

# Поле

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

<b>Поле</b>	<b>6</b>
Типы полей и операций с полями	6
Добавить поле	6
Реализовать валидацию поля	11
Установить для поля значение по умолчанию	11
Настроить обязательность для поля	13
Настроить фильтрацию значений справочного поля	13
Настроить условия блокировки поля	15
Настроить исключения блокировки поля	17
Настроить условия отображения поля	18
Добавить автонумерацию к полю	19
Добавить информационную кнопку к полю	20
Добавить всплывающую подсказку к полю	20
Вычислить разницу дат в полях	20
<b>Добавить поле на страницу записи с использованием новой колонки</b>	<b>21</b>
1. Создать схему замещающего объекта	21
2. Создать схему замещающей модели представления страницы активности	22
Результат выполнения примера	25
<b>Добавить поле на страницу записи с использованием существующей колонки</b>	<b>25</b>
Создать схему замещающей модели представления страницы контакта	26
Результат выполнения примера	27
<b>Добавить поле с изображением на страницу записи</b>	<b>28</b>
1. Создать схему замещающего объекта	28
2. Создать схему замещающей модели представления страницы статьи базы знаний	30
Результат выполнения примера	36
<b>Добавить вычисляемое поле на страницу записи</b>	<b>36</b>
1. Создать схему замещающего объекта	36
2. Создать схему замещающей модели представления страницы заказа	38
Результат выполнения примера	42
<b>Добавить мультивалютное поле на страницу записи</b>	<b>42</b>
1. Создать схему замещающего объекта	42
2. Создать схему замещающей модели представления страницы проекта	45
Результат выполнения примера	50
<b>Реализовать валидацию поля типа [Дата/Время] на странице записи</b>	<b>51</b>
Создать схему замещающей модели представления страницы продажи	51
Результат выполнения примера	54

<b>Реализовать валидацию поля типа [Строка] на странице записи</b>	<b>55</b>
Создать схему замещающей модели представления страницы контакта	55
Результат выполнения примера	58
<b>Установить значение по умолчанию для поля на странице записи</b>	<b>58</b>
Создать схему замещающей модели представления страницы проекта	59
Результат выполнения примера	61
<b>Настроить обязательность для поля на странице записи</b>	<b>61</b>
Создать схему замещающей модели представления страницы контакта	62
Результат выполнения примера	64
<b>Настроить фильтрацию значений справочного поля на странице записи</b>	<b>65</b>
Создать схему замещающей модели представления страницы контрагента	65
Результат выполнения примера	68
<b>Настроить фильтрацию значений связанных справочных полей на странице записи</b>	<b>69</b>
Создать схему замещающей модели представления страницы контакта	69
Результат выполнения примера	73
<b>Настроить условия блокировки поля на странице записи</b>	<b>78</b>
Создать схему замещающей модели представления страницы контакта	78
Результат выполнения примера	81
<b>Настроить исключения блокировки полей на странице записи</b>	<b>81</b>
Создать схему замещающей модели представления страницы счета	81
Результат выполнения примера	84
<b>Настроить условия отображения поля на странице записи</b>	<b>85</b>
1. Создать схему замещающего объекта	85
2. Создать схему замещающей модели представления страницы активности	87
Результат выполнения примера	91
<b>Добавить автонумерацию к полю на странице добавления записи (front-end)</b>	<b>92</b>
1. Создать системные настройки	92
2. Создать схему замещающей модели представления страницы продукта	93
Результат выполнения примера	95
<b>Добавить автонумерацию к полю на странице добавления записи (back-end)</b>	<b>96</b>
1. Создать системные настройки	96
2. Создать схему замещающего объекта	97
Результат выполнения примера	104
<b>Добавить информационную кнопку к полю на странице записи</b>	<b>104</b>
Создать схему замещающей модели представления страницы контакта	105
Результат выполнения примера	108
<b>Добавить всплывающую подсказку к полю на странице записи</b>	<b>108</b>
Создать схему замещающей модели представления страницы контакта	108
Результат выполнения примера	111



# Поле



## Типы полей и операций с полями

**Типы** полей, которые предоставляет Creatio:

- Простое поле.
- Поле с изображением.
- Вычисляемое поле.
- Мультивалютное поле.

**Операции с полями**, которые позволяет выполнять Creatio:

- Реализовать валидацию поля.
- Установить для поля значение по умолчанию.
- Настроить обязательность для поля.
- Настроить фильтрацию значений справочного поля.
- Настроить условия блокировки поля.
- Настроить исключения блокировки полей.
- Настроить условия отображения поля.
- Добавить автонумерацию к полю.
- Добавить информационную кнопку к полю.
- Добавить всплывающую подсказку к полю.
- Вычислить разницу дат в полях.

## Добавить поле

**Инструменты**, которые позволяют добавить поле:

- Мастер разделов.
- Creatio IDE.

## Добавить простое поле

**Способы** добавления простого поля:

- С использованием существующей колонки.
- С использованием новой колонки.

## Добавить простое поле с использованием мастера разделов

Чтобы добавить простое поле **с использованием мастера разделов**, воспользуйтесь инструкцией, которая приведена в статье [Настроить поля страницы](#).

## Добавить простое поле с использованием Creatio IDE

Чтобы добавить простое поле **с использованием существующей колонки** с помощью Creatio IDE:

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схеме замещающей модели представления настройте расположение поля. Для этого в массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля на странице.

**Способы** добавления простого поля с использованием новой колонки с помощью Creatio IDE:

- Создать схему замещающего объекта и добавить в нее колонку.
- Создать схему замещающей модели представления страницы записи, на которой размещено поле. Затем создать атрибут в схеме модели представления и добавить поле к созданной виртуальной колонке.

Ниже рассмотрен один из способов.

Чтобы добавить простое поле **с использованием новой колонки** с помощью Creatio IDE:

1. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающего объекта добавьте колонку, которая соответствует полю схемы замещающей модели представления страницы. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
3. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
4. При необходимости, в схему замещающей модели представления добавьте локализуемую строку, которая содержит название поля. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
5. В схеме замещающей модели представления настройте расположение поля. Для этого в массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля на странице.

## Добавить поле с изображением

1. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающего объекта добавьте колонку типа [ Ссылка на изображение ] ([ Image Link ]), которая соответствует полю схемы замещающей модели представления страницы. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).

3. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
4. При необходимости, в схему замещающей модели представления добавьте локализуемую строку, которая содержит название поля. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
5. В коллекцию изображений схемы добавьте изображение.
6. В схеме замещающей модели представления настройте **поле с изображением**.
  - a. В свойстве `methods` реализуйте **методы**:
    - Метод, который получает изображение по ссылке.
    - Метод, который вызывается перед открытием диалогового окна выбора изображения.
    - Метод, который вызывается при изменении изображения.
    - Метод, который сохраняет ссылку на измененное изображение в колонке объекта.
  - f. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля. Поле с изображением добавляется на страницу с использованием вспомогательного контейнера-обертки `PhotoContainer` с классом `"image-edit-container"`. В свойстве `values` массива модификаций `diff` реализуйте свойства:
    - `getSrcMethod()` — получает изображение по ссылке.
    - `onPhotoChange()` — вызывается при изменении изображения.
    - `beforeFileSelected()` — вызывается перед открытием диалогового окна выбора изображения.
    - `readonly` — определяет возможность модификации изображения.
    - `generator` — генератор элемента управления. Для поля с изображением укажите `ImageCustomGeneratorV2.generateCustomImageControl`.

## Добавить вычисляемое поле

**Вычисляемое поле** — это элемент управления страницы записи, значение которого вычисляется в зависимости от состояния или значений других элементов управления этой страницы.

В Creatio работа вычисляемых полей основана через подписку на события изменения атрибутов схемы модели представления страницы. При изменении значений колонок схемы объекта должно измениться значение текущей колонки.

Чтобы **добавить вычисляемое поле** на страницу записи:

1. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающего объекта добавьте колонку, которая соответствует полю схемы замещающей модели представления страницы. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
3. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных](#)



[элементов.](#)

4. В схеме замещающей модели представления настройте **вычисляемое поле**.

- a. В свойство `attributes` добавьте вычисляемую колонку (атрибут), для которой планируется установить зависимость. Для этого атрибута объявите свойство `dependencies`, которое содержит массив конфигурационных объектов. Свойства свойства `dependencies`:
  - `columns` — массив имен колонок, от значений которых зависит значение текущей колонки.
  - `methodName` — имя метода-обработчика, который вызывается при изменении значения хотя бы одной из перечисленных колонок.
- d. В свойстве `methods` реализуйте:
  - Метод-обработчик события изменения колонки, от которой зависит вычисляемая колонка.
  - `onEntityInitialized()` — переопределенный базовый виртуальный метод. Срабатывает после выполнения инициализации схемы объекта страницы записи. В метод `onEntityInitialized()` добавьте вызов метода-обработчика, который обеспечит расчет вычисляемого поля в момент открытия страницы записи, а не только после изменений в колонках-зависимостях.
- g. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля.

## Добавить мультивалютное поле

**Мультивалютное поле** — элемент управления страницы записи, значение которого вычисляется в зависимости от состояния или значений других элементов управления этой страницы.

Мультивалютное поле **позволяет**:

- Вводить денежную сумму.
- Указывать валюту введенной денежной суммы.
- Фиксировать эквивалент суммы в базовой валюте, которая задана в настройках системы.

При изменении валюты введенная сумма автоматически пересчитывается с учетом обменных курсов валют.

Чтобы **добавить мультивалютное поле** на страницу записи:

1. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающего объекта добавьте **колонки**:
  - Колонку типа [ Справочник ] ([ Lookup ]) для хранения валюты.
  - Колонку курса валюты.
  - Колонку для хранения общей суммы в выбранной валюте.
  - Колонку для хранения суммы в базовой валюте.

В схеме объекта может быть определена только одна колонка — для хранения общей суммы в выбранной валюте. Остальные колонки могут являться виртуальными, если бизнес-задача не

предполагает хранения в базе данных их значений. Они могут быть определены как атрибуты в схеме модели представления.

Для добавления колонки воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).

3. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
4. В объявлении класса модели представления в качестве зависимостей добавьте модули `MoneyModule`, `MultiCurrencyEdit`, `MultiCurrencyEditUtilities`.
5. В схеме замещающей модели представления настройте **мультивалютное поле**.

a. В свойство `attributes` добавьте **атрибуты**:

- Валюта.
- Курс валюты.
- Общая сумма.
- Сумма в базовой валюте.
- Коллекция курсов валют.
- Коллекция для кнопки выбора валюты.

Для атрибутов объявите свойство `dependencies`, которое содержит массив конфигурационных объектов. Свойства свойства `dependencies`:

- `columns` — массив имен колонок, от значений которых зависит значение текущей колонки.
- `methodName` — имя метода-обработчика, который вызывается при изменении значения хотя бы одной из перечисленных колонок.

j. В свойство `mixins` добавьте МИКСИН `MultiCurrencyEditUtilities`.

k. В свойстве `methods` реализуйте **методы**:

- `onEntityInitialized()` — переопределяет базовый виртуальный метод. Срабатывает после выполнения инициализации схемы объекта страницы записи.
- `setCurrencyRate()` — устанавливает курс валюты.
- `recalculateAmount()` — пересчитывает общую сумму.
- `recalculatePrimaryAmount()` — пересчитывает сумму в базовой валюте.
- `onVirtualCurrencyChange()` — метод-обработчик изменения виртуальной колонки валюты.

q. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля. В свойстве `values` массива модификаций `diff` реализуйте свойства:

- `primaryAmount` — наименование колонки, которая содержит сумму в базовой валюте.
- `currency` — наименование колонки, которая ссылается на справочник валют.
- `rate` — наименование колонки, которая содержит курс валюты.

- `generator` — генератор элемента управления. Для мультивалютного поля укажите `MultiCurrencyEditViewGenerator.generate`.

## Реализовать валидацию поля

**Валидация** — проверка значений заполненных полей на соответствие установленным требованиям. Валидация значений полей страницы в Creatio реализуется на уровне колонок моделей представления страниц. Логика проверки значения поля выполняется в пользовательском методе-валидаторе.

Чтобы **реализовать валидацию поля**:

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. При необходимости, в схему замещающей модели представления добавьте локализуемую строку, которая содержит сообщение валидации. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
3. В схеме замещающей модели представления настройте **валидацию поля**.

Для этого в свойстве `methods` реализуйте **методы**:

- Метод-валидатор, который определяет выполнение условия. **Валидатор** — метод модели представления, в котором выполняется анализ значения колонки модели представления на соответствие бизнес-требованиям. Этот метод должен возвращать объект с результатами валидации.
  - Если **валидация поля успешна**, то метод-валидатор возвращает объект с пустой строкой.
  - Если **валидация поля неуспешна**, то метод-валидатор возвращает объект из свойством `invalidMessage`. Свойство `invalidMessage` содержит строку с сообщением, которое отображается под полем при попытке ввести в него некорректное значение и в информационном окне при попытке сохранить страницу с полем, которое не прошло валидацию.
- `setValidationConfig()` — переопределенный базовый метод, в котором метод-валидатор привязан к соответствующей колонке схемы замещающей модели представления страницы записи. Метод `setValidationConfig()` вызывает метод `addColumnValidator()`. **Параметры** метода `addColumnValidator()`:
  - Имя колонки модели представления, к которой привязывается валидатор.
  - Имя метода-валидатора значения колонки.

Если валидация поля реализуется в схеме замещающей модели представления базовой страницы, то для корректной инициализации валидаторов полей базовой страницы перед вызовом метода `addColumnValidator()` добавьте вызов родительской реализации метода `setValidationConfig()`.

## Установить для поля значение по умолчанию

**Способы** установки для поля значения по умолчанию, которые предоставляет Creatio:

- **На уровне колонок бизнес-объектов в схеме замещающего объекта.**

При создании нового объекта некоторые поля страницы необходимо заполнить соответствующими значениями. В этом случае для колонок объекта, которые соответствуют этим полям, в дизайнера объектов укажите эти значения в качестве значений по умолчанию.

- **В исходном коде схемы замещающей модели представления страницы записи.**

В некоторых случаях невозможно установить значение по умолчанию с помощью свойств колонки объекта. Например, это могут быть вычисляемые значения, которые рассчитываются по значениям других колонок объекта. В таком случае, установить для поля значение по умолчанию можно только программными средствами.

**Виды** значений по умолчанию для полей страницы записи, которые устанавливаются на уровне колонок бизнес-объектов в схеме замещающего объекта, представлены в таблице ниже.

Виды значений по умолчанию

Вид значения по умолчанию	Описание
[ <i>Константа</i> ] ([ <i>Constant</i> ])	Возможные <b>типы</b> колонок, для которых можно установить константу в качестве значения по умолчанию: <ul style="list-style-type: none"> <li>• [ <i>Строка</i> ] ([ <i>String</i> ]).</li> <li>• [ <i>Число</i> ] ([ <i>Number</i> ]).</li> <li>• [ <i>Справочник</i> ] ([ <i>Lookup</i> ]).</li> <li>• [ <i>Логическое</i> ] ([ <i>Boolean</i> ]).</li> </ul>
[ <i>Системная настройка</i> ] ([ <i>System setting</i> ])	Системные настройки содержатся в разделе [ <i>Системные настройки</i> ] ([ <i>System settings</i> ]), в который можно добавить пользовательскую системную настройку. Значение системной настройки устанавливается на уровне пользователя, а не на уровне приложения.
[ <i>Системная переменная</i> ] ([ <i>System variable</i> ])	<b>Системные переменные</b> — глобальные переменные, которые хранят информацию о настройках системы. Значение системной переменной устанавливается на уровне ядра приложения, а не на уровне пользователя.

Чтобы **установить для поля значение по умолчанию**:

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схеме замещающей модели представления настройте для поля **значение по умолчанию**.

Для этого в свойстве `methods` реализуйте **методы**:

- `onEntityInitialized()` — переопределенный базовый виртуальный метод. Срабатывает после окончания инициализации схемы объекта. В метод `onEntityInitialized()` добавьте вызов метода-обработчика, который обеспечит установку значения поля в момент открытия страницы записи.

- Метод-обработчик, который рассчитывает значение поля.

## Настроить обязательность для поля

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В объявлении класса модели представления в качестве зависимостей добавьте модули `BusinessRuleModule` и `ConfigurationConstants`.
3. В схеме замещающей модели представления настройте **обязательность для поля**. Для этого в свойстве `rules`:
  - В свойстве `ruleType` укажите значение `BINDPARAMETER`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType`.
  - В свойстве `property` укажите значение `REQUIRED`, которое устанавливает обязательность заполнения колонки. Свойства бизнес-правила `BINDPARAMETER` представлены перечислением `BusinessRuleModule.enums.Property`.
  - В массиве `conditions` укажите условия выполнения бизнес-правила.

## Настроить фильтрацию значений справочного поля

**Способы** настройки фильтрации значений справочного поля, которые предоставляет Creatio:

- С использованием бизнес-правила [ *FILTRATION* ].
- Явное указание фильтров в описании колонки в свойстве `attributes`.

Чтобы **настроить фильтрацию значений справочного поля** с использованием бизнес-правила [ *FILTRATION* ]:

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В объявлении класса модели представления в качестве зависимостей добавьте модуль `BusinessRuleModule`.
3. В схеме замещающей модели представления настройте **фильтрацию значений справочного поля**.
  - a. В свойстве `rules`:
    - В свойстве `ruleType` укажите значение `FILTRATION`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType`.
    - В свойстве `autocomplete` укажите значение `true`, которое выполняет обратную фильтрацию.
  - d. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля.

Настройка фильтрации значений справочного поля явным указанием фильтров в описании колонки используется для применения произвольной фильтрации, сортировки и добавления дополнительных колонок в запрос при отображении выпадающего списка.

Чтобы **настроить фильтрацию значений справочного поля** явным указанием фильтров в описании колонки:

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В объявлении класса модели представления в качестве зависимостей добавьте модуль `BusinessRuleModule`.
3. В схеме замещающей модели представления настройте **фильтрацию значений справочного поля**. Для этого в свойстве `attributes`:
  - В свойстве `dataValueType` укажите значение `LOOKUP`, которое устанавливает тип данных колонки. Типы данных колонки представлены перечислением `Terrasoft.core.enums.DataValueType`.
  - В свойстве `lookupListConfig` укажите конфигурационный объект поля-справочника.

**Опциональные свойства** свойства `lookupListConfig`:

- `columns` — массив имен колонок, которые добавлены к запросу дополнительно к колонке `[Id]` и первичной для отображения колонки.
- `orders` — массив конфигурационных объектов, которые определяют сортировку данных при отображении.
- Свойство, которое задает фильтрацию:
  - `filter` — метод, который возвращает объект класса `Terrasoft.BaseFilter` или его наследника. Объект применяется к запросу.
  - `filters` — массив методов, которые возвращают коллекцию класса `Terrasoft.FilterGroup`.

Фильтры добавляются в коллекцию с помощью метода `add()`. **Параметры** метода `add()` представлены в таблице ниже.

Параметры метода `add()`

Параметр	Тип данных	Описание
<code>key</code>	<code>String</code>	Ключ.
<code>item</code>	<code>Mixed</code>	<p>Элемент.</p> <p>В качестве параметра <code>item</code> выступает объект класса <code>Terrasoft.BaseFilter</code> или его наследника. Настройка фильтров описана в статье <a href="#">Операции с данными (front-end)</a>.</p> <p>По умолчанию фильтры в коллекции объединяются с использованием логической операции <code>AND</code>. Если необходимо применить операцию <code>OR</code>, то ее нужно явно указать в свойстве <code>logicalOperation</code> объекта <code>Terrasoft.FilterGroup</code>.</p>
<code>index</code>	<code>Number</code>	Индекс для вставки. Если индекс не указан, то он не учитывается.

## Настроить условия блокировки поля

**Назначение** блокировки полей страницы записи — одновременная блокировка всех полей и деталей на странице записи при выполнении соответствующего условия. Блокировка полей страницы записи позволяет решить задачу без написания большого количества бизнес-правил.

**Типы** деталей, к полям которых можно применить блокировку:

- Деталь с реестром.
- Деталь с редактируемым реестром.
- Деталь с полями.

Чтобы **настроить условия блокировки поля**:

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В объявлении класса модели представления в качестве зависимостей добавьте модуль `BusinessRuleModule`.
3. В схеме замещающей модели представления настройте **условия блокировки поля**.
  - a. В свойстве `attributes` включите блокировку с помощью атрибута `IsModelItemsEnabled`.

**Способы** включения блокировки поля:

- Для атрибута `IsModelItemsEnabled` установите значение `false`.
- Для атрибута `IsModelItemsEnabled` установите значение по умолчанию.
- Для детали с полями используйте атрибут `IsEnabled`.

Включение блокировки полей страницы записи (способ 1)

```
this.set("IsModelItemsEnabled", false);
```

Включение блокировки полей страницы записи (способ 2)

```
"IsModelItemsEnabled": {
  dataValueType: Terrasoft.DataValueType.BOOLEAN,
  value: true,
  dependencies: [{
    columns: ["PaymentStatus"],
    methodName: "setCardLockoutStatus"
  }]
}
```

e. В свойство `rules`:

- В свойстве `ruleType` укажите значение `BINDPARAMETER`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType`.
- В свойстве `property` укажите значение `ENABLED`, которое устанавливает доступность колонки. Свойства бизнес-правила `BINDPARAMETER` представлены перечислением `BusinessRuleModule.enums.Property`.
- В массиве `conditions` укажите условия выполнения бизнес-правила.

i. В массив модификаций `diff` добавьте конфигурационный объект:

- В свойстве `name` для блокировки всех полей страницы записи укажите глобальный контейнер `CardContentWrapper`.
- В свойстве `generator` свойства `values` укажите генератор `DisableControlsGenerator` для тех контейнеров, в которых планируется блокировать поля.

#### Пример настройки массива модификаций `diff`

```
diff: /**SCHEMA_DIFF*/[
  {
    "operation": "merge",
    "name": "CardContentWrapper",
    "values": {
      "generator": "DisableControlsGenerator.generatePartial"
    }
  }
]/**SCHEMA_DIFF*/
```



У деталей блокируются кнопки и элементы меню, которые отвечают за выполнение операций над записью. При этом в детали с редактируемым реестром остается возможность перейти на страницу объекта, на которой в соответствии с бизнес-правилами будут заблокированы поля.

Поле не будет заблокировано, если для поля в массиве модификаций `diff` или в бизнес-правиле существует привязка для свойства `enabled`.

## Настроить исключения блокировки поля

Creatio предоставляет возможность исключить блокировку для полей и деталей.

Чтобы **настроить исключения блокировки поля**:

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схеме замещающей модели представления настройте **исключения блокировки поля**.
  - a. В свойстве `attributes` включите блокировку с помощью атрибута `IsModelItemsEnabled`.
  - b. В свойстве `methods` реализуйте **методы**, которые возвращают списки полей и деталей, которые не должны быть заблокированы:
    - `getDisableExclusionsColumnTags()` — исключает блокировку колонки.
    - `getDisableExclusionsDetailSchemaNames()` — исключает блокировку детали.
    - `isModelItemEnabled()` — исключает блокировку колонки. Сложная логика исключений. Метод вызывается для каждого поля. Получает на вход имя и возвращают признак доступности поля.
    - `isDetailEnabled()` — исключает блокировку детали. Сложная логика исключений. Метод вызывается для каждой детали. Получает на вход имя и возвращают признак доступности детали.

Пример исключения блокировки поля и детали (способ 1)

```
getDisableExclusionsColumnTags: function() {
    return ["SomeField"];
}
getDisableExclusionsDetailSchemaNames: function() {
    return ["SomeDetailV2"]
}
```

Пример исключения поля и детали (способ 2)

```
isModelItemEnabled: function(fieldName) {
    var condition = this.get("SomeConditionAttribute");
    if (fieldName === "ExampleField" || condition) {
        return true;
    }
}
```

```

    }
    return this.callParent(arguments);
}

isDetailEnabled: function(detailName) {
    if (detailName === "ExampleDetail") {
        var exampleDate = this.get("Date");
        var dateNow = new Date(this.Ext.Date.now());
        var condition = this.Ext.Date.isDate(exampleDate) && exampleDate >= dateNow;
        return condition;
    }
    return this.callParent(arguments);
}

```

- g. В массив модификаций `diff` добавьте конфигурационный объект с настройками контейнера `CardContentWrapper`, в котором планируется блокировать поля.

Чтобы **отключить блокировку полей**, на странице отключения функциональности Feature toggle используйте соответствующий переключатель опции `CompleteCardLockout`. Страница функциональности описана в статье [Механизм отключения функциональности Feature Toggle](#).

## Настроить условия отображения поля

1. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающего объекта добавьте колонку, которая соответствует полю схемы замещающей модели представления страницы. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
3. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
4. В схеме замещающей модели представления настройте **условия отображения поля**.
  - a. В свойстве `rules`:
    - В свойстве `ruleType` укажите значение `BINDPARAMETER`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType`.
    - В свойстве `property` укажите значение `VISIBLE`, которое устанавливает видимость колонки. Свойства бизнес-правила `BINDPARAMETER` представлены перечислением `BusinessRuleModule.enums.Property`.
    - В массиве `conditions` укажите условия выполнения бизнес-правила.
  - e. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля.

## Добавить автонумерацию к полю

**Автонумерация поля** — автоматическая генерация номера записи в заданном шаблоне. Автонумерация реализована в разделах [ *Документы* ] ([ *Documents* ]), [ *Счета* ] ([ *Invoices* ]) и [ *Договоры* ] ([ *Contracts* ]).

**Способы** добавления автонумерацию к полю:

- На стороне front-end.
- На стороне back-end.

### Добавить автонумерацию к полю на стороне front-end

#### 1. Создайте **системные настройки**:

- `[Entity]CodeMask` — маска номера объекта.
- `[Entity]LastNumber` — текущий номер объекта.

`Entity` — наименование объекта, к колонке которого планируется применить автонумерацию. Например, `InvoiceCodeMask` — маска номера счета и `InvoiceLastNumber` — текущий номер счета.

#### 2. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).

#### 3. В схеме замещающей модели представления настройте **автонумерацию к полю на стороне front-end**.

Для этого в свойстве `methods` реализуйте метод `onEntityInitialized()` — переопределенный базовый виртуальный метод. Срабатывает после окончания инициализации схемы объекта. В метод `onEntityInitialized()` добавьте вызов метода-обработчика `getIncrementCode()` базовой схемы страницы записи `BasePageV2`, который присвоит сгенерированный номер полю [ *Код* ] ([ *Code* ]).

**Параметры** метода `getIncrementCode()`:

- `callback` — функция, которая будет вызвана при получении ответа от сервиса. Ответ необходимо передать в соответствующую колонку (атрибут).
- `scope` — контекст, в котором будет вызвана функция `callback` (необязательный параметр).

### Добавить автонумерацию к полю на стороне back-end

#### 1. Создайте **системные настройки**:

- `[Entity]CodeMask` — маска номера объекта.
- `[Entity]LastNumber` — текущий номер объекта.

`Entity` — наименование объекта, к колонке которого планируется применить автонумерацию. Например, `InvoiceCodeMask` — маска номера счета и `InvoiceLastNumber` — текущий номер счета.

#### 2. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).

#### 3. В схему замещающего объекта добавьте событие [ *Перед добавлением записи* ] ([ *Before record added* ])

)).

4. В бизнес-процессе реализуйте событийный подпроцесс.

## Добавить информационную кнопку к полю

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. При необходимости, в схему замещающей модели представления добавьте локализуемую строку, которая содержит название поля. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
3. В схеме замещающей модели представления **добавьте информационную кнопку к полю**.  
Для этого в массив модификаций `diff` добавьте конфигурационный объект с настройками расположения информационной кнопки к полю на странице. В свойстве `values` массива модификаций `diff` реализуйте свойство `itemType`, которому укажите значение `Terrasoft.ViewItemType.INFORMATION_BUTTON`.

## Добавить всплывающую подсказку к полю

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. При необходимости, в схему замещающей модели представления добавьте локализуемую строку, которая содержит название поля. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
3. В схеме замещающей модели представления **добавьте информационную кнопку к полю**.  
Для этого в массив модификаций `diff` добавьте конфигурационный объект с настройками расположения всплывающей подсказки к полю на странице. В свойстве `tip` массива модификаций `diff` настройте всплывающую подсказку.

## Вычислить разницу дат в полях

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схеме замещающей модели представления **вычислите разницу дат в полях**.  
Для этого в свойстве `methods` реализуйте **методы**:
  - `onEntityInitialized()` — переопределяет базовый виртуальный метод. Срабатывает после выполнения инициализации схемы объекта страницы записи.
  - `setEndDate()` — вспомогательный метод для установки даты. В метод `setEndDate()` добавьте вызов метода `getDate()`, который получает дату.

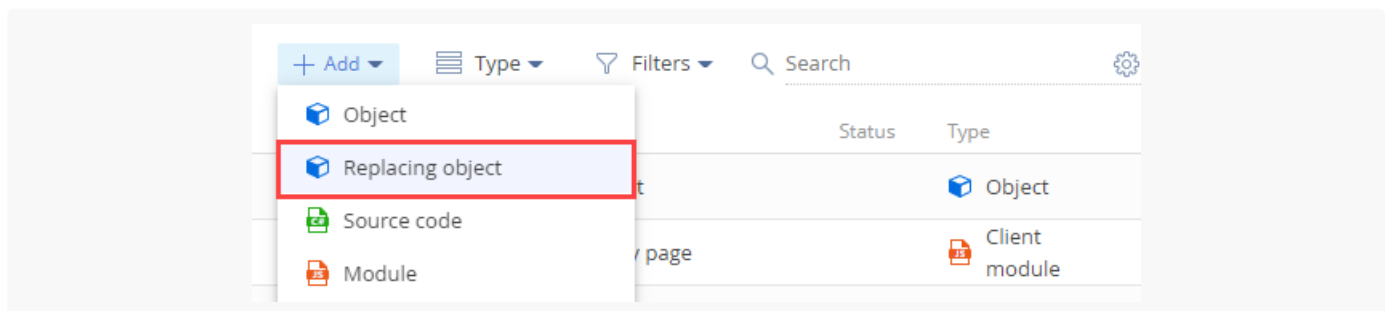
# Добавить поле на страницу записи с использованием новой колонки

 Средний

**Пример.** Добавить поле [ Место встречи ] ([ *Meeting place* ]) на страницу активности. Предварительно добавить соответствующую колонку в схему объекта активности.

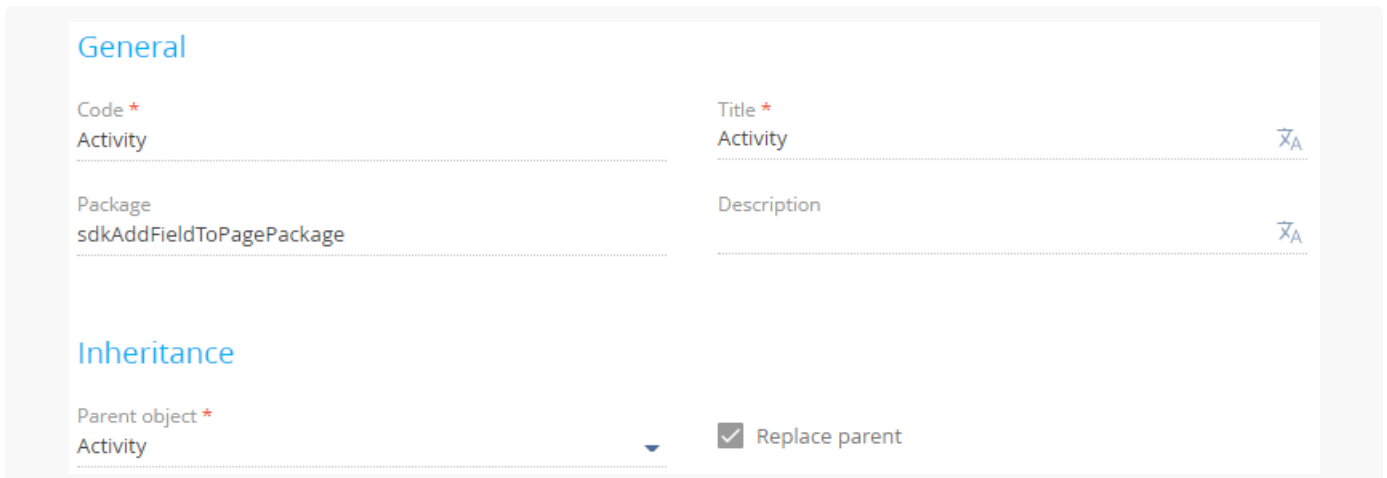
## 1. Создать схему замещающего объекта

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



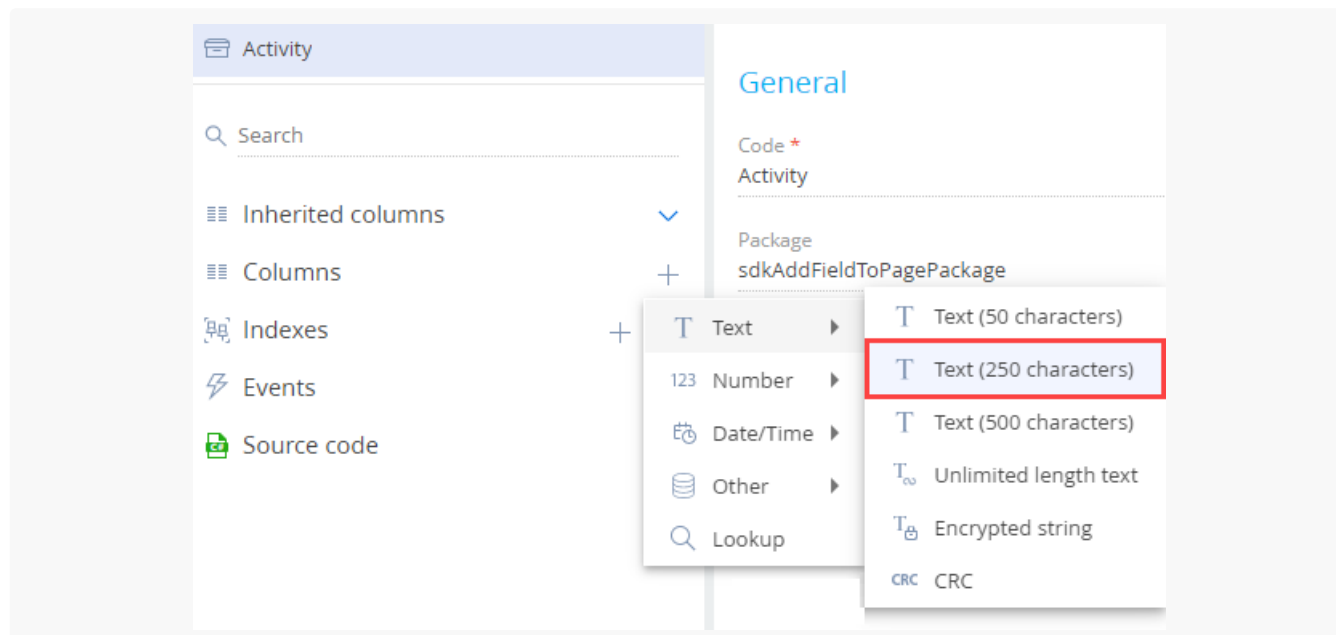
3. Заполните **свойства схемы**.

- [ Код ] ([ *Code* ]) — "Activity".
- [ Заголовок ] ([ *Title* ]) — "Активность" ("Activity").
- [ Родительский объект ] ([ *Parent object* ]) — выберите "Activity".

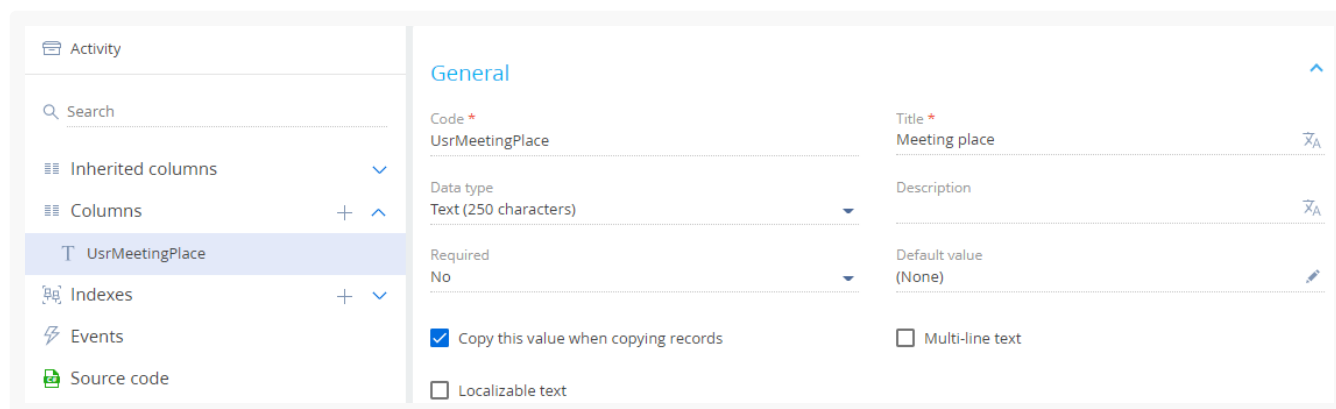

 A screenshot of a configuration form with two main sections. The 'General' section contains fields for 'Code \*' (with the value 'Activity'), 'Package' (with the value 'sdkAddFieldToPagePackage'), 'Title \*' (with the value 'Activity'), and 'Description'. The 'Inheritance' section contains a 'Parent object \*' dropdown menu (set to 'Activity') and a checked checkbox labeled 'Replace parent'.

4. В схему добавьте **колонку**.

- a. В контекстном меню узла [ Колонки ] ([ Columns ]) структуры объекта нажмите **+**.
- b. В выпадающем меню нажмите [ Строка ] —> [ Строка (250 символов) ] ([ Text ] —> [ Text (250 characters) ]).



- c. Заполните **свойства добавляемой колонки**.
  - [ Код ] ([ Code ]) — "UsrMeetingPlace".
  - [ Заголовок ] ([ Title ]) — "Место встречи" ("Meeting place").

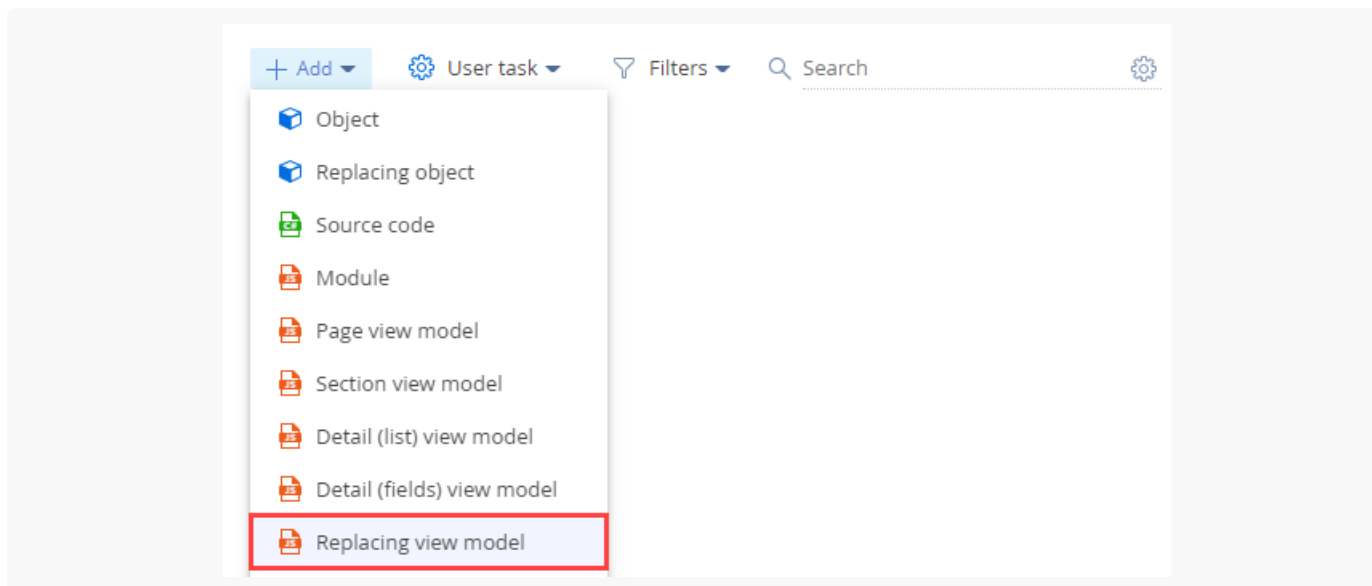


5. На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]), а затем [ Опубликовать ] ([ Publish ]).

## 2. Создать схему замещающей модели представления страницы активности

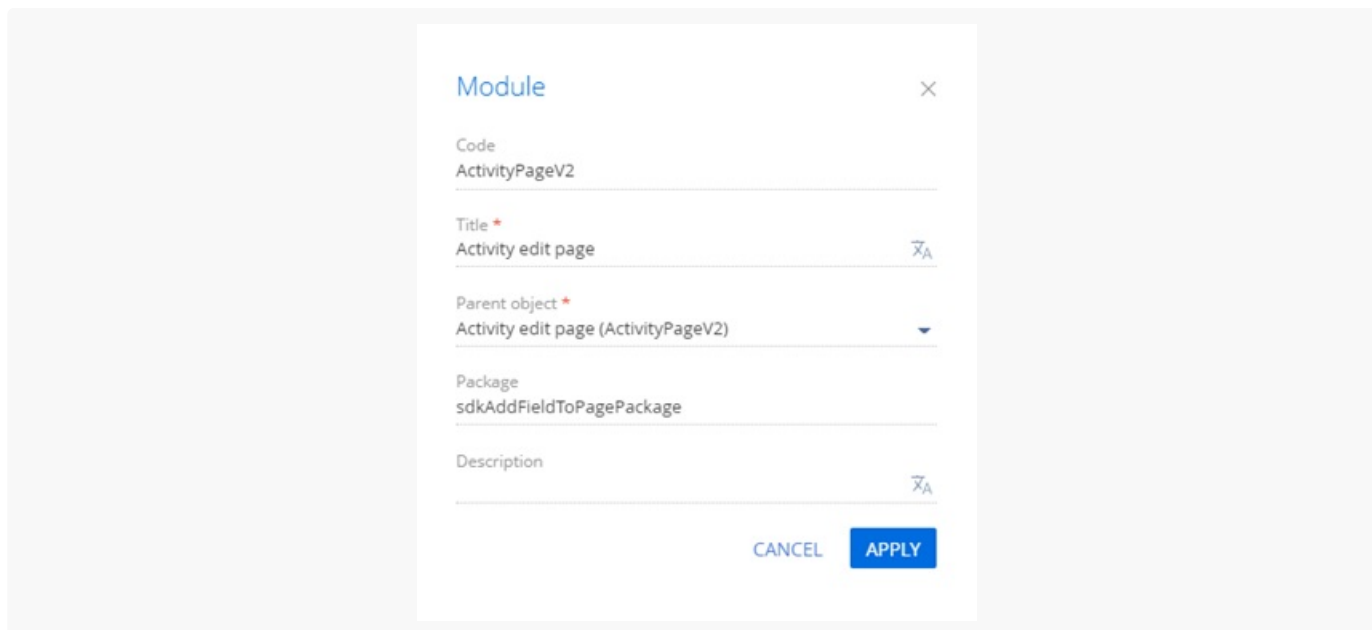
1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель

представления ] ([ Add ] —> [ Replacing view model ]).



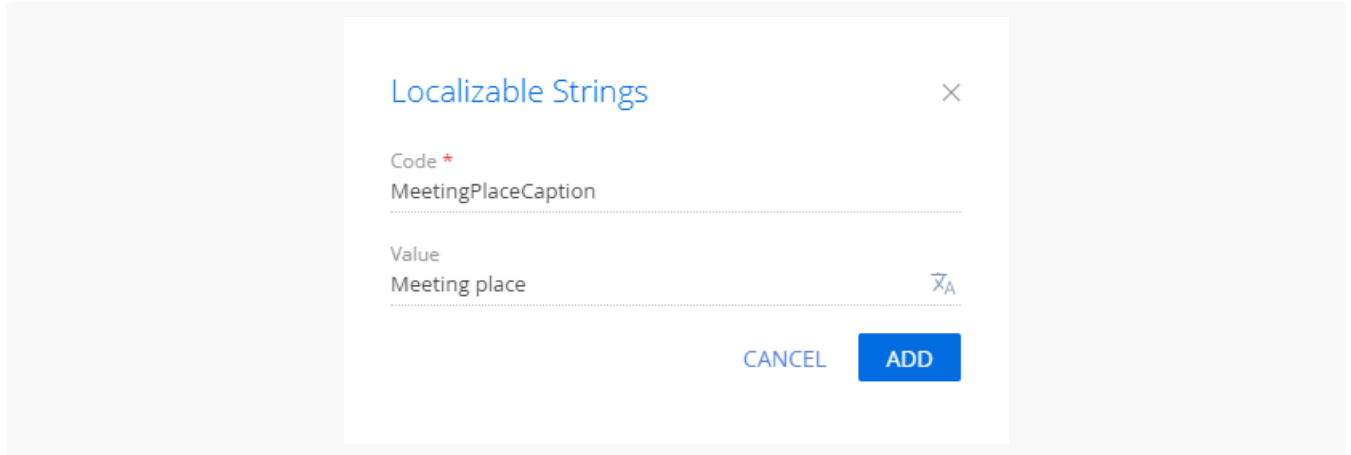
### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "ActivityPageV2".
- [ Заголовок ] ([ Title ]) — "Страница редактирования активности" ("Activity edit page").
- [ Родительский объект ] ([ Parent object ]) — выберите "ActivityPageV2".



### 4. Добавьте **локализуемую строку**.

- В контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
  - [ Код ] ([ Code ]) — "MeetingPlaceCaption".
  - [ Значение ] ([ Value ]) — "Место встречи" ("Meeting place").



е. Для добавления локализуемой строки нажмите [ *Добавить* ] ([ *Add* ]).

5. Настройте **расположение поля**. Для этого в массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля на странице.

Исходный код схемы замещающей модели представления страницы активности представлен ниже.

#### ActivityPageV2

```
define("ActivityPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Activity",
        /* Отображение поля на странице записи. */
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для добавления на страницу пользовательского поля. */
            {
                /* Выполняется операция добавления элемента на страницу. */
                "operation": "insert",
                /* Мета-имя родительского контейнера, в который добавляется поле. */
                "parentName": "Header",
                /* Поле добавляется в коллекцию элементов родительского элемента. */
                "propertyName": "items",
                /* Мета-имя добавляемого поля. */
                "name": "UsrMeetingPlace",
                /* Свойства, передаваемые в конструктор элемента. */
                "values": {
                    /* Привязка заголовка поля к локализуемой строке схемы. */
                    "caption": {"bindTo": "Resources.Strings.MeetingPlaceCaption"},
                    /* Настройка расположения поля. */
                    "layout": {
                        /* Номер столбца. */
                        "column": 0,
                        /* Номер строки. */
                        "row": 5,
                        /* Диапазон занимаемых столбцов. */
                        "colSpan": 12
                    }
                }
            }
        ]
    };
});
```



```

    }
  }
}
]/**SCHEMA_DIFF*/
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Очистите кэш браузера.
2. Обновите страницу раздела [ Активности ] ([ Activities ]).

В результате выполнения примера на страницу активности добавлено поле [ Место встречи ] ([ Meeting place ]).

The screenshot shows the 'Test task' form in the Creatio system. The form has a header with the title 'Test task', a search bar, and the Creatio logo. Below the header are buttons for 'CLOSE', 'ACTIONS', and a plus icon. The main form area contains several fields: 'Subject' (Test task), 'Start' (11/15/2021, 7:00 AM), 'Due' (11/15/2021, 7:30 AM), 'Status' (Not started), 'Show in calendar' (checked), 'Role' (empty), 'Owner' (Marina Kysla), 'Reporter' (Marina Kysla), 'Priority' (Medium), and 'Category' (To do). A red box highlights the 'Meeting place' field, which is a new field added to the form.

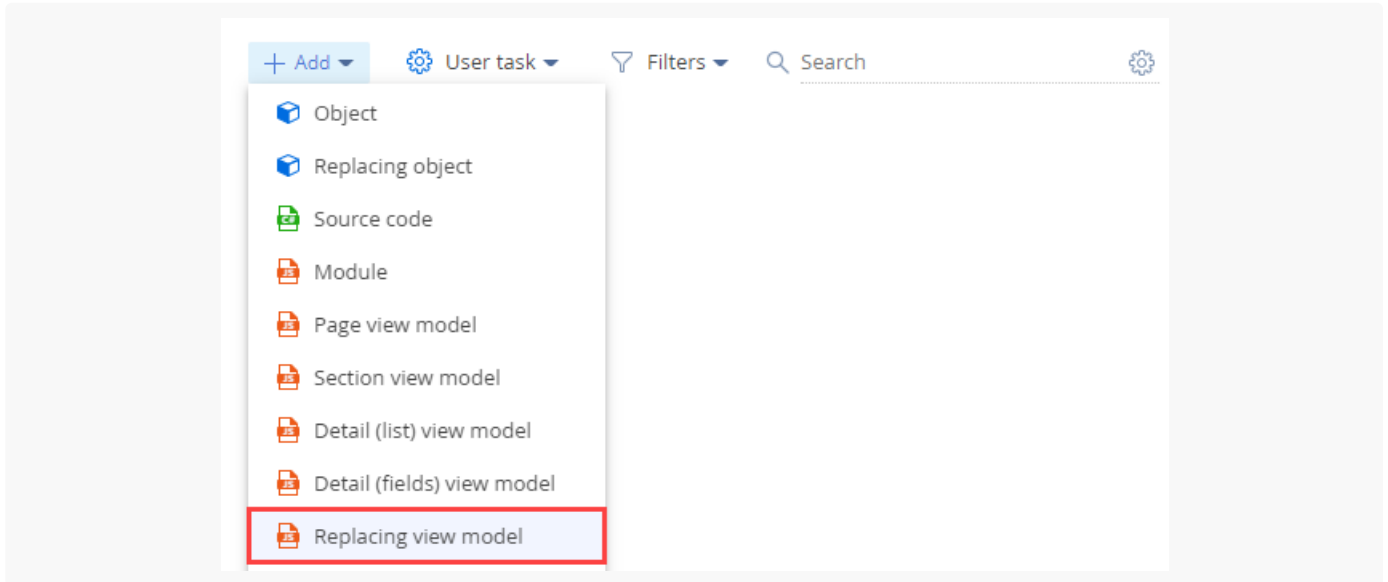
## Добавить поле на страницу записи с использованием существующей колонки

Средний

**Пример.** Добавить поле [ Страна ] ([ Country ]) в профиль контакта страницы контакта. Колонка, которая соответствует полю [ Страна ] ([ Country ]) страницы контакта, уже присутствует в схеме объекта контакта.

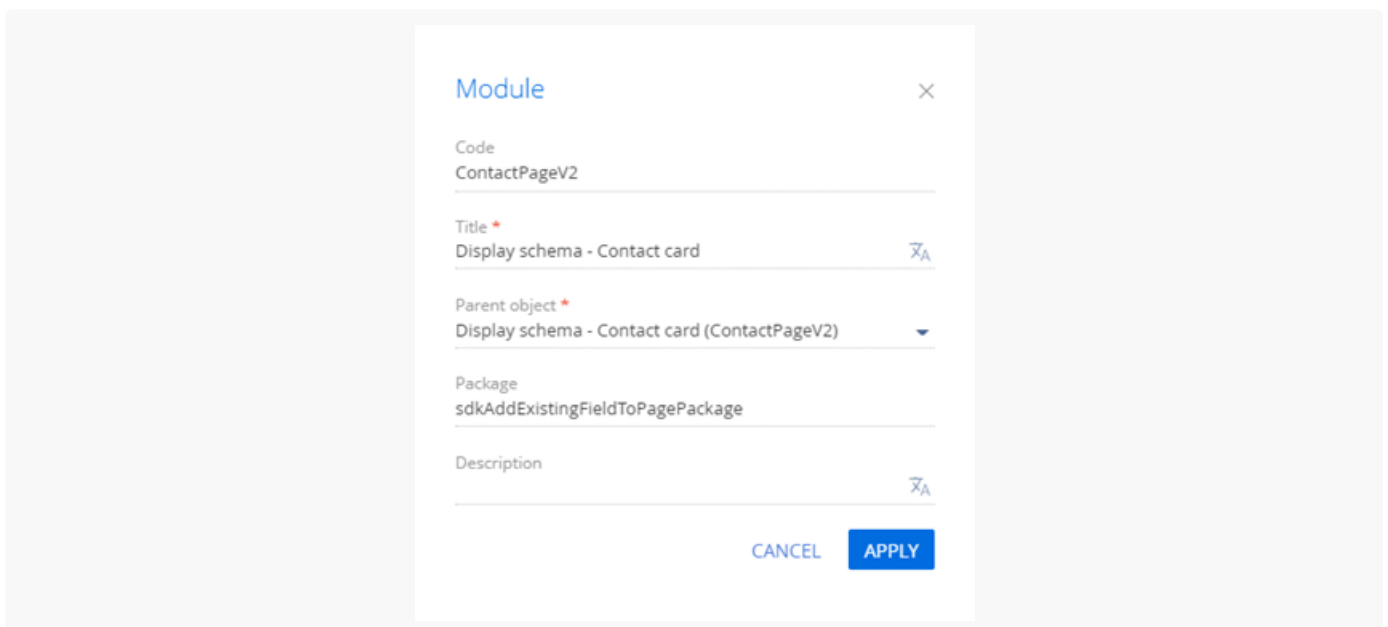
## Создать схему замещающей модели представления страницы контакта

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



### 3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "ContactPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Схема отображения карточки контакта" ("Display schema - Contact card").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "ContactPageV2".



4. Настройте **расположение поля**. Для этого в массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля на странице.

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

#### ContactPageV2

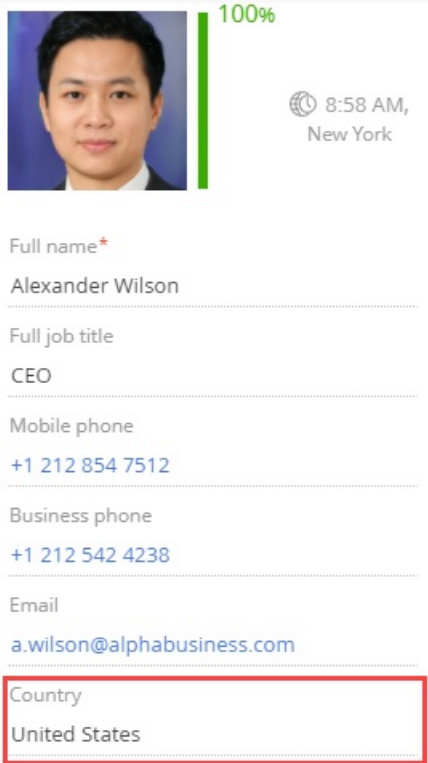
```
define("ContactPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение поля на странице записи. */
        diff: [
            /* Метаданные для добавления на страницу поля [Страна]. */
            {
                /* Выполняется операция добавления элемента на страницу. */
                "operation": "insert",
                /* Мета-имя родительского контейнера, в который добавляется поле. */
                "parentName": "ProfileContainer",
                /* Поле добавляется в коллекцию элементов родительского элемента. */
                "propertyName": "items",
                /* Мета-имя добавляемого поля. */
                "name": "Country",
                /* Свойства, передаваемые в конструктор элемента. */
                "values": {
                    /* Тип поля – справочник. */
                    "contentType": Terrasoft.ContentType.LOOKUP,
                    /* Настройка расположения поля. */
                    "layout": {
                        /* Номер столбца. */
                        "column": 0,
                        /* Номер строки. */
                        "row": 6,
                        /* Диапазон занимаемых столбцов. */
                        "colSpan": 24
                    }
                }
            }
        ]
    };
});
```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ *Контакты* ] ([ *Contacts* ]).

В результате выполнения примера в профиль контакта страницы контакта добавлено поле [ Страна ] ([ Country ]).



100%

8:58 AM,  
New York

Full name\*

Alexander Wilson

Full job title

CEO

Mobile phone

+1 212 854 7512

Business phone

+1 212 542 4238

Email

a.wilson@alphabusiness.com

Country

United States

## Добавить поле с изображением на страницу записи

 Сложный

**Пример.** Добавить поле с изображением на страницу статьи базы знаний. Использовать изображение, которое приведено ниже.

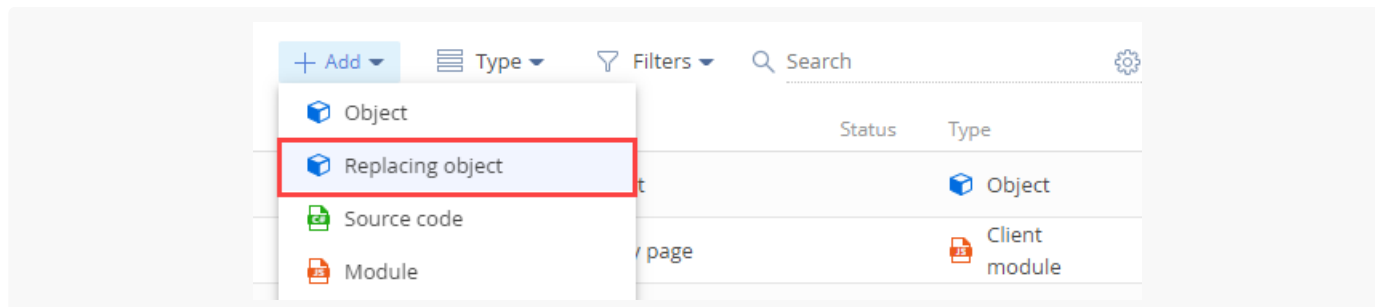


### 1. Создать схему замещающего объекта

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который

будет добавлена схема.

2. На панели инструментов реестра раздела нажмите [ *Добавить* ] —> [ *Замещающий объект* ] ([ *Add* ] —> [ *Replacing object* ]).



3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "KnowledgeBase".
- [ *Заголовок* ] ([ *Title* ]) — "Статья базы знаний" ("Knowledge base article").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "KnowledgeBase".

**General**

Code \*  
KnowledgeBase

Title \*  
Knowledge base article

Package  
sdkAddFieldWithImageToPagePackage

Description

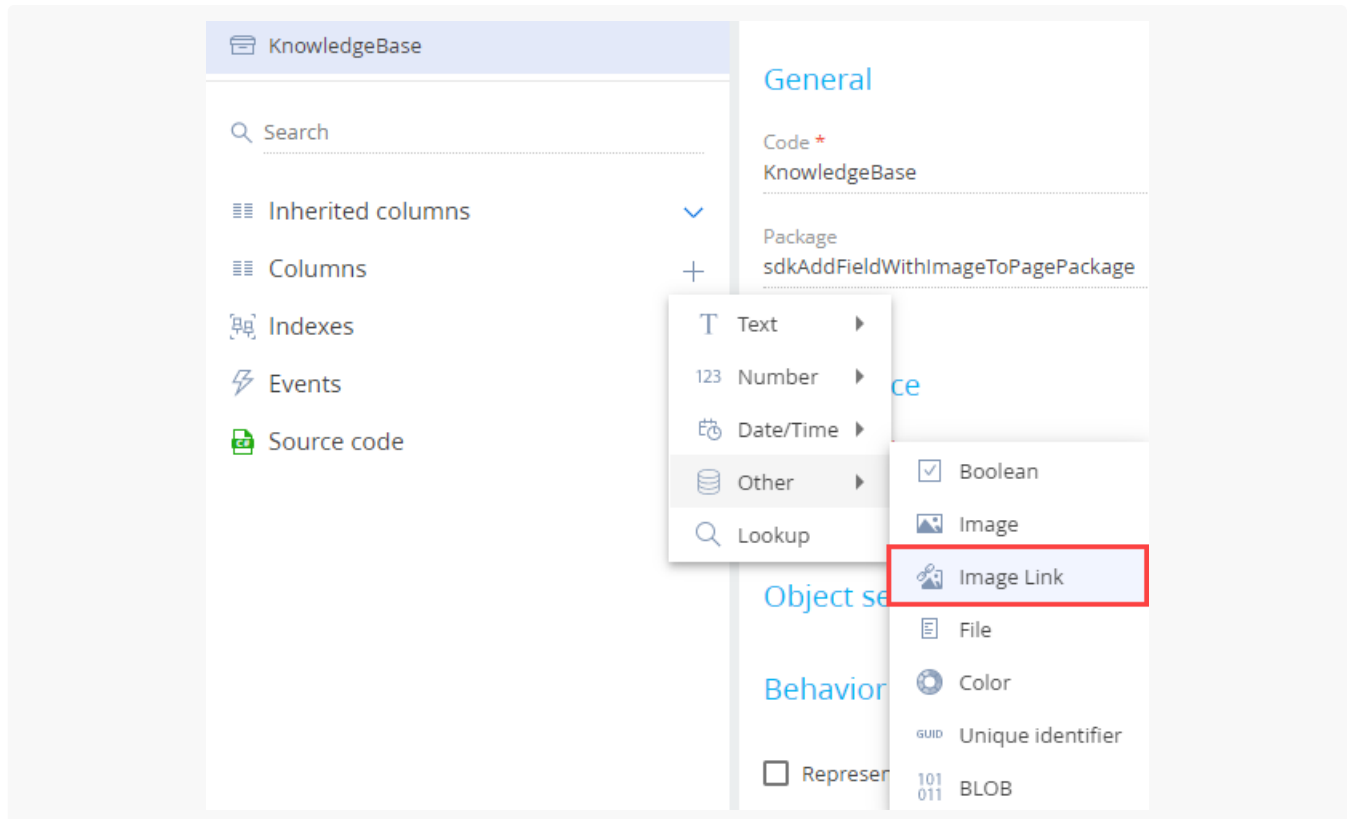
**Inheritance**

Parent object \*  
KnowledgeBase

☒ Replace parent

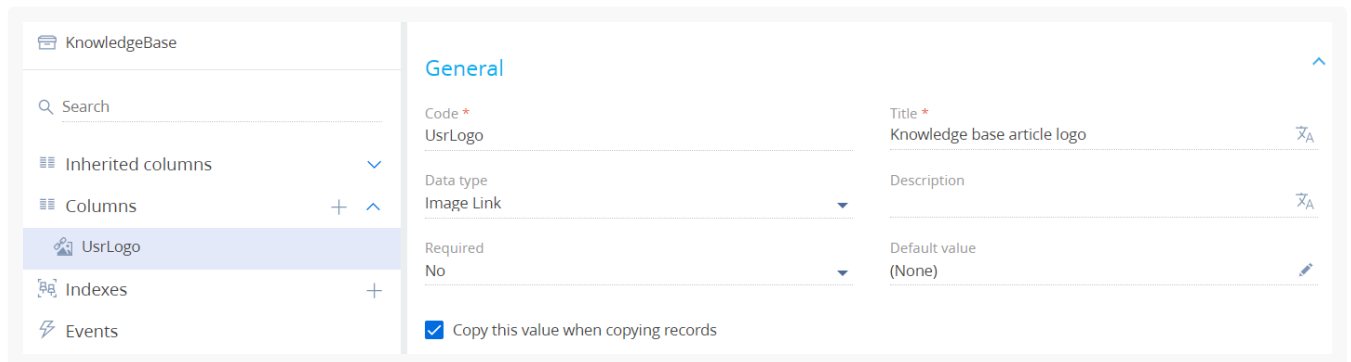
4. В схему добавьте **колонку**.

- a. В контекстном меню узла [ *Колонки* ] ([ *Columns* ]) структуры объекта нажмите +.
- b. В выпадающем меню нажмите [ *Другие* ] —> [ *Ссылка на изображение* ] ([ *Other* ] —> [ *Image Link* ]).



с. Заполните **свойства добавляемой колонки**.

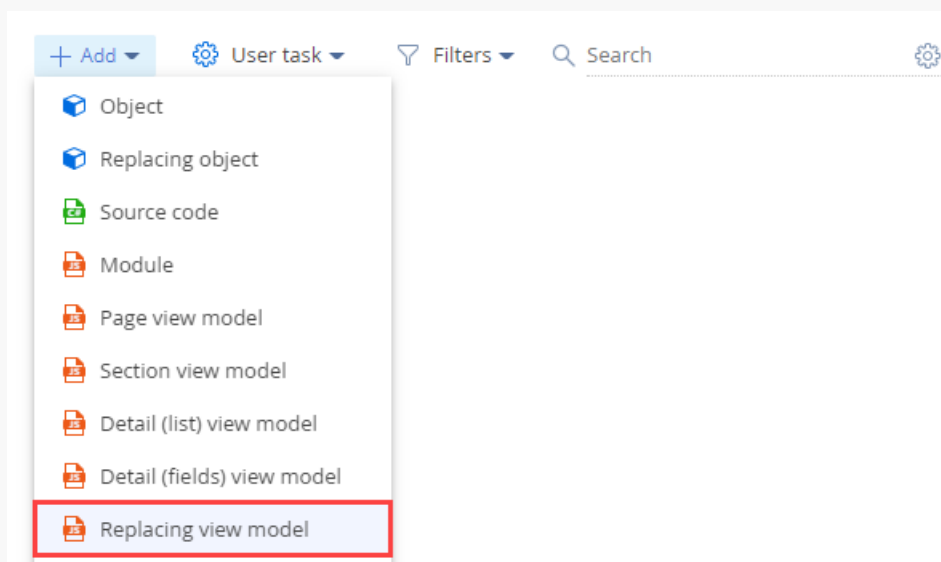
- [ Код ] ([ Code ]) — "UsrLogo".
- [ Заголовок ] ([ Title ]) — "Логотип статьи базы знаний" ("Knowledge base article logo").



5. На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]), а затем [ Опубликовать ] ([ Publish ]).

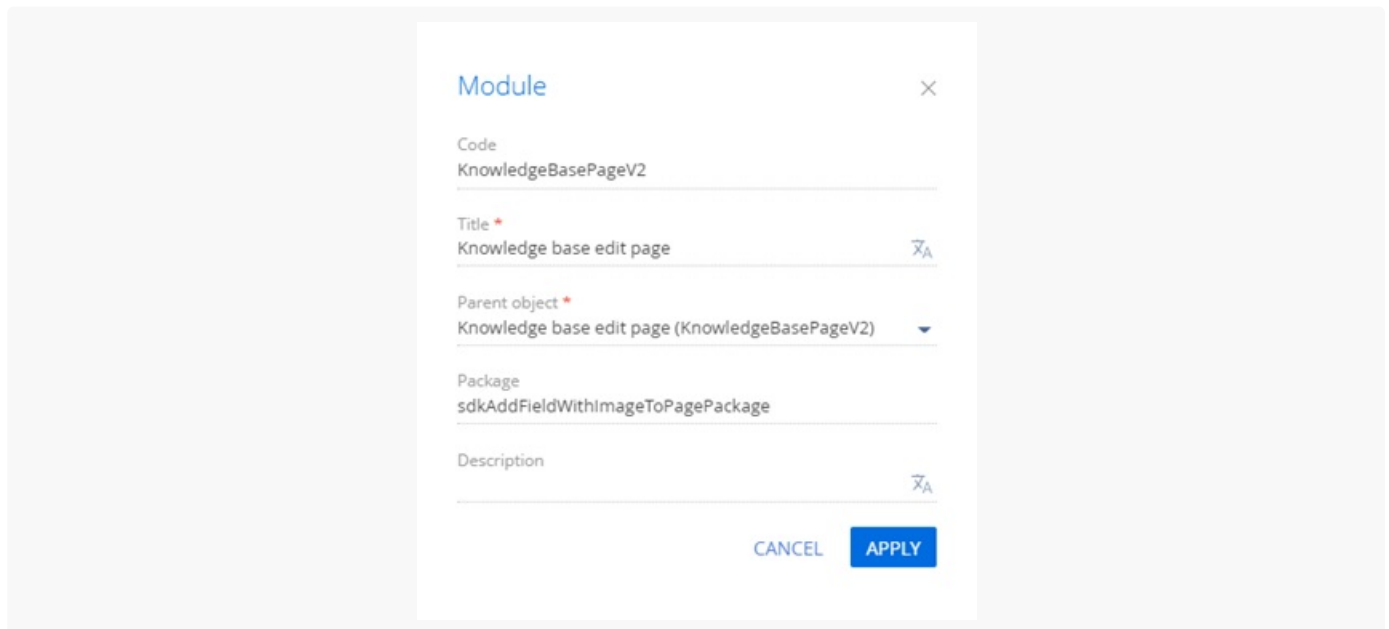
## 2. Создать схему замещающей модели представления страницы статьи базы знаний

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ Add ] —> [ Replacing view model ]).



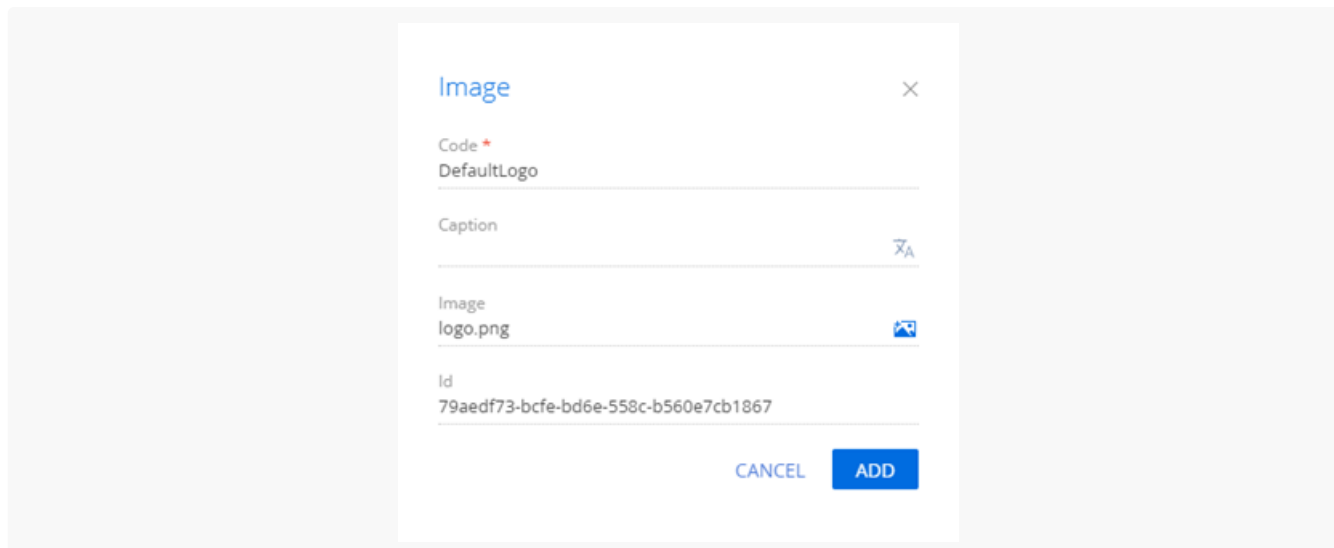
### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "KnowledgeBasePageV2".
- [ Заголовок ] ([ Title ]) — "Knowledge base edit page".
- [ Родительский объект ] ([ Parent object ]) — выберите "KnowledgeBasePageV2".



### 4. Добавьте **изображение**.

- В контекстном меню узла [ Изображения ] ([ Images ]) нажмите кнопку **+**.
- Заполните **свойства изображения**.
  - [ Код ] ([ Code ]) — "DefaultLogo".
  - [ Изображение ] ([ Image ]) — выберите файл с изображением.



е. Для добавления изображения нажмите [ *Добавить* ] ([ *Add* ]).

5. В объявлении класса модели представления в качестве зависимостей добавьте модули

`KnowledgeBasePageV2Resources` и `ConfigurationConstants`.

6. Настройте **расположение поля с изображением**.

Поле с изображением планируется разместить в верхней части страницы статьи базы знаний. Добавление поля с изображением может нарушить расположение полей базовой страницы. Чтобы этого избежать, кроме размещения поля с изображением, необходимо дополнительно изменить расположение существующих полей, которые находятся в верхней части страницы. Это поля [ *Название* ] ([ *Name* ]), [ *Тип* ] ([ *Type* ]), [ *Изменил* ] ([ *Modified By* ]).

а. В свойстве `methods` реализуйте **методы**:

- `getPhotoSrcMethod()` — получает изображение по ссылке.
- `beforePhotoFileSelected()` — вызывается перед открытием диалогового окна выбора изображения.
- `onPhotoChange()` — вызывается при изменении изображения.
- `onPhotoUploaded()` — сохраняет ссылку на измененное изображение в колонке объекта.

ф. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля с изображением и существующих полей на странице. Поле с изображением добавляется на страницу с использованием вспомогательного контейнера-обертки `PhotoContainer` с классом `"image-edit-container"`.

Исходный код схемы замещающей модели представления страницы статьи базы знаний представлен ниже.

`KnowledgeBasePageV2`

```
define("KnowledgeBasePageV2", ["KnowledgeBasePageV2Resources", "ConfigurationConstants"], fun
    return {
        /* Название схемы объекта страницы записи. */
```



```

entitySchemaName: "KnowledgeBase",
/* Методы модели представления страницы раздела. */
methods: {
    /* Вызывается перед открытием диалогового окна выбора изображения. */
    beforePhotoFileSelected: function() {
        return true;
    },
    /* Получает изображение по ссылке. */
    getPhotoSrcMethod: function() {
        /* Получает ссылку на изображение из колонки объекта. */
        var imageColumnValue = this.get("UsrLogo");
        /* Если ссылка установлена, то метод возвращает url файла с изображением. */
        if (imageColumnValue) {
            return this.getSchemaImageUrl(imageColumnValue);
        }
        /* Если ссылка не установлена, то возвращает изображение по умолчанию. */
        return this.Terrasoft.ImageUrlBuilder.getUrl(this.get("Resources.Images.DefaultImage"));
    },
    /* Обработывает изменение изображения.
    photo – файл с изображением. */
    onPhotoChange: function(photo) {
        if (!photo) {
            this.set("UsrLogo", null);
            return;
        }
        /* Выполняет загрузку файла в базу данных. По окончании загрузки вызывается метод
        this.Terrasoft.ImageApi.upload({
            file: photo,
            onComplete: this.onPhotoUploaded,
            onError: this.Terrasoft.emptyFn,
            scope: this
        });
    },
    /* Сохраняет ссылку на измененное изображение.
    imageId – Id сохраненного файла из базы данных. */
    onPhotoUploaded: function(imageId) {
        var imageData = {
            value: imageId,
            displayValue: "Image"
        };
        /* Колонке изображения присваивается ссылка на изображение. */
        this.set("UsrLogo", imageData);
    },
    /* Отображение поля на странице раздела. */
    diff: /**SCHEMA_DIFF*/[
        /* Метаданные для добавления на страницу записи пользовательского поля с изображением. */
        {
            /* Выполняется операция добавления элемента на страницу. */

```

```

"operation": "insert",
/* Мета-имя родительского контейнера, в который добавляется поле. */
"parentName": "Header",
/* Изображение добавляется в коллекцию элементов родительского элемента. */
"propertyName": "items",
/* Мета-имя добавляемого изображения. */
"name": "PhotoContainer",
/* Свойства, передаваемые в конструктор элемента. */
"values": {
    /* Тип добавляемого элемента – контейнер. */
    "itemType": Terrasoft.ViewItemType.CONTAINER,
    /* Имя CSS класса. */
    "wrapClass": ["image-edit-container"],
    /* Настройка расположения изображения. */
    "layout": {
        /* Номер столбца. */
        "column": 0,
        /* Номер строки. */
        "row": 0,
        /* Диапазон занимаемых строк. */
        "rowSpan": 3,
        /* Диапазон занимаемых столбцов. */
        "colSpan": 3
    },
    /* Массив дочерних элементов. */
    "items": []
}
},
/* Поле [UsrLogo] – поле с логотипом контрагента. */
{
    "operation": "insert",
    "parentName": "PhotoContainer",
    "propertyName": "items",
    "name": "UsrLogo",
    "values": {
        /* Метод, который получает изображение по ссылке. */
        "getSrcMethod": "getPhotoSrcMethod",
        /* Метод, который вызывается при изменении изображения. */
        "onPhotoChange": "onPhotoChange",
        /* Метод, который вызывается перед вызовом диалогового окна выбора изобра
        "beforeFileSelected": "beforePhotoFileSelected",
        /* Свойство, которое определяет возможность редактирования изображения. *
        "readonly": false,
        /* View-генератор элемента управления. */
        "generator": "ImageCustomGeneratorV2.generateCustomImageControl"
    }
},
/* Изменение расположения поля [Name]. */

```

```

{
    /* Выполняется операция изменения существующего элемента. */
    "operation": "merge",
    "name": "Name",
    "parentName": "Header",
    "propertyName": "items",
    "values": {
        "bindTo": "Name",
        "layout": {
            "column": 3,
            "row": 0,
            "colSpan": 20
        }
    }
},
/* Изменение расположения поля [ModifiedBy]. */
{
    "operation": "merge",
    "name": "ModifiedBy",
    "parentName": "Header",
    "propertyName": "items",
    "values": {
        "bindTo": "ModifiedBy",
        "layout": {
            "column": 3,
            "row": 2,
            "colSpan": 20
        }
    }
},
/* Изменение расположения поля [Type]. */
{
    "operation": "merge",
    "name": "Type",
    "parentName": "Header",
    "propertyName": "items",
    "values": {
        "bindTo": "Type",
        "layout": {
            "column": 3,
            "row": 1,
            "colSpan": 20
        },
        "contentType": Terrasoft.ContentType.ENUM
    }
}
]/**SCHEMA_DIFF*/
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ База знаний ] ([ Knowledge base ]).

В результате выполнения примера на страницу статьи базы знаний добавлено поле с изображением. Изображение можно изменить или удалить.

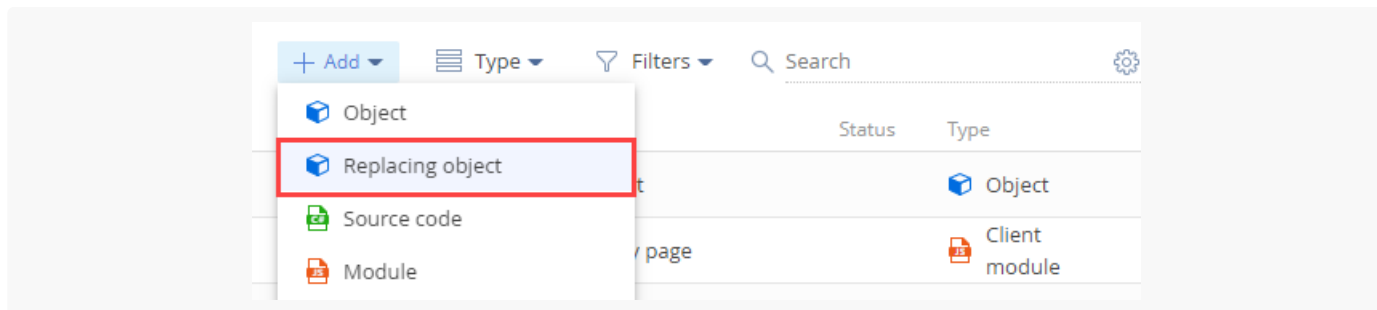
## Добавить вычисляемое поле на страницу записи

Средний

**Пример.** Добавить вычисляемое поле [ Остаток для оплаты ] ([ Payment balance ]) на страницу заказа. В поле отображается разница между суммой заказа (поле [ Итого ] ([ Total ])) и суммой оплаты (поле [ Сумма оплаты ] ([ Payment amount ])).

### 1. Создать схему замещающего объекта

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

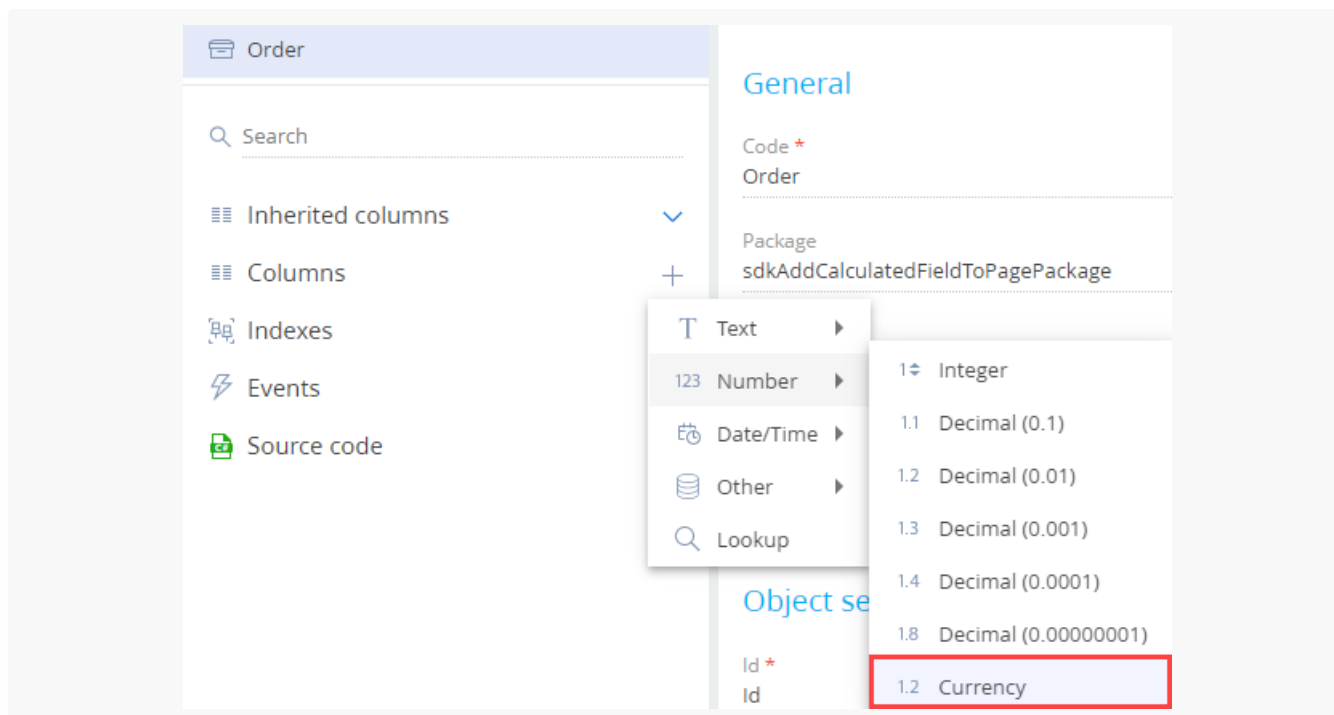


### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "Order".
- [ Заголовок ] ([ Title ]) — "Заказ" ("Order").
- [ Родительский объект ] ([ Parent object ]) — выберите "Order".

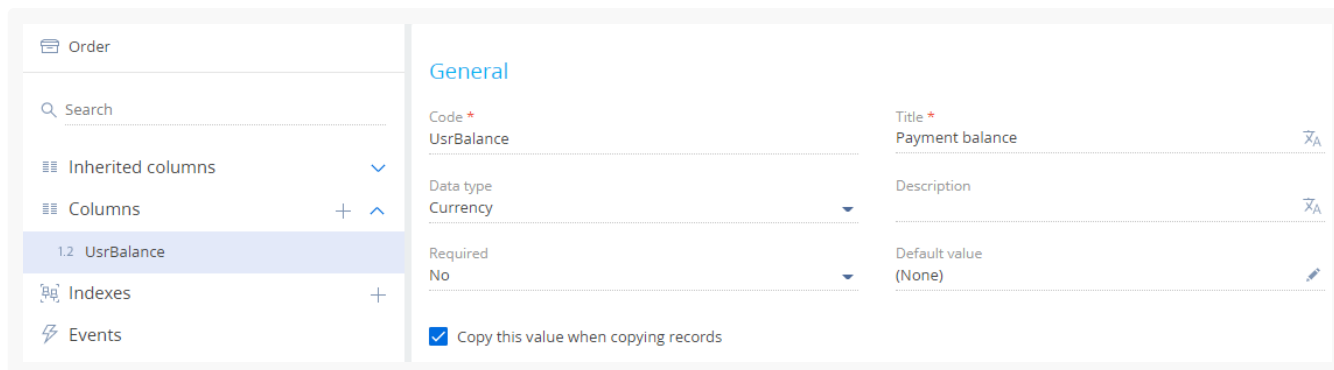
### 4. В схему добавьте **колонку**.

- В контекстном меню узла [ Колонки ] ([ Columns ]) структуры объекта нажмите **+**.
- В выпадающем меню нажмите [ Число ] —> [ Деньги ] ([ Number ] —> [ Currency ]).



с. Заполните **свойства добавляемой колонки**.

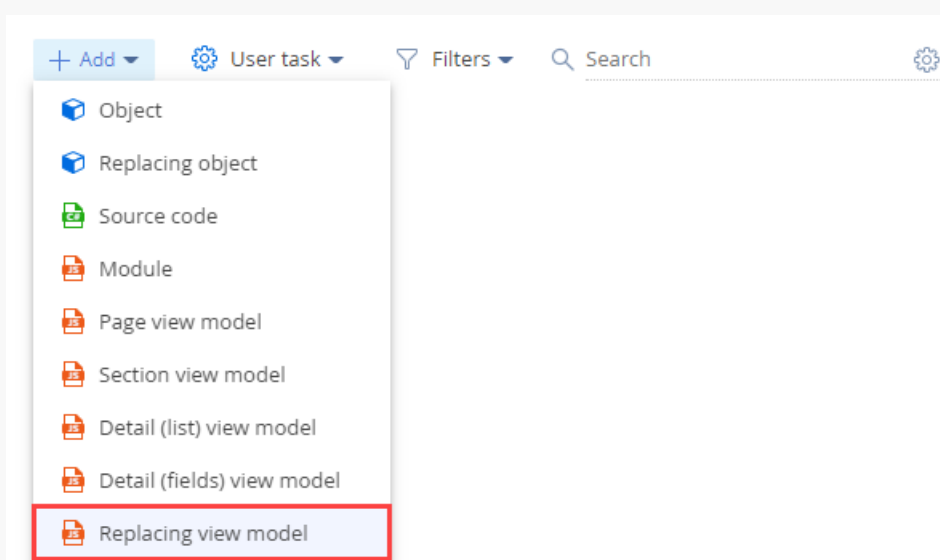
- [ Код ] ([ Code ]) — "UsrBalance".
- [ Заголовок ] ([ Title ]) — "Остаток для оплаты" ("Payment balance").



5. На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]), а затем [ Опубликовать ] ([ Publish ]).

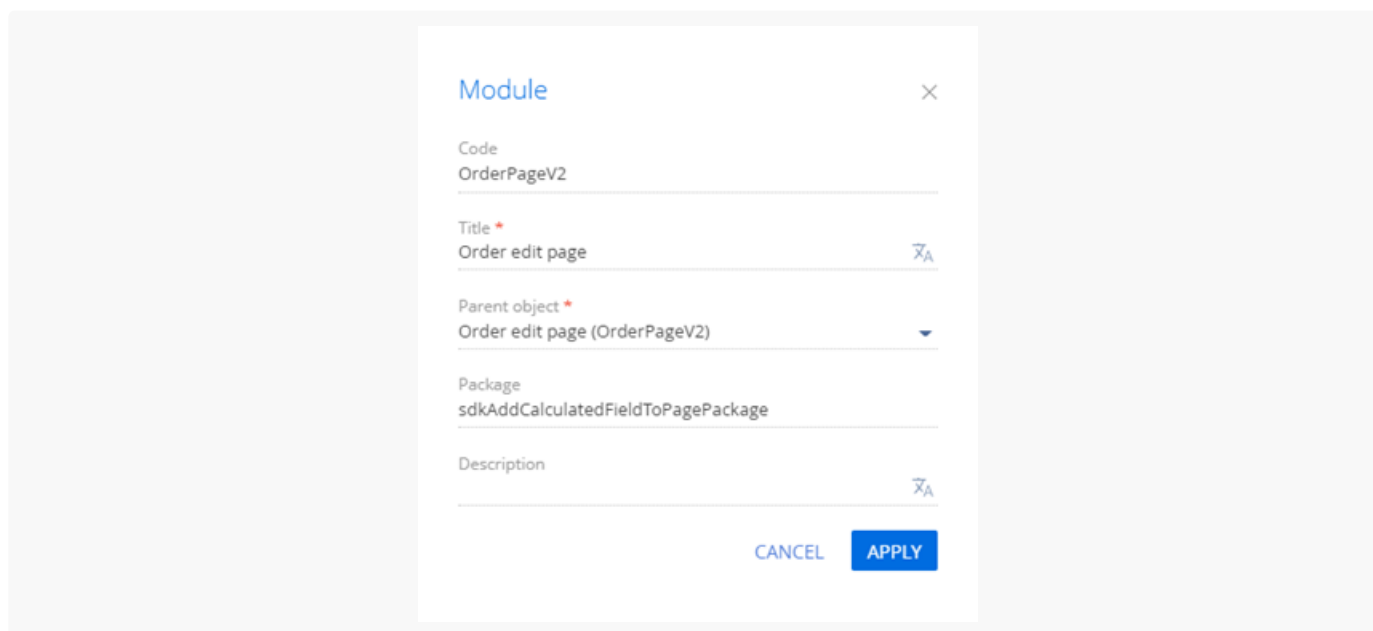
## 2. Создать схему замещающей модели представления страницы заказа

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ Add ] —> [ Replacing view model ]).



### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "OrderPageV2".
- [ Заголовок ] ([ Title ]) — "Страница редактирования заказа" ("Order edit page").
- [ Родительский объект ] ([ Parent object ]) — выберите "OrderPageV2".



### 4. Настройте **расположение вычисляемого поля**.

- В свойство `attributes` добавьте атрибут `UsrBalance`, в котором укажите зависимость от колонок [ Amount ] и [ PaymentAmount ], а также метод-обработчик `calculateBalance()`.
- В свойстве `methods` реализуйте **методы**:
  - `calculateBalance()` — метод-обработчик события изменения колонок [ Amount ] и [ PaymentAmount ]. Используется для расчета значения колонки [ UsrBalance ], которая указана в атрибуте `UsrBalance`.

В методе-обработчике необходимо учесть тип данных, который будет получен в результате выполнения и отображен в вычисляемом поле. Например, тип данных [ *Дробное число (0.01)* ] ([ *Decimal (0.01)* ]) предполагает число с двумя знаками после запятой. В таком случае перед записью результата в поле объекта необходимо преобразовать тип данных с помощью функции `toFixed()`. Исходный код примера преобразования типа данных представлен ниже.

#### Пример преобразования типа данных

```
/* Расчет разницы между значениями в колонках [Amount] и [PaymentAmount]. */
var result = amount - paymentAmount;
/* Результат расчета присваивается в качестве значения колонке [UsrBalance]. */
this.set("UsrBalance", result.toFixed(2));
```

- `onEntityInitialized()` — переопределяет базовый виртуальный метод. Срабатывает после выполнения инициализации схемы объекта страницы записи. В метод `onEntityInitialized()` добавьте вызов метода-обработчика `calculateBalance()`, который обеспечит расчет суммы остатка для оплаты в момент открытия страницы записи, а не только после изменений в колонках-зависимостях.

е. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения вычисляемого поля.

Исходный код схемы замещающей модели представления страницы заказа представлен ниже.

#### OrderPageV2

```
define("OrderPageV2", [], function() {
  return {
    /* Название схемы объекта страницы записи. */
    entitySchemaName: "Order",
    details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
    /* Атрибуты модели представления. */
    attributes: {
      /* Название атрибута модели представления. */
      "UsrBalance": {
        /* Тип данных колонки модели представления. */
        dataValueType: Terrasoft.DataValueType.FLOAT,
        /* Массив конфигурационных объектов, которые определяют зависимости колонки [
        dependencies: [
          {
            /* Значение колонки [UsrBalance] зависит от значений колонок [Amount]
            columns: ["Amount", "PaymentAmount"],
            /* Метод-обработчик, который вызывается при изменении значения одной
            methodName: "calculateBalance"
          }
        ]
      }
    }
  }
}
```



```

},
/* Методы модели представления страницы записи. */
methods: {
  /* Переопределение базового метода Terrasoft.BasePageV2.onEntityInitialized, кото
onEntityInitialized: function() {
  /* Вызывается родительская реализация метода. */
  this.callParent(arguments);
  /* Вызов метода-обработчика, который рассчитывает значение колонки [UsrBalance]
  this.calculateBalance();
},
/* Метод-обработчик, который рассчитывает значение колонки [UsrBalance]. */
calculateBalance: function() {
  /* Проверка выполнения инициализации колонок [Amount] и [PaymentAmount] в мом
  var amount = this.get("Amount");
  if (!amount) {
    amount = 0;
  }
  var paymentAmount = this.get("PaymentAmount");
  if (!paymentAmount) {
    paymentAmount = 0;
  }
  /* Расчет разницы между значениями колонок [Amount] и [PaymentAmount]. */
  var result = amount - paymentAmount;
  /* Результат расчета присваивается в качестве значения колонке [UsrBalance].
  this.set("UsrBalance", result);
}
},
/* Отображение поля на странице записи. */
diff: /**SCHEMA_DIFF*/[
  /* Метаданные для добавления на страницу записи вычисляемого поля. */
  {
    /* Выполняется операция добавления элемента на страницу. */
    "operation": "insert",
    /* Мета-имя родительского контейнера, в который добавляется поле. */
    "parentName": "Header",
    /* Поле добавляется в коллекцию элементов родительского элемента. */
    "propertyName": "items",
    /* Мета-имя добавляемого поля. */
    "name": "UsrBalance",
    /* Свойства, передаваемые в конструктор элемента. */
    "values": {
      /* Привязка значения элемента управления к колонке модели представления.
      "bindTo": "UsrBalance",
      /* Настройка расположения поля. */
      "layout": {
        /* Номер столбца. */
        "column": 12,
        /* Номер строки. */
        "row": 2,

```

```

        /* Диапазон занимаемых столбцов. */
        "colSpan": 12
    }
}
}
]/**SCHEMA_DIFF*/
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Очистите кэш браузера.
2. Обновите страницу раздела [ Заказы ] ([ Orders ]).

В результате выполнения примера на страницу заказа добавлено вычисляемое поле [ Остаток для оплаты ] ([ Payment balance ]). Значение поля — разница между суммой заказа (поле [ Итого ] ([ Total ])) и суммой оплаты (поле [ Сумма оплаты ] ([ Payment amount ])).

ORD-26

What can I do for you? >

Creatio 7.18.4.1532

SAVE CANCEL ACTIONS PRINT VIEW

Customer\* Streamline Development

Status 4. Completed

Total, \$ 13,850.00

Payment amount, \$ 13,000.00

Payment balance 850.00

## Добавить мультивалютное поле на страницу записи

Сложный

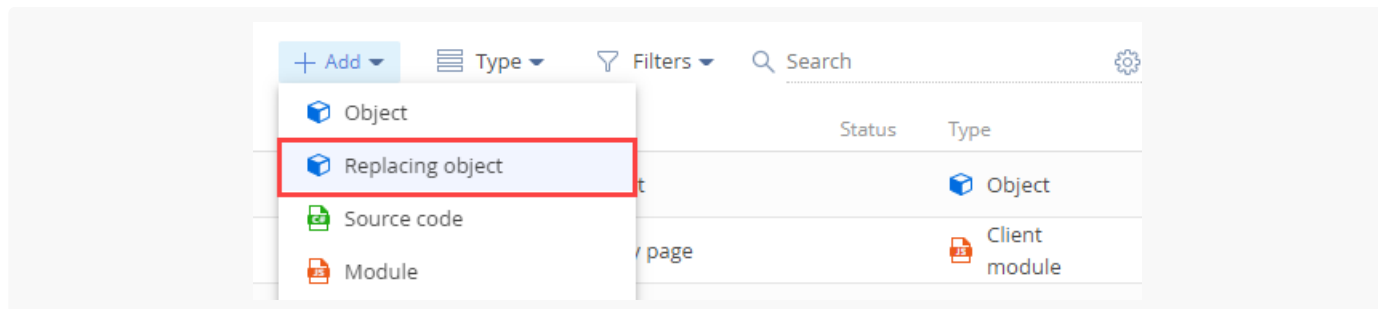
**Пример.** Добавить мультивалютное поле [ Общая сумма ] ([ Amount ]) на страницу проекта.

### 1. Создать схему замещающего объекта

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который

будет добавлена схема.

2. На панели инструментов реестра раздела нажмите [ *Добавить* ] —> [ *Замещающий объект* ] ([ *Add* ] —> [ *Replacing object* ]).

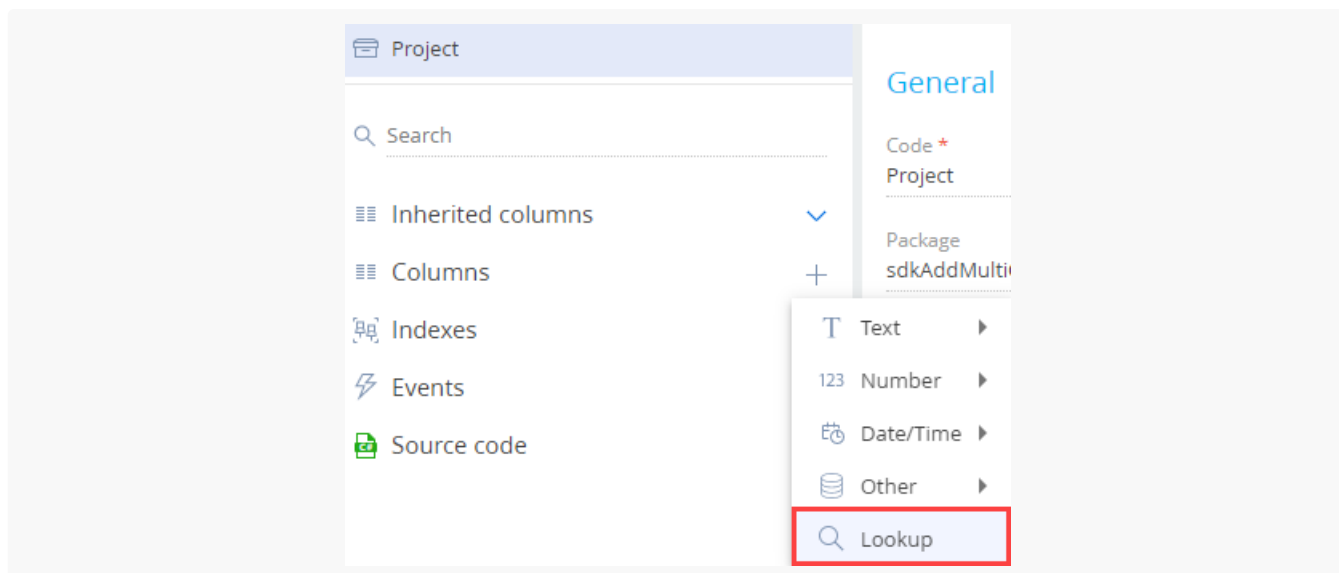


3. Заполните **свойства схемы**.


- [ *Код* ] ([ *Code* ]) — "Project".
- [ *Заголовок* ] ([ *Title* ]) — "Проект" ("Project").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "Project".

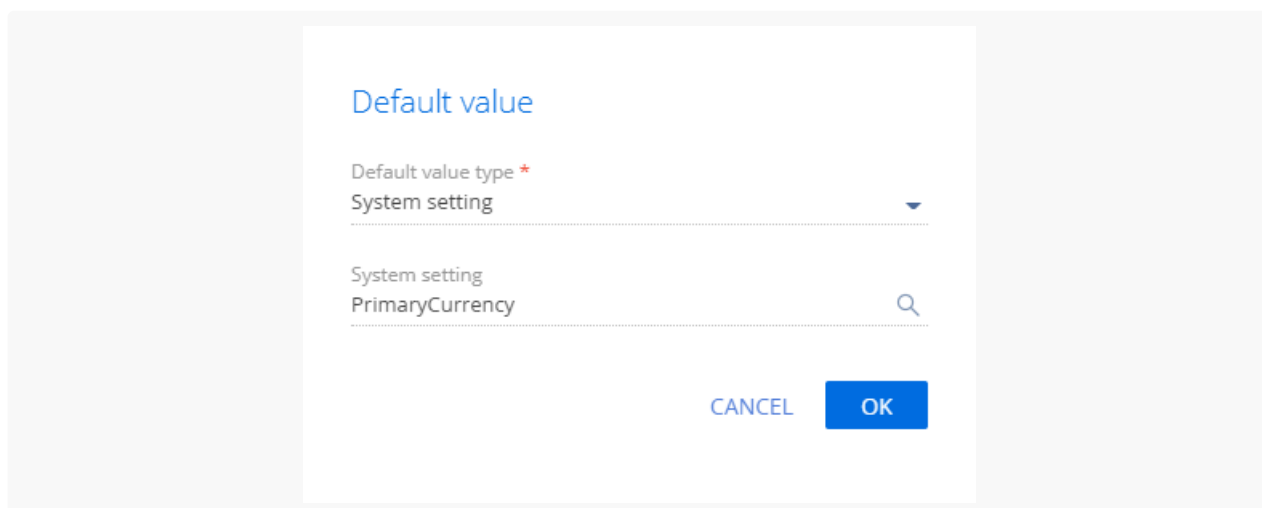
4. В схему добавьте **колонку**.

- a. В контекстном меню узла [ *Колонки* ] ([ *Columns* ]) структуры объекта нажмите +.
- b. В выпадающем меню нажмите [ *Справочник* ] ([ *Lookup* ]).



с. Заполните **свойства добавляемой колонки**.

- [ Код ] ([ Code ]) — "UsrCurrency".
- [ Заголовок ] ([ Title ]) — "Валюта" ("Currency").
- [ Справочник ] ([ Lookup ]) — выберите "Currency".
- [ Значение по умолчанию ] ([ Default value ]).
  - В поле [ Значение по умолчанию ] ([ Default value ]) нажмите .
  - [ Тип значения ] ([ Default value type ]) — выберите "Системная настройка" ("System setting").
  - [ Системная настройка ] ([ System setting ]) — выберите "Базовая валюта" ("Base currency", код PrimaryCurrency ).



Свойства колонки представлены на рисунке ниже.

The screenshot shows the 'Project' configuration window. On the left, a sidebar lists 'Inherited columns' and 'Columns'. Under 'Columns', 'UsrCurrency' is selected. The main area is divided into 'General' and 'Data source' sections. In the 'General' section, 'Code' is 'UsrCurrency', 'Data type' is 'Lookup', 'Required' is 'No', and 'Copy this value when copying records' is checked. In the 'Data source' section, 'Lookup' is 'Currency' and 'Do not control integrity' is unchecked.

5. Аналогично добавьте колонки, которые содержат общую сумму, сумму в базовой валюте и курс валют. Свойства колонок приведены в таблице ниже.

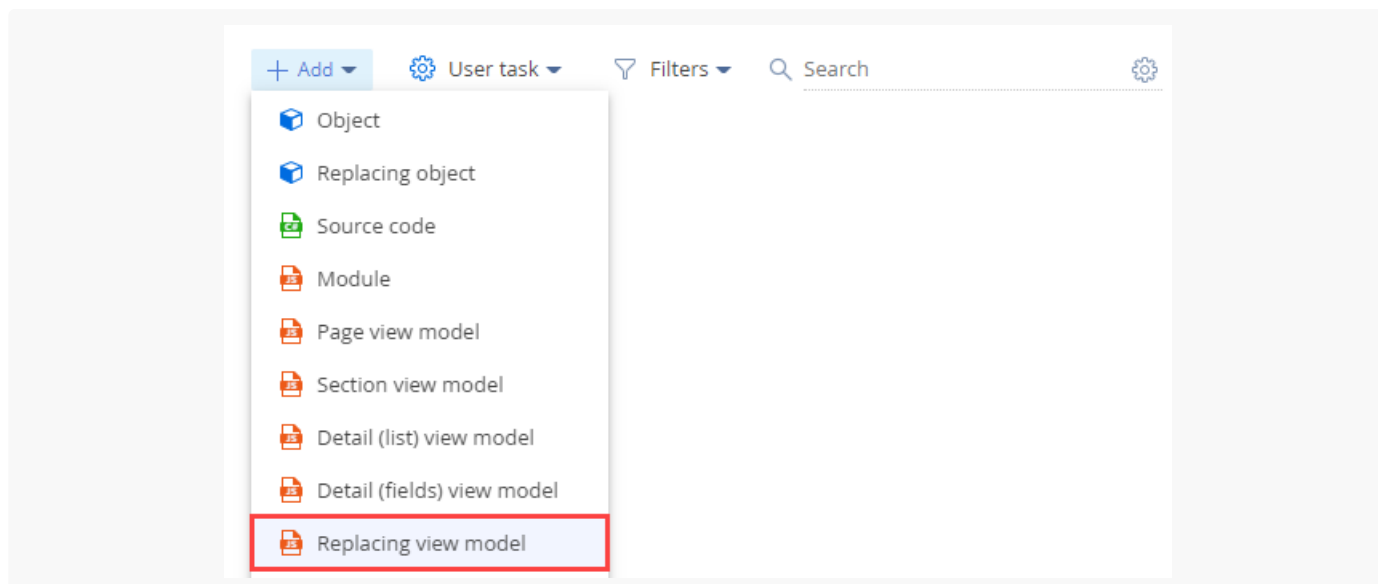
Свойства добавляемых колонок

[ Тип данных ] ([ Data type ])	[ Код ] ([ Code ])	[ Заголовок ] ([ Title ])
[ Деньги ] ([ Currency ])	"UsrAmount"	"Общая сумма" ("Amount")
[ Деньги ] ([ Currency ])	"UsrPrimaryAmount"	"Сумма в базовой валюте" ("Amount, base currency")
[ Дробное число (0,0001) ] ([ Decimal (0.0001) ])	"UsrCurrencyRate"	"Курс валют" ("Exchange rate")

6. На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]), а затем [ Опубликовать ] ([ Publish ]).

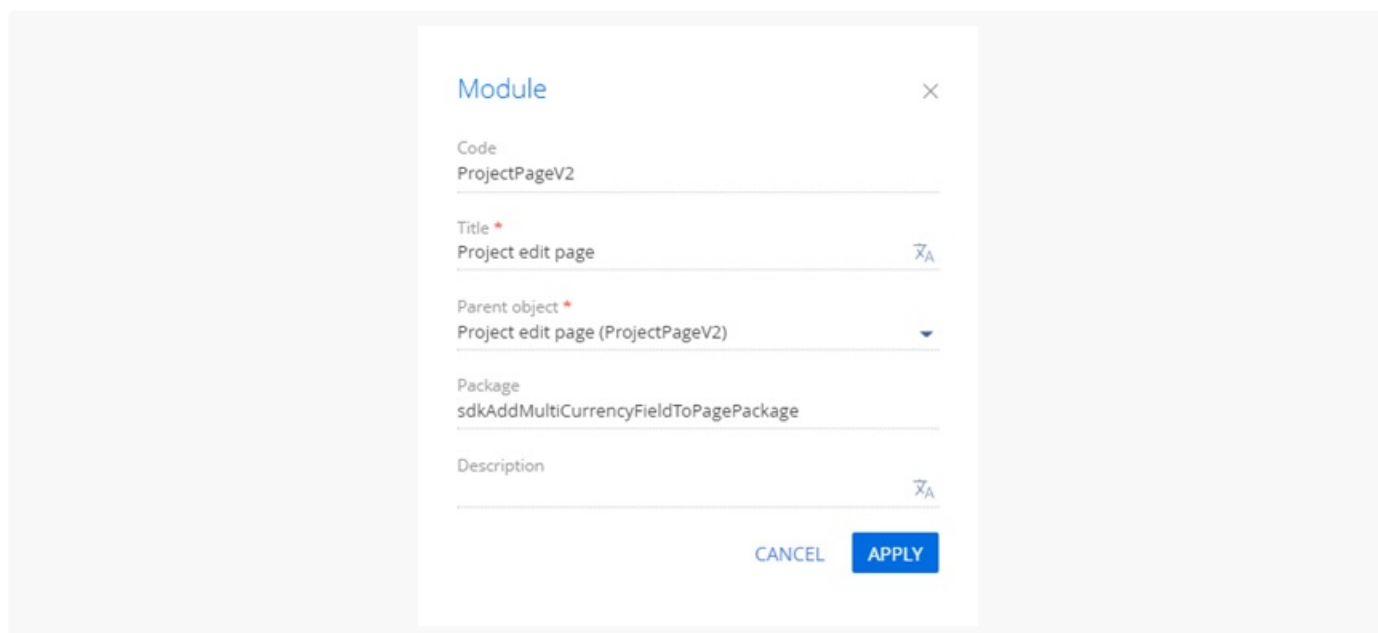
## 2. Создать схему замещающей модели представления страницы проекта

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ Add ] —> [ Replacing view model ]).



### 3. Заполните **свойства схемы**.

- [ Код ] ([ *Code* ]) — "ProjectPageV2".
- [ Заголовок ] ([ *Title* ]) — "Страница редактирования проекта" ("Project edit page").
- [ Родительский объект ] ([ *Parent object* ]) — выберите "ProjectPageV2".



4. В объявлении класса модели представления в качестве зависимостей добавьте модули `MoneyModule`, `MultiCurrencyEdit`, `MultiCurrencyEditUtilities`.

### 5. Настройте **расположение мультивалютного поля**.

a. В свойство `attributes` добавьте **атрибуты**:

- `UsrCurrency` — валюта. Соответствует колонке [ *UsrCurrency* ].
- `UsrCurrencyRate` — курс валюты. Соответствует колонке [ *UsrCurrencyRate* ].

- `UsrAmount` — общая сумма. Соответствует колонке [ *UsrAmount* ].
  - `UsrPrimaryAmount` — сумма в базовой валюте. Соответствует колонке [ *UsrPrimaryAmount* ].
  - `Currency` — валюта. Соответствует колонке [ *Currency* ]. Это колонка, с которой взаимодействует мультивалютный модуль. В атрибуте `Currency` объявите виртуальную колонку, которую с помощью метода-обработчика свяжите с колонкой [ *UsrCurrency* ].
  - `CurrencyRateList` — коллекция курсов валют. Предназначен для корректной работы мультивалютного модуля.
  - `CurrencyButtonMenuList` — коллекция для кнопки выбора валюты. Предназначен для корректной работы мультивалютного модуля.
- i. В свойство `mixins` добавьте миксин `MultiCurrencyEditUtilities`.
- j. В свойстве `methods` реализуйте **методы**:
- `onEntityInitialized()` — переопределяет базовый виртуальный метод. Срабатывает после выполнения инициализации схемы объекта страницы записи.
  - `setCurrencyRate()` — устанавливает курс валюты.
  - `recalculateAmount()` — пересчитывает общую сумму.
  - `recalculatePrimaryAmount()` — пересчитывает сумму в базовой валюте.
  - `onVirtualCurrencyChange()` — метод-обработчик изменения виртуальной колонки валюты.
- p. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения мультивалютного поля.

Исходный код схемы замещающей модели представления страницы проекта представлен ниже.

#### ProjectPageV2

```
/* В качестве зависимостей укажите модули MoneyModule, MultiCurrencyEdit и MultiCurrencyEditU
define("ProjectPageV2", ["MoneyModule", "MultiCurrencyEdit", "MultiCurrencyEditUtilities"],
  function(MoneyModule, MultiCurrencyEdit, MultiCurrencyEditUtilities) {
    return {
      /* Название схемы объекта страницы записи. */
      entitySchemaName: "Project",
      /* Атрибуты модели представления. */
      attributes: {
        /* Валюта. */
        "UsrCurrency": {
          /* Тип данных колонки модели представления. */
          "dataValueType": this.Terrasoft.DataValueType.LOOKUP,
          /* Конфигурация справочника валют. */
          "lookupListConfig": {
            "columns": ["Division", "Symbol"]
          }
        },
        /* Курс. */
```

```

"UsrCurrencyRate": {
    "dataValueType": this.Terrasoft.DataValueType.FLOAT,
    /* Массив конфигурационных объектов, которые определяют зависимости колон
    "dependencies": [
        {
            /* Значение колонки [UsrCurrencyRate] зависит от значения колонки
            "columns": ["UsrCurrency"],
            /* Метод-обработчик. */
            "methodName": "setCurrencyRate"
        }
    ]
},
/* Общая сумма. */
"UsrAmount": {
    "dataValueType": this.Terrasoft.DataValueType.FLOAT,
    "dependencies": [
        {
            "columns": ["UsrCurrencyRate", "UsrCurrency"],
            "methodName": "recalculateAmount"
        }
    ]
},
/* Сумма в базовой валюте. */
"UsrPrimaryAmount": {
    "dependencies": [
        {
            "columns": ["UsrAmount"],
            "methodName": "recalculatePrimaryAmount"
        }
    ]
},
/* Валюта – виртуальная колонка для совместимости с модулем MultiCurrencyEdit
"Currency": {
    "type": this.Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
    "dataValueType": this.Terrasoft.DataValueType.LOOKUP,
    "lookupListConfig": {
        "columns": ["Division"]
    },
    "dependencies": [
        {
            "columns": ["Currency"],
            "methodName": "onVirtualCurrencyChange"
        }
    ]
},
/* Коллекция курсов валют. */
"CurrencyRateList": {
    dataValueType: this.Terrasoft.DataValueType.COLLECTION,
    value: this.Ext.create("Terrasoft.Collection")
}

```



```

    },
    /* Коллекция для кнопки выбора валюты. */
    "CurrencyButtonMenuList": {
        dataValueType: this.Terrasoft.DataValueType.COLLECTION,
        value: this.Ext.create("Terrasoft.BaseViewModelCollection")
    }
},
/* Миксины модели представления. */
mixins: {
    /* Миксин управления мультивалютностью на странице записи. */
    MultiCurrencyEditUtilities: "Terrasoft.MultiCurrencyEditUtilities"
},
/* Методы модели представления страницы записи. */
methods: {
    /* Переопределение базового метода Terrasoft.BasePageV2.onEntityInitialized()
    onEntityInitialized: function() {
        /* Вызывается родительская реализация метода. */
        this.callParent(arguments);
        this.set("Currency", this.get("UsrCurrency"), {silent: true});
        /* Инициализация миксина управления мультивалютностью. */
        this.mixins.MultiCurrencyEditUtilities.init.call(this);
    },
    /* Устанавливает курс валюты. */
    setCurrencyRate: function() {
        /* Загружает курс валют на дату начала проекта. */
        MoneyModule.LoadCurrencyRate.call(this, "UsrCurrency", "UsrCurrencyRate",
    },
    /* Пересчитывает сумму. */
    recalculateAmount: function() {
        var currency = this.get("UsrCurrency");
        var division = currency ? currency.Division : null;
        MoneyModule.RecalcCurrencyValue.call(this, "UsrCurrencyRate", "UsrAmount"
    },
    /* Пересчитывает сумму в базовой валюте. */
    recalculatePrimaryAmount: function() {
        var currency = this.get("UsrCurrency");
        var division = currency ? currency.Division : null;
        MoneyModule.RecalcBaseValue.call(this, "UsrCurrencyRate", "UsrAmount", "U
    },
    /* Обработчик изменения виртуальной колонки валюты. */
    onVirtualCurrencyChange: function() {
        var currency = this.get("Currency");
        this.set("UsrCurrency", currency);
    }
},
/* Отображение поля на странице записи. */
diff: /**SCHEMA_DIFF*/[
    /* Метаданные для добавления на страницу записи мультивалютного поля. */

```

```

{
    /* Выполняется операция добавления элемента на страницу. */
    "operation": "insert",
    /* Мета-имя родительского контейнера, в который добавляется поле. */
    "parentName": "Header",
    /* Поле добавляется в коллекцию элементов родительского элемента. */
    "propertyName": "items",
    /* Мета-имя добавляемого поля. */
    "name": "UsrAmount",
    /* Свойства, передаваемые в конструктор элемента. */
    "values": {
        /* Привязка значения элемента управления к колонке модели представлен
        "bindTo": "UsrAmount",
        /* Настройка расположения поля. */
        "layout": {
            /* Номер столбца. */
            "column": 0,
            /* Номер строки. */
            "row": 2,
            /* Диапазон занимаемых столбцов. */
            "colSpan": 12
        },
        /* Наименование колонки, которая содержит сумму в базовой валюте. */
        "primaryAmount": "UsrPrimaryAmount",
        /* Наименование колонки, которая содержит валюту суммы. */
        "currency": "UsrCurrency",
        /* Наименование колонки, которая содержит курс валюты. */
        "rate": "UsrCurrencyRate",
        /* Свойство, которое определяет доступность для редактирования поля с
        "primaryAmountEnabled": false,
        /* Генератор представления элемента управления. */
        "generator": "MultiCurrencyEditViewGenerator.generate"
    }
}
]/**SCHEMA_DIFF*/
};
});

```

6. На панели инструментов дизайнера нажмите [ *Сохранить* ] ([ *Save* ]).

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ *Проекты* ] ([ *Projects* ]).

В результате выполнения примера на страницу проекта добавлено мультивалютное поле [ *Общая сумма* ] ([ *Amount* ]).

Software Documentation Development

What can I do for you? >

7.18.4.1532

VIEW

SAVE CANCEL ACTIONS

Name\* Software Documentation Development

Status\* In progress

Owner\* Sarah M. Richards

Amount, aud 20.01

aud

€

hrn.

rub.

\$

GENERAL INFO STRUCTURE FINANCIAL INDICATORS HISTORY ATTACHMENTS AND NOTES

Contact Andrew Wayne

Type\* Maintenance

Значение поля автоматически пересчитывается после выбора валюты в выпадающем списке.

Software Documentation Development

What can I do for you? >

7.18.4.1532

VIEW

SAVE CANCEL ACTIONS

Name\* Software Documentation Development

Status\* In progress

Owner\* Sarah M. Richards

Amount, \$ 16.00

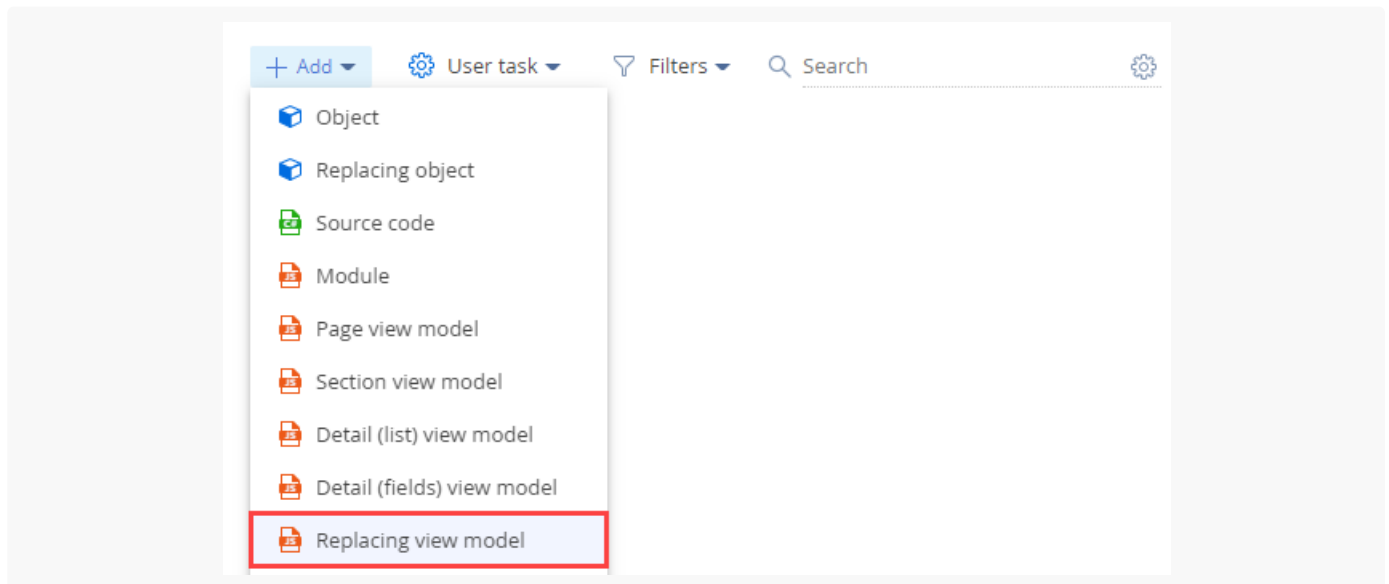
## Реализовать валидацию поля типа [Дата/Время] на странице записи

Средний

**Пример.** Реализовать валидацию поля [ Дата создания ] ([ Created on ]) типа [ Дата/Время ] ([ Date/Time ]) на странице продажи. Значение поля [ Дата создания ] ([ Created on ]) должно быть меньше значения поля [ Дата закрытия ] ([ Closed on ]).

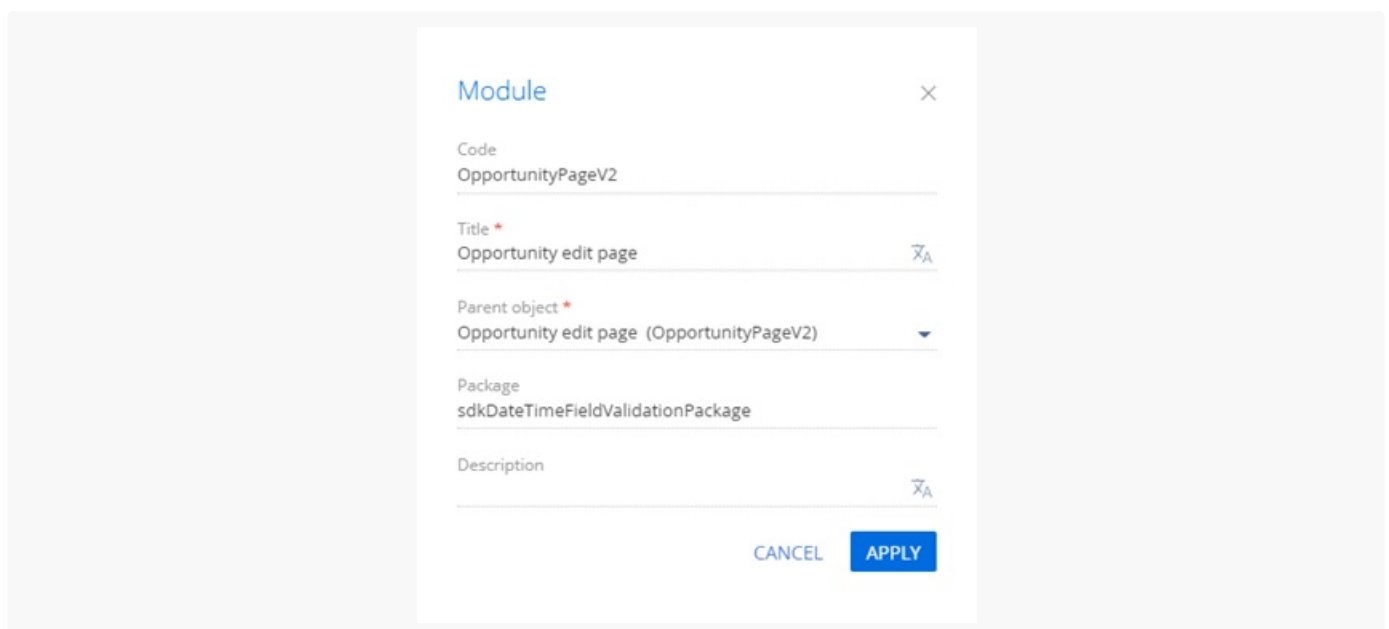
## Создать схему замещающей модели представления страницы продажи

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



3. Заполните **свойства схемы**.

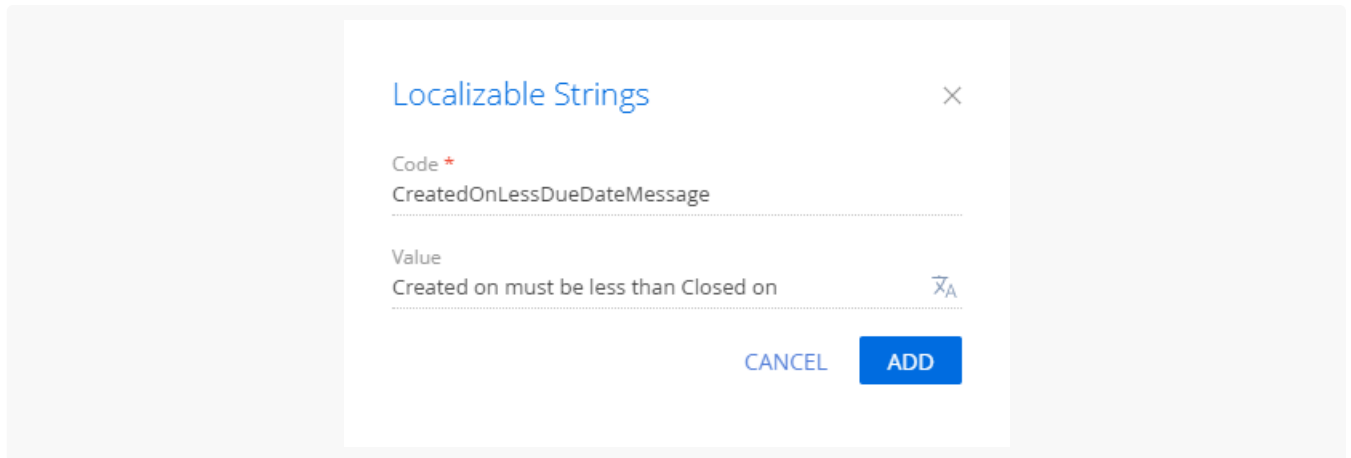
- [ *Код* ] ([ *Code* ]) — "OpportunityPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Страница редактирования продажи" ("Opportunity edit page").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "OpportunityPageV2".



4. Добавьте **локализуемую строку**.

- a. В контекстном меню узла [ *Локализуемые строки* ] ([ *Localizable strings* ]) нажмите кнопку [+](#).
- b. Заполните **свойства локализуемой строки**.

- [ Код ] ([ Code ]) — "CreatedOnLessDueDateMessage".
- [ Значение ] ([ Value ]) — "Дата создания должна быть меньше даты закрытия" ("Created on must be less than Closed on").



е. Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).

## 5. Реализуйте **валидацию поля типа [ Дата/Время ]** ([ Date/Time ]).

Для этого в свойстве `methods` реализуйте **методы**:

- `dueDateValidator()` — метод-валидатор, который определяет выполнение условия.
- `setValidationConfig()` — переопределенный базовый метод, в котором метод-валидатор привязан к колонкам [ DueDate ] и [ CreatedOn ].

Исходный код схемы замещающей модели представления страницы продажи представлен ниже.

### OpportunityPageV2

```
define("OpportunityPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Opportunity",
        /* Методы модели представления страницы раздела. */
        methods: {
            /* Метод-валидатор значения колонок [DueDate] и [CreatedOn]. */
            dueDateValidator: function() {
                /* Переменная для хранения сообщения об ошибке валидации. */
                var invalidMessage = "";
                /* Проверка значений колонок [DueDate] и [CreatedOn]. */
                if (this.get("DueDate") < this.get("CreatedOn")) {
                    /* Если значение колонки [DueDate] меньше значения колонки [CreatedOn], т
                    invalidMessage = this.get("Resources.Strings.CreatedOnLessDueDateMessage"
                }
                /* Объект, свойство которого содержит сообщение об ошибке валидации. Если вал
                return {
                    /* Сообщение об ошибке валидации. */
```

```

        invalidMessage: invalidMessage
    };
},
/* Переопределение базового метода, который инициализирует пользовательские валид
setValidationConfig: function() {
    /* Вызывает инициализацию валидаторов родительской модели представления. */
    this.callParent(arguments);
    /* Для колонки [DueDate] добавляется метод-валидатор dueDateValidator(). */
    this.addColumnValidator("DueDate", this.dueDateValidator);
    /* Для колонки [CreatedOn] добавляется метод-валидатор dueDateValidator(). */
    this.addColumnValidator("CreatedOn", this.dueDateValidator);
}
}
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Обновите страницу раздела [ Продажи ] ([ Opportunities ]).
2. В поле [ Дата закрытия ] ([ Closed on ]) выберите дату, значение которой меньше значения поля [ Дата создания ] ([ Created on ]).

В результате выполнения примера на странице продажи отображается соответствующее предупреждение.

При попытке сохранить продажу, у которой значение поля [ Дата закрытия ] ([ Closed on ]) меньше значения поля [ Дата создания ] ([ Created on ]) отображается информационное сообщение.

Field "Closed on": Created on must be less than  
Closed on

OK

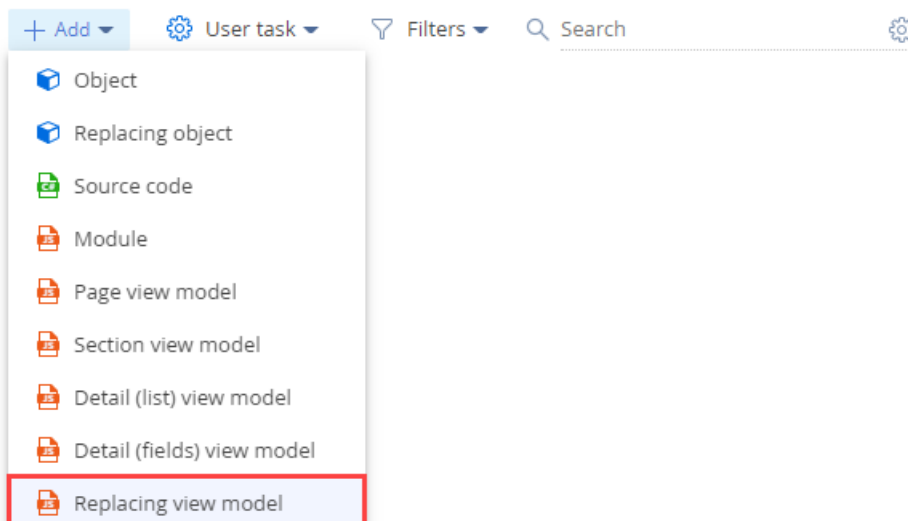
## Реализовать валидацию поля типа [Строка] на странице записи

 Средний

**Пример.** Реализовать валидацию поля [ Рабочий телефон ] ([ Business phone ]) типа [ Строка ] ([ String ]) на странице контакта. Значение поля [ Рабочий телефон ] ([ Business phone ]) должно соответствовать маске `+44 xxx xxx xxxx`.

## Создать схему замещающей модели представления страницы контакта

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ Add ] —> [ Replacing view model ]).



3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "ContactPageV2".

- [ Заголовок ] ([ Title ]) — "Схема отображения карточки контакта" ("Display schema - Contact card").
- [ Родительский объект ] ([ Parent object ]) — выберите "ContactPageV2".

The screenshot shows a 'Module' dialog box with the following fields:

- Code:** ContactPageV2
- Title:** Display schema - Contact card
- Parent object:** Display schema - Contact card (ContactPageV2)
- Package:** sdkStringFieldValidationPackage
- Description:** (empty)

Buttons: CANCEL, APPLY

4. Добавьте **локализуемую строку**.

- В контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
  - [ Код ] ([ Code ]) — "InvalidPhoneFormatMessage".
  - [ Значение ] ([ Value ]) — "Введите номер в формате: +44 XXX XXX XXXX" ("Enter the number in format +44 XXX XXX XXXX").

The screenshot shows a 'Localizable Strings' dialog box with the following fields:

- Code:** InvalidPhoneFormatMessage
- Value:** Enter the number in format +44 XXX XXX XXXX

Buttons: CANCEL, ADD

- Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).
5. В объявлении класса модели представления в качестве зависимостей добавьте модуль `ConfigurationConstants`.
6. Реализуйте **валидацию поля типа** [ Строка ] ([ String ]).

Для этого в свойстве `methods` реализуйте **методы**:



- `phoneValidator()` — метод-валидатор, который определяет выполнение условия.
- `setValidationConfig()` — переопределенный базовый метод, в котором метод-валидатор привязан к колонке [ *Phone* ].

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

#### ContactPageV2

```
define("ContactPageV2", ["ConfigurationConstants"], function(ConfigurationConstants) {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Методы модели представления страницы записи. */
        methods: {
            /* Переопределение базового метода, который инициализирует пользовательские валид
            setValidationConfig: function() {
                /* Вызывает инициализацию валидаторов родительской модели представления. */
                this.callParent(arguments);
                /* Для колонки [Phone] добавляется метод-валидатор phoneValidator(). */
                this.addColumnValidator("Phone", this.phoneValidator);
            },
            /* Метод-валидатор значения колонки [Phone]. */
            phoneValidator: function(value) {
                /* Переменная для хранения сообщения об ошибке валидации. */
                var invalidMessage = "";
                /* Переменная для хранения результата проверки номера. */
                var isValid = true;
                /* Переменная для хранения номера телефона. */
                var number = value || this.get("Phone");
                /* Определение правильности формата номера с помощью регулярного выражения. */
                isValid = (Ext.isEmpty(number) ||
                    new RegExp("^\\+44\\s[0-9]{3}\\s[0-9]{3}\\s[0-9]{4}$").test(number));
                /* Если формат номера неправильный, то заполняется сообщение об ошибке. */
                if (!isValid) {
                    invalidMessage = this.get("Resources.Strings.InvalidPhoneFormatMessage");
                }
                /* Объект, свойство которого содержит сообщение об ошибке валидации. Если вал
                return {
                    invalidMessage: invalidMessage
                };
            }
        }
    };
});
```

7. На панели инструментов дизайнера нажмите [ *Сохранить* ] ([ *Save* ]).

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Обновите страницу раздела [ *Контакты* ] ([ *Contacts* ]).
2. В поле [ *Рабочий телефон* ] ([ *Business phone* ]) введите номер телефона, который не соответствует маске `+44 xxx xxx xxxx`.

В результате выполнения примера на странице контакта отображается соответствующее предупреждение.

Alexander Wilson

What can I do for you? >

7.18.4.1532

SAVE CANCEL ACTIONS VIEW

100%

12:28 AM, New York

Full name\*  
Alexander Wilson

Full job title  
CEO

Mobile phone  
+1 212 854 7512

Business phone  
+1 212 542 4238  
Enter the number in format +44 XXX XXX XXXX

Next Steps (8)

Define the problem with Laserjet Pro CP1025nw 10/2/2021   Marina Kysla	Analyze the case 10/4/2021   William Walker
Meet with Wilson in his office 10/4/2021   Marina Kysla	Call back the customer. Inform him about the case resolution. 10/4/2021   William Walker
Conference call (Alpha Business) 10/5/2021   Marina Kysla	Call back to the client. Inform about case resolution 10/6/2021   Marina Kysla

show all

При попытке сохранить контакт, у которого номер телефона не соответствует маске `+44 xxx xxx xxxx`, отображается информационное сообщение.

Field "Business phone": Enter the number in format  
+44 XXX XXX XXXX

OK

## Установить значение по умолчанию для

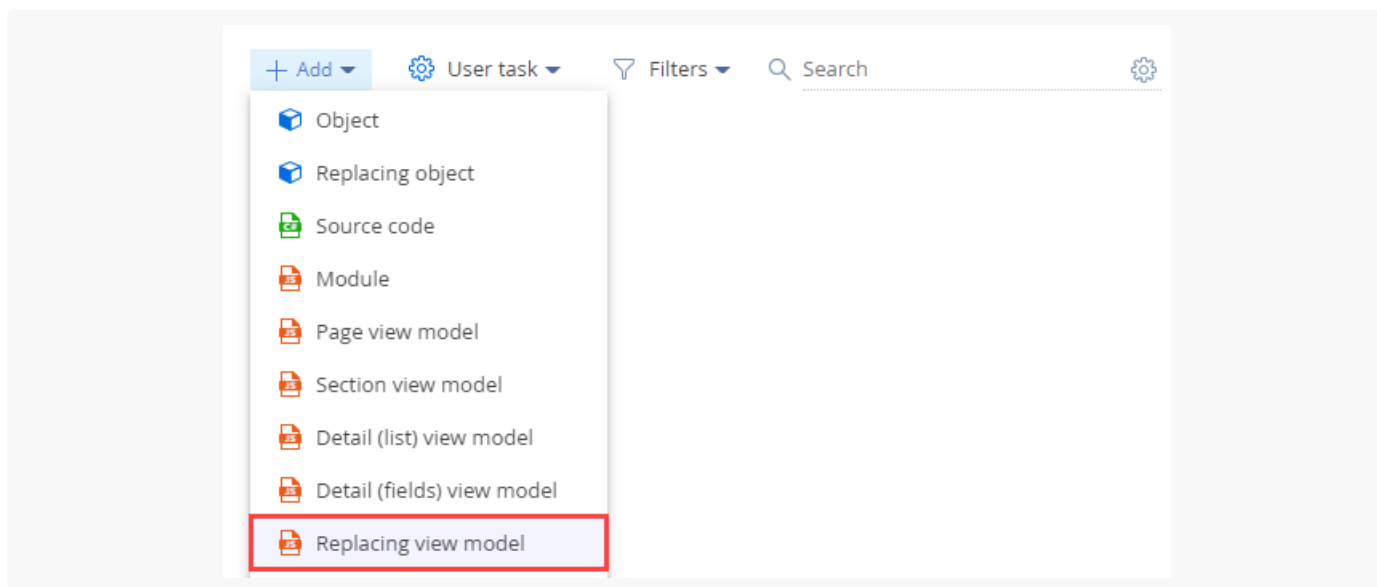
# поля на странице записи

Средний

**Пример.** Установить значение по умолчанию для поля [ *Крайний срок* ] ([ *Deadline* ]) на странице добавления проекта. Значение поля [ *Крайний срок* ] ([ *Deadline* ]) должно быть на 10 дней больше значения поля [ *Начало* ] ([ *Start* ]).

## Создать схему замещающей модели представления страницы проекта

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



3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "ProjectPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Страница редактирования проекта" ("Project edit page").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "ProjectPageV2".

#### 4. Настройте **логику заполнения поля**.

Для этого в свойстве `methods` реализуйте **методы**:

- `onEntityInitialized()` — переопределенный базовый виртуальный метод. Срабатывает после окончания инициализации схемы объекта. В метод `onEntityInitialized()` добавьте вызов метода-обработчика `setDeadline()`, который обеспечит установку значения поля [ *Крайний срок* ] ([ *Deadline* ]) в момент открытия страницы записи.
- `setDeadline()` — метод-обработчик, который рассчитывает значение поля [ *Крайний срок* ] ([ *Deadline* ]).

Исходный код схемы замещающей модели представления страницы проекта представлен ниже.

##### ProjectPageV2

```
define("ProjectPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Project",
        /* Методы модели представления страницы записи. */
        methods: {
            /* Переопределение базового метода Terrasoft.BasePageV2.onEntityInitialized, кото
            onEntityInitialized: function() {
                /* Вызывается родительская реализация метода. */
                this.callParent(arguments);
                /* Вызов метода-обработчика, который рассчитывает значение колонки [Deadline]
                this.setDeadline();
            },
            /* Метод-обработчик, который рассчитывает значение колонки [Deadline]. */
            setDeadline: function() {
                /* Значение колонки [Deadline]. */
                var deadline = this.get("Deadline");
```

```

/* Проверяет установку режима новой записи. */
var newmode = this.isNewMode();
/* Если значение не установлено и режим новой записи установлен. */
if (!deadline && newmode) {
    /* Получает значение колонки [StartDate]. */
    var newDate = new Date(this.get("StartDate"));
    newDate.setDate(newDate.getDate() + 10);
    /* Установка значения колонки [Deadline]. */
    this.set("Deadline", newDate);
}
}
}
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ Проекты ] ([ Projects ]).

В результате выполнения примера на странице добавления проекта значение поля [ Крайний срок ] ([ Deadline ]) устанавливается на 10 дней больше значения поля [ Начало ] ([ Start ]).

The screenshot shows a form with the following fields and values:

- Account: [Empty]
- Contact: [Empty]
- Completion %: [Empty]
- Calculate automatically: ☐
- Start: 11/19/2021 (highlighted with a red box)
- End: [Empty]
- Type\*: [Empty]
- Duration: 0 min
- Deadline: 11/29/2021 (highlighted with a red box)

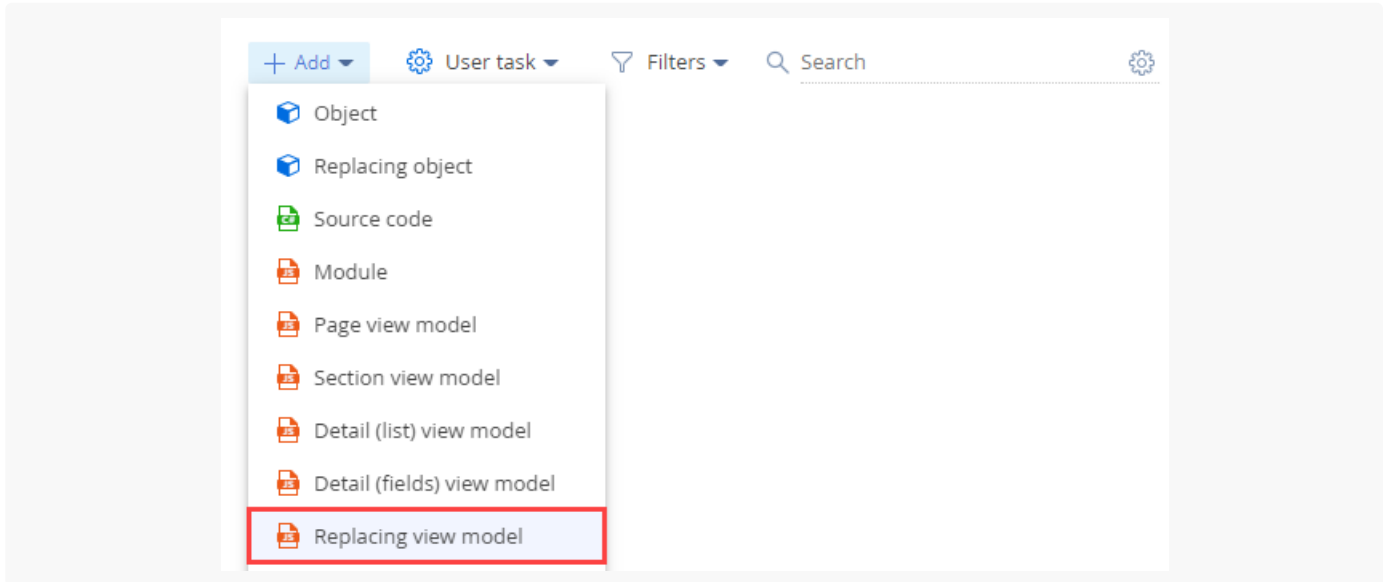
## Настроить обязательность для поля на странице записи

Средний

**Пример.** Настроить обязательность для поля [ Рабочий телефон ] ([ Business phone ]) страницы контакта. Поле обязательное для контакта типа "Клиент" ("Customer") (т. е. в поле [ Тип контакта ] ([ Type ]) выбрано значение "Клиент" ("Customer")).

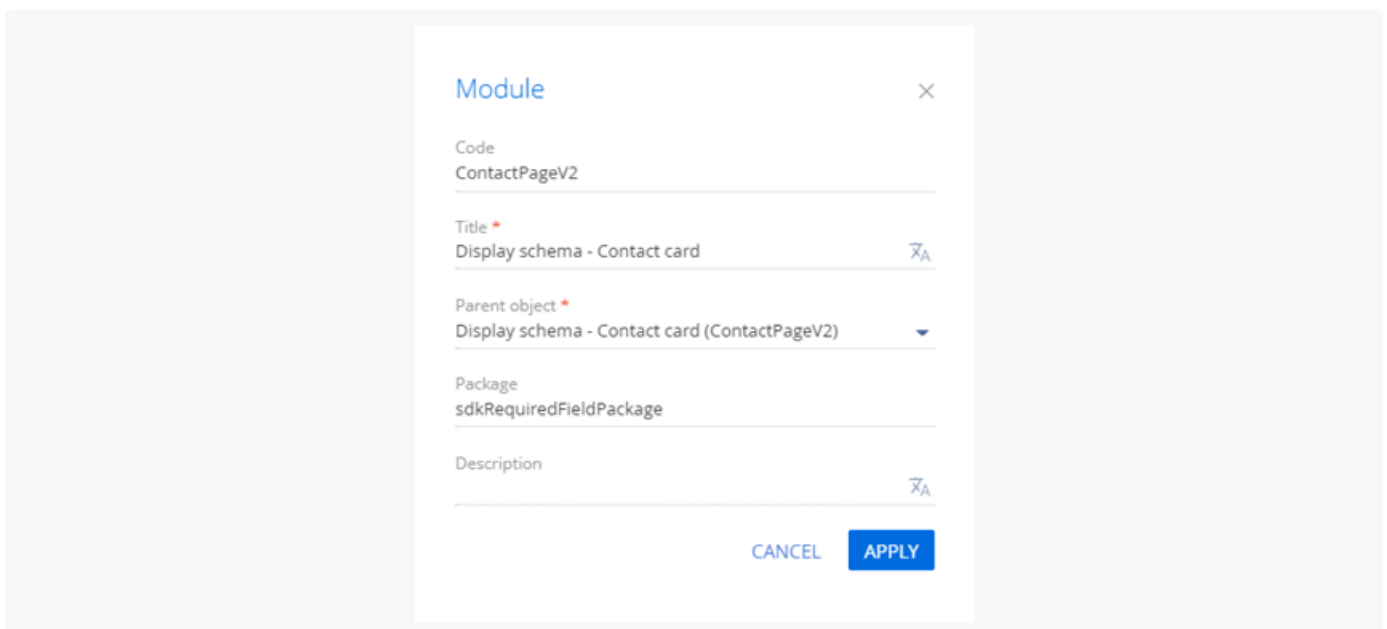
## Создать схему замещающей модели представления страницы контакта

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



### 3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "ContactPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Схема отображения карточки контакта" ("Display schema - Contact card").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "ContactPageV2".



## 4. В объявлении класса модели представления в качестве зависимостей добавьте модули

`BusinessRuleModule` и `ConfigurationConstants`.

5. Реализуйте **обязательность поля**.

Для этого задайте свойство `rules` для колонки [ *Phone* ]:

- В свойстве `ruleType` укажите значение `BINDPARAMETER`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType`.
- В свойстве `property` укажите значение `REQUIRED`, которое устанавливает обязательность заполнения колонки. Свойства бизнес-правила `BINDPARAMETER` представлены перечислением `BusinessRuleModule.enums.Property`.
- В массиве `conditions` укажите условия выполнения бизнес-правила. Значение колонки [ *Type* ] должно быть равно конфигурационной константе `ConfigurationConstants.ContactType.Client`, которая содержит идентификатор записи "Клиент" ("Customer") справочника [ *Типы контактов* ] ([ *Contact types* ]).

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

**ContactPageV2**

```
/* В качестве зависимостей укажите модули BusinessRuleModule и ConfigurationConstants. */
define("ContactPageV2", ["BusinessRuleModule", "ConfigurationConstants"],
    function(BusinessRuleModule, ConfigurationConstants) {
        return {
            /* Название схемы объекта страницы записи. */
            entitySchemaName: "Contact",
            /* Бизнес-правила модели представления страницы записи. */
            rules: {
                /* Набор правил для колонки [Phone] модели представления. */
                "Phone": {
                    /* Зависимость обязательности поля [Phone] от значения в поле [Type]. */
                    "BindParameterRequiredAccountByType": {
                        /* Тип правила BINDPARAMETER. */
                        "ruleType": BusinessRuleModule.enums.RuleType.BINDPARAMETER,
                        /* Правило регулирует свойство REQUIRED. */
                        "property": BusinessRuleModule.enums.Property.REQUIRED,
                        /* Массив условий для срабатывания правила. Определяет равно ли значе
                        "conditions": [{
                            /* Выражение левой части условия. */
                            "leftExpression": {
                                /* Тип выражения – атрибут (колонка) модели представления. */
                                "type": BusinessRuleModule.enums.ValueType.ATTRIBUTE,
                                /* Название колонки модели представления, значение которой ср
                                "attribute": "Type"
                            },
                            /* Тип операции сравнения – равно. */
                            "comparisonType": Terrasoft.ComparisonType.EQUAL,
```

```

/* Выражение правой части условия. */
"rightExpression": {
  /* Тип выражения – константное значение. */
  "type": BusinessRuleModule.enums.ValueType.CONSTANT,
  /* Значение, с которым сравнивается выражение левой части. */
  "value": ConfigurationConstants.ContactType.Client
}
}
}
}
}
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Обновите страницу раздела [ Контакты ] ([ Contacts ]).
2. При необходимости, в поле [ Тип контакта ] ([ Type ]) страницы контакта выберите "Клиент" ("Customer").

В результате выполнения примера поле [ Рабочий телефон ] ([ Business phone ]) является обязательным для контакта типа "Клиент" ("Customer").

The screenshot displays a contact management interface. On the left, a contact profile for Andrew Wayne is shown with fields for Full name, Full job title, Mobile phone, Business phone (highlighted with a red box), Email, and Country. The Business phone field contains the value +44 141 429 1595. On the right, a 'NEXT STEPS (4)' section lists tasks such as 'Work on the case', 'Work on the incident', 'Prepare specifications', and 'Visit: Andrew Wayne, Apex Solutions'. Below this, a 'CONTACT INFO' tab is active, showing details like Type (Customer), Title (Mr.), Recipient's name (Dr. Wayne), Age (49), Owner (Marina Kysla), Gender (Male), and Preferred language (English (United States)).

При попытке сохранить контакт типа "Клиент" ("Customer"), у которого не заполнено поле [ Рабочий телефон ] ([ Business phone ]), отображается информационное сообщение.



Field "Business phone": Enter a value

OK

Поле [ Рабочий телефон ] ([ Business phone ]) является необязательным для другого типа контакта (например, "Сотрудник" ("Employee")).

100% 10:18 AM, Glasgow

Full name\*  
Andrew Wayne

Full job title  
CEO

Mobile phone  
+44 141 258 9878

Business phone  
+44 141 429 1595

Email  
a.wayne@apex.co.uk

Country  
United Kingdom

NEXT STEPS (4)

- Work on the case  
10/4/2021 | Megan Lewis
- Work on the incident  
10/6/2021 | Marina Kysla
- Prepare specifications  
10/13/2021 | Marina Kysla
- Visit: Andrew Wayne, Apex Solutions  
10/26/2021 | Symon Clarke

CONTACT INFO CONNECTED TO MAINTENANCE TIMELINE ENGAGEMENT WEBSITE EVENTS

Type Employee

Title Mr.

Recipient's name Dr. Wayne

Age 49

Owner Marina Kysla

Gender Male

Preferred language English (United States)

## Настроить фильтрацию значений справочного поля на странице записи

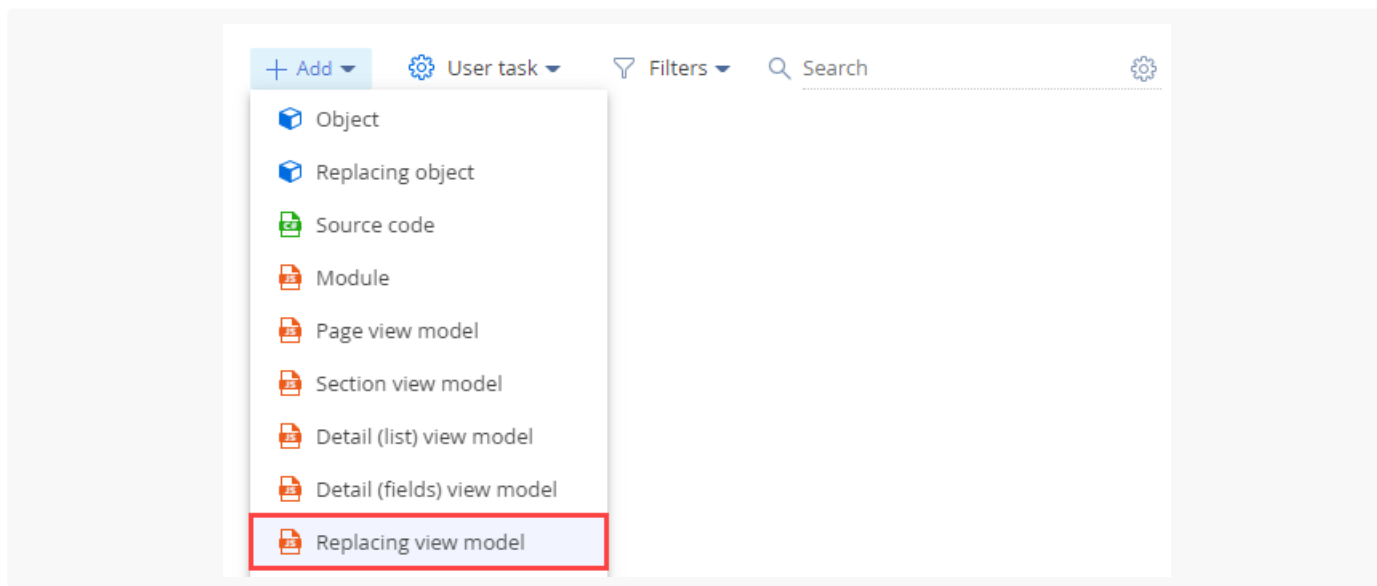
Средний

**Пример.** Настроить фильтрацию контактов, которые доступны для выбора при заполнении справочного поля [ Ответственный ] ([ Owner ]) страницы контрагента. В окне выбора контактов отображать:

- Контакты, которые имеют связанных пользователей системы.
- Активные контакты.

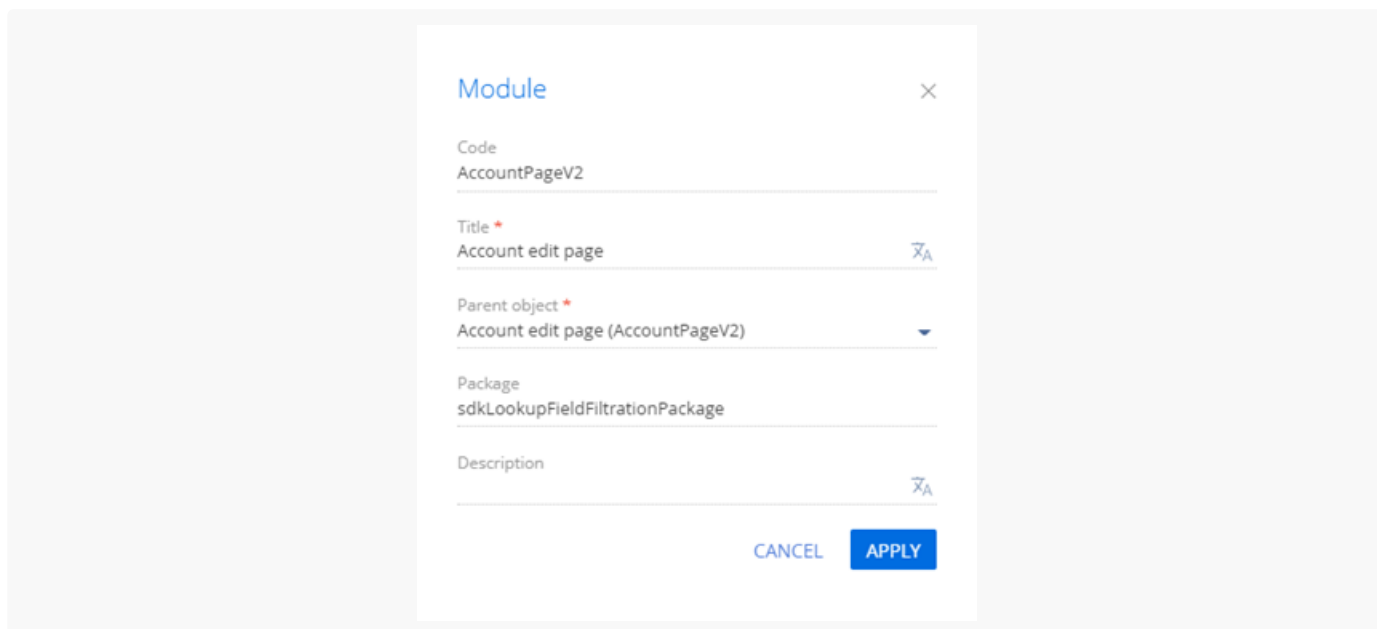
## Создать схему замещающей модели представления страницы контрагента

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



3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "AccountPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Страница редактирования контрагента" ("Account edit page").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "AccountPageV2".



4. Реализуйте **фильтрацию значений справочного поля**.

Для этого задайте свойство `attributes` для колонки [ *Owner* ]:

- В свойстве `dataValueType` укажите значение `LOOKUP`, которое устанавливает тип данных колонки. Типы данных колонки представлены перечислением `Terrasoft.core.enums.DataValueType`.
- В свойстве `lookupListConfig` укажите конфигурационный объект поля-справочника.
- В массиве `filters` укажите функцию, которая возвращает коллекцию фильтров.

Исходный код схемы замещающей модели представления страницы контрагента представлен ниже.

#### AccountPageV2

```
define("AccountPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        "entitySchemaName": "Account",
        /* Атрибуты модели представления. */
        "attributes": {
            /* Колонка модели представления. */
            "Owner": {
                /* Тип данных колонки модели представления. */
                "dataValueType": Terrasoft.DataValueType.LOOKUP,
                /* Конфигурационный объект атрибута типа LOOKUP. */
                "lookupListConfig": {
                    /* Массив фильтров, которые применяются к запросу для формирования данных */
                    "filters": [
                        function() {
                            var filterGroup = Ext.create("Terrasoft.FilterGroup");
                            /* Добавляет фильтр "IsUser" в результирующую коллекцию фильтров.
                               Выбирает все записи из корневой схемы Contact, к которой присоеди
                               filterGroup.add("IsUser", Terrasoft.createColumnIsNotNullFilter("
                               /* Добавляет фильтр "IsActive" в результирующую коллекцию фильтро
                               Выбирает все записи из корневой схемы [Contact], к которой присое
                               filterGroup.add("IsActive",
                                    Terrasoft.createColumnFilterWithParameter(
                                        Terrasoft.ComparisonType.EQUAL,
                                        "[SysAdminUnit:Contact].Active",
                                        true));
                            return filterGroup;
                        }
                    ]
                }
            }
        }
    };
});
```

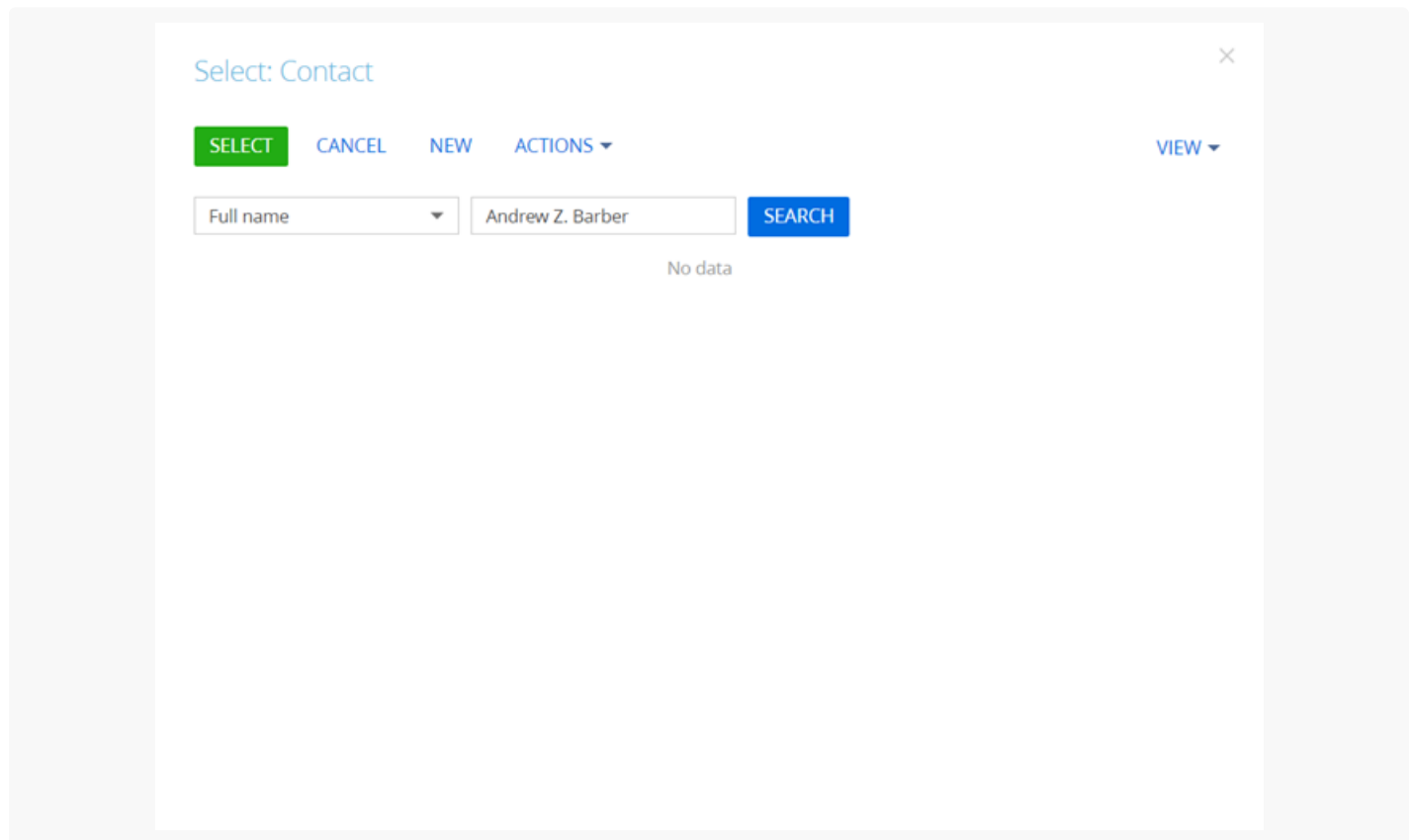
5. На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ *Контрагенты* ] ([ *Accounts* ]).

В результате выполнения примера при заполнении справочного поля [ *Ответственный* ] ([ *Owner* ]) настроена фильтрация контактов на странице контрагента.

Например, контакт Andrew Z. Barber не доступен для выбора в справочном поле [ *Ответственный* ] ([ *Owner* ]) на странице контрагента, поскольку является неактивным.



Контакт Sheldon Mallen не доступен для выбора в справочном поле [ *Ответственный* ] ([ *Owner* ]) на странице контрагента, поскольку не имеет связанного пользователя системы.

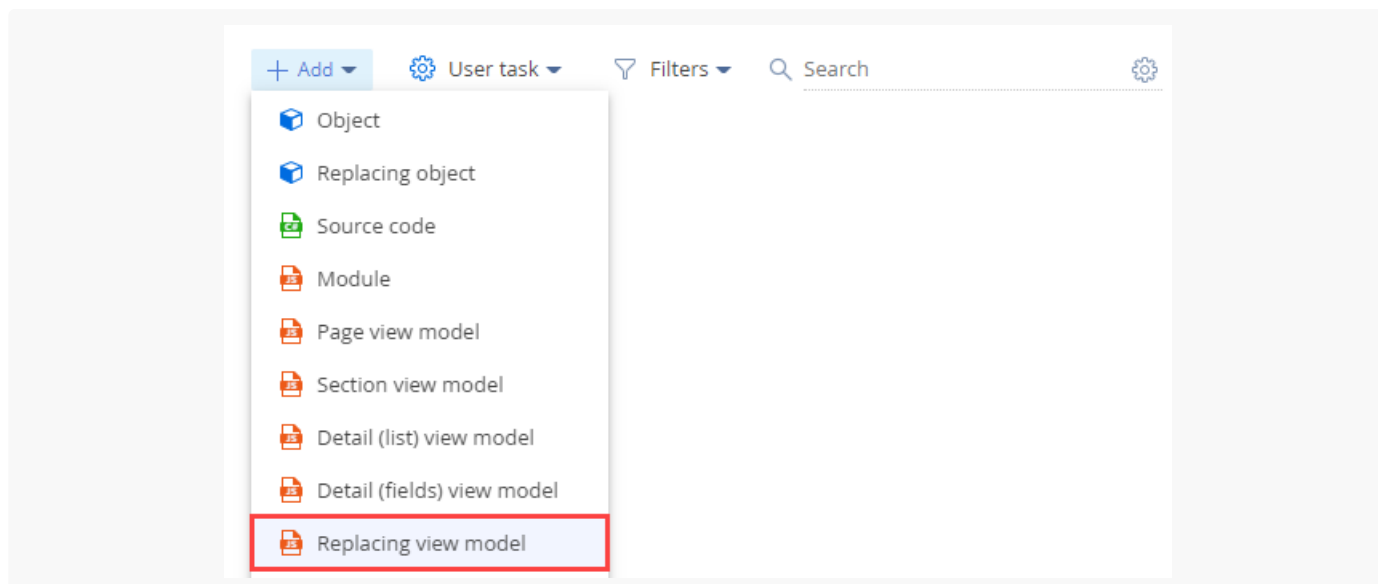
## Настроить фильтрацию значений связанных справочных полей на странице записи

Средний

**Пример.** Настроить фильтрацию полей [ Страна ] ([ Country ]), [ Область/штат ] ([ State/province ]), [ Город ] ([ City ]) страницы контакта. Перечень доступных для выбора областей/штатов зависит от страны, выбранной в поле [ Страна ] ([ Country ]). Перечень доступных для выбора городов зависит от области/штата, выбранного в поле [ Область/штат ] ([ State/province ]).

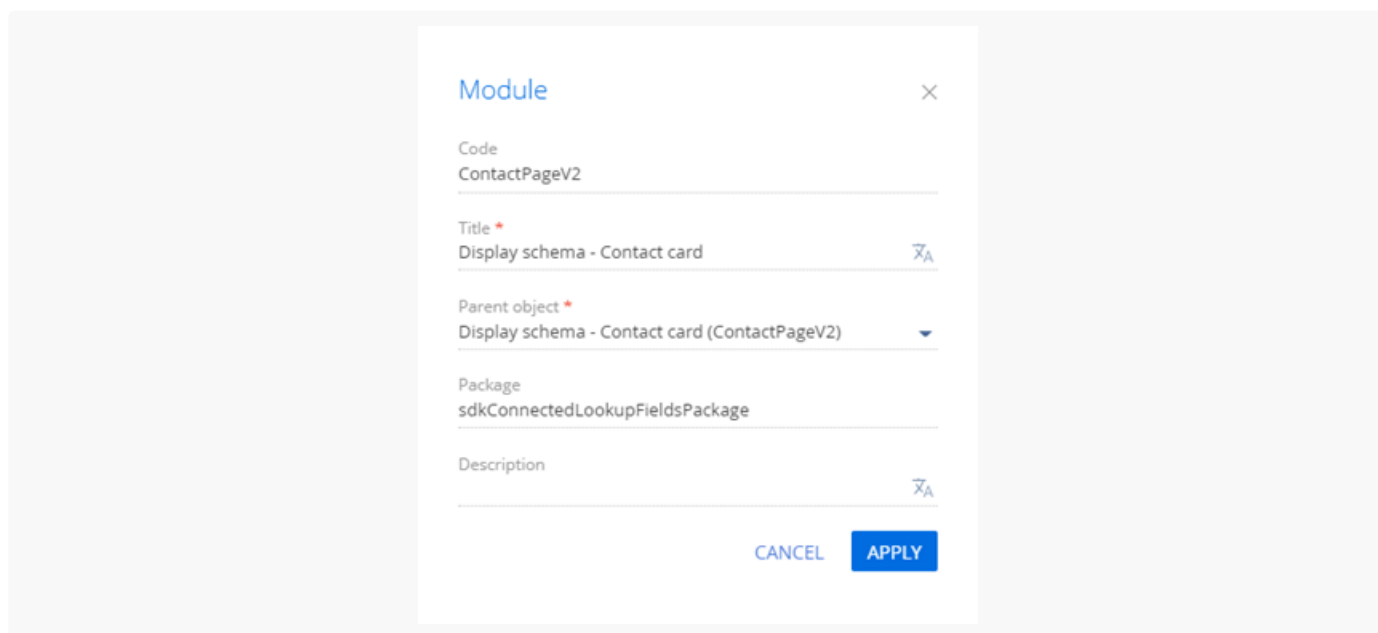
## Создать схему замещающей модели представления страницы контакта

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ Add ] —> [ Replacing view model ]).



### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "ContactPageV2".
- [ Заголовок ] ([ Title ]) — "Схема отображения карточки контакта" ("Display schema - Contact card").
- [ Родительский объект ] ([ Parent object ]) — выберите "ContactPageV2".



4. В объявлении класса модели представления в качестве зависимостей добавьте модуль `BusinessRuleModule`.

### 5. Реализуйте **фильтрацию значений связанных справочных полей**.

a. В свойство `rules` для колонок [ City ] и [ Region ]:

- В свойстве `ruleType` укажите значение `FILTRATION`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType`.
- В свойстве `autocomplete` укажите значение `true`, которое выполняет обратную фильтрацию,

т. е. автозаполнение полей [ Страна ] ([ Country ]) и [ Область/штат ] ([ State/province ]) в зависимости от выбранного города.

- d. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения полей [ Страна ] ([ Country ]), [ Область/штат ] ([ State/province ]), [ Город ] ([ City ]).

В базовой схеме страницы контакта определено правило фильтрации городов в зависимости от указанной для контакта страны. Чтобы получить возможность выбрать город из страны, которая отличается от указанной для контакта, то необходимо добавить поле [ Страна ] ([ Country ]).

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

#### ContactPageV2

```
/* В качестве зависимостей укажите модуль BusinessRuleModule. */
define("ContactPageV2", ["BusinessRuleModule"],
    function(BusinessRuleModule) {
        return {
            /* Название схемы объекта страницы записи. */
            entitySchemaName: "Contact",
            /* Бизнес-правила модели представления страницы записи. */
            rules: {
                /* Набор правил для колонки [City] модели представления. */
                "City": {
                    /* Правило фильтрации колонки [City] по значению колонки [Region]. */
                    "FiltrationCityByRegion": {
                        /* Тип правила FILTRATION. */
                        "ruleType": BusinessRuleModule.enums.RuleType.FILTRATION,
                        /* Выполняется обратная фильтрация. */
                        "autocomplete": true,
                        /* Выполняется очистка значения при изменении значения колонки [Region] */
                        "autoClean": true,
                        /* Путь к колонке для фильтрации в справочной схеме [City], на которую */
                        "baseAttributePatch": "Region",
                        /* Тип операции сравнения в фильтре. */
                        "comparisonType": Terrasoft.ComparisonType.EQUAL,
                        /* Тип выражения – атрибут (колонка) модели представления. */
                        "type": BusinessRuleModule.enums.ValueType.ATTRIBUTE,
                        /* Название колонки модели представления, значение которой сравнивает */
                        "attribute": "Region"
                    }
                },
                /* Набор правил для колонки [Region] модели представления. */
                "Region": {
                    "FiltrationRegionByCountry": {
                        "ruleType": BusinessRuleModule.enums.RuleType.FILTRATION,
                        "autocomplete": true,
                        "autoClean": true,
                        "baseAttributePatch": "Country",
```

```

        "comparisonType": Terrasoft.ComparisonType.EQUAL,
        "type": BusinessRuleModule.enums.ValueType.ATTRIBUTE,
        "attribute": "Country"
    }
}
},
/* Отображение полей на странице записи. */
diff: [
    /* Метаданные для добавления на страницу записи поля [Country]. */
    {
        /* Выполняется операция добавления элемента на страницу. */
        "operation": "insert",
        /* Мета-имя родительского контейнера, в который добавляется поле. */
        "parentName": "ProfileContainer",
        /* Поле добавляется в коллекцию элементов родительского элемента. */
        "propertyName": "items",
        /* Мета-имя добавляемого поля. */
        "name": "Country",
        /* Свойства, передаваемые в конструктор элемента. */
        "values": {
            /* Тип поля – справочник. */
            "contentType": Terrasoft.ContentType.LOOKUP,
            /* Настройка расположения поля. */
            "layout": {
                /* Номер столбца. */
                "column": 0,
                /* Номер строки. */
                "row": 6,
                /* Диапазон занимаемых столбцов. */
                "colSpan": 24
            }
        }
    },
    /* Метаданные для добавления на страницу записи поля [Region]. */
    {
        "operation": "insert",
        "parentName": "ProfileContainer",
        "propertyName": "items",
        "name": "Region",
        "values": {
            "contentType": Terrasoft.ContentType.LOOKUP,
            "layout": {
                "column": 0,
                "row": 7,
                "colSpan": 24
            }
        }
    },
    /* Метаданные для добавления на страницу записи поля [City]. */

```



```

        {
            "operation": "insert",
            "parentName": "ProfileContainer",
            "propertyName": "items",
            "name": "City",
            "values": {
                "contentType": Terrasoft.ContentType.LOOKUP,
                "layout": {
                    "column": 0,
                    "row": 8,
                    "colSpan": 24
                }
            }
        }
    ]
};
});


```

6. На панели инструментов дизайнера нажмите [ *Сохранить* ] ([ *Save* ]).


## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ *Контакты* ] ([ *Contacts* ]).

В результате выполнения примера в профиль контакта на страницу контакта добавлены связанные справочные поля [ *Страна* ] ([ *Country* ]), [ *Область/штат* ] ([ *State/province* ]), [ *Город* ] ([ *City* ]).



100%

 2:51 AM,  
New York

Full name\*

Alexander Wilson

Full job title

CEO

Mobile phone

+1 212 854 7512

Business phone\*

+44 123 456 7890

Email

a.wilson@alphabusiness.com

Country

United States

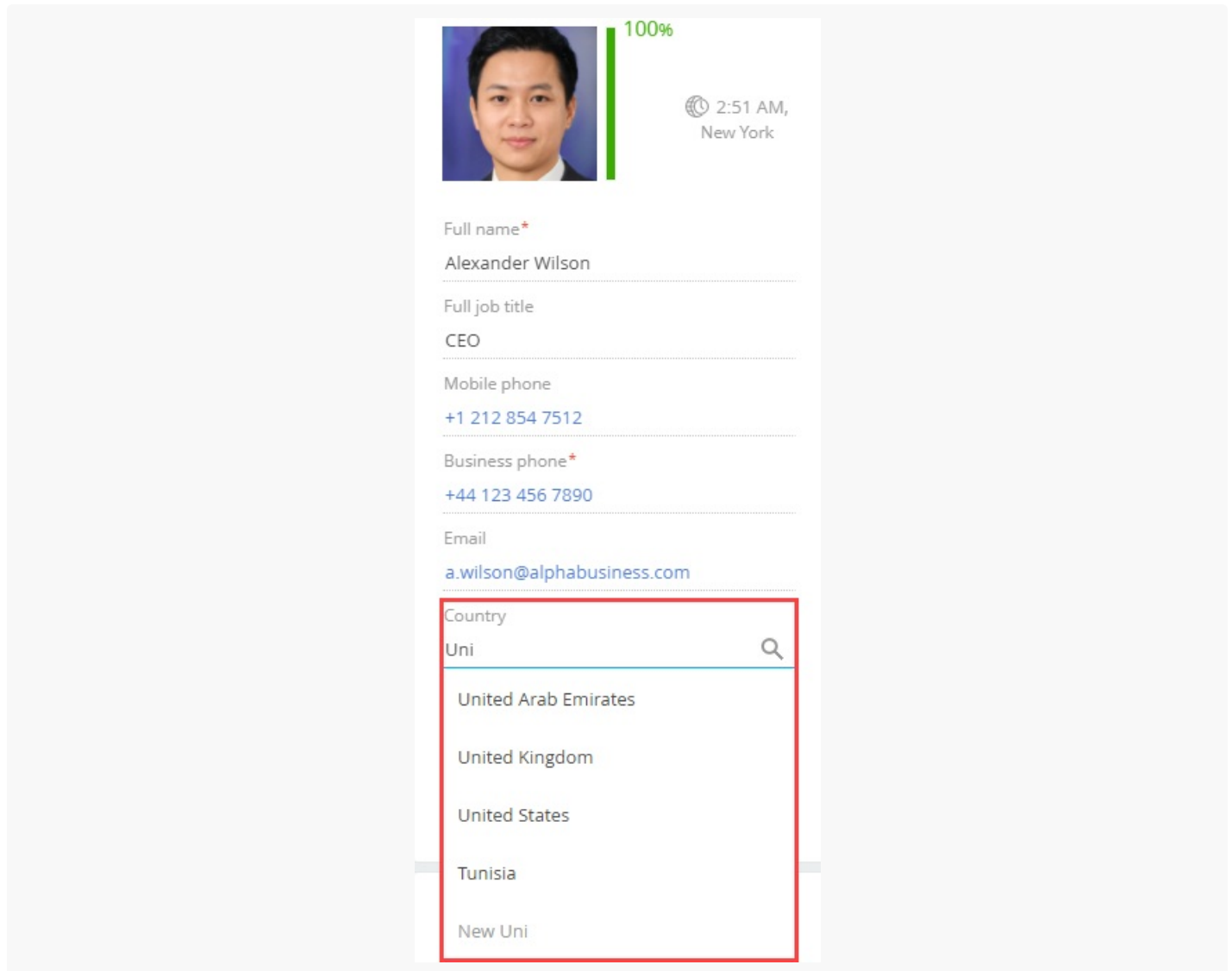
State/province

New York

City

New York

В профиле контакта можно изменить страну контакта.



The screenshot shows a user profile form for Alexander Wilson. At the top, there is a profile picture, a green progress bar at 100%, and a clock icon indicating the time as 2:51 AM in New York. Below the picture, the form fields are as follows:

- Full name\***: Alexander Wilson
- Full job title**: CEO
- Mobile phone**: +1 212 854 7512
- Business phone\***: +44 123 456 7890
- Email**: a.wilson@alphabusiness.com
- Country**: A dropdown menu is open, showing a search bar with the text "Uni" and a magnifying glass icon. The dropdown list contains the following options: United Arab Emirates, United Kingdom, United States, Tunisia, and New Uni.

Также фильтрация выполняется в окне выбора страны.

Select: Countries

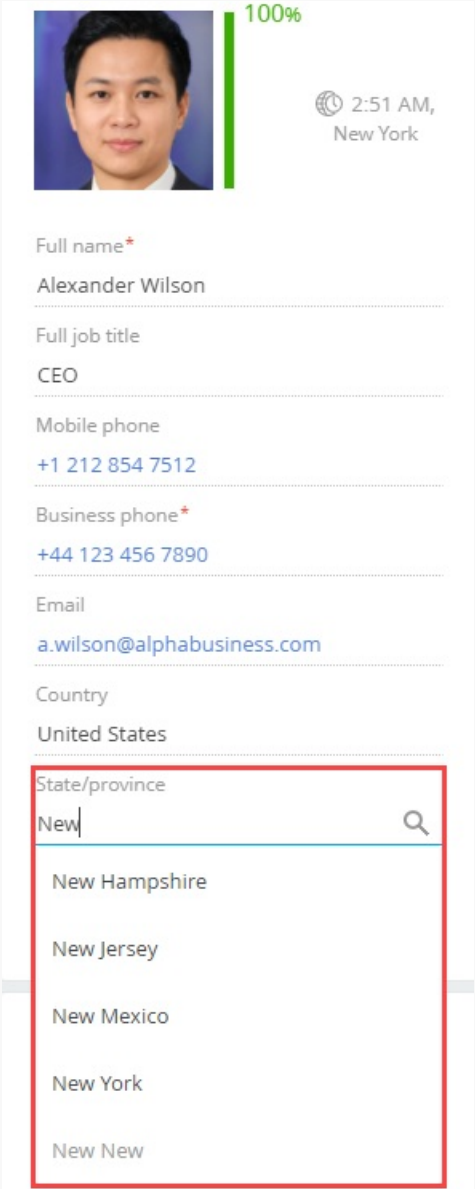
SELECT CANCEL NEW ACTIONS VIEW

Name Uni SEARCH

Name ^

- Tunisia
- United Arab Emirates
- United Kingdom
- United States

Перечень доступных для выбора областей/штатов зависит от страны, выбранной в поле [ Страна ] ([ Country ]). Фильтрация выполняется как в поле ввода значения, так и в окне выбора области/штата.



100%

2:51 AM,  
New York

Full name\*

Alexander Wilson

Full job title

CEO

Mobile phone

+1 212 854 7512

Business phone\*

+44 123 456 7890

Email

a.wilson@alphabusiness.com

Country

United States

State/province

New

New Hampshire

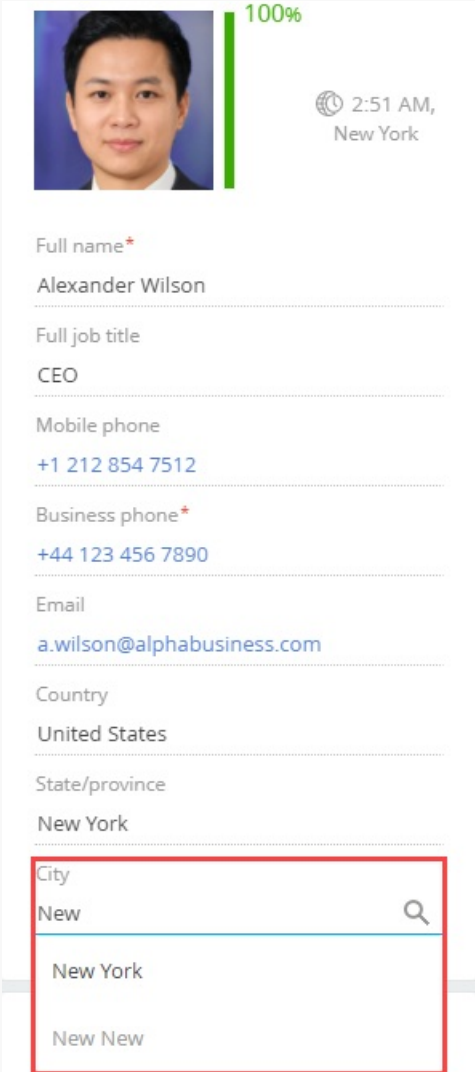
New Jersey

New Mexico

New York

New New

Перечень доступных для выбора городов зависит от области/штата, выбранного в поле [ *Область/штат* ] ([ *State/province* ]). Фильтрация выполняется как в поле ввода значения, так и в окне выбора города.



100%

2:51 AM,  
New York

Full name\*

Alexander Wilson

Full job title

CEO

Mobile phone

+1 212 854 7512

Business phone\*

+44 123 456 7890

Email

a.wilson@alphabusiness.com

Country

United States

State/province

New York

City

New

New York

New New

## Настроить условия блокировки поля на странице записи

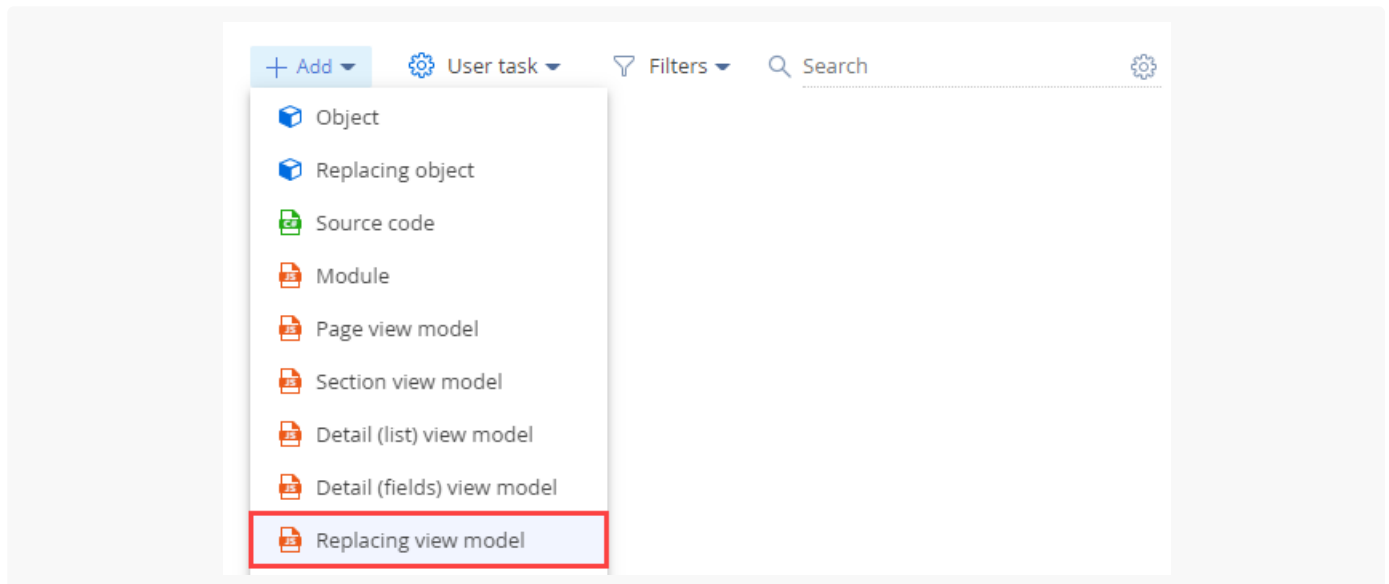
 Средний

**Пример.** Настроить блокировку поля [ *Рабочий телефон* ] ([ *Business phone* ]) страницы контакта. Поле заблокировано при отсутствии значения в поле [ *Мобильный телефон* ] ([ *Mobile phone* ]).

## Создать схему замещающей модели представления страницы контакта

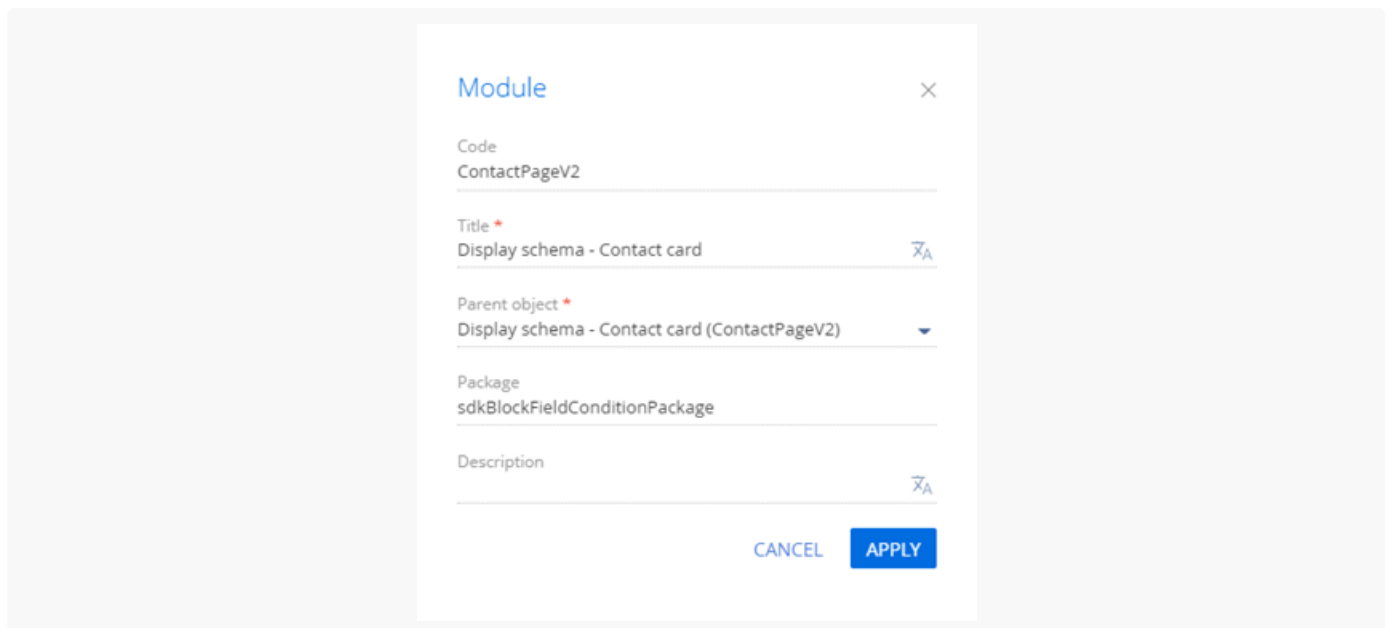
1. [Перейдите в раздел \[ Конфигурация \]](#) ([ *Configuration* ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ *Добавить* ] —> [ *Замещающая модель* ]

представления ] ([ *Add* ] —> [ *Replacing view model* ]).



### 3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "ContactPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Схема отображения карточки контакта" ("Display schema - Contact card").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "ContactPageV2".



### 4. В объявлении класса модели представления в качестве зависимостей добавьте модуль `BusinessRuleModule`.

### 5. Реализуйте **условия блокировки поля**.

a. В свойство `rules` для колонки [ *Phone* ]:

- В свойстве `ruleType` укажите значение `BINDPARAMETER`, которое задает тип бизнес-правила. Типы

правил представлены перечислением `BusinessRuleModule.enums.RuleType`.

- В свойстве `property` укажите значение `ENABLED`, которое устанавливает доступность колонки. Свойства бизнес-правила `BINDPARAMETER` представлены перечислением `BusinessRuleModule.enums.Property`.
- В массиве `conditions` укажите условия выполнения бизнес-правила. Значение колонки `[ MobilePhone ]` не должно быть пустым.

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

#### ContactPageV2

```
/* В качестве зависимостей укажите модуль BusinessRuleModule. */
define("ContactPageV2", ["BusinessRuleModule"], function(BusinessRuleModule) {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Бизнес-правила модели представления страницы записи. */
        rules: {
            /* Набор правил для колонки [Phone] модели представления. */
            "Phone": {
                /* Зависимость обязательности поля [Phone] от значения в поле [MobilePhone].
                "BindParameterEnabledPhoneByMobile": {
                    /* Тип правила BINDPARAMETER. */
                    "ruleType": BusinessRuleModule.enums.RuleType.BINDPARAMETER,
                    /* Правило регулирует свойство ENABLED. */
                    "property": BusinessRuleModule.enums.Property.ENABLED,
                    /* Массив условий для срабатывания правила. Определяет установлено ли зна
                    "conditions": [{
                        /* Выражение левой части условия. */
                        "leftExpression": {
                            /* Тип выражения – атрибут (колонка) модели представления. */
                            "type": BusinessRuleModule.enums.ValueType.ATTRIBUTE,
                            /* Название колонки модели представления, значение которой сравни
                            "attribute": "MobilePhone"
                        },
                        /* Тип операции сравнения – не равно. */
                        "comparisonType": Terrasoft.ComparisonType.NOT_EQUAL,
                        /* Выражение правой части условия. */
                        "rightExpression": {
                            /* Тип выражения – константное значение. */
                            "type": BusinessRuleModule.enums.ValueType.CONSTANT,
                            /* Значение, с которым сравнивается выражение левой части. */
                            "value": ""
                        }
                    }
                }
            }
        }
    }
}
```



```

    }
  };
});

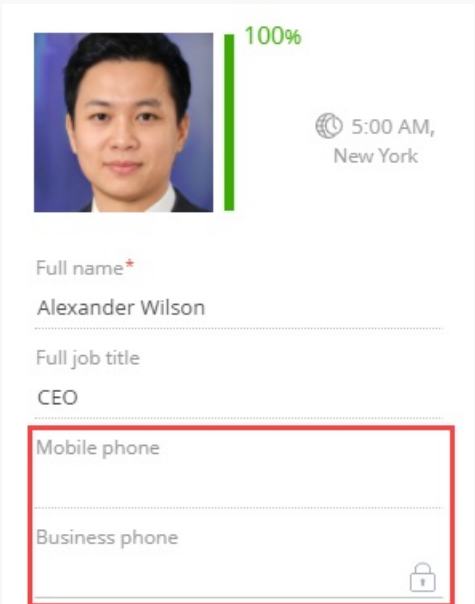
```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ Контакты ] ([ Contacts ]).

В результате выполнения примера на странице контакта поле [ Рабочий телефон ] ([ Business phone ]) заблокировано при отсутствии значения в поле [ Мобильный телефон ] ([ Mobile phone ]).



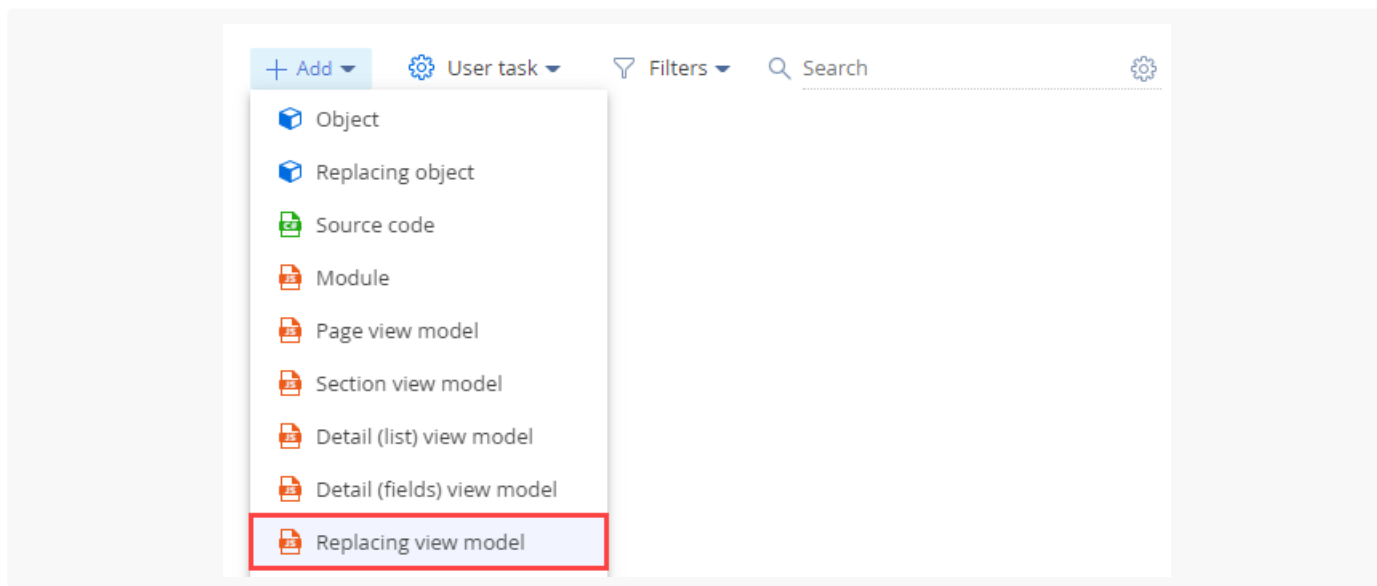
## Настроить исключения блокировки полей на странице записи

 Средний

**Пример.** Настроить блокировку полей на странице счета. Поля заблокированы для полностью оплаченного счета (т. е. в поле [ Состояние оплаты ] ([ Payment status ]) выбрано значение "Оплачен полностью" ("Paid")). Не блокируются поля [ Состояние оплаты ] ([ Payment status ]) и деталь [ Активности ] ([ Activities ]).

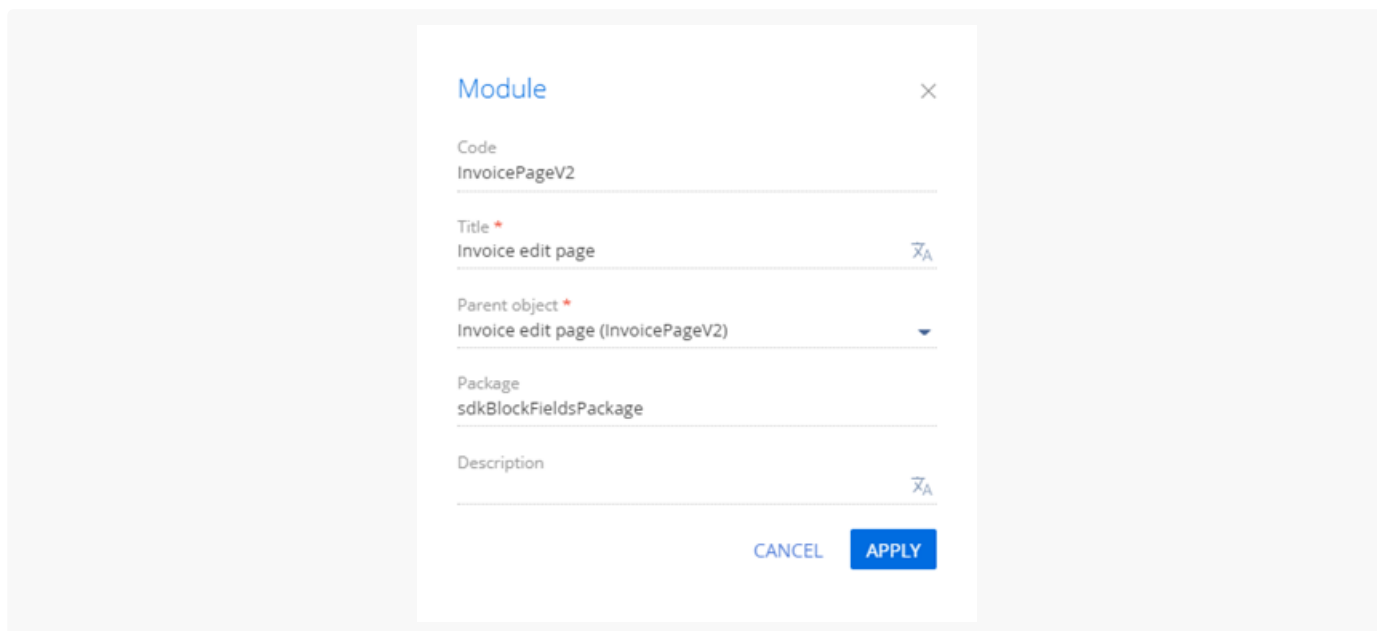
## Создать схему замещающей модели представления страницы счета

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



3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "InvoicePageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Страница редактирования счета" ("Invoice edit page").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "InvoicePageV2".



4. В объявлении класса модели представления в качестве зависимостей добавьте модуль `InvoiceConfigurationConstants`.
5. Реализуйте **исключения и условия блокировки полей**.

- a. В свойство `attributes` добавьте атрибут `IsModelItemsEnabled`, который включает механизм блокировки полей.
- b. В свойстве `methods` реализуйте **методы**:
  - `getDisableExclusionsColumnTags()` — исключает блокировку колонки.
  - `getDisableExclusionsDetailSchemaNames()` — исключает блокировку детали.
  - `setCardLockoutStatus()` — настраивает условия блокировки полей.
  - `onEntityInitialized()` — переопределяет базовый виртуальный метод. Срабатывает после выполнения инициализации схемы объекта страницы записи.
- g. В массив модификаций `diff` добавьте конфигурационный объект с настройками контейнера `CardContentWrapper`, в котором планируется заблокировать поля.

Исходный код схемы замещающей модели представления страницы счета представлен ниже.

#### InvoicePageV2

```
/* В качестве зависимостей укажите модуль InvoiceConfigurationConstants. */
define("InvoicePageV2", ["InvoiceConfigurationConstants"], function(InvoiceConfigurationConst
  return {
    /* Название схемы объекта страницы записи. */
    entitySchemaName: "Invoice",
    /* Атрибуты модели представления. */
    attributes: {
      "IsModelItemsEnabled": {
        /* Тип данных колонки модели представления. */
        dataValueType: Terrasoft.DataValueType.BOOLEAN,
        value: true,
        /* Массив конфигурационных объектов, которые определяют зависимости атрибута
        dependencies: [{
          /* Значение колонки [IsModelItemsEnabled] зависит от значения колонки [Pa
          columns: ["PaymentStatus"],
          /* Метод-обработчик. */
          methodName: "setCardLockoutStatus"
        }]
      }
    },
    /* Методы модели представления страницы записи. */
    methods: {
      /* Исключение блокировки колонки [PaymentStatus]. */
      getDisableExclusionsColumnTags: function() {
        return ["PaymentStatus"];
      },
      /* Исключение блокировки детали [ActivityDetailV2]. */
      getDisableExclusionsDetailSchemaNames: function() {
        return ["ActivityDetailV2"];
      },
    },
  },
);
```

```

/* Настройка условий блокировки полей. */
setCardLockoutStatus: function() {
    var state = this.get("PaymentStatus");
    if (state.value === InvoiceConfigurationConstants.Invoice.PaymentStatus.Paid)
        this.set("IsModelItemsEnabled", false);
    } else {
        this.set("IsModelItemsEnabled", true);
    }
},
/* Переопределение базового метода Terrasoft.BasePageV2.onEntityInitialized(). */
onEntityInitialized: function() {
    /* Вызывается родительская реализация метода. */
    this.callParent(arguments);
    this.setCardLockoutStatus();
}
},
details: /**SCHEMA_DETAILS*/{/**SCHEMA_DETAILS*/,
/* Отображение контейнера блокировки на странице записи. */
diff: /**SCHEMA_DIFF*/[
{
    /* Выполняется операция изменения существующего элемента. */
    "operation": "merge",
    /* Мета-имя родительского контейнера, в котором блокируются поля. */
    "name": "CardContentWrapper",
    /* Свойства, передаваемые в конструктор элемента. */
    "values": {
        /* Генератор представления элемента управления. */
        "generator": "DisableControlsGenerator.generatePartial"
    }
}
]/**SCHEMA_DIFF*/
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ Счета ] ([ Invoices ]).

В результате выполнения примера на странице счета, у которого в поле [ Состояние оплаты ] ([ Payment status ]) выбрано значение "Оплачен полностью" ("Paid"), заблокировано большинство полей.

Незаблокированными остаются:

- Поле [ Состояние оплаты ] ([ Payment status ]).
- Деталь [ Активности ] ([ Activities ]).
- Поля, для которых в свойстве `enabled` массива модификаций `diff` указано значение `true`.

INV-8

What can I do for you? >

Creatio 7.18.4.1532

CLOSE ACTIONS PRINT VIEW

Number INV-8 Date\* 9/29/2021

Owner\* Marina Kysla Order ORD-11

< GENERAL INFORMATION PRODUCTS APPROVALS HISTORY ATTACHMENTS AND NOTES >

Customer\* Alpha Business

Supplier Our company

Customer details Partners, USD

Supplier details For invoices (USD)

Amount

Amount, \$ 3,000.00

Payment

Payment status\* Paid

Paid on 10/29/2021

Payment amount, \$ 3,000.00

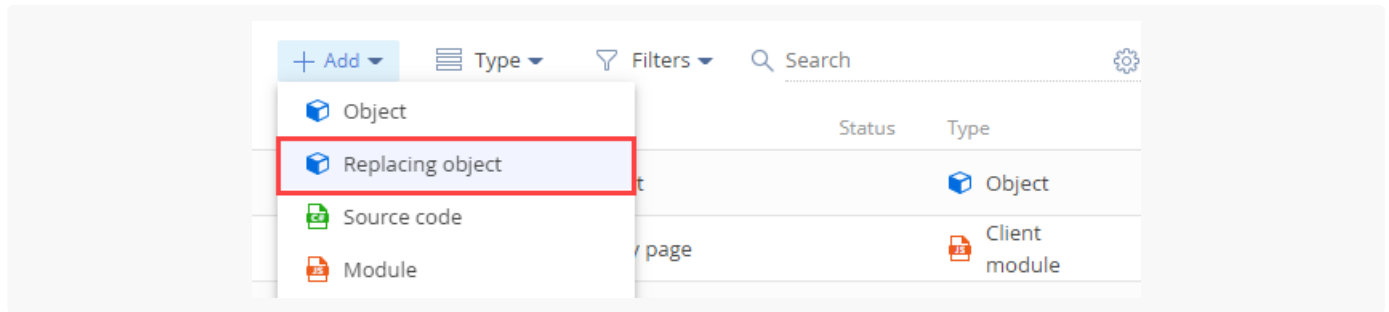
## Настроить условия отображения поля на странице записи

Средний

**Пример.** Настроить условия отображения поля [ Место встречи ] ([ Meeting place ]) на странице активности. Поле отображается для активности категории "Встреча" ("Meeting") (т. е. в поле [ Категория ] ([ Category ]) выбрано значение "Встреча" ("Meeting")).

### 1. Создать схему замещающего объекта

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

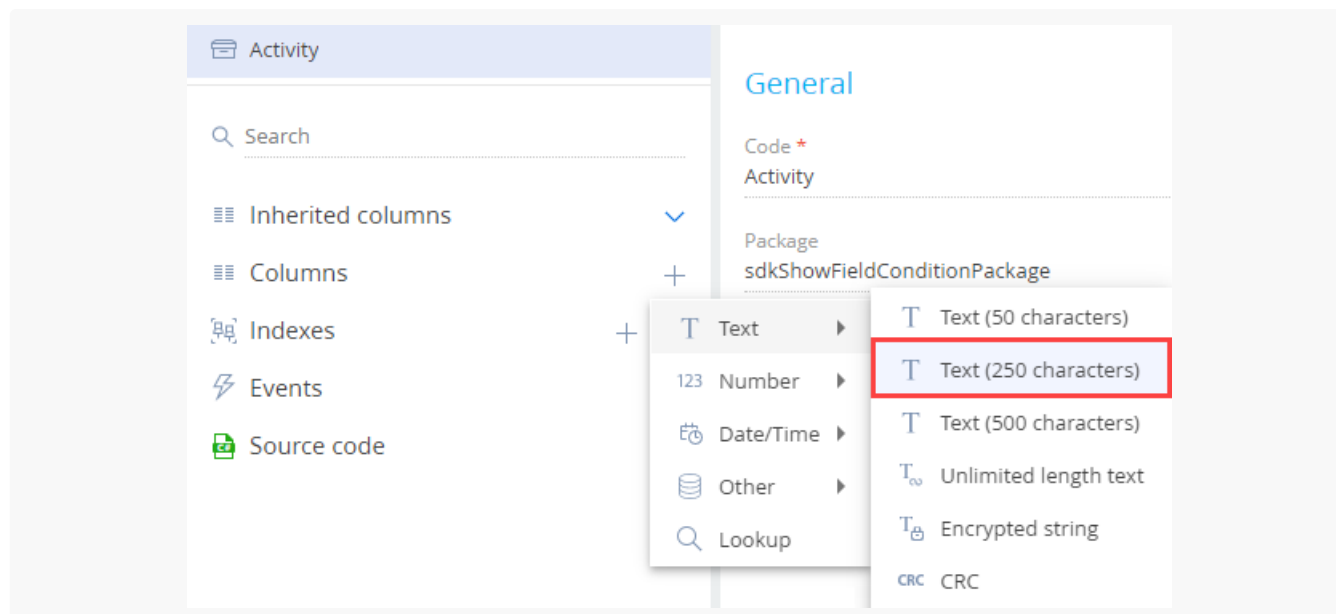


### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "Activity".
- [ Заголовок ] ([ Title ]) — "Активность" ("Activity").
- [ Родительский объект ] ([ Parent object ]) — выберите "Activity".

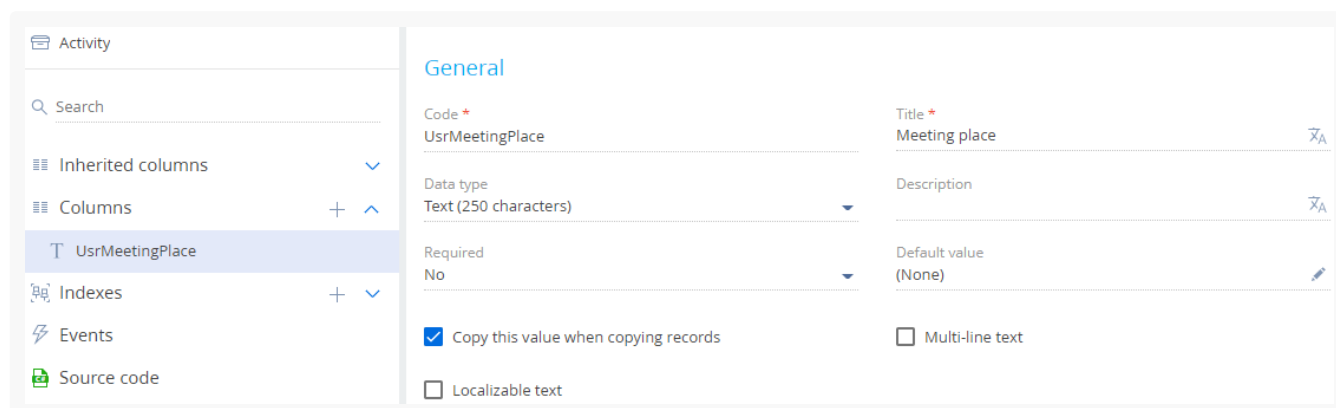
### 4. В схему добавьте **колонку**.

- В контекстном меню узла [ Колонки ] ([ Columns ]) структуры объекта нажмите **+**.
- В выпадающем меню нажмите [ Строка ] —> [ Строка (250 символов) ] ([ Text ] —> [ Text (250 characters) ]).



с. Заполните **свойства добавляемой колонки**.

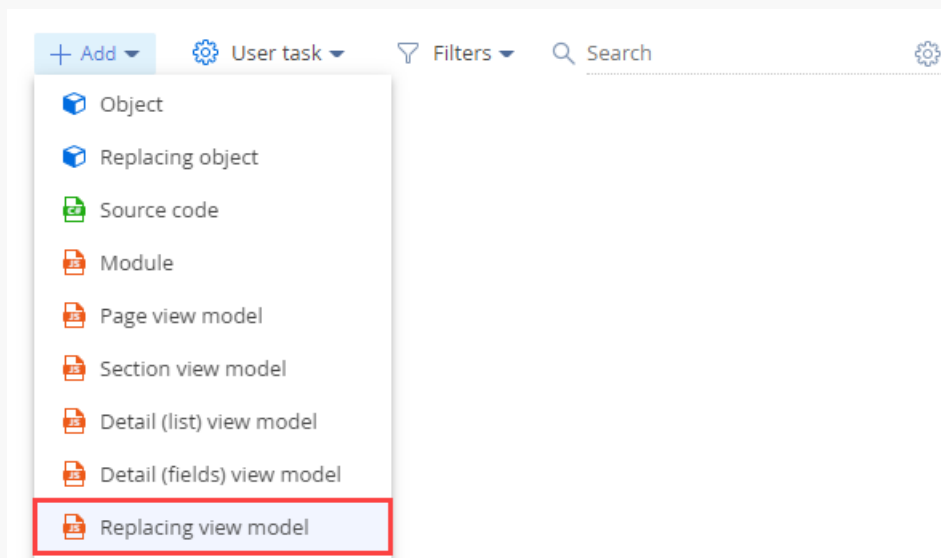
- [ Код ] ([ Code ]) — "UsrMeetingPlace".
- [ Заголовок ] ([ Title ]) — "Место встречи" ("Meeting place").



5. На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]), а затем [ Опубликовать ] ([ Publish ]).

## 2. Создать схему замещающей модели представления страницы активности


1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ Add ] —> [ Replacing view model ]).



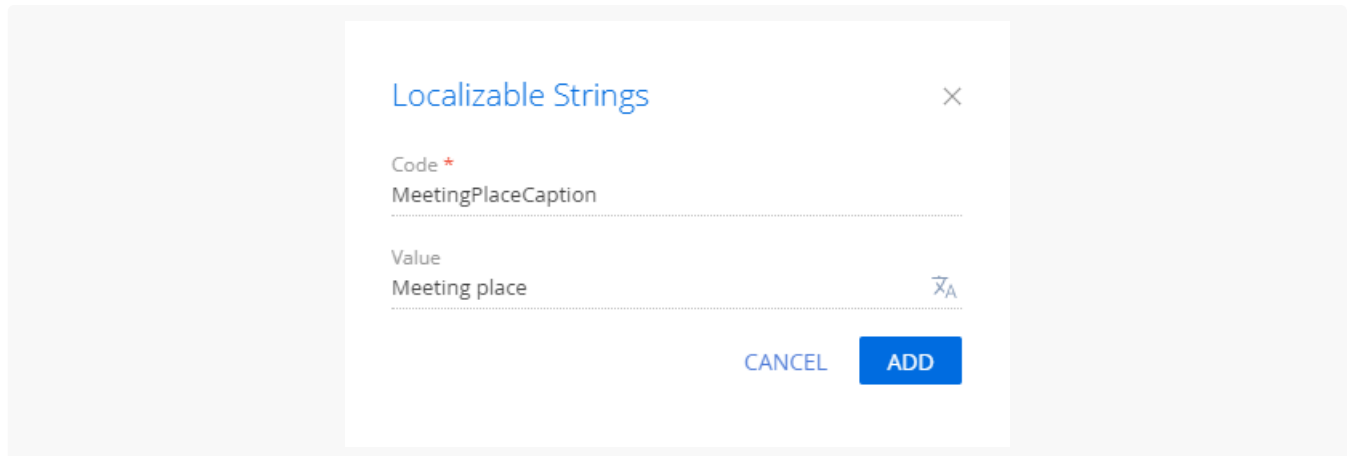
### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "ActivityPageV2".
- [ Заголовок ] ([ Title ]) — "Страница редактирования активности" ("Activity edit page").
- [ Родительский объект ] ([ Parent object ]) — выберите "ActivityPageV2".

### 4. Добавьте **локализуемую строку**.

- В контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) нажмите кнопку .
- Заполните **свойства локализуемой строки**.
  - [ Код ] ([ Code ]) — "MeetingPlaceCaption".
  - [ Значение ] ([ Value ]) — "Место встречи" ("Meeting place").





е. Для добавления локализуемой строки нажмите [ *Добавить* ] ([ *Add* ]).

5. В объявлении класса модели представления в качестве зависимостей добавьте модули `BusinessRuleModule` и `ConfigurationConstants`.

6. Реализуйте **условия отображения поля**.

а. В свойство `rules` для колонки [ *UsrMeetingPlace* ]:

а. В свойстве `ruleType` укажите значение `BINDPARAMETER`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType`.

б. В свойстве `property` укажите значение `VISIBLE`, которое устанавливает видимость колонки. Свойства бизнес-правила `BINDPARAMETER` представлены перечислением `BusinessRuleModule.enums.Property`.

с. В массиве `conditions` укажите условия выполнения бизнес-правила. Значение колонки [ *ActivityCategory* ] должно быть равно конфигурационной константе `ConfigurationConstants.Activity.ActivityCategory.Meeting`, которая содержит идентификатор записи "Встреча" ("Meeting") справочника [ *Категории активностей* ] ([ *Activity categories* ]).

б. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля [ *Место встречи* ] ([ *Meeting place* ]).

Исходный код схемы замещающей модели представления страницы активности представлен ниже.

#### ActivityPageV2

```
/* В качестве зависимостей укажите модули BusinessRuleModule и ConfigurationConstants. */
define("ActivityPageV2", ["BusinessRuleModule", "ConfigurationConstants"],
    function(BusinessRuleModule, ConfigurationConstants) {
        return {
            /* Название схемы объекта страницы записи. */
            entitySchemaName: "Activity",
            /* Отображение поля на странице записи. */
            diff: /**SCHEMA_DIFF*/[
                /* Метаданные для добавления на страницу записи поля [UsrMeetingPlace]. */
            ]
        }
    })
```

```

/* Выполняется операция добавления элемента на страницу. */
"operation": "insert",
/* Мета-имя родительского контейнера, в который добавляется поле. */
"parentName": "Header",
/* Поле добавляется в коллекцию элементов родительского элемента. */
"propertyName": "items",
/* Мета-имя добавляемого поля. */
"name": "UsrMeetingPlace",
/* Свойства, передаваемые в конструктор элемента. */
"values": {
    /* Привязка заголовка поля к локализуемой строке схемы. */
    "caption": {"bindTo": "Resources.Strings.MeetingPlaceCaption"},
    /* Настройка расположения поля. */
    "layout": {
        /* Номер столбца. */
        "column": 0,
        /* Номер строки. */
        "row": 5,
        /* Диапазон занимаемых столбцов. */
        "colSpan": 12
    }
}
}
]/**SCHEMA_DIFF*/,
/* Бизнес-правила модели представления страницы записи. */
rules: {
    /* Набор правил для колонки [UsrMeetingPlace] модели представления. */
    "UsrMeetingPlace": {
        /* Зависимость видимости поля [UsrMeetingPlace] от значения в поле [Activ
        "BindParametrVisiblePlaceByType": {
            /* Тип правила BINDPARAMETER. */
            "ruleType": BusinessRuleModule.enums.RuleType.BINDPARAMETER,
            /* Правило регулирует свойство VISIBLE. */
            "property": BusinessRuleModule.enums.Property.VISIBLE,
            /* Массив условий для срабатывания правила. Определяет равно ли значе
            "conditions": [{
                /* Выражение левой части условия. */
                "leftExpression": {
                    /* Тип выражения – атрибут (колонка) модели представления. */
                    "type": BusinessRuleModule.enums.ValueType.ATTRIBUTE,
                    /* Название колонки модели представления, значение которой ср
                    "attribute": "ActivityCategory"
                },
                /* Тип операции сравнения – равно. */
                "comparisonType": Terrasoft.ComparisonType.EQUAL,
                /* Выражение правой части условия. */
                "rightExpression": {
                    /* Тип выражения – константное значение. */
                    "type": BusinessRuleModule.enums.ValueType.CONSTANT,

```

```

/* Значение, с которым сравнивается выражение левой части. */
"value": ConfigurationConstants.Activity.ActivityCategory.Mee
    }
  }
}
}
};
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Обновите страницу раздела [ Активности ] ([ Activities ]).
2. При необходимости, в поле [ Категория ] ([ Category ]) страницы активности выберите значение "Встреча" ("Meeting").

В результате выполнения примера поле [ Место встречи ] ([ Meeting place ]) отображается для активности категории "Встреча" ("Meeting").

The screenshot shows the 'Test task' form in the Creatio system. The form includes a header with the title 'Test task', a search bar, and the Creatio logo. Below the header are buttons for 'SAVE', 'CANCEL', 'ACTIONS', and a 'VIEW' button. The main form area contains several fields: 'Subject' (Test task), 'Start' (11/23/2021, 10:00 AM), 'Due' (11/23/2021, 10:30 AM), 'Status' (Not started), 'Role', 'Owner' (Marina Kysla), 'Reporter' (Marina Kysla), 'Priority' (Medium), 'Meeting place' (highlighted with a red box), and 'Category' (Meeting, highlighted with a red box). The 'Show in calendar' checkbox is checked.

Поле [ Место встречи ] ([ Meeting place ]) не отображается для другой категории активности (например, "Выполнить" ("To do")).

Test task

What can I do for you? >

SAVE CANCEL ACTIONS

VIEW

Subject\* Test task

Start\* 11/23/2021 10:00 AM

Due\* 11/23/2021 10:30 AM

Status\* Not started

Show in calendar ☒

Role

Owner Marina Kysla

Reporter\* Marina Kysla

Priority\* Medium

Category\* To do

# Добавить автонумерацию к полю на странице добавления записи (front-end)

Сложный


**Пример.** Добавить автонумерацию к полю [ Код ] ([ Code ]) страницы добавления продукта.  
Шаблон номера: ART\_0000N, где N = 1, 2, и т. д. Автонумерацию реализовать на стороне front-end.

## 1. Создать системные настройки

1. Создайте [системную настройку](#) с маской кода продукта.

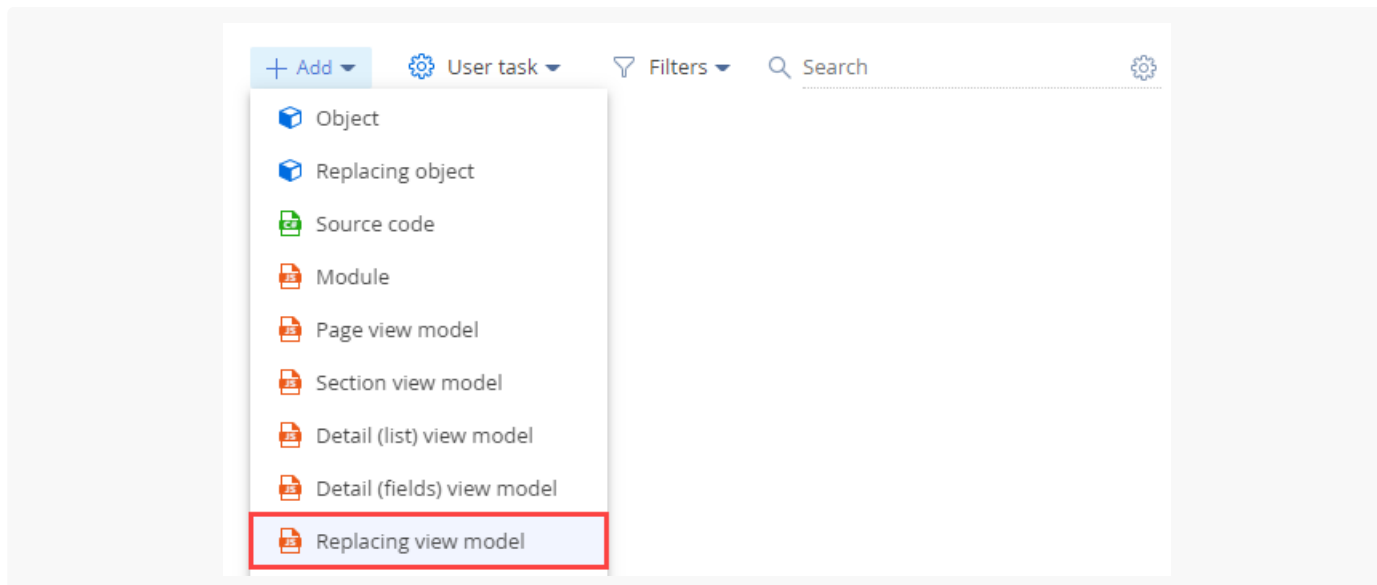
- Перейдите в дизайнер системы по кнопке .
- В блоке [ Настройка системы ] ([ System setup ]) перейдите по ссылке [ Системные настройки ] ([ System settings ]).
- На панели инструментов раздела нажмите на кнопку [ Добавить настройку ] ([ Add setting ]).
- Заполните **свойства системной настройки**.
  - [ Название ] ([ Name ]) — "Маска кода продукта" ("Product code mask").
  - [ Код ] ([ Code ]) — "ProductCodeMask".
  - [ Тип ] ([ Type ]) — выберите "Строка неограниченной длины" ("Unlimited length text").
  - [ Значение по умолчанию ] ([ Default value ]) — "ART\_{0:00000}".

2. Создайте [системную настройку](#) с текущим кодом продукта.

- a. Перейдите в дизайнер системы по кнопке .
- b. В блоке [ *Настройка системы* ] ([ *System setup* ]) перейдите по ссылке [ *Системные настройки* ] ([ *System settings* ]).
- c. На панели инструментов раздела нажмите на кнопку [ *Добавить настройку* ] ([ *Add setting* ]).
- d. Заполните **свойства системной настройки**.
  - [ *Название* ] ([ *Name* ]) — "Текущий код продукта" ("Product last number").
  - [ *Код* ] ([ *Code* ]) — "ProductLastNumber".
  - [ *Тип* ] ([ *Type* ]) — выберите "Целое число" ("Integer").

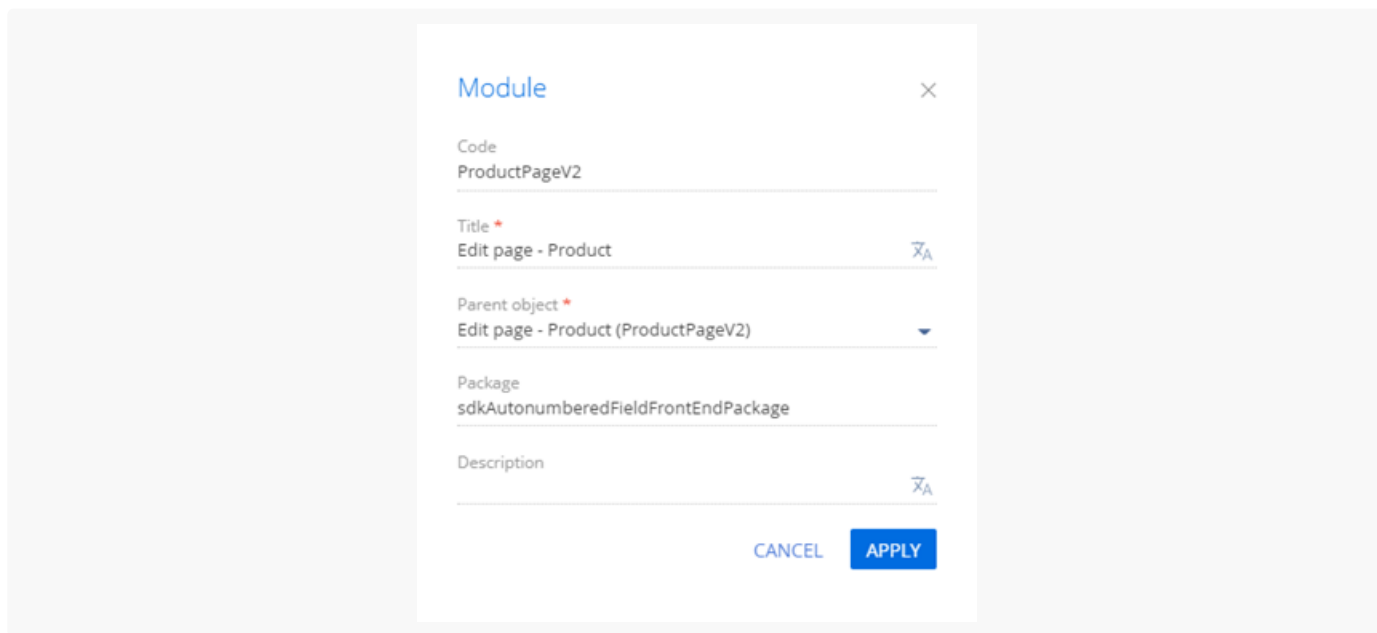
## 2. Создать схему замещающей модели представления страницы продукта

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



### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "ProductPageV2".
- [ Заголовок ] ([ Title ]) — "Страница редактирования продукта" ("Edit page - Product").
- [ Родительский объект ] ([ Parent object ]) — выберите "ProductPageV2".



### 4. Реализуйте **автонумерацию поля**.

Для этого в свойстве `methods` реализуйте метод `onEntityInitialized()` — переопределенный базовый виртуальный метод. Срабатывает после окончания инициализации схемы объекта. В метод `onEntityInitialized()` добавьте вызов метода-обработчика `getIncrementCode()`, который присвоит сгенерированный номер полю [ Код ] ([ Code ]).

Исходный код схемы замещающей модели представления страницы продукта представлен ниже.

**ProductPageV2**

```

define("ProductPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Product",
        /* Методы модели представления страницы записи. */
        methods: {
            /* Переопределение базового метода Terrasoft.BasePageV2.onEntityInitialized, кото
            onEntityInitialized: function() {
                /* Вызывается родительская реализация метода. */
                this.callParent(arguments);
                /* Код генерируется, если создается новый элемент или копия существующего. */
                if (this.isAddMode() || this.isCopyMode()) {
                    /* Вызов базового метода Terrasoft.BasePageV2.getIncrementCode, который г
                    this.getIncrementCode(function(response) {
                        /* Сгенерированный номер возвращается в колонку [Code]. */
                        this.set("Code", response);
                    });
                }
            }
        }
    };
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Очистите кэш браузера.
2. Обновите страницу раздела [ *Продукты* ] ([ *Products* ]).

В результате выполнения примера добавлена автонумерацию к полю [ *Код* ] ([ *Code* ]) страницы добавления продукта.

The screenshot shows a web interface for creating a new record. At the top, there's a header with the text 'New record', a search bar with the placeholder 'What can I do for you?', and the 'Creatio' logo with version '7.18.4.1532'. Below the header are four buttons: 'SAVE' (green), 'CANCEL' (blue), 'ACTIONS' (blue with a dropdown arrow), and 'VIEW' (blue with a dropdown arrow). The main form area has a placeholder image on the left. To its right are several fields: 'Name\*' (with a red asterisk), 'Code' (with a green checkmark icon and a red box around it, containing the text 'ART\_00001'), 'Link' (with a green checkmark icon), 'Owner' (with a green checkmark icon and the text 'Marina Kysla'), and 'Inactive' (with a green checkmark icon and an unchecked checkbox).

# Добавить автонумерацию к полю на странице добавления записи (back-end)



**Пример.** Добавить автонумерацию к полю [ Код ] ([ Code ]) страницы добавления продукта.  
Шаблон номера: ART\_0000N, где N = 1, 2, и т. д. Автонумерацию реализовать на стороне back-end.

## 1. Создать системные настройки

1. Создайте [системную настройку](#) с маской кода продукта.

- Перейдите в дизайнер системы по кнопке
- В блоке [ Настройка системы ] ([ System setup ]) перейдите по ссылке [ Системные настройки ] ([ System settings ]).
- На панели инструментов раздела нажмите на кнопку [ Добавить настройку ] ([ Add setting ]).
- Заполните **свойства системной настройки**.
  - [ Название ] ([ Name ]) — "Маска кода продукта" ("Product code mask").
  - [ Код ] ([ Code ]) — "ProductCodeMask".
  - [ Тип ] ([ Type ]) — выберите "Строка неограниченной длины" ("Unlimited length text").
  - [ Значение по умолчанию ] ([ Default value ]) — "ART\_{0:00000}".

2. Создайте [системную настройку](#) с текущим кодом продукта.

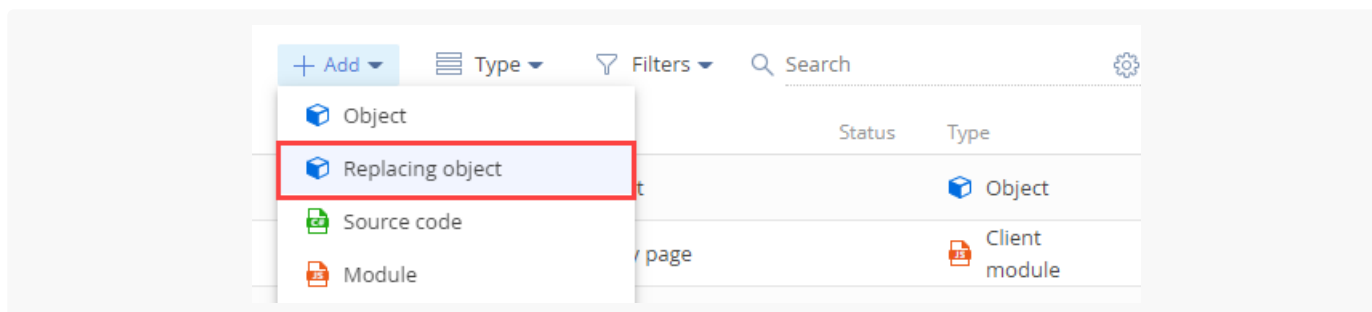
- Перейдите в дизайнер системы по кнопке
- В блоке [ Настройка системы ] ([ System setup ]) перейдите по ссылке [ Системные настройки ] ([ System settings ]).
- На панели инструментов раздела нажмите на кнопку [ Добавить настройку ] ([ Add setting ]).
- Заполните **свойства системной настройки**.



- [ *Название* ] ([ *Name* ]) — "Текущий код продукта" ("Product last number").
- [ *Код* ] ([ *Code* ]) — "ProductLastNumber".
- [ *Тип* ] ([ *Type* ]) — выберите "Целое число" ("Integer").

## 2. Создать схему замещающего объекта

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



### 3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "Product".
- [ *Заголовок* ] ([ *Title* ]) — "Продукт" ("Product").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "Product".

**Product**

Search

- Inherited columns
- Columns
- Indexes
- Events
- Source code

**General**

Code\*  
Product

Package  
sdkAutonumberedFieldBackEndPackage

Title\*  
Product

Description

**Inheritance**

Parent object\*  
Product

☒ Replace parent

#### 4. В схему добавьте **событие**.

- Перейдите в узел [ События ] ([ Events ]) структуры объекта.
- В блоке [ Добавление ] ([ Adding ]) установите признак [ Перед добавлением записи ] ([ Before record added ]). Событию присвоено имя `ProductInserting`.

**Product**

Search

- Inherited columns
- Columns
- Indexes
- Events
- Source code

**Adding**

☒ Before record added ☐ After record added

ProductInserting

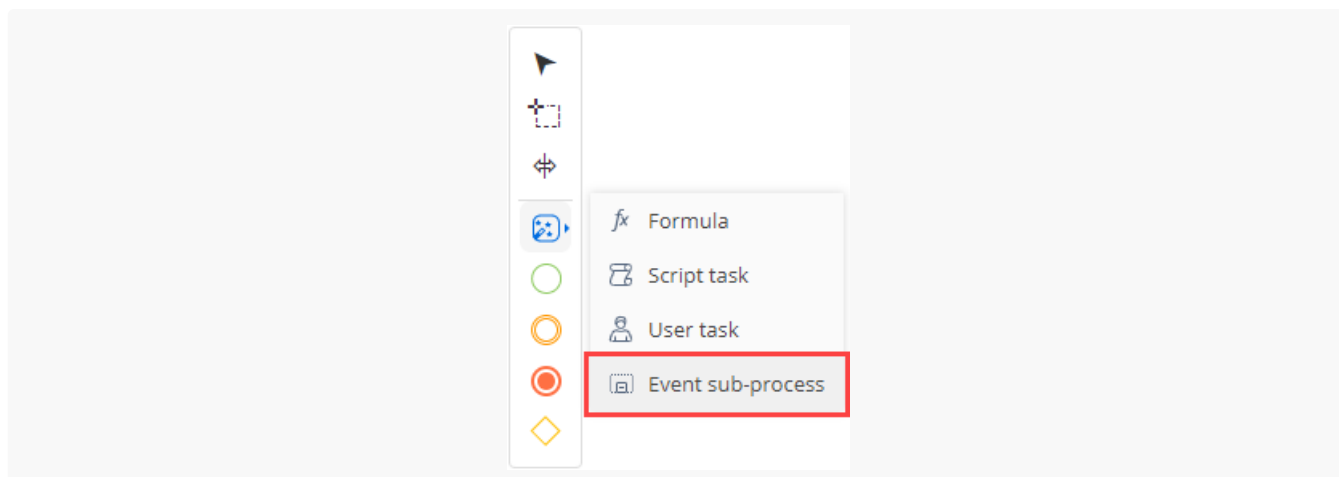
**Saving**

☐ Validating ☐ Before record saved

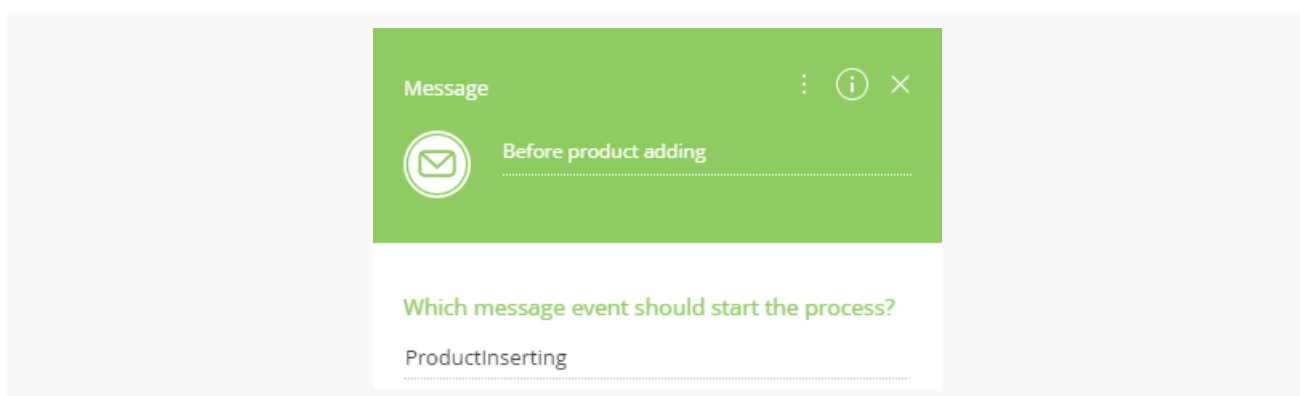
- На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]).

#### 5. Реализуйте **событийный подпроцесс**.

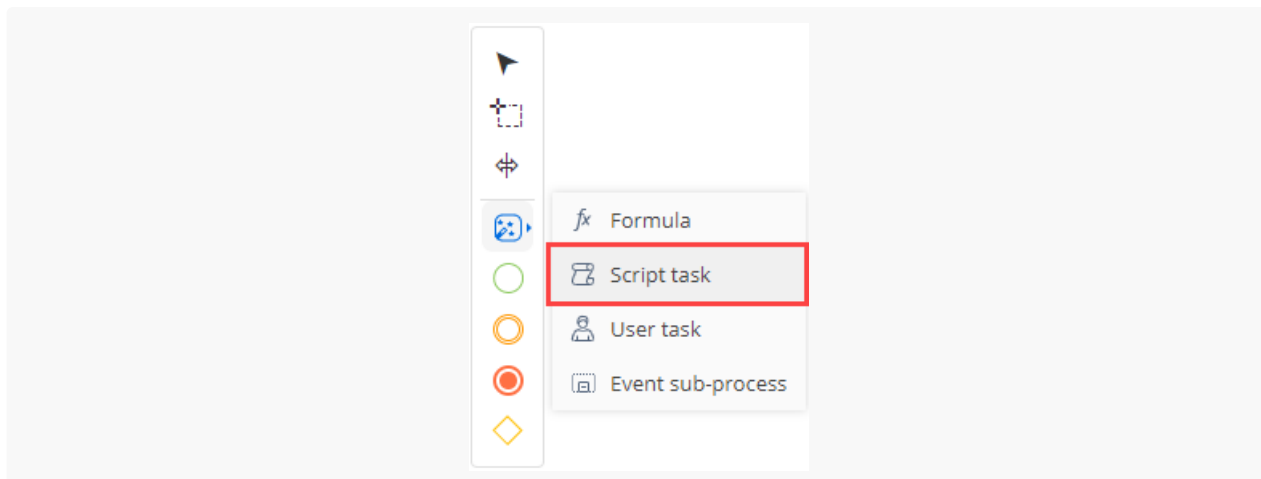
- На панели инструментов дизайнера объектов нажмите [ Открыть процесс ] ([ Open process ]).
- В области элементов дизайнера нажмите [ Действия системы ] ([ System actions ]) и разместите элемент [ Событийный подпроцесс ] ([ Event sub-process ]) в рабочей области дизайнера процессов.



- c. На панели настройки элементов заполните свойство [ Заголовок ] ([ Title ]) — "Product Inserting Sub-process".
- d. Настройте **элементы событийного подпроцесса**.
  - a. Настройте **начальное событие** [ Сообщение ] ([ Message ]).
    - [ Заголовок ] ([ Title ]) — "Before product adding".
    - [ При получении какого сообщения запускать процесс? ] ([ Which message event should start the process? ]) — "ProductInserting".



- d. Добавьте **логический оператор** [ Исключающее "ИЛИ" ] ([ Exclusive gateway (OR) ]).  
 Для этого в меню начального события [ Сообщение ] ([ Message ]) выберите [ Исключающее "ИЛИ" ] ([ Exclusive gateway (OR) ]).
- e. Добавьте **действие системы** [ Задание-сценарий ] ([ Script task ]).
  - a. В области элементов дизайнера нажмите [ Действия системы ] ([ System actions ]) и разместите действие системы [ Задание-сценарий ] ([ Script task ]) в рабочей области подпроцесса.



- b. Действию системы [ *Задание-сценарий* ] ([ *Script task* ]) добавьте имя "Определить схему объекта для генерации номера" ("Get entity schema to generate number").
- c. Добавьте код действия системы [ *Задание-сценарий* ] ([ *Script task* ]).

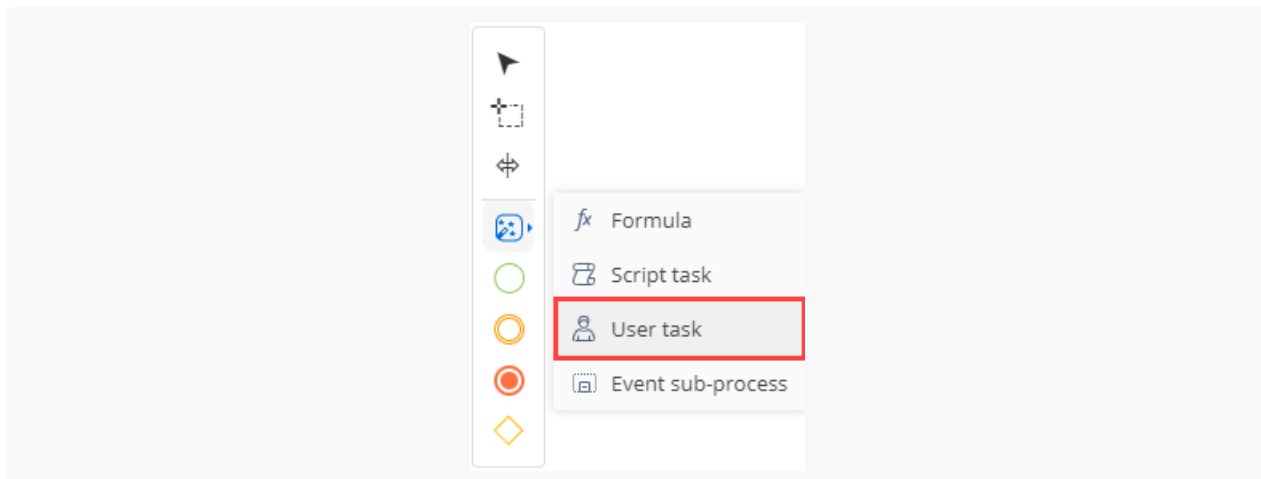
#### Код действия системы [ *Задание-сценарий* ] ([ *Script task* ])

```
/* Установка схемы для генерации номера. */
UserTask1.EntitySchema = Entity.Schema;
return true;
```

UserTask1 — код действия системы [ *Выполнить действие процесса* ] ([ *User task* ])

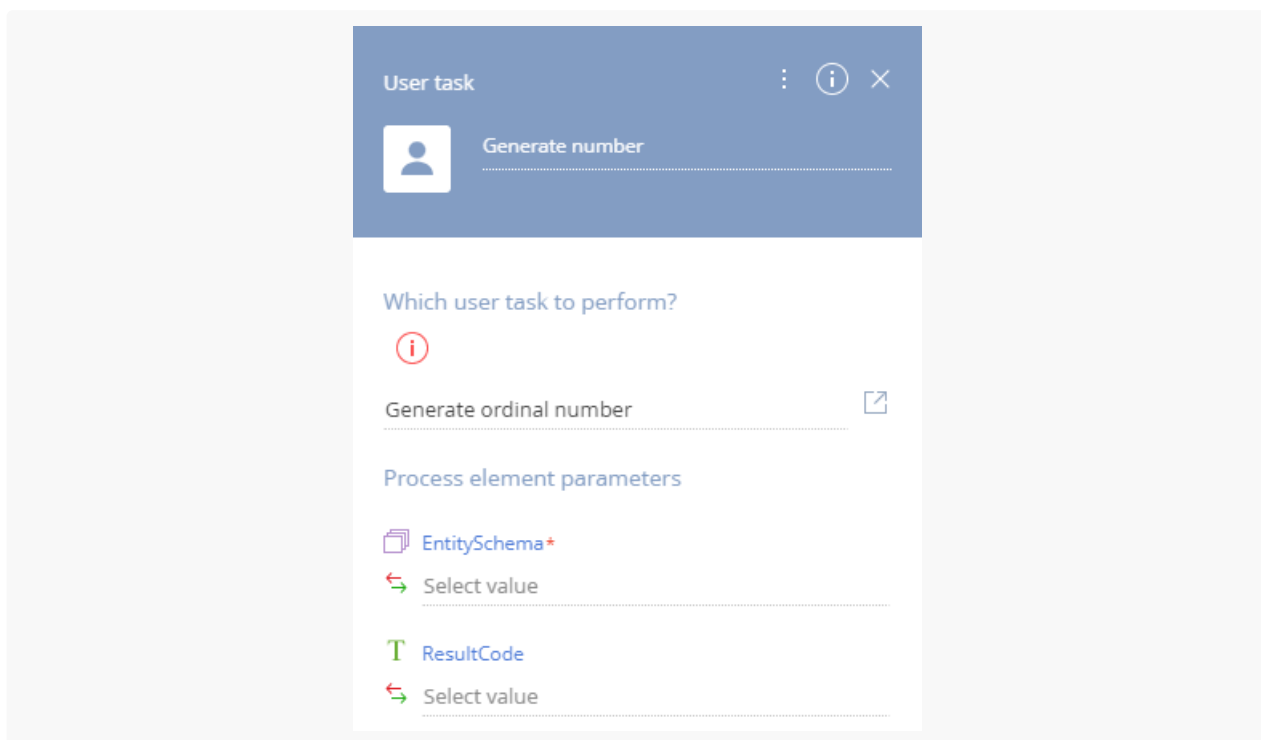
Выполнить генерацию номера (Generate number), настройка которого описана на [следующем шаге](#). Изменить код можно в расширенном режиме настройки действия системы [ *Выполнить действие процесса* ] ([ *User task* ]).

- d. На панели инструментов дизайнера процессов нажмите [ *Сохранить* ] ([ *Save* ]).
- f. Добавьте **действие системы** [ *Выполнить действие процесса* ] ([ *User task* ]).
  - a. В области элементов дизайнера нажмите [ *Действия системы* ] ([ *System actions* ]) и разместите действие системы [ *Выполнить действие процесса* ] ([ *User task* ]) в рабочей области подпроцесса.



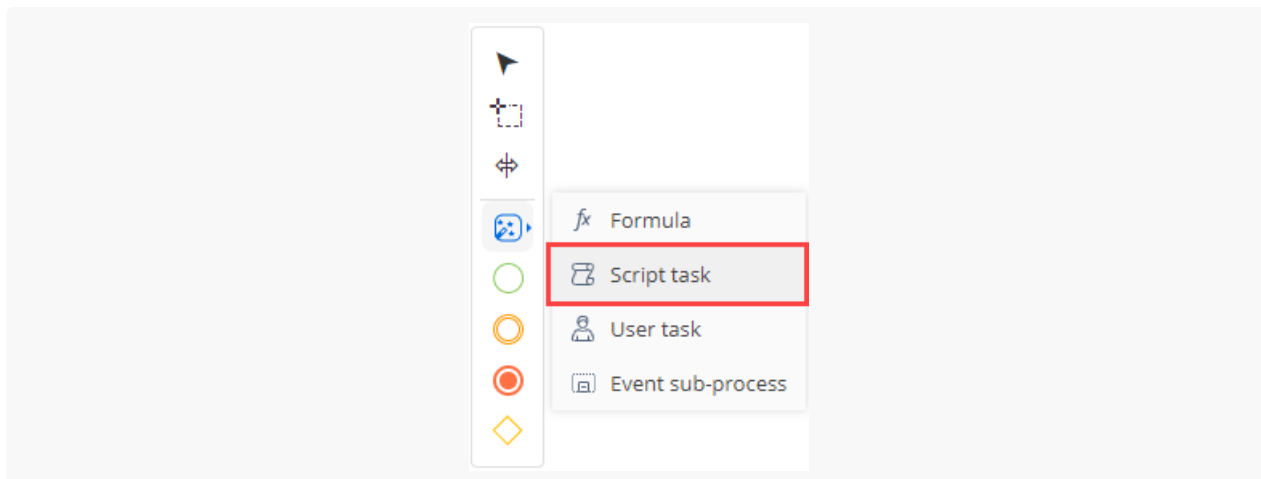
b. Заполните **свойства действия системы**.

- [ *Заголовок* ] ([ *Title* ]) — "Выполнить генерацию номера" ("Generate number").
- [ *Какое пользовательское действие выполнить?* ] ([ *Which user task to perform?* ]) — выберите "Generate ordinal number". Системное действие генерирует текущий порядковый номер в соответствии с маской, которая установлена в системной настройке `ProductCodeMask`.



g. Добавьте **действие системы** [ *Задание-сценарий* ] ([ *Script task* ]).

- В области элементов дизайнера нажмите [ *Действия системы* ] ([ *System actions* ]) и разместите действие системы [ *Задание-сценарий* ] ([ *Script task* ]) в рабочей области подпроцесса.



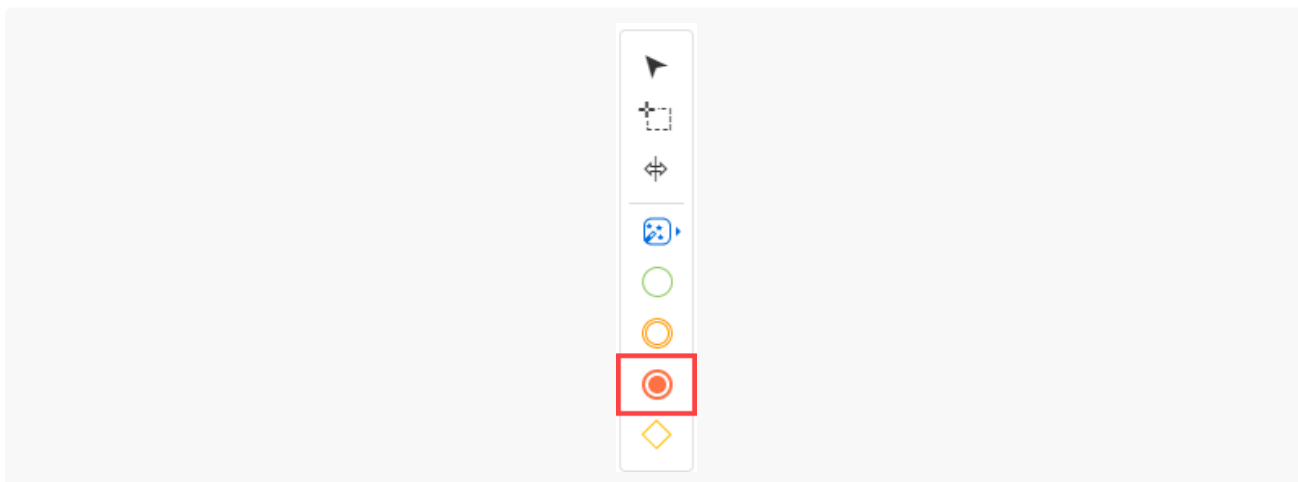
- b. Действию системы [ *Задание-сценарий* ] ([ *Script task* ]) добавьте имя "Записать полученный номер в колонку объекта" ("Save number to entity column").
- c. Добавьте код действия системы [ *Задание-сценарий* ] ([ *Script task* ]).

#### Код действия системы [ *Задание-сценарий* ] ([ *Script task* ])


```
Entity.SetColumnValue("Code", UserTask1.ResultCode);
return true;
```

- d. На панели инструментов дизайнера процессов нажмите [ *Сохранить* ] ([ *Save* ]).
- h. Добавьте **событие** [ *Останов* ] ([ *Terminate* ]).


Для этого в области элементов дизайнера нажмите [ *Останов* ] ([ *Terminate* ]) и разместите событие в рабочей области подпроцесса.



- e. Настройте **потоки**.
  - a. Настройте **условный поток** между логическим оператором [ *Исключающее "ИЛИ"* ] ([ *Exclusive gateway (OR)* ]) и действием системы [ *Определить схему объекта для генерации номера* ] ([ *Get entity schema to generate number* ]).
  - a. В меню логического оператора [ *Исключающее "ИЛИ"* ] ([ *Exclusive gateway (OR)* ]) нажмите на

кнопку  и соедините логический оператор [ *Исключающее "ИЛИ"* ] ([ *Exclusive gateway (OR)* ]) с действием системы [ *Определить схему объекта для генерации номера* ] ([ *Get entity schema to generate number* ]).



b. Заполните **свойства условного потока**.

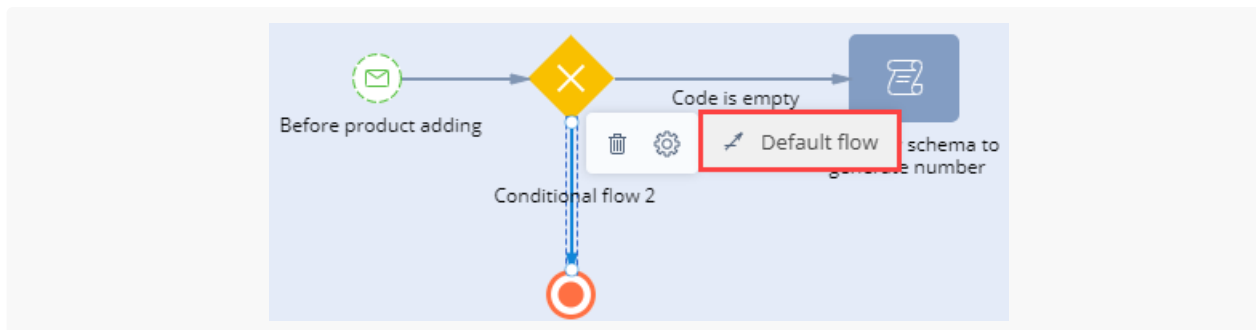
- [ *Заголовок* ] ([ *Title* ]) — "Код не заполнен" ("Code is empty").
- [ *Условие перехода* ] ([ *Condition to move down the flow* ]).
- На панели настройки элементов в свойстве [ *Условие перехода* ] ([ *Condition to move down the flow* ]) нажмите кнопку .
- Задайте формулу.

```
string.IsNullOrEmpty(Entity.GetTypedColumnValue<string>("Code"))
```



- Сохраните изменения.

b. Настройте **поток по умолчанию** между логическим оператором [ *Исключающее "ИЛИ"* ] ([ *Exclusive gateway (OR)* ]) и событием [ *Останов* ] ([ *Terminate* ]).

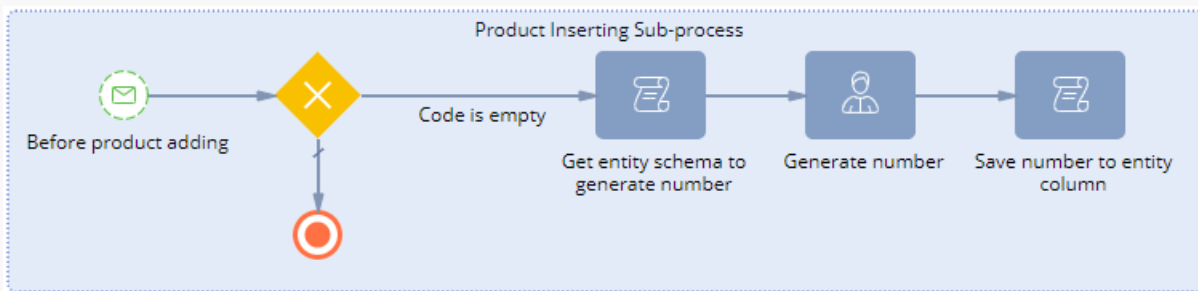
- В меню логического оператора [ *Исключающее "ИЛИ"* ] ([ *Exclusive gateway (OR)* ]) нажмите на кнопку  и соедините логический оператор [ *Исключающее "ИЛИ"* ] ([ *Exclusive gateway (OR)* ]) с событием [ *Останов* ] ([ *Terminate* ]).
- Трансформируйте поток управления в поток по умолчанию. Для этого в меню потока нажмите  —> [ *Поток по умолчанию* ] ([ *Default flow* ]).



c. Настройте **потоки управления**.

- В меню действия системы [ *Определить схему объекта для генерации номера* ] ([ *Get entity schema to generate number* ]) нажмите на кнопку  и соедините действие системы [ *Определить схему объекта для генерации номера* ] ([ *Get entity schema to generate number* ]) с действием системы [ *Выполнить генерацию номера* ] ([ *Generate number* ]).
- В меню действия системы [ *Выполнить генерацию номера* ] ([ *Generate number* ]) нажмите на кнопку  и соедините действие системы [ *Выполнить генерацию номера* ] ([ *Generate number* ]) с действием системы [ *Записать полученный номер в колонку объекта* ] ([ *Save number to entity column* ]).

Событийный подпроцесс представлен на рисунке ниже.



6. На панели инструментов дизайнера процессов нажмите [ Сохранить ] ([ Save ]), а затем [ Опубликовать ] ([ Publish ]).

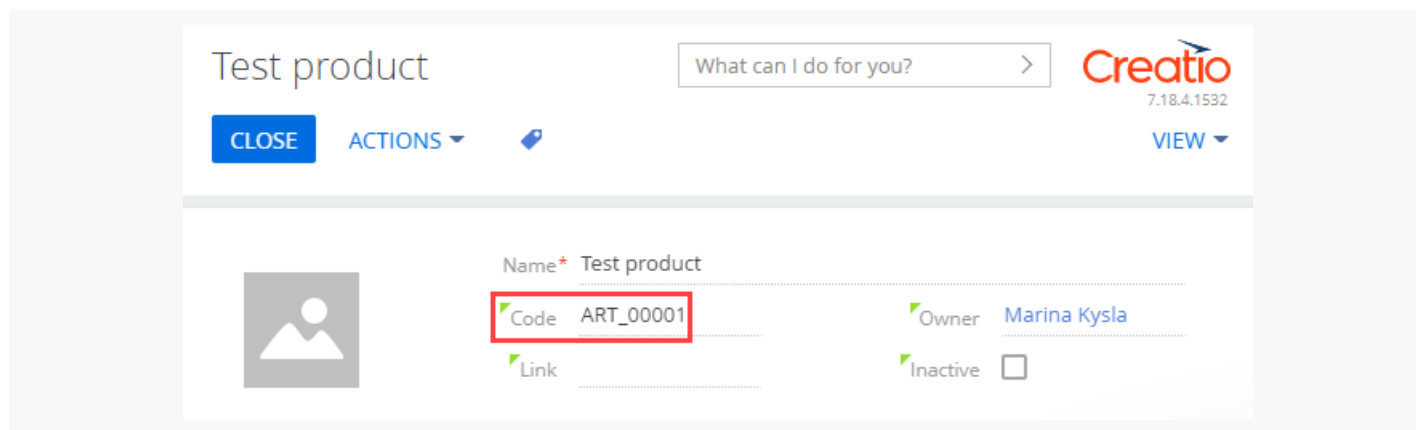
## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Очистите кэш браузера.
2. Обновите страницу раздела [ Продукты ] ([ Products ]).
3. Добавьте и сохраните новый продукт (например, `Test product`).

Автогенерация кода и его сохранение в колонку выполняется на стороне сервера при возникновении события [ Перед сохранением записи ] ([ Before Record Saved ]), которое возникает на стороне сервера после отправки запроса на добавление записи из front-end части. Поэтому значение кода невозможно сразу отобразить на странице добавления продукта. Номер отобразится после сохранения продукта.

В результате выполнения примера добавлена автонумерацию к полю [ Код ] ([ Code ]) страницы продукта.



## Добавить информационную кнопку к полю на странице записи

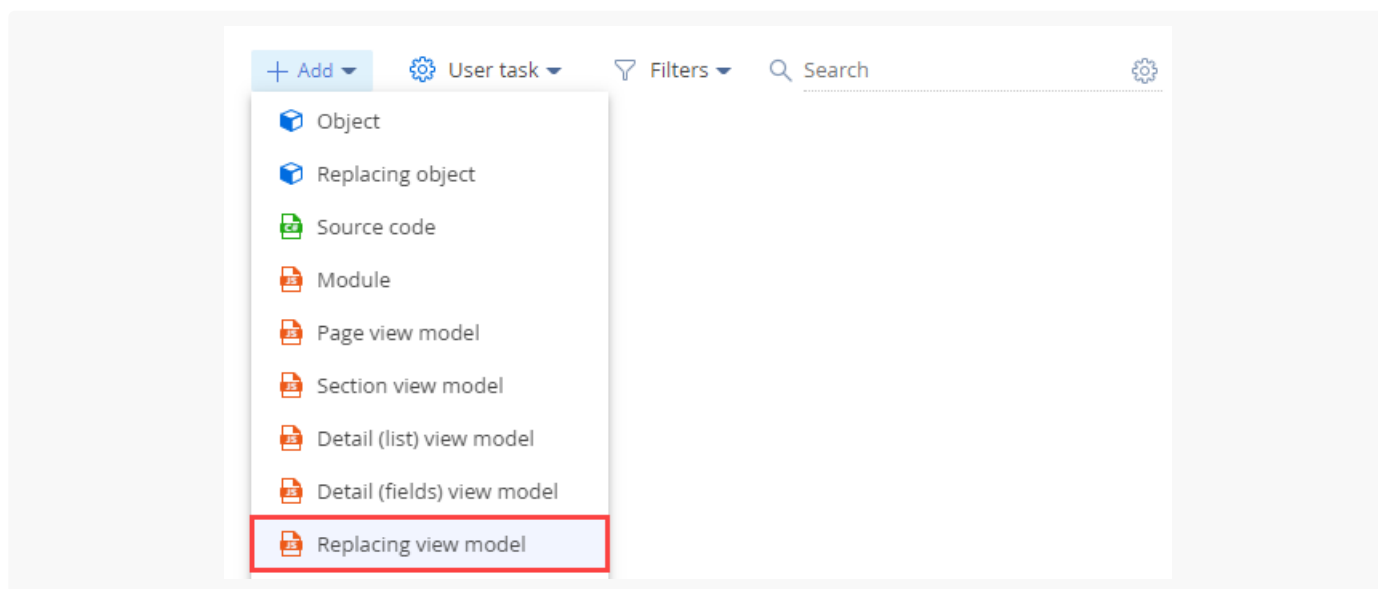
 Средний



**Пример.** Добавить информационную кнопку к полю [ *ФИО* ] ([ *Full name* ]) в профиль контакта страницы контакта. К информационной кнопке добавить всплывающую подсказку.

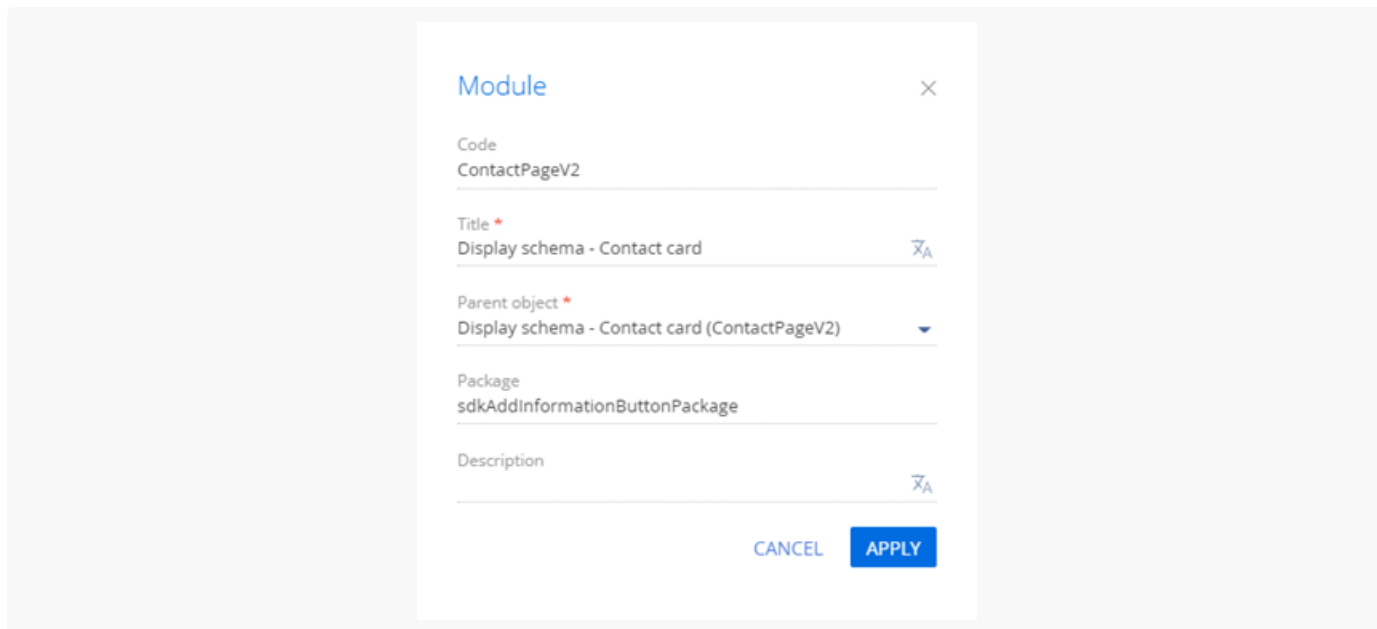
## Создать схему замещающей модели представления страницы контакта

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



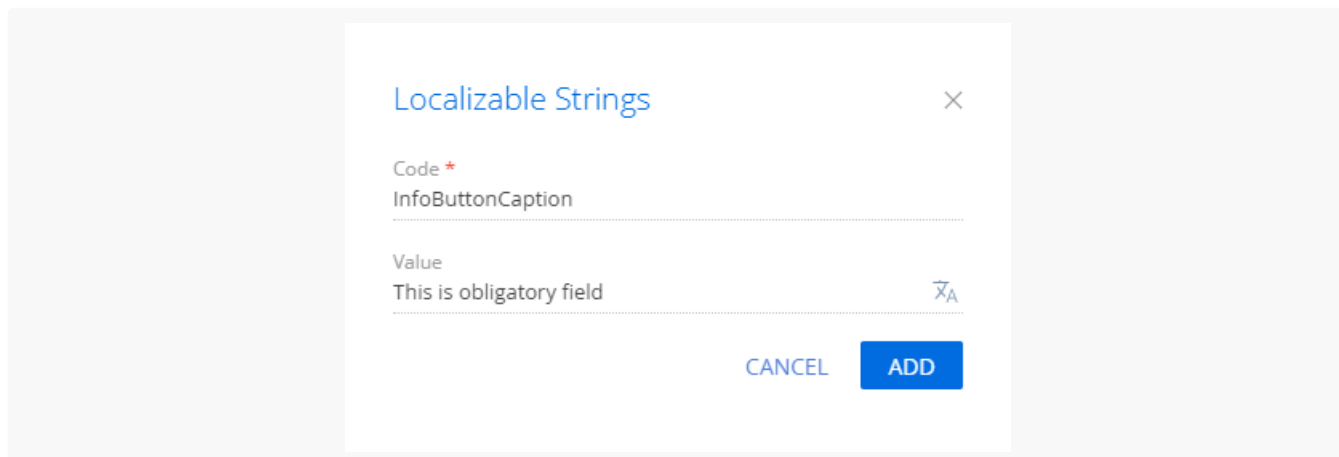
3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "ContactPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Схема отображения карточки контакта" ("Display schema - Contact card").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "ContactPageV2".



4. Добавьте **локализуемую строку**.

- a. В контекстном меню узла [ *Локализуемые строки* ] ([ *Localizable strings* ]) нажмите кнопку **+**.
- b. Заполните **свойства локализуемой строки**.
  - [ *Код* ] ([ *Code* ]) — "InfoButtonCaption".
  - [ *Значение* ] ([ *Value* ]) — "Это обязательное поле" ("This is obligatory field").



- e. Для добавления локализуемой строки нажмите [ *Добавить* ] ([ *Add* ]).

5. Реализуйте **информационную кнопку**.

Для этого в массив модификаций `diff` добавьте конфигурационный объект с настройками расположения информационной кнопки к полю на странице.

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

**ContactPageV2**

```
define("ContactPageV2", [], function () {
```

```

return {
    /* Название схемы объекта страницы записи. */
    entitySchemaName: "Contact",
    /* Отображение информационной кнопки к полю на странице записи. */
    diff: /**SCHEMA_DIFF*/[
        /* Метаданные для добавления текста информационной кнопки. */
        {
            /* Выполняется операция изменения существующего элемента. */
            "operation": "merge",
            /* Мета-имя родительского контейнера, в который добавляется информационная кн
            "parentName": "ProfileContainer",
            /* Информационная кнопка добавляется в коллекцию элементов родительского элем
            "propertyName": "items",
            /* Мета-имя изменяемого поля. */
            "name": "AccountName",
            /* Свойства, передаваемые в конструктор элемента. */
            "values": {
                /* Настройка расположения информационной кнопки. */
                "layout": {
                    /* Номер столбца. */
                    "column": 0,
                    /* Номер строки. */
                    "row": 1,
                    /* Диапазон занимаемых столбцов. */
                    "colSpan": 22,
                    /* Диапазон занимаемых строк. */
                    "rowSpan": 1
                }
            }
        },
        {
            /* Выполняется операция добавления элемента на страницу. */
            "operation": "insert",
            "parentName": "ProfileContainer",
            "propertyName": "items",
            "name": "SimpleInfoButton",
            "values": {
                "layout": {
                    "column": 22,
                    "row": 1,
                    "colSpan": 1,
                    "rowSpan": 1
                },
                /* Тип добавляемого элемента – информационная кнопка. */
                "itemType": Terrasoft.ViewItemType.INFORMATION_BUTTON,
                /* Текст подсказки. */
                "content": { "bindTo": "Resources.Strings.InfoButtonCaption" }
            }
        }
    ]
}

```

```

    ]/**SCHEMA_DIFF*/
  };
});

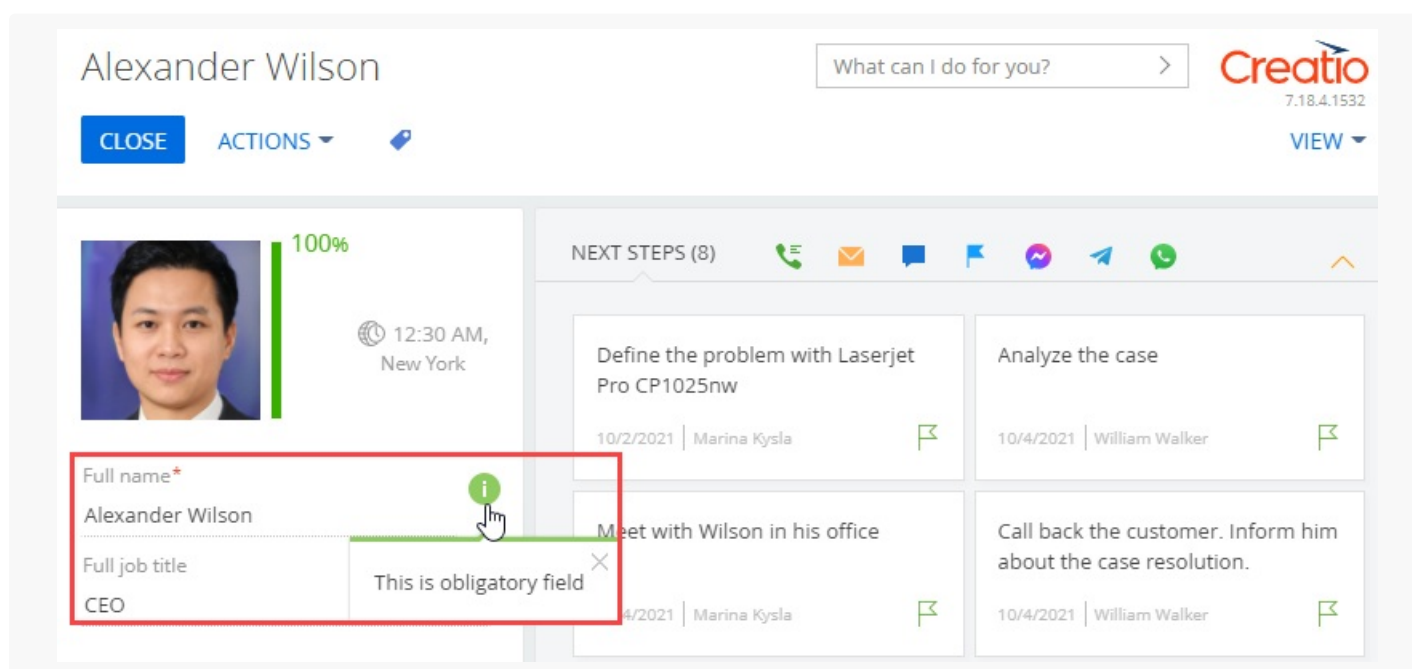
```

6. На панели инструментов дизайнера нажмите [ *Сохранить* ] ([ *Save* ]).

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ *Контакты* ] ([ *Contacts* ]).

В результате выполнения примера в профиль контакта страницы контакта добавлена информационная кнопка к полю [ *ФИО* ] ([ *Full name* ]).



## Добавить всплывающую подсказку к полю на странице записи

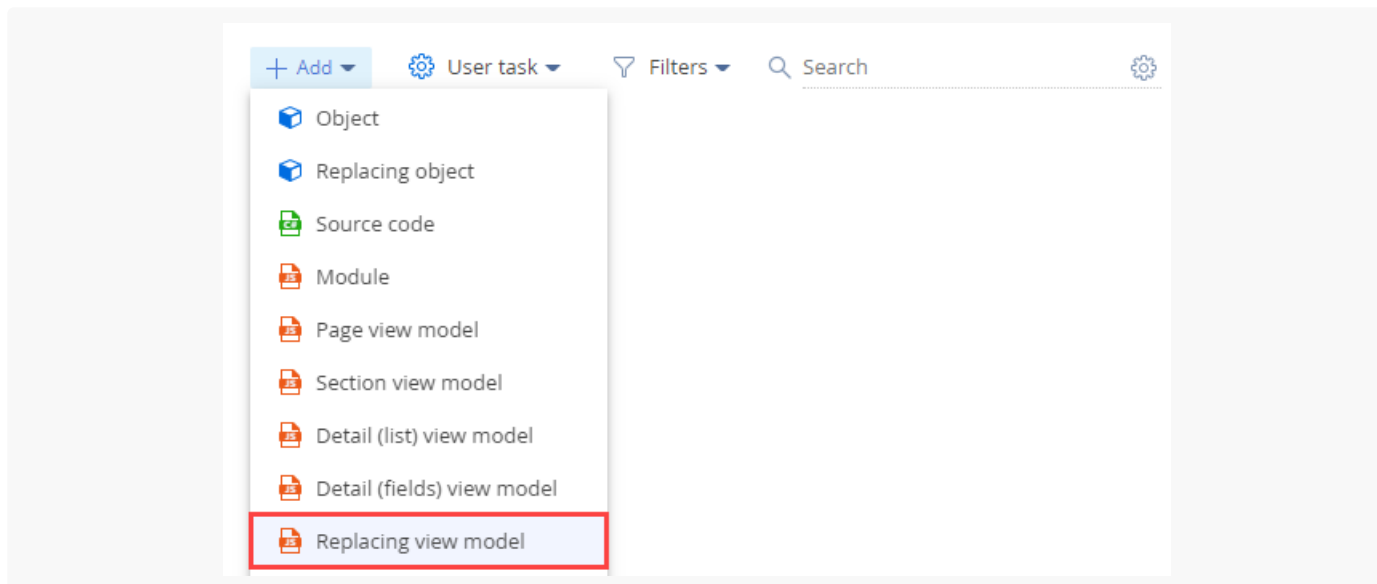
 Средний

**Пример.** Добавить всплывающую подсказку к полю [ *Тип* ] ([ *Type* ]) страницы контакта.

## Создать схему замещающей модели представления страницы контакта

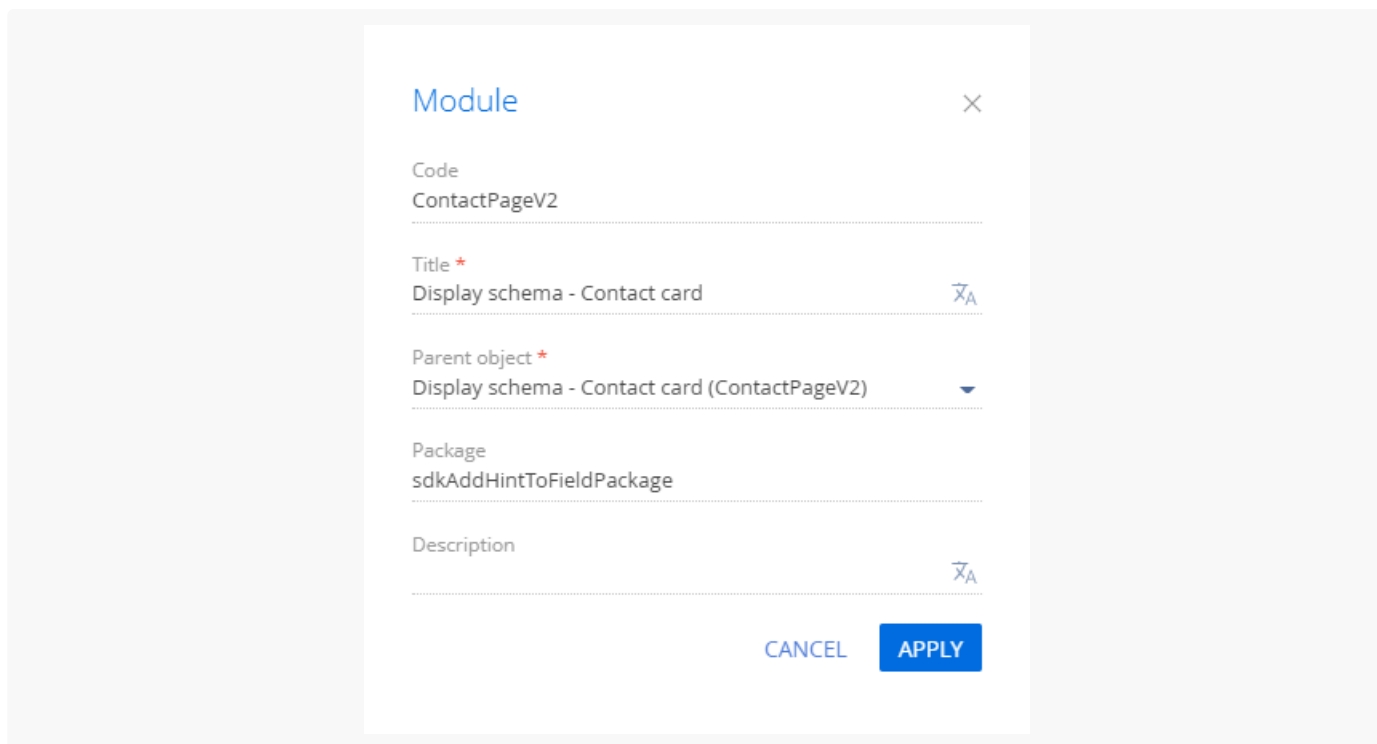
1. [Перейдите в раздел \[ \*Конфигурация\* \]](#) ([ *Configuration* ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.

2. На панели инструментов реестра раздела нажмите [ *Добавить* ] —> [ *Замещающая модель представления* ] ([ *Add* ] —> [ *Replacing view model* ]).



3. Заполните **свойства схемы**.

- [ *Код* ] ([ *Code* ]) — "ContactPageV2".
- [ *Заголовок* ] ([ *Title* ]) — "Схема отображения карточки контакта" ("Display schema – Contact card").
- [ *Родительский объект* ] ([ *Parent object* ]) — выберите "ContactPageV2".

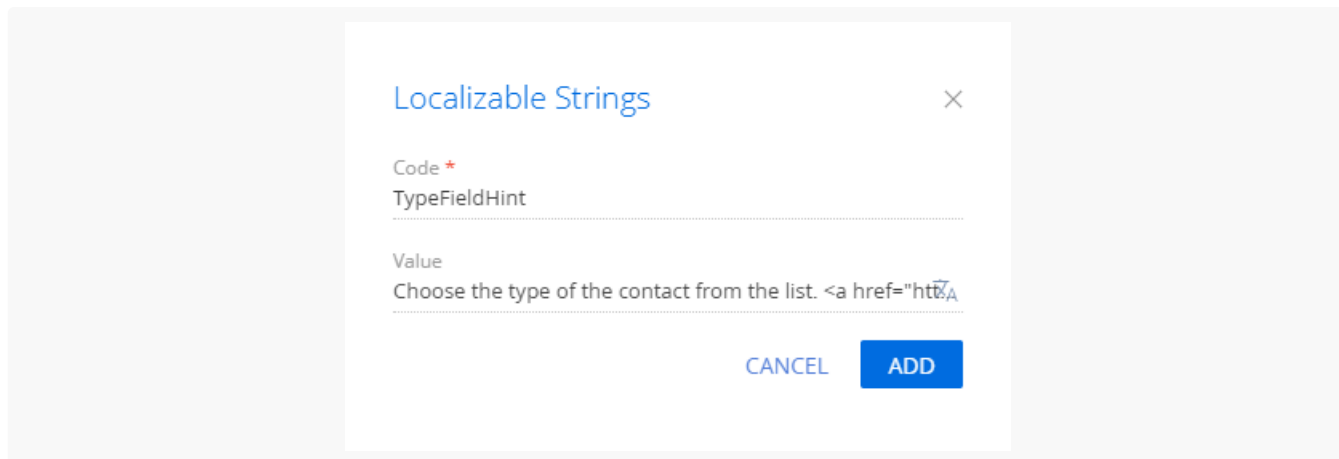


4. Добавьте **локализуемую строку**, которая содержит текст подсказки.

- a. В контекстном меню узла [ *Локализуемые строки* ] ([ *Localizable strings* ]) нажмите кнопку .

b. Заполните **свойства локализуемой строки**.

- [ Код ] ([ Code ]) — "TypeFieldHint".
- [ Значение ] ([ Value ]) — "Выберите из списка тип контакта. <a href='\"https://academy.terrasoft.ua/docs/user/bazis\_platformy/interfejs/stranitsy\_zapisey/stranicy\_zapisej\" target='\"\_blank\">Узнать больше</a>" ("Choose the type of the contact from the list. <a href='\"https://academy.creatio.com/docs/user/platform\_basics/user\_interface/record\_pages\_shortcut/record\_pages\" target='\"\_blank\">Read more</a>").



e. Для добавления локализуемой строки нажмите [ *Добавить* ] ([ Add ]).

5. Настройте **всплывающую подсказку к полю** [ Тип ] ([ Type ]) страницы контакта. Для этого в массив модификаций `diff` добавьте конфигурационный объект поля на странице.

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

## ContactPageV2

```
define("ContactPageV2", [], function () {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение всплывающей подсказки. */
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для добавления к полю всплывающей подсказки. */
            {
                /* Выполняется операция изменения существующего элемента. */
                "operation": "merge",
                /* Мета-имя изменяемого поля. */
                "name": "Type",
                /* Мета-имя родительского контейнера, в котором изменяется поле. */
                "parentName": "ContactGeneralInfoBlock",
                /* Поле изменяется изменяется в коллекции элементов родительского элемента. */
                "propertyName": "items",
                /* Свойства, передаваемые в конструктор элемента. */
                "values": {
```

```

/* Свойство поля, которое отвечает за отображение подсказки. */
"tip": {
  /* Текст подсказки. */
  "content": { "bindTo": "Resources.Strings.TypeFieldHint" },
  /* Режим отображения подсказки.
  По умолчанию режим WIDE - толщина зеленой полосы, которая отображает
  "displayMode": Terrasoft.controls.TipEnums.displayMode.WIDE
  }
}
}
]/**SCHEMA_DIFF*/
});
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ Контакты ] ([ Contacts ]).

В результате выполнения примера к полю [ Тип ] ([ Type ]) страницы контакта добавлена всплывающая подсказка.

Andrew Z. Barber

What can I do for you? > Creatio 7.18.4.1532 VIEW

SAVE CANCEL ACTIONS

100% 11:41 PM -1d, Seattle

Full name\* Andrew Z. Barber

Full job title Project Manager

Mobile phone +1 206 587 1036

Business phone +1 206 480 3801

NEXT STEPS (1)

Contact customer, specify need, budget, decision-making role.

10/3/2021 | Valerie E. Murphy

Choose the type of the contact from the list. [Read more](#)

Type Customer

Owner Marina Kysla

Title Mr.

Gender Male

Recipient's name Barber

Preferred language

Age 46

MAINTENANCE TIMELINE ENGAGEMENT >