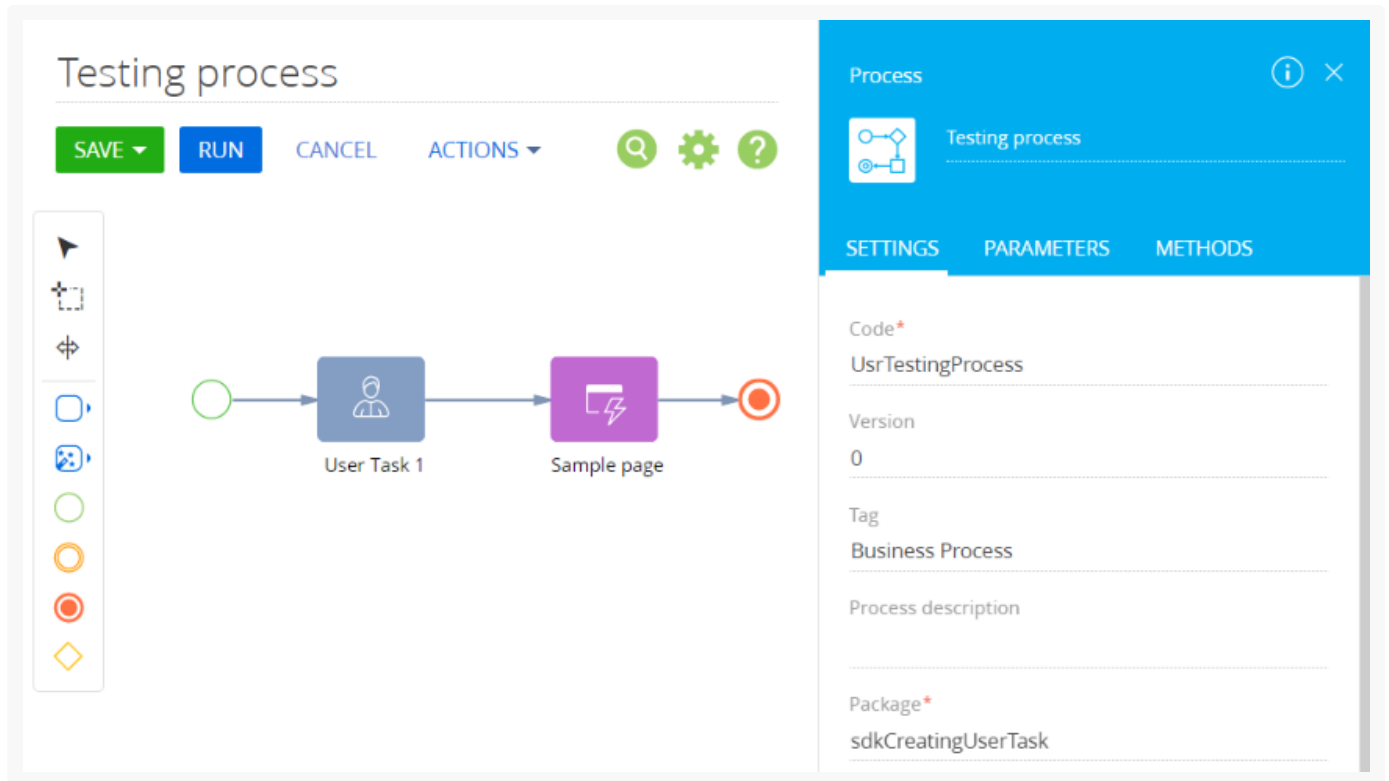
Recommendation to user

User hints

Which record to connect the page to?

Connected object

Бизнес-процесс представлен на рисунке ниже.



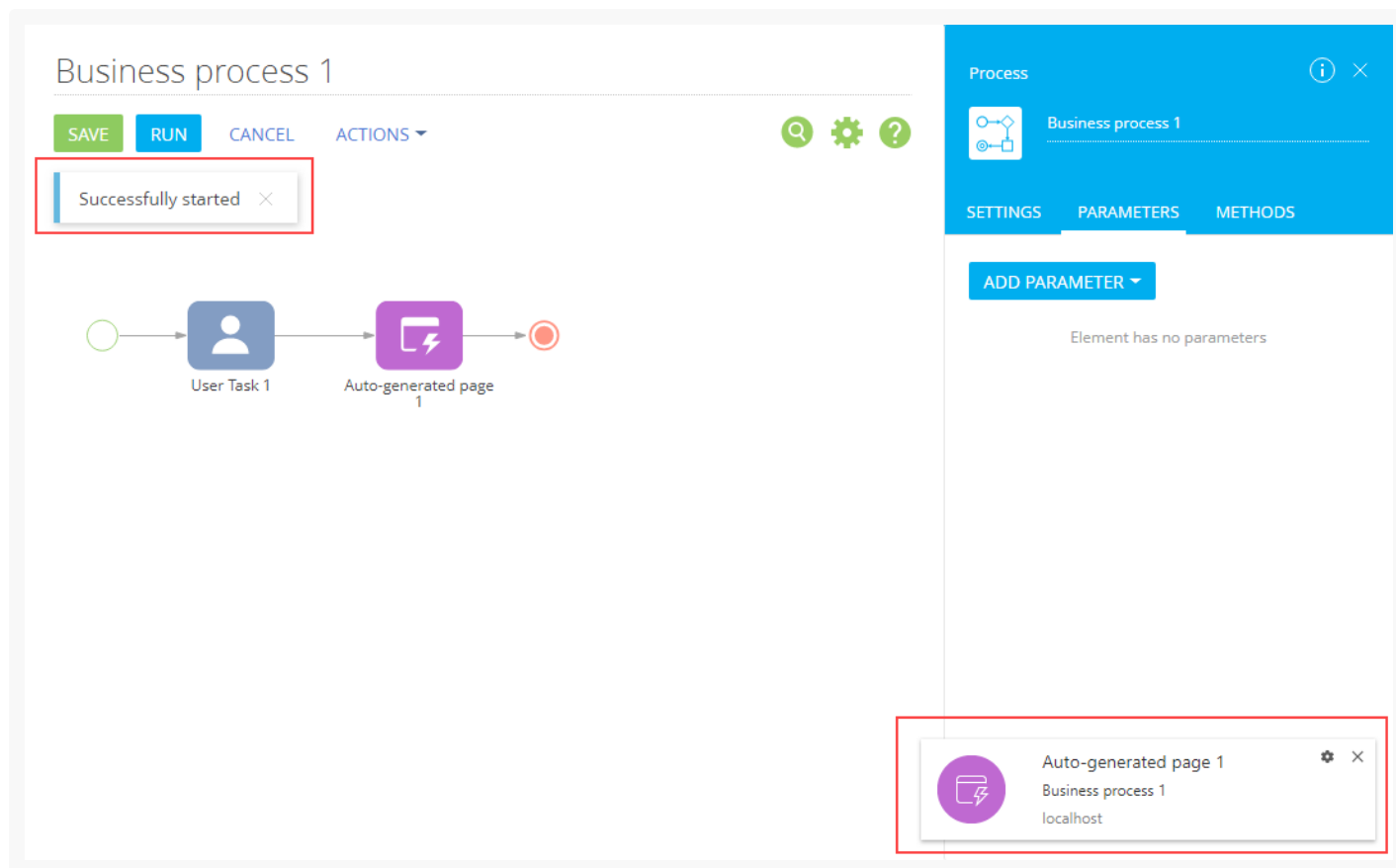
5. На панели инструментов дизайнера процессов нажмите [Сохранить] ([Save]).

6. После сохранения процесса, запустите его на выполнение.

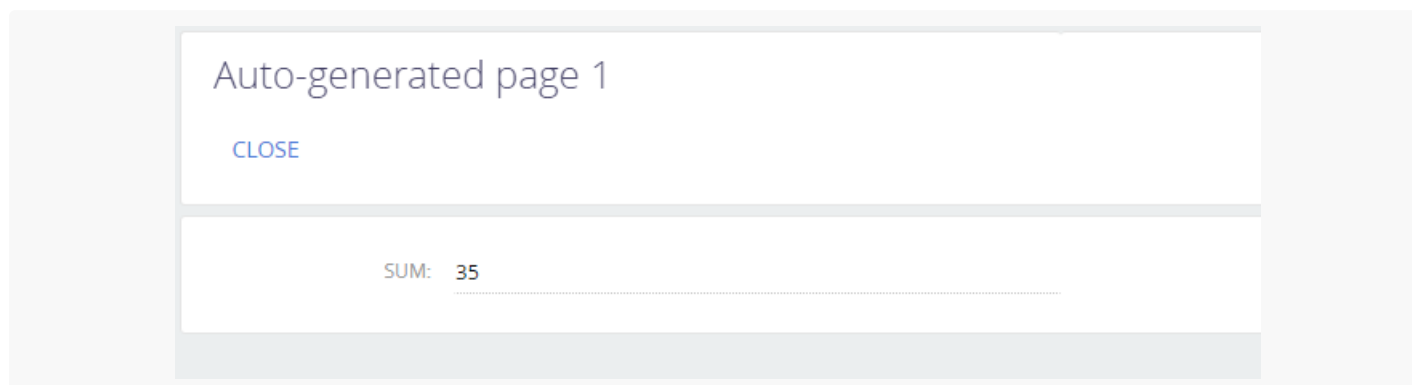
Результат выполнения примера

После запуска бизнес процесса появится сообщение, нажав на которое можно перейти к странице отображения результатов бизнес-процесса.

Сообщение о запуске бизнес-процесса



В результате выполнения бизнес-процесса Testing process отображается страница, которая вычисляет сумму значений, заданный в качестве параметров бизнес-процесса.



Добавить действие процесса на вкладку [Элементы процесса]

Если созданный пользовательский элемент действия процесса планируется часто использовать, то для удобства его можно добавить на вкладку Элементы процесса (Process elements) дизайнера процессов. Для этого в базе данных выполните следующий SQL-скрипт.

SQL-скрипт

MSSQL

```
-- UsrSampleProcessUserTask – название схемы действия процесса.
insert into SysProcessUserTask(SysUserTaskSchemaUid, Caption)
select s.Uid, s.Caption from SysSchema s
where s.Name = 'UsrSampleProcessUserTask'
```

PostgreSQL

```
-- UsrSampleProcessUserTask – название схемы действия процесса.
INSERT INTO SysProcessUserTask (SysUserTaskSchemaUid, Caption)
VALUES
(
    SELECT s.Uid, s.Caption FROM SysSchema AS s
    WHERE s.Name = 'UsrSampleProcessUserTask'
)
```

После перезагрузки приложения (или его компиляции) элемент отобразится на вкладке.

Пользовательский элемент на вкладке

