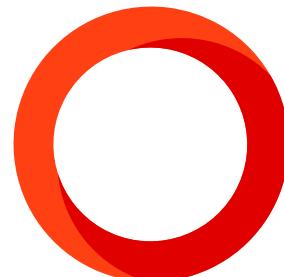
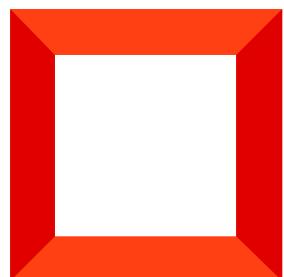
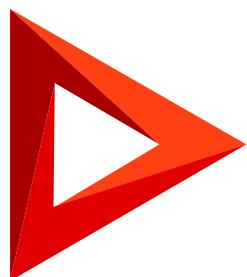


Страница записи

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

Страница записи	8
Контейнеры страницы записи	8
Создать страницу записи	9
Настроить страницу записи	10
Добавить пользовательское действие на страницу записи	11
Добавить действие на страницу записи	12
1. Создать схему замещающей модели представления страницы заказа	12
2. Создать схему замещающей модели представления раздела	16
Результат выполнения примера	18
Схема BasePageV2	19
Сообщения	19
Атрибуты	21
Методы	23
Схема BaseEntityPage	26
Сообщения	26
Атрибуты	26
Миксины	27
Методы	27
Панель действий	29
Добавить панель действий на страницу	30
Добавить новый канал на панель действий	30
Добавить пользовательское действие верификации	30
Добавить панель действий	32
Описание примера	32
Исходный код	32
Алгоритм реализации примера	32
Добавить новый канал на панель действий	36
Описание примера	36
Исходный код	36
Алгоритм выполнения примера	36
Добавить мультиязычные шаблоны email-сообщений	43
Описание примера	44
Исходный код	44
Предварительные настройки	44
Алгоритм реализации примера	45
Создать пользовательскую страницу действия верификации	50

1. Создать схему страницы действия верификации	51
2. Настроить представление страницы	52
2. Использовать созданную схему в бизнес-процессе	52
Результат выполнения примера	53
Поле	53
Типы полей и операций с полями	54
Добавить поле	54
Реализовать валидацию поля	58
Установить для поля значение по умолчанию	59
Настроить обязательность для поля	60
Настроить фильтрацию значений справочного поля	61
Настроить условия блокировки поля	62
Настроить исключения блокировки поля	64
Настроить условия отображения поля	66
Добавить автонумерацию к полю	66
Добавить информационную кнопку к полю	67
Добавить всплывающую подсказку к полю	67
Вычислить разницу дат в полях	68
Добавить поле на страницу записи с использованием новой колонки	68
1. Создать схему замещающего объекта	68
2. Создать схему замещающей модели представления страницы активности	70
Результат выполнения примера	73
Добавить поле на страницу записи с использованием существующей колонки	73
Создать схему замещающей модели представления страницы контакта	74
Результат выполнения примера	75
Добавить поле с изображением на страницу записи	76
1. Создать схему замещающего объекта	76
2. Создать схему замещающей модели представления страницы статьи базы знаний	78
Результат выполнения примера	84
Добавить вычисляемое поле на страницу записи	84
1. Создать схему замещающего объекта	84
2. Создать схему замещающей модели представления страницы заказа	86
Результат выполнения примера	90
Добавить мультивалютное поле на страницу записи	90
1. Создать схему замещающего объекта	90
2. Создать схему замещающей модели представления страницы проекта	93
Результат выполнения примера	98
Реализовать валидацию поля типа [Дата/Время] на странице записи	99
Создать схему замещающей модели представления страницы продажи	99

Результат выполнения примера	102
Реализовать валидацию поля типа [Строка] на странице записи	103
Создать схему замещающей модели представления страницы контакта	103
Результат выполнения примера	106
Установить значение по умолчанию для поля на странице записи	106
Создать схему замещающей модели представления страницы проекта	107
Результат выполнения примера	109
Настроить обязательность для поля на странице записи	109
Создать схему замещающей модели представления страницы контакта	110
Результат выполнения примера	112
Настроить фильтрацию значений справочного поля на странице записи	113
Создать схему замещающей модели представления страницы контрагента	113
Результат выполнения примера	116
Настроить фильтрацию значений связанных справочных полей на странице записи	117
Создать схему замещающей модели представления страницы контакта	117
Результат выполнения примера	121
Настроить условия блокировки поля на странице записи	126
Создать схему замещающей модели представления страницы контакта	126
Результат выполнения примера	129
Настроить исключения блокировки полей на странице записи	129
Создать схему замещающей модели представления страницы счета	129
Результат выполнения примера	132
Настроить условия отображения поля на странице записи	133
1. Создать схему замещающего объекта	133
2. Создать схему замещающей модели представления страницы активности	135
Результат выполнения примера	139
Добавить автонумерацию к полю на странице добавления записи (front-end)	140
1. Создать системные настройки	140
2. Создать схему замещающей модели представления страницы продукта	141
Результат выполнения примера	143
Добавить автонумерацию к полю на странице добавления записи (back-end)	144
1. Создать системные настройки	144
2. Создать схему замещающего объекта	145
Результат выполнения примера	152
Добавить информационную кнопку к полю на странице записи	152
Создать схему замещающей модели представления страницы контакта	153
Результат выполнения примера	156
Добавить всплывающую подсказку к полю на странице записи	156
Создать схему замещающей модели представления страницы контакта	156

Результат выполнения примера	159
Кнопка	159
Контейнеры кнопок	160
Добавить кнопку	162
Добавить к кнопке всплывающую подсказку	163
Добавить кнопку на панель инструментов раздела	164
Создать схему замещающей модели представления раздела	165
Результат выполнения примера	168
Добавить кнопку на панель инструментов страницы записи в совмещенном режиме	169
Создать схему замещающей модели представления раздела	170
Результат выполнения примера	174
Добавить кнопку на панель инструментов страницы добавления записи	176
1. Изменить способ добавления контрагента	176
2. Создать схему замещающей модели представления страницы контрагента	176
Результат выполнения примера	179
Добавить кнопку выбора цвета на страницу записи	181
1. Создать схему замещающего объекта	181
2. Создать схему замещающей модели представления страницы контрагента	183
Результат выполнения примера	185
Добавить к кнопке всплывающую подсказку	186
Создать схему замещающей модели представления страницы контакта	186
Результат выполнения примера	190
Добавить кнопку в строку записи раздела	190
Создать схему замещающей модели представления раздела	190
Результат выполнения примера	194
Свойство diff объекта кнопки	194
Свойства	194
Хронология	197
Таблицы базы данных	197
Добавление вкладки Хронология в раздел	200
Добавить базовую хронологию в раздел	200
1. Создать SQL-сценарий	200
2. Установить данные SQL-сценария	202
Результат выполнения примера	203
Создать хронологию, связанную с пользовательским разделом	203
1. Создать раздел Книги (Books)	204
2. Создать модуль представления плитки	205
3. Создать модуль модели представления плитки	208
4. Настроить отображение плитки	209

5. Изменить привязку плитки	212
Результат выполнения примера	213
Профиль связанной сущности	213
Добавить профиль связанной сущности	214
Добавить пользовательский профиль связанной сущности на страницу записи	216
1. Создать схему модели представления связанного профиля контрагента	216
2. Создать схему замещающей модели представления страницы контакта	218
Результат выполнения примера	221
Схема BaseProfileSchema	221
Атрибуты	222
Сообщения	222
Методы	223
Схема BaseMultipleProfileSchema	223
Атрибуты	224
Сообщения	224
Схема BaseRelatedProfileSchema	224
Сообщения	225
HTML-элемент iframe	225
Добавить HTML-элемент iframe	226
Создать схему замещающей модели представления страницы записи	226
Результат выполнения примера	228

Страница записи



Основы

Страница записи — элемент интерфейса, который хранит информацию о бизнес-объектах приложения в виде полей, вкладок, деталей и дашбордов. Имя страницы записи соответствует имени объекта системы (например, страница контрагента, страница контакта и т. д.). **Назначение** страницы записи — работа с записями [реестра раздела](#). Каждый раздел приложения содержит одну или несколько страниц записей.

Каждая страница записи представлена схемой [клиентского модуля](#). Например, страница контакта сконфигурирована в схеме `ContactPageV2` пакета `UIv2`. Функциональность базовой страницы записи реализована в схеме `BasePageV2` пакета `NUI`. Все схемы страниц записи должны наследовать схему `BasePageV2`.

Виды страницы записи:

- **Страница добавления записи** — используется для создания записи реестра раздела.
- **Страница редактирования записи** — используется для редактирования существующей записи реестра раздела.

Контейнеры страницы записи

Элементы пользовательского интерфейса приложения, которые относятся к странице записи, размещены в соответствующих контейнерах. Контейнеры конфигурируются в базовой схеме страницы записи или схеме замещающей страницы записи. Не зависят от вида страницы записи.

На заметку. В приложении используются мета-имена html-контейнеров. На основании мета-имен приложение формирует фактические идентификаторы соответствующих html-элементов страницы записи.

Основные **контейнеры** страницы записи представлены на рисунке ниже.

The screenshot illustrates the layout of a CRM application's record page. At the top right is the **ActionButtonsContainer**, which includes a **CLOSE** button, an **ACTIONS** dropdown, and a **VIEW** button. To the left is the **LeftModulesContainer**, which contains fields for **Name*** (Vertigo Systems), **Type** (Customer), **Owner** (Mary King), **Web** (www.vertigosys.com), and **Primary phone** (+44 (20) 3427 1374). In the center is the **ActionDashboardContainer**, which displays a message: "You don't have any tasks yet" with a "Press F above to add a task" instruction. Below the dashboard is the **TabsContainer**, which has tabs for **ACCOUNT INFO** (selected), **CONTACTS AND STRUCTURE**, **MAINTENANCE**, **TIMELINE**, and **CONNECTED TO**. The **ACCOUNT INFO** tab shows details like **Also known as**, **Code** (109), **Segmentation** (No. of employees: 101-200, Annual revenue: 21 - 30 million), and **Business entity** (Corp.).

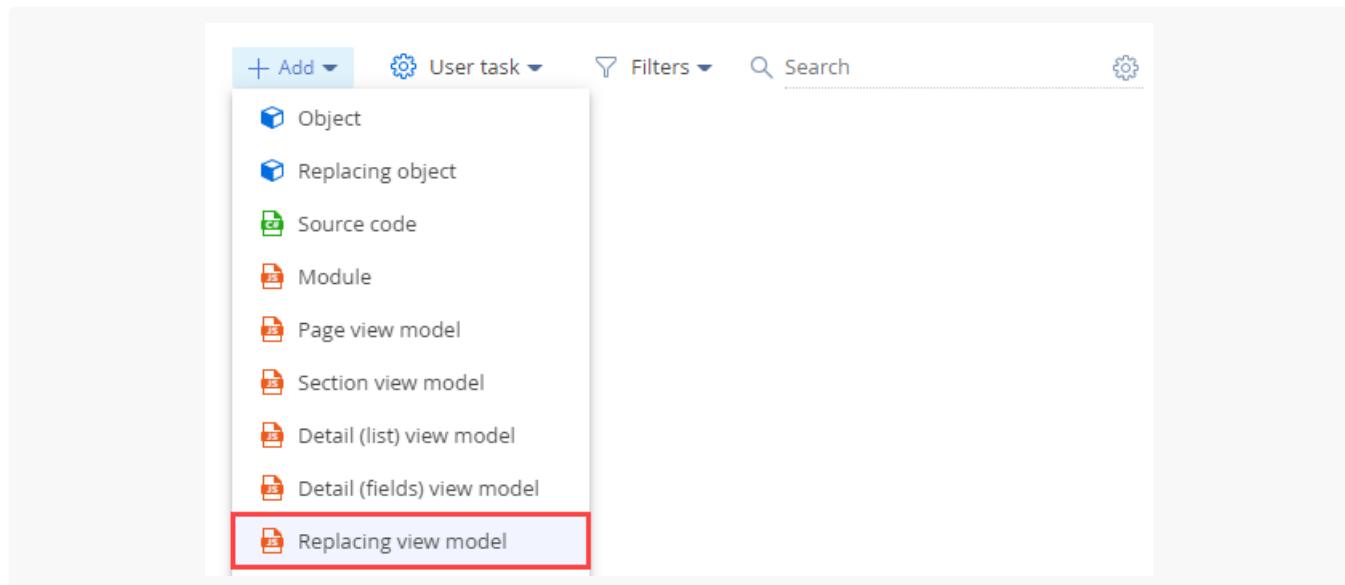
- Контейнер кнопок действий (`ActionButtonsContainer`) — содержит кнопки действий страницы записи.
- Контейнер левой части страницы записи (`LeftModulesContainer`) — содержит основные поля ввода и редактирования данных.
- Контейнер панели действий (`ActionDashboardContainer`) — содержит панель действий и полосу стадий.
- Контейнер вкладок (`TabsContainer`) — содержит вкладки с полями ввода и редактирования, которые сгруппированы по признаку, например, месту работы.

Создать страницу записи

- Перейдите в раздел [[Конфигурация](#)] ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [[Добавить](#)] —> и выберите **вид схемы модели представления**.
 - Для **создания новой страницы записи** выберите [[Модель представления страницы](#)] ([*Page view model*]).

The screenshot shows the configuration interface for creating a new page view model. On the left is a sidebar with buttons for **Add**, **Multi actions**, **Type**, **Filters**, **Search**, and a gear icon. The main area lists objects with columns for **Status**, **Type**, **Object**, and **Modified on**. One object is selected: "User edit page" (Status: **Active**, Type: **Client module**, Object: **editPage**, Modified on: **11/11/2021, 7:40:37 AM**). At the bottom of the sidebar, the **Page view model** option is highlighted with a red box.

- Для замещения существующей страницы записи выберите [Замещающая модель представления] ([Replacing view model]).



3. Заполните **свойства схемы**.

- [Код] ([Code]) — название схемы (обязательное свойство). Должно содержать префикс (по умолчанию `Usr`), указанный в системной настройке [Префикс названия объекта] (код [*SchemaNamePrefix*]).
- [Заголовок] ([Title]) — локализуемый заголовок схемы (обязательное свойство).
- [Родительский объект] ([Parent object]) — выберите схему страницы записи, функциональность которой необходимо наследовать.
 - Для **создания новой страницы записи** выберите "BasePageV2".
 - Для **замещения существующей страницы записи** выберите схему страницы записи, которую планируется заменить. Например, чтобы в приложении отображалась пользовательская страница контрагента, выберите схему `AccountPageV2` пакета `UIv2`.

4. Реализуйте логику страницы записи.

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

Настроить страницу записи

Настройка страницы записи выполняется с помощью **элементов управления**.

Действия для настройки страницы записи, которые позволяет выполнять приложение:

- Добавлять стандартные элементы управления страницы.
- Изменять стандартные элементы управления страницы.
- Добавлять пользовательские элементы управления страницы.

Основные **элементы управления** страницы:

- Панель действий. Настройка панели действий страницы записи описана в статье [Панель действий](#).
- Поле. Настройка полей страницы записи описана в статье [Поле](#).
- Кнопка. Настройка кнопок страницы записи описана в статье [Кнопка](#).

Добавить пользовательское действие на страницу записи

Creatio предоставляет возможность добавления пользовательского действия в выпадающее меню [Действия] ([Actions]) страницы записи, которое реализовано в схеме `BasePageV2` базовой страницы записи.

Чтобы **добавить пользовательское действие на страницу записи**:

1. Создайте страницу записи или замещающую страницу записи. Для создания страницы записи выполните шаги 1-3 [инструкции по созданию страницы записи](#).
2. Переопределите защищенный виртуальный метод `getActions()`, который возвращает перечень действий страницы. Перечень действий страницы является экземпляром класса `Terrasoft.BaseViewModelCollection`. Каждое действие — это модель представления.
3. В метод `addItem()` в качестве параметра передайте callback-метод `getButtonMenuItem()`. Метод `addItem()` добавляет в коллекцию пользовательское действие.
4. В callback-метод `getButtonMenuItem()` в качестве параметра передайте конфигурационный объект. Метод `getButtonMenuItem()` создает экземпляр модели представления действия.
5. Реализуйте конфигурационный объект действия, который позволяет явно задать свойства модели представления действий или использовать базовый механизм привязки.

Важно. При замещении базовой страницы записи в методе `getActions()` схемы замещающей модели представления вызовите метод `this.callParent(arguments)`, который возвращает коллекцию действий родительской страницы записи.

Шаблон добавления пользовательского действия на страницу записи приведен ниже.

Шаблон добавления пользовательского действия на страницу записи

```
/**
 * Возвращает коллекцию действий страницы записи.
 * @protected
 * @virtual
 * @return {Terrasoft.BaseViewModelCollection} Возвращает коллекцию действий страницы записи.
 */
getActions: function() {
    /* Список действий — экземпляр Terrasoft.BaseViewModelCollection. */
    var actionMenuItems = this.Ext.create("Terrasoft.BaseViewModelCollection");
    /* Добавление действия в коллекцию. В качестве callback-метода передается метод, который инстанцирует конфигуратор действия. */
    actionMenuItems.addItem(this.getButtonMenuItem({
        /* Конфигурационный объект настройки действия. */
        ...
    }));
}
```

```

});  

/* Возвращает новую коллекцию действий. */  

return actionMenuItems;  

}

```

Добавить действие на страницу записи

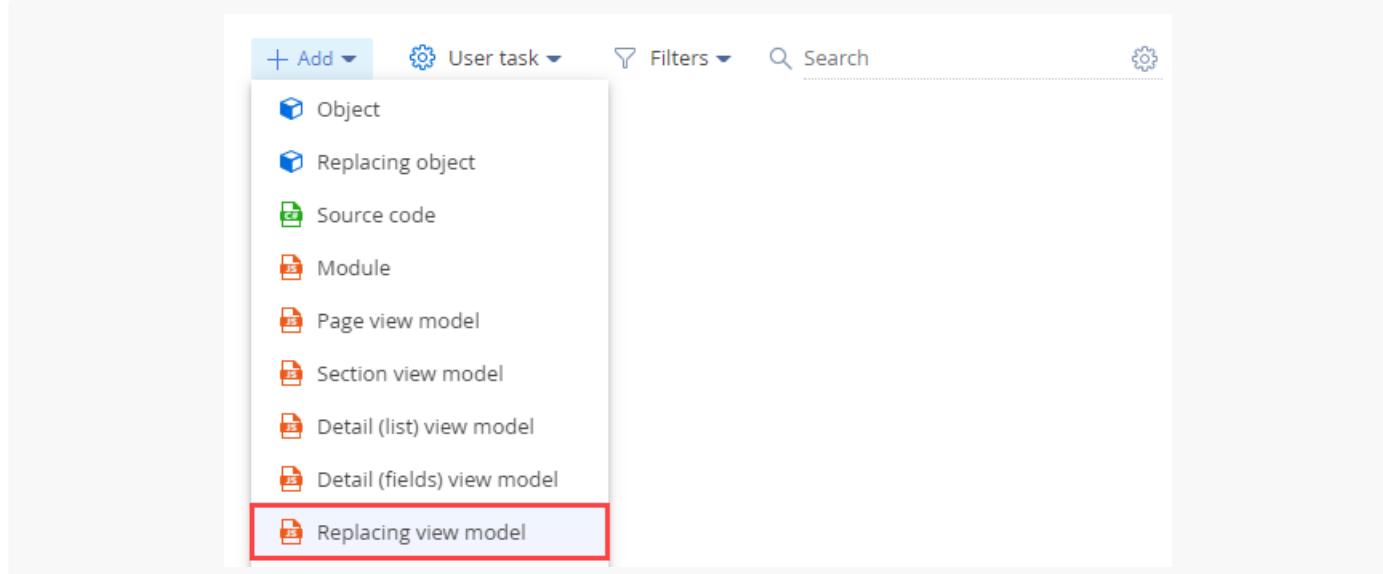


Пример реализован для продуктов линейки Sales Creatio.

Пример. На страницу заказа добавить действие [Показать дату выполнения] ([*Show execution date*]), которое в информационном окне отображает планируемую дату выполнения заказа. Действие активно для заказов, которые находятся на стадии [Исполнение] ([*In progress*]).

1. Создать схему замещающей модели представления страницы заказа

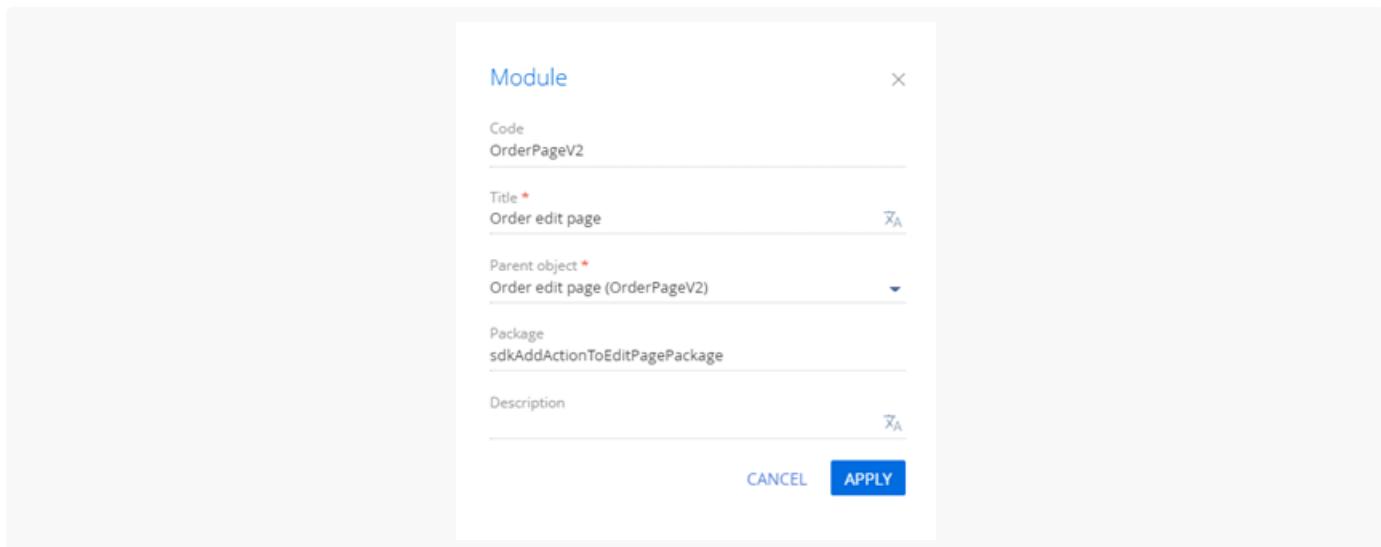
- Перейдите в раздел [Конфигурация] ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [Добавить] —> [Замещающая модель представления] ([*Add*] —> [*Replacing view model*]).



3. Заполните **свойства схемы**.

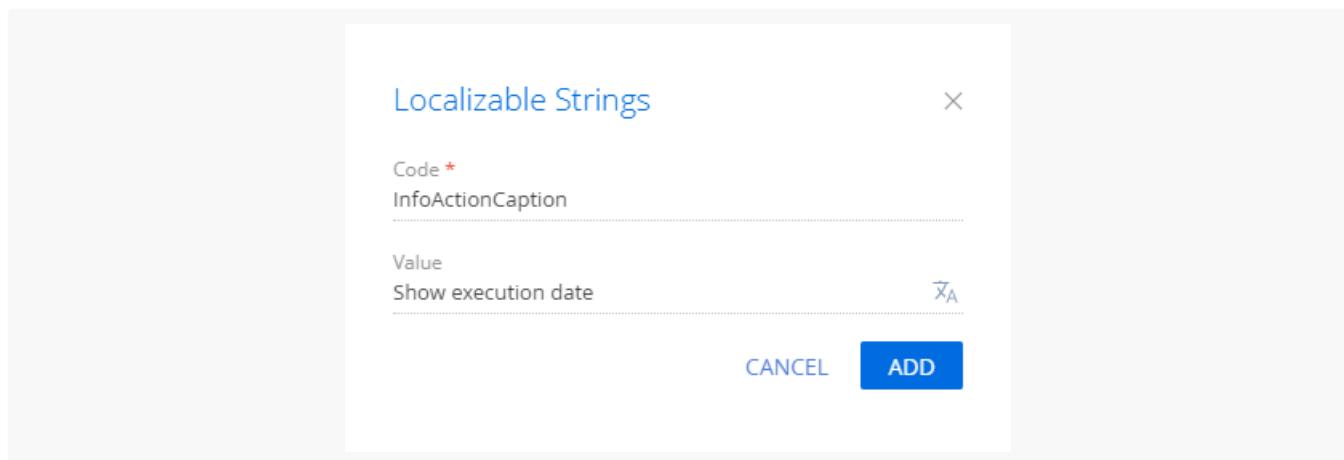
- [Код] ([*Code*]) — "OrderPageV2".
- [Заголовок] ([*Title*]) — "Страница редактирования заказа" ("Order edit page").

- [Родительский объект] ([Parent object]) — выберите "OrderPageV2".



4. Добавьте **локализуемую строку** с текстом пункта меню, который планируется добавить.

- В контекстном меню узла [Локализуемые строки] ([Localizable strings]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
 - [Код] ([Code]) — "InfoActionCaption".
 - [Значение] ([Value]) — "Показать дату выполнения" ("Show execution date").



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

5. Реализуйте **логику работы пункта меню**.

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

- `isRunning()` — проверяет находится ли заказ на стадии [Исполнение] ([In progress]) и определяет доступность добавленного пункта.
- `showOrderInfo()` — метод-обработчик действия. В информационном окне отображает планируемую дату завершения заказа. Действие страницы записи применяется к конкретному объекту, который открыт на странице. Для доступа к значениям полей объекта страницы записи в методе-

обработчики действия необходимо использовать методы модели представления `get()` (получить значение) и `set()` (установить значение).

- `getActions()` — переопределенный базовый метод. Возвращает коллекцию действий замещающей страницы.

Исходный код схемы замещающей модели представления страницы заказа представлен ниже.

OrderPageV2

```
define("OrderPageV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstants) {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Order",
        /* Методы модели представления страницы записи. */
        methods: {
            /* Проверяет стадию заказа для определения доступности пункта меню. */
            isRunning: function() {
                /* Метод возвращает true, если статус заказа [Исполнение], иначе возвращает false. */
                if (this.get("Status")) {
                    return this.get("Status").value === OrderConfigurationConstants.Order.OrderStatuses.InProgress;
                }
                return false;
            },
            /* Метод-обработчик действия. В информационном окне отображает дату выполнения заказа. */
            showOrderInfo: function() {
                /* Получает дату выполнения заказа. */
                var dueDate = this.get("DueDate");
                /* Отображает информационное окно. */
                this.showInformationDialog(dueDate);
            },
            /* Переопределяет базовый виртуальный метод, который возвращает коллекцию действий. */
            getActions: function() {
                /* Вызывается родительская реализация метода для получения коллекции прориниций. */
                var actionMenuItems = this.callParent(arguments);
                /* Добавляет линию-разделитель. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                /* Добавляет пункт меню в список действий страницы записи. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    /* Привязывает заголовок пункта меню к локализуемой строке схемы. */
                    "Caption": {bindTo: "Resources.Strings.InfoActionCaption"},
                    /* Привязывает метод-обработчик действия. */
                    "Tag": "showOrderInfo",
                    /* Привязывает свойство доступности пункта меню к значению, которое возвращается методом isRunning. */
                    "Enabled": {bindTo: "isRunning"}
                }));
            }
        }
    };
});
```

```

        return actionMenuItems;
    }
}
);
});

```

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

В результате на страницу заказа, который находится на стадии [Исполнение] ([In progress]), добавлено действие [Показать дату выполнения] ([Show execution date]).

The screenshot shows the 'ORD-1' order details page in the Creatio application. The top navigation bar includes 'What can I do for you?' and the 'Creatio 7.18.4.1532' logo. The main content area displays the order summary with fields like 'Total, \$' (2,725.00) and 'Payment amount, \$' (0.00). Below the summary, there are tabs for 'SUMMARY', 'HISTORY', 'APPROVALS', and 'GENERAL INFORMATION'. A red box highlights the 'Actions' dropdown menu, which contains options such as 'Unfollow the feed', 'Set up access rights', 'Send for approval', 'New invoice based on this order', 'New contract based on order', and 'Show execution date'. The 'Show execution date' option is also highlighted with a red box. At the bottom, a table lists the order items with columns for Product, Price, Quantity, Unit of measure, Discount, %, and Total.

Product	Price	Quantity	Unit of measure	Discount, %	Total
Asus R9280X-DC2T-3GD5	450.00	3.000	pieces	0.00	1,350.00
CRM bundle	185.00	5.000	licenses	0.00	925.00
Windows 10 Pro	150.00	3.000	pieces	0.00	450.00

В режиме вертикального представления реестра пользовательское действие страницы не отображается.

The screenshot shows a web-based application interface for managing orders. On the left, there's a sidebar with a list of orders: ORD-1 (selected), ORD-2, and ORD-3. The main content area is for ORD-1, displaying various tabs like SUMMARY, HISTORY, APPROVALS, and GENERAL INFO. A prominent feature is a table showing product details with columns for Product, Price, Quantity, Unit of measur..., Discount, %, and Total. The 'Actions' dropdown menu is open, highlighting the 'New contract based on order' option.

Для корректного отображения действия страницы в схему замещающей модели представления раздела необходимо добавить:

- Локализуемую строку, которая содержит текст пункта меню.
- Метод, который определяет доступность пункта меню.

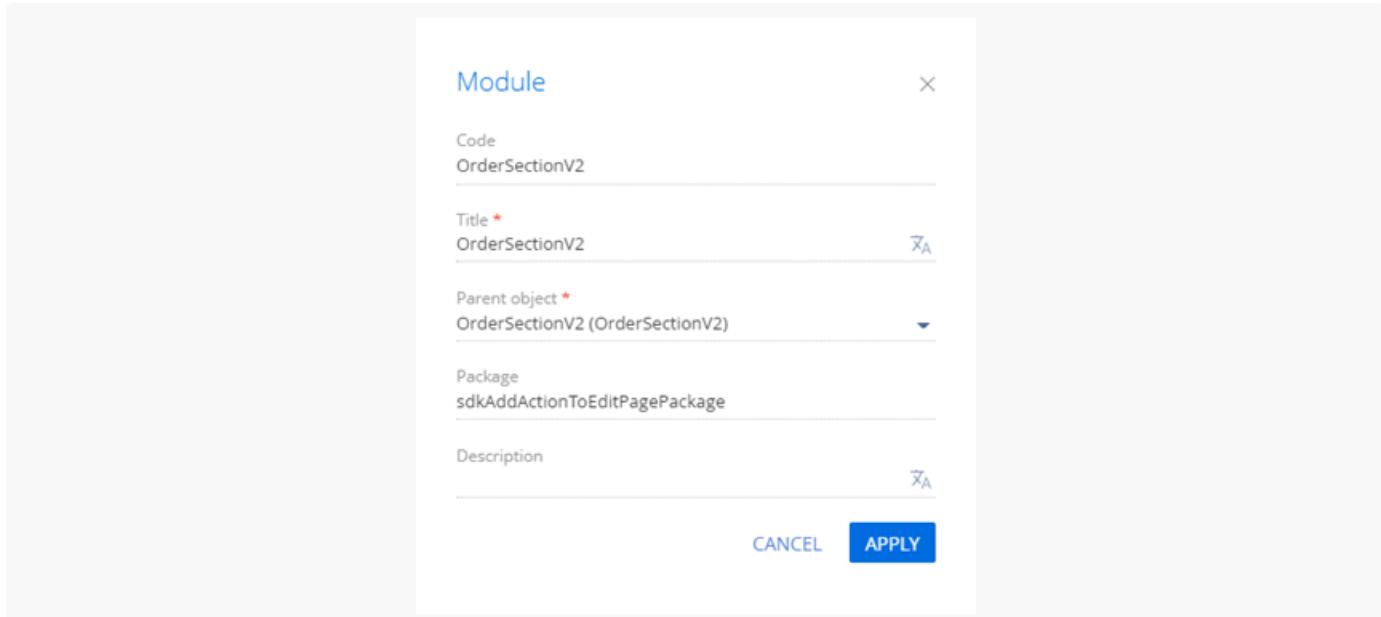
2. Создать схему замещающей модели представления раздела

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]).

The screenshot shows the 'User task' section of the configuration interface. A red box highlights the 'Replacing view model' option in the list of available models, which also includes Object, Replacing object, Source code, Module, Page view model, Section view model, Detail (list) view model, and Detail (fields) view model.

3. Заполните **свойства схемы**.

- [Код] ([*Code*]) — "OrderSectionV2".
- [Заголовок] ([*Title*]) — "Раздел заказа" ("Order section").
- [Родительский объект] ([*Parent object*]) — выберите "OrderSectionV2".



4. Добавьте **локализуемую строку** с текстом пункта меню, который планируется добавить. Для этого выполните шаг 4 [алгоритма создания схемы замещающей модели представления страницы заказа](#).
5. Реализуйте **логику работы пункта меню**. Для этого в свойстве `methods` реализуйте **метод** `isRunning()`, который проверяет находится ли заказ на стадии [*Исполнение*] ([*In progress*]) и определяет доступность добавленного пункта.

Исходный код схемы замещающей модели представления страницы раздела представлен ниже.

```
OrderSectionV2

define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
    return {
        /* Название схемы страницы раздела. */
        entitySchemaName: "Order",
        /* Методы модели представления страницы раздела. */
        methods: {
            /* Проверяет стадию заказа для определения доступности пункта меню. */
            isRunning: function(activeRowId) {
                activeRowId = this.get("ActiveRow");
                /* Получает коллекцию данных списочного представления реестра раздела. */
                var gridData = this.get("GridData");
                /* Получает модель выбранного заказа по заданному значению первичной колонки. */
                var selectedOrder = gridData.get(activeRowId);
                /* Получает свойства модели – статуса выбранного заказа. */
                var selectedOrderStatus = selectedOrder.get("Status");
            }
        }
    }
});
```

```

    /* Метод возвращает true, если статус заказа [Исполнение], иначе возвращает false
    return selectedOrderStatus.value === OrderConfigurationConstants.Order.OrderStatus.InProgress;
}
};

});
});

```

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

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

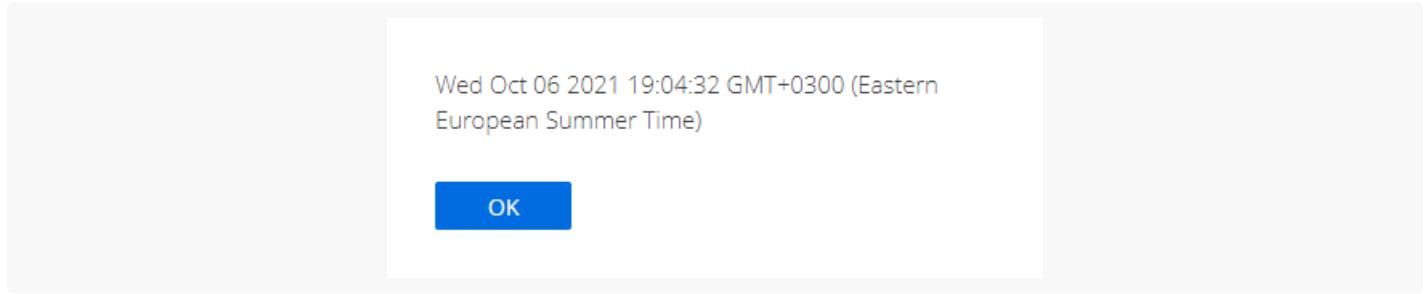
1. Очистите кэш браузера.
2. Обновите страницу раздела [Заказы] ([Orders]).

В результате выполнения примера на страницу заказа добавлено действие [Показать дату выполнения] ([Show execution date]).

Если заказ находится на стадии [Исполнение] ([In progress]), то действие [Показать дату выполнения] ([Show execution date]) активно.

Product	Price	Quantity	Unit of measur...	Discount, %	Total
Asus R9280X-DC2T-3GD5	450.00	3.000	pieces	0.00	1,350.00
CRM bundle	185.00	5.000	licenses	0.00	925.00
Windows 10 Pro	150.00	3.000	pieces	0.00	450.00

В результате выбора действия [Показать дату выполнения] ([Show execution date]), в информационном окне отображается планируемая дата выполнения заказа.



Если заказ не находится на стадии [Исполнение] ([In progress]), то действие [Показать дату выполнения] ([Show execution date]) неактивно.

The screenshot shows the Orders module interface. On the left, there's a list of orders: ORD-2 (Status: 4. Completed), ORD-1 (Status: 3. In progress), and ORD-30. The ORD-2 card is open, displaying its details. In the 'Actions' dropdown menu, the 'Show execution date' option is highlighted with a red box. The menu also contains other options like 'Unfollow the feed', 'Set up access rights', 'Send for approval', 'New invoice based on this order', and 'New contract based on order'.

ORD-2	
Total	2,233.30
Status	4. Completed
ORD-1	
Total	2,725.00
Status	3. In progress
ORD-30	
Account	Apex Solutions

ORD-2

SUMMARY HISTORY APPROVALS GENERAL INFORMATION ATTACHMENTS AND NC >

Product	Price	Quantity	Unit of measure	Discount, %	Total
Microsoft Office English	100.00	10.000	pieces	0.00	1,000.00
Installing software	100.00	8.000	hours	0.00	800.00
Gamepad Logitech F310 Gamepad	43.33	10.000	pieces	0.00	433.30

Схема BasePageV2



BasePageV2 — базовая схема карточки. Реализована в пакете `nui`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#).

Сообщения

Сообщения базовой карточки

Название	Режим	Направление	Описание
UpdatePage HeaderCaption	Адресное	Публикация	Обновляет заголовок страницы.
GridRowChanged	Адресное	Подписка	Получает идентификатор выбранной в реестре записи при ее изменении.
UpdateCard Property	Адресное	Подписка	Изменяет значение параметра модели.

<code>UpdateCardHeader</code>	Адресное	Подписка	Обновляет заголовок карточки.
<code>CloseCard</code>	Адресное	Публикация	Закрывает карточку.
<code>OpenCard</code>	Адресное	Подписка	Открывает карточку.
<code>OpenCardInChain</code>	Адресное	Публикация	Открывает цепочку карточек.
<code>GetCardState</code>	Адресное	Подписка	Возвращает состояние карточки.
<code>IsCardChanged</code>	Адресное	Публикация	Сообщает об изменении карточки.
<code>GetActiveViewName</code>	Адресное	Публикация	Получает имя активного представления.
<code>GetMiniPageMasterEntityInfo</code>	Адресное	Подписка	Возвращает информацию об основной сущности мини-карточки.
<code>GetPageTips</code>	Адресное	Подписка	Возвращает подсказки страницы.
<code>GetColumnInfo</code>	Адресное	Подписка	Возвращает информацию колонки.
<code>GetEntityColumnChanges</code>	Широковещательное	Публикация	Отправляет информацию колонки сущности при ее изменении.
<code>ReloadSectionRow</code>	Адресное	Публикация	Перезагружает строку раздела в соответствии со значением основного столбца.
<code>ValidateCard</code>	Адресное	Подписка	Запускает проверку валидности карточки.
<code>ReInitializeActionsDashboard</code>	Адресное	Публикация	Запускает повторную инициализацию панели действий.
<code>ReInitializeActions</code>	Адресное	Подписка	Обновляет конфиг панели действий.

Dashboard			
ReloadDashboardItems	Широковещательное	Публикация	Перезагружает элементы дашбордов.
ReloadDashboardItemsPTP	Адресное	Публикация	Перезагружает элементы дашбордов для текущей страницы.
CanChangeHistoryState	Широковещательное	Подписка	Разрешает или запрещает изменение текущего состояния истории.
IsEntityChanged	Адресное	Подписка	Возвращает измененную сущность.
IsDcmFilterColumnChanged	Адресное	Подписка	Возвращает <code>true</code> , если изменены отфильтрованные колонки.
UpdateParentLookupDisplayValue	Широковещательное	Двунаправленное	Обновляет значение родительской записи справочника по конфигу.
UpdateParentLookupDisplayValue	Широковещательное	Двунаправленное	Указывает на необходимость перезагрузки данных при следующем запуске.

Режимы сообщений представлены перечислением `Terrasoft.core.enums.MessageMode`, а направления сообщений — перечислением `Terrasoft.core.enums.MessageDirectionType`. Перечисление `MessageMode` описано в [Библиотеке JS классов](#). Перечисление `MessageDirectionType` описано в [Библиотеке JS классов](#).

Атрибуты

`IsLeftModulesContainerVisible` BOOLEAN

Признак видимости контейнера `LeftModulesContainer`.

`IsActionDashboardContainerVisible` BOOLEAN

Признак видимости контейнера `ActionDashboardContainer`.

`HasActiveDcm` BOOLEAN

Признак видимости контейнера `DcmActionsDashboardContainer`.

`ActionsDashboardAttributes` `CUSTOM_OBJECT`

Пользовательские атрибуты контейнера `DcmActionsDashboardContainer`.

`IsPageHeaderVisible` `BOOLEAN`

Флаг видимости заголовка страницы.

`ActiveTabName` `TEXT`

Сохранить имя активной вкладки.

`GridDataViewName` `TEXT`

Имя представления `GridData`.

`AnalyticsDataViewName` `TEXT`

Имя представления `AnalyticsData`.

`IsCardOpenedAttribute` `STRING`

Атрибут тела карточки, когда карточки отображена или скрыта.

`IsMainHeaderVisibleAttribute` `STRING`

Атрибут тела карточки, когда основной заголовок отображен или скрыт.

`PageHeaderColumnNames` `CUSTOM_OBJECT`

Массив имен колонок заголовка страницы.

`IsNotAvailable` `BOOLEAN`

Признак недоступности страницы.

`CanCustomize` `BOOLEAN`

Признак возможности кастомизации страницы.

`Operation ENUM`

Операции карточки.

`EntityReloadScheduled BOOLEAN`

Признак необходимости перезагрузки сущности при следующем запуске.

Типы данных атрибутов представлены перечислением `Terrasoft.core.enums.DataValueType`.

Перечисление `DataValueType` описано в [Библиотеке JS классов](#).

Методы

`onDiscardChangesClick(callback, scope)`

Обрабатывает нажатие кнопки [Отменить] ([*Discard*]).

Параметры

<code>{String} [callback]</code>	Функция обратного вызова.
<code>{Terrasoft.BaseViewModel} [scope]</code>	Контекст выполнения метода.

`addChange DataViewOptions(viewOptions)`

Добавляет представления точек переключения в выпадающий список кнопки [Вид] ([*View*]).

Параметры

<code>{Terrasoft.BaseViewModelCollection} viewOptions</code>	Пункты выпадающего списка кнопки [Вид] ([<i>View</i>]).
--	---

`addSectionDesignerViewOptions(viewOptions)`

Добавляет пункт [Открыть мастер раздела] ([*Open section wizard*]) в выпадающий список кнопки [Вид] ([*View*]).

Параметры

<code>{Terrasoft.BaseViewModelCollection} viewOptions</code>	Пункты выпадающего списка кнопки [Вид] ([<i>View</i>]).
--	---

`getReportFilters()`

Возвращает коллекцию фильтров для запроса.

`initPageHeaderColumnNames()`

Инициализировать имена колонок заголовка страницы.

`getParameters(parameters)`

Возвращает значения параметров `ViewModel`.

Параметры

<code>{Array} parameters</code>	Имена параметров.
---------------------------------	-------------------

`setParameters(parameters)`

Задает параметры `ViewModel`.

Параметры

<code>{Object} parameters</code>	Значения параметров.
----------------------------------	----------------------

`getLookupModuleId()`

Возвращает идентификатор модуля страницы справочника.

`onReloadCard(defaultValues)`

Обработчик сообщения `ReloadCard`. Перезагружает данные сущности карточки.

Параметры

<code>{Object[]} defaultValues</code>	Массив значений по умолчанию.
---------------------------------------	-------------------------------

`onGetColumnInfo(columnName)`

Возвращает информацию колонки.

Параметры

<code>{String} columnName</code>	Имя колонки.
----------------------------------	--------------

`getTabsContainerVisible()`

Возвращает статус видимости вкладок контейнера.

`getPrintMenuItemVisible(reportId)`

Возвращает статус видимости пунктов выпадающего списка (т. е. отчетов) кнопки [Печать] ([Print]).

Параметры

{String} reportId	Идентификатор отчета.
-------------------	-----------------------

`getDataViews()`

Получает представление раздела.

`runProcess(tag)`

Запустить бизнес-процесс с помощью кнопки запуска глобальных бизнес-процессов.

Параметры

{Object} tag	Идентификатор схемы бизнес-процесса.
--------------	--------------------------------------

`runProcessWithParameters(config)`

Запустить бизнес-процесс с параметрами.

Параметры

{Object} config	Конфигурационный объект.
-----------------	--------------------------

`onCanBeDestroyed(cacheKey)`

Проверяет наличие несохраненных данных. Измените `config.result` из кэша, если данные изменены, но не сохранены.

Параметры

{String} cacheKey	Ключ конфигурационного объекта в кэше.
-------------------	--

```
onValidateCard()
```

Отображается сообщение о некорректности, если карточка невалидна.

Схема BaseEntityPage js



Основы

`BaseEntityPage` — базовая схема страницы записи. Реализована в пакете `NUI`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Все схемы страниц записей должны наследовать схему `BaseEntityPage`.

Сообщения

Сообщения базовой страницы записи

Название	Режим	Направление	Описание
<code>UpdateDetail</code>	Адресное	Публикация	Сообщает детали изменения карточки.
<code>CardModuleResponse</code>	Адресное	Двунаправленное	Сообщает об изменении страницы записи.
<code>CardRendered</code>	Широковещательное	Двунаправленное	Сообщает об обработке страницы записи.
<code>EntityInitialized</code>	Широковещательное	Двунаправленное	Сообщает о выполнении инициализации объекта и отправляет информацию об объекте.
<code>ShowProcessPage</code>	Адресное	Публикация	Отображает страницу процесса.

Режимы сообщений представлены перечислением `Terrasoft.core.enums.MessageMode`, а направления сообщений — перечислением `Terrasoft.core.enums.MessageDirectionType`. Перечисление `MessageMode` описано в [Библиотеке JS классов](#). Перечисление `MessageDirectionType` описано в [Библиотеке JS классов](#).

Атрибуты

`IsEntityInitialized` BOOLEAN

Используется для подготовки сущности.

DefaultValues ARRAY

Массив значений по умолчанию для объекта.

IsModelItemsEnabled BOOLEAN

Признак доступности элементов модели.

DetailsConfig CUSTOM_OBJECT

Детали конфигурации.

Типы данных атрибутов представлены перечислением `Terrasoft.core.enums.DataValueType`.Перечисление `DataValueType` описано в [Библиотеке JS классов](#).

Миксины

EntityResponseValidationMixin `Terrasoft.EntityResponseValidationMixin`Проверяет ответ сервера. Если `false`, то генерирует сообщение об ошибке и запускает диалог.

Методы

init(callback, scope)

Инициализирует страницу записи.

Параметры

<code>{Function} callback</code>	Функция обратного вызова.
<code>{Object} scope</code>	Контекст выполнения метода.

saveDetailsInChain(next)

Сохраняет детали в цепочку.

Параметры

<code>{Function} next</code>	Функция обратного вызова.
------------------------------	---------------------------

getDetailId(detailName)

Возвращает идентификатор детали.

Параметры

{String} detailName	Имя детали.
---------------------	-------------

`loadDetail(config)`

Загружает деталь. Если деталь загружена, то повторно ее отображает.

Параметры

{Object} config	Конфигурация детали.
-----------------	----------------------

`saveCheckCanEditRight(callback, scope)`

Проверяет наличие прав пользователя на редактирование.

Параметры

{Function} callback	Функция обратного вызова.
{Object} scope	Контекст выполнения метода.

`getLookupDisplayValueQuery(config)`

Формирует поисковый запрос для отображаемого значения.

Параметры

{Object} config	Конфигурация детали.
-----------------	----------------------

`loadLookupDisplayValue(name, value, callback, scope)`

Заполняет справочные поля.

Параметры

{String} name	Имя схемы объекта.
{String} value	Значение объекта.
{Function} callback	Функция обратного вызова.
{Object} scope	Контекст выполнения метода.

canAutoCleanDependentColumns()

Возвращает `true`, если бизнес-правило может очищать колонку объекта.

Панель действий



Основы

Панель действий предназначена для отображения информации о текущем состоянии работы с записью. Она состоит из двух частей:

- **Индикатор стадий** (1) — показывает состояние этапов бизнес-процесса в тех разделах, где работа с записями выполняется с использованием бизнес-процесса.
- **Панель действий** (2):
 - Позволяет перейти к выполнению активности, работе с email-сообщениями или с лентой, не покидая раздел.
 - Отображает созданные по бизнес-процессу активности, которые находятся в неконечном состоянии и связаны с объектом раздела по соответствующему полю.
 - Может отображать автогенерируемую страницу, преднастроенную страницу, вопрос и страницу объекта в виде задач.

The screenshot shows the Creatio application interface. On the left, there is a vertical sidebar with various icons and a list of properties for a record: Customer need*, Additional service, Registration method, Landing page, Created on 3/20/2016 at 2:57 PM, Account Alpha Business, Web www.alphabusiness.co.uk, and Industry. The main area is divided into two main sections: 'ActionDashboardContainer' (top) and 'ContentContainer' (bottom). The 'ActionDashboardContainer' contains a green header bar with 'Distribution' and 'NEXT STEPS (0)' buttons, along with icons for phone, email, messages, and file. The 'ContentContainer' below it has tabs for 'LEAD INFO', 'CUSTOMER NEED DETAILS', 'HISTORY', 'ATTACHMENTS AND NOTES', and 'FEED'. Under 'LEAD INFO', there are fields for Contact name (Jordan), Account name (AlphaBusiness), Web (www.alphabusiness.com), Job title, Mobile phone, Email, and Country (United Kingdom). At the bottom of the ContentContainer, there is a link to 'Similar leads...'. On the right side of the interface, there are several circular icons for different actions like phone, email, messages, and notifications, with some showing a count of 22.

Панель действий расположена в контейнере `ActionDashboardContainer` страницы записи раздела. Индикатор стадий расположен во вложенном контейнере `HeaderContainer`, а панель действий — в `ContentContainer`.

Расположение элементов панели действий конфигурируется схемой модели представления `BaseActionsDashboard` и унаследованной схемой `SectionActionsDashboard` пакета `ActionsDashboard`.

Добавить панель действий на страницу

1. Создайте схему модели представления, унаследованную от `SectionActionsDashboard`.
2. Создайте схему замещающей модели представления страницы.
3. В свойстве `modules` схемы замещающей модели представления страницы выполните настройку модуля.
4. В свойстве `diff` схемы замещающей модели представления страницы добавьте модуль на страницу.

Добавить новый канал на панель действий

Каналы в панели действий — это способ коммуникации с контактом. Канал создается для каждого раздела, в котором он подключен, например, для обращения, контакта или лида.

Чтобы **добавить новый канал на панель действий**:

1. Создайте класс-наследник базового класса `BaseMessagePublisher`.
2. Создайте схему замещающей модели представления `SectionActionsDashboard`.
3. В свойстве `diff` схемы замещающей модели представления укажите операцию `insert` для вставки вкладки `CallsMessageTab` и контейнера сообщений, а также укажите модуль, который будет отрисовываться в данном канале на одной из вкладок.
4. Переопределите методы:
 - `getSectionPublishers()` — добавляет созданный канал в список издателей сообщений.
 - `getExtendedConfig()` — определяет параметры вкладки.
5. Создайте модуль-контейнер для прорисовки в панели действий страницы, в которой будет реализована логика добавляемого канала.
6. Создайте схему модели представления, в которой будет реализована логика канала. В качестве родительской схемы установите `BaseMessagePublisherPage`.

Добавить пользовательское действие верификации

Элемент процесса [*Действие верификации*] используется в продуктах линейки *Financial Services Creation* при верификации заявки сотрудником компании. С его помощью можно создать проверку данных в **кредитной заявке** — набор необходимых действий верификации, которые должен провести ответственный сотрудник. При помощи этого элемента можно реализовать процесс принятия решения по кредитной заявке. От результата выполнения действия верификации зависит дальнейшее ветвление бизнес-процесса.

При создании настраиваемой страницы действия верификации, например, в бизнес-процессе [Подтверждение заявки] ([Approve application]), есть возможность выбрать заранее созданную страницу с названием [Преднастроенная страница верификации] ([Preconfigured verification page]).

The screenshot shows the configuration of a validation item for an application approval process. On the left, there's a process diagram with nodes like 'Validation item: Approve loan issuance', 'Change application data', 'Add contract', 'Add contract parameters', 'Read contract id', and 'Connect contract with application'. A red box highlights the 'Execute on page*' dropdown, which is set to 'Preconfigured verification page'. On the right, a modal window titled 'Validation' shows the selected validation item: 'Approve loan issuance' for application '#Application#'. It also displays fields for 'How to perform validation?' (set to 'Application approval') and 'Who performs validation?' (set to 'Group of employees').

Сама страница отображается, например, после нажатия на кнопку [Завершить] ([Complete]) активности [Согласовать выдачу кредита] ([Approve loan issuance]), которая создается при переходе заявки на стадию [Верификация] ([Validation]).

The screenshot shows the verification page for loan issuance approval. At the top, it lists the steps: Product selection, Filling in the ..., Validation, Settlement, and Closed successfully. Below that, it shows 'NEXT STEPS (1)' with icons for phone, email, message, and flag. A main box contains the action 'Approve loan issuance' with a sub-section also labeled 'Approve loan issuance'. A large green 'COMPLETE' button is visible on the right.

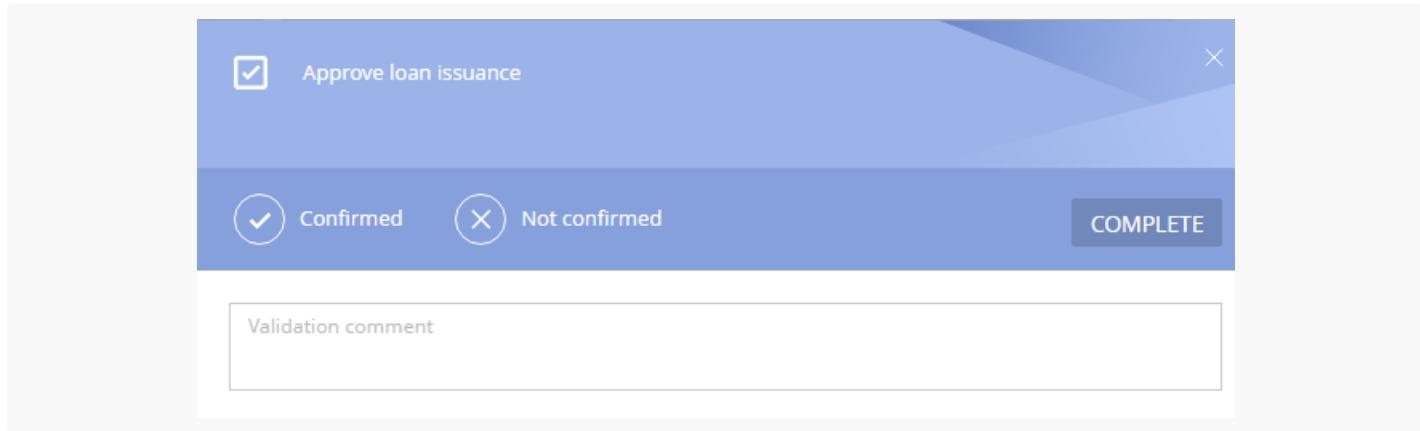
Структурные элементы преднастроенной страницы верификации:

- Кнопки выбора результата выполнения действия верификации.
- Поле [Комментарий] — содержит комментарий к действию верификации.
- Деталь [Сценарий разговора] — содержит скрипт-подсказку для верификатора. Доступна только в режиме чтения.
- Деталь [Файлы и ссылки] — содержит прикрепленные к действию верификации файлы и ссылки.

Доступна только в режиме чтения.

- Деталь [Результаты проверок] — содержит контрольные вопросы и ответы на них.

Важно. Если деталь не содержит прикрепленные данные, то она не отображается на странице.



Creatio предоставляет возможность создавать пользовательские страницы верификации, наследующие преднастроенную.

Чтобы создать **пользовательскую страницу верификации**:

1. Создайте клиентскую схему страницы действия верификации.
2. Используйте созданную схему в бизнес-процессе.

Добавить панель действий

 Средний

Описание примера

Добавить инструментальную панель действий на страницу заказа.

Исходный код

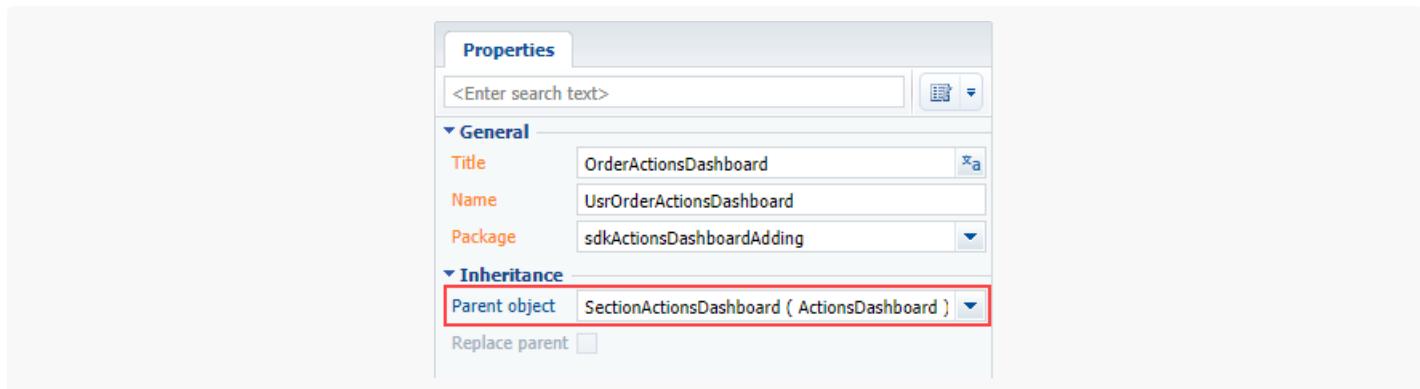
Пакет с реализацией примера можно скачать по [ссылке](#).

Алгоритм реализации примера

1. Создать схему модели представления OrderActionsDashboard

В качестве родительского объекта укажите схему `SectionActionsDashboard` (рис. 1).

Рис. 1. — Свойства схемы модели представления



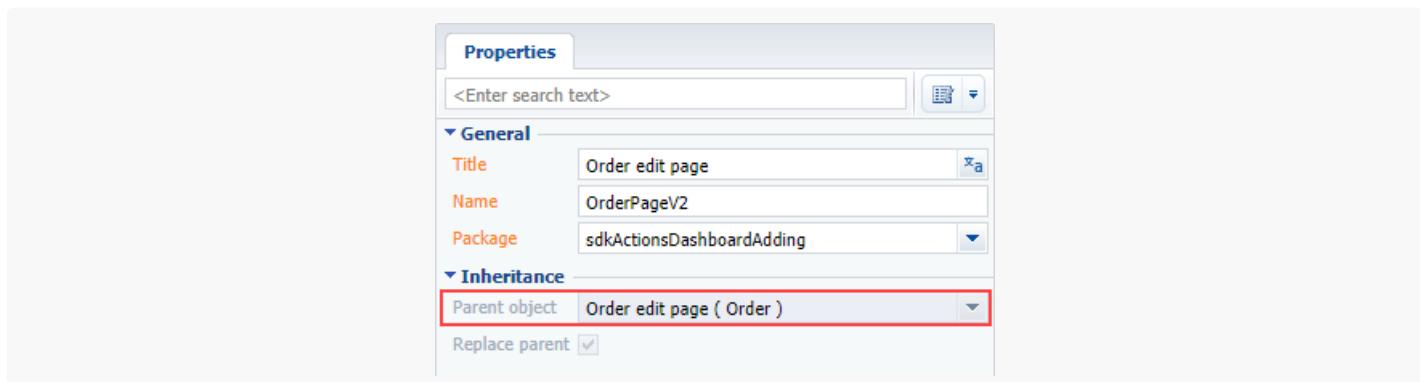
Исходный код схемы модели представления:

```
define("UsrOrderActionsDashboard", [], function () {
    return {
        details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
        methods: {},
        diff: /**SCHEMA_DIFF*/[]/**SCHEMA_DIFF*/
    };
});
```

2. Создать замещающую страницу заказа

Создайте схему замещающей модели представления, в которой в качестве родительского объекта укажите [Страница редактирования заказа] ([Order edit page], OrderPageV2) (рис. 2). Процесс создания схемы замещающей модели представления описан в статье "[Создать клиентскую схему](#)".

Рис. 2. — Свойства схемы замещающей модели представления страницы записи



3. В коллекцию modules схемы страницы добавить конфигурационный объект с настройками модуля

На вкладку исходного кода добавьте код замещающего модуля страницы. В нем в коллекцию modules модели представления добавьте конфигурационный объект с настройками модуля.

4. В массив diff добавить конфигурационный объект с настройками расположения модуля на странице

Исходный код схемы замещающей модели представления:

```
define("OrderPageV2", [],
  function () {
    return {
      entitySchemaName: "Order",
      attributes: {},
      modules: /**SCHEMA_MODULES*/{
        "ActionsDashboardModule": {
          "config": {
            "isSchemaConfigInitialized": true,
            // Имя схемы.
            "schemaName": "UsrOrderActionsDashboard",
            "useHistoryState": false,
            "parameters": {
              // Конфигурационный объект модели представления.
              "viewModelConfig": {
                // Имя сущности страницы.
                "entitySchemaName": "Order",
                // Конфигурационный объект блока Actions.
                "actionsConfig": {
                  // Имя схемы для загрузки элементов в Actions.
                  "schemaName": "OrderStatus",
                  // Имя колонки в родительской схеме, ссылающейся на схему, с
                  // Если не указана, берет значение равное schemaName.
                  "columnName": "Status",
                  // Имя колонки для сортировки элементов.
                  "orderColumnName": "Position",
                  // Имя колонки для сортировки элементов в меню элемента.
                  "innerOrderColumnName": "Position"
                },
                // Отвечает за отображение модуля панели действий, значение [true]
                "useDashboard": true,
                // Отвечает за отображение блока Content, значение [true] по умолчанию.
                "contentVisible": true,
                // Отвечает за отображение блока Header, значение [true] по умолчанию.
                "headerVisible": true,
                // Конфигурационный объект элементов панели.
                "dashboardConfig": {
                  // Связь активностей с объектом страницы.
                  "Activity": {
                    // Имя колонки объекта страницы.
                    "masterColumnName": "Id",
                    // Имя колонки в объекте [Activity].
                    "referenceColumnName": "Order"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
)
```

```
        }
    }
}
}

}/**SCHEMA_MODULES*/,
details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
methods: {},
diff: /**SCHEMA_DIFF*/[
{
    "operation": "insert",
    "name": "ActionsDashboardModule",
    "parentName": "ActionDashboardContainer",
    "propertyName": "items",
    "values": {
        "classes": { wrapClassName: ["actions-dashboard-module"] },
        "itemType": Terrasoft.ViewItemType.MODULE
    }
}
]/**SCHEMA_DIFF*/
};

});
});
```

После сохранения схемы и обновления страницы приложения на странице заказа появится инструментальная панель действий, которая будет отображать состояние заказа, а также связанные с ним незавершенные активности (рис. 3).

Рис. 3. — Демонстрация результата выполнения примера

ORD-41

What can I do for you? >

Creatio

CLOSE ACTIONS > VIEW >

1. Draft > 2. Confirmation > 3. In progress > 4. Completed > 5. Canceled

NEXT STEPS (2)

Prepare documents for customer
5/9/2018 | Supervisor

Call to customer
5/9/2018 | Supervisor

Customer* IT-Plus
Status 1. Draft
Total, \$ 5,389.81
Payment amount, \$ 5,389.81

PRODUCTS ORDER DETAILS DELIVERY SUMMARY HISTORY GENERAL INFORMATION APPROV >

Products + : Items: 1 Total: \$ 5,389.81

Product	Price	Quantity	Unit of measure	Discount, %	Total

Добавить новый канал на панель действий

Сложный

Описание примера

Добавить новый пользовательский канал в инструментальную панель действий страницы контакта. Канал должен полностью повторять функциональность канала фиксации результатов звонка (канал `CallMessagePublisher`).

Исходный код

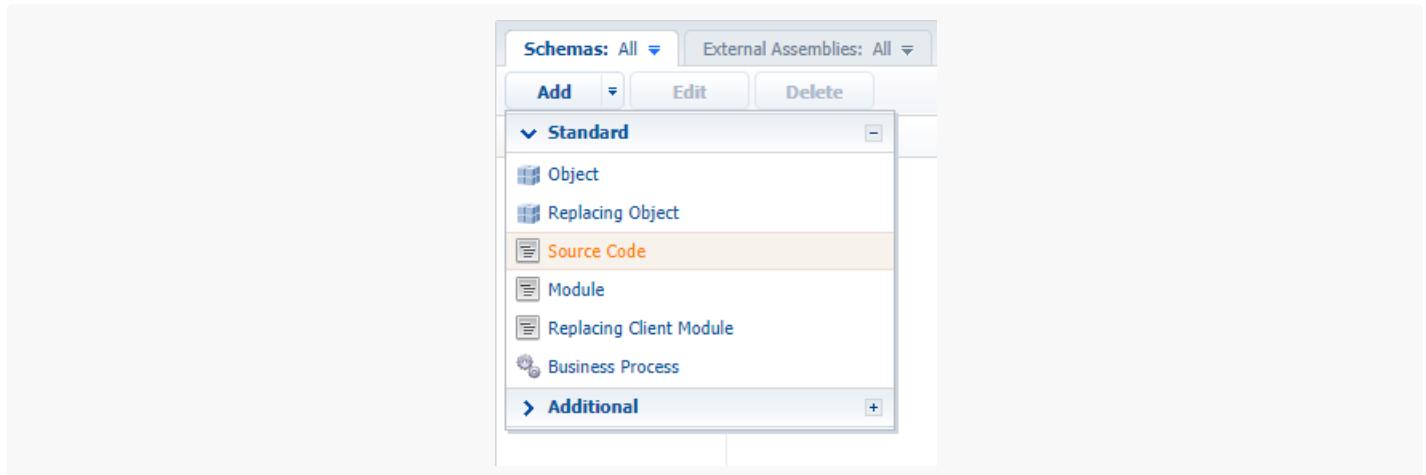
Пакет с реализацией примера можно скачать по [ссылке](#).

Алгоритм выполнения примера

1. Добавить схему исходного кода `UsrCallsMessagePublisher`

Для создания схемы исходного кода в разделе [Конфигурация] на вкладке [Схемы] выполните пункт меню [Добавить] — [Исходный код] (рис. 1).

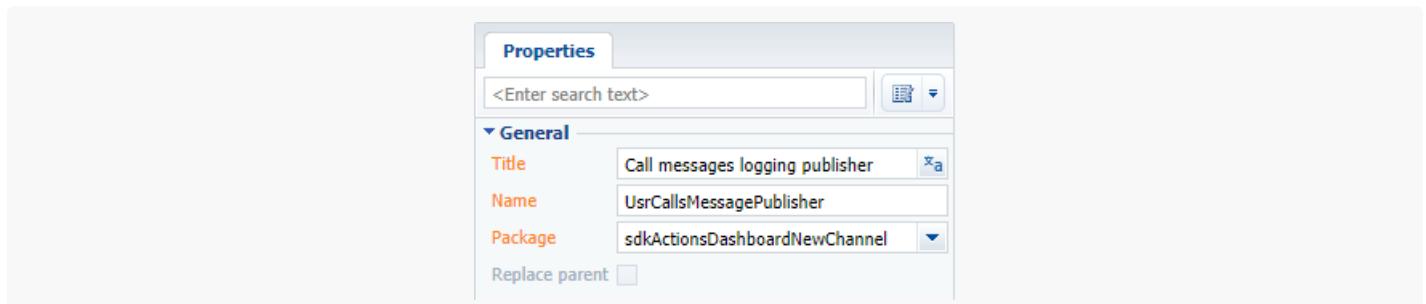
Рис. 1. — Добавление схемы исходного кода



Для созданной схемы укажите (рис. 2):

- [Заголовок] ([Title]) — "Издатель сообщений логирования звонка" (Call message logging publisher);
- [Название] ([Name]) — "UsrCallsMessagePublisher".

Рис. 2. — Свойства схемы исходного кода



В созданной схеме в пространстве имен `Terrasoft.Configuration` добавьте новый класс `CallsMessagePublisher`, наследуемый от класса `BaseMessagePublisher`. Класс `BaseMessagePublisher` содержит базовую логику сохранения объекта в базу данных и базовую логику обработчиков событий. Класс-наследник будет содержать логику для конкретного отправителя, например, заполнение колонок объекта `Activity` и последующую отправку сообщения.

Для реализации нового класса `CallsMessagePublisher` в созданную схему добавьте следующий исходный код:

```

using System.Collections.Generic;
using Terrasoft.Core;

namespace Terrasoft.Configuration
{
    // Класс-наследник BaseMessagePublisher.
  
```

```

public class CallsMessagePublisher : BaseMessagePublisher
{
    // Конструктор класса.
    public CallsMessagePublisher(UserConnection userConnection, Dictionary<string, string> entityFieldsData)
        : base(userConnection, entityFieldsData) {
        //Схема, с которой будет работать CallsMessagePublisher.
        EntitySchemaName = "Activity";
    }
}
}

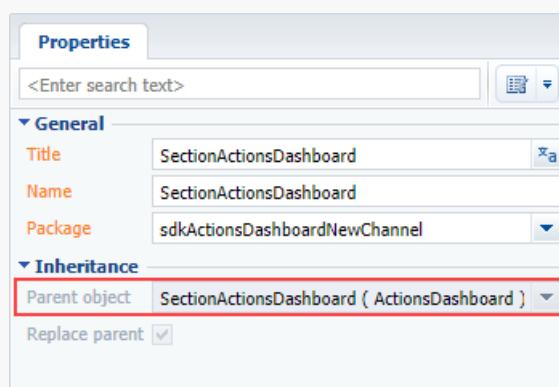
```

После этого сохраните и опубликуйте схему.

2. Создать схему замещающей модели представления SectionActionsDashboard

Создайте схему замещающей модели представления, в которой в качестве родительского объекта укажите `SectionActionsDashboard` (рис. 3). Процесс создания схемы замещающей модели представления описан в статье "[Создать клиентскую схему](#)".

Рис. 3. — Свойства схемы замещающей модели представления



На заметку.

Если нужно добавить канал только в одну страницу записи, то необходимо создать новый модуль с названием `имя_разделаSectionActionsDashboard` (например, `BooksSectionActionsDashboard`) и в качестве родительской схемы указать `SectionActionsDashboard` того раздела, в котором будет размещена страница записи.

В свойстве `diff` схемы замещающей модели представления установите операции вставки вкладки `CallsMessageTab` и контейнера сообщений, а также укажите модуль, который будет отрисовываться в данном канале на одной из вкладок. После этого новый канал будет виден на страницах записей тех разделов, в которых подключен `SectionActionsDashboard`.

В свойстве `methods` переопределите метод `getSectionPublishers()`, который добавит созданный канал в

список издателей сообщений, и метод `getExtendedConfig()`, в котором определяются параметры вкладки.

Чтобы метод `getExtendedConfig()` отработал корректно, загрузите изображение иконки канала и укажите ее в параметре `ImageSrc`. Файл изображения иконки, используемый в данном примере, можно скачать [здесь](#).

Переопределите метод `onGetRecordInfoForPublisher()` и добавьте метод `getContactEntityParameterValue()`, определяющие значение контакта из страницы записи раздела, в котором находится панель действий.

Исходный код схемы замещающей модели представления:

```
define("SectionActionsDashboard", ["SectionActionsDashboardResources", "UsrCallsMessagePublisher"]
function(resources) {
    return {
        attributes: {},
        messages: {},
        methods: {
            // Метод задает настройки отображения вкладки канала в панели действий.
            getExtendedConfig: function() {
                // Вызов родительского метода.
                var config = this.callParent(arguments);
                var lcziImages = resources.localizableImages;
                config.CallsMessageTab = {
                    // Изображение вкладки.
                    "ImageSrc": this.Terrasoft.ImageUrlBuilder.getUrl(lcziImages.CallsMessageTab),
                    // Значение маркера.
                    "MarkerValue": "calls-message-tab",
                    // Выравнивание.
                    "Align": this.Terrasoft.Align.RIGHT,
                    // Тэг.
                    "Tag": "UsrCalls"
                };
                return config;
            },
            // Переопределяет родительский и добавляет значение контакта из страницы записи
            // раздела, в котором находится панель действий.
            onGetRecordInfoForPublisher: function() {
                var info = this.callParent(arguments);
                info.additionalInfo.contact = this.getContactEntityParameterValue(info.relat
                return info;
            },
            // Определяет значение контакта из страницы записи раздела,
            // в котором находится панель действий.
            getContactEntityParameterValue: function(relationSchemaName) {
                var contact;
                if (relationSchemaName === "Contact") {
                    var id = this.getMasterEntityParameterValue("Id");
                    var name = this.getMasterEntityParameterValue("Name");
                    if (id & name) {
                        contact = {value: id, displayValue: name};
                    }
                }
                return contact;
            }
        }
    }
});
```

```

        }
    } else {
        contact = this.getMasterEntityParameterValue("Contact");
    }
    return contact;
},
//Добавляет созданный канал в список издателей сообщений.
getSectionPublishers: function() {
    var publishers = this.callParent(arguments);
    publishers.push("UsrCalls");
    return publishers;
}
},
// Массив модификаций, с помощью которых строится представление модуля в интерфейсе
diff: /**SCHEMA_DIFF*/[
    // Добавление вкладки CallsMessageTab.
    {
        // Тип операции – вставка.
        "operation": "insert",
        // Название вкладки.
        "name": "CallsMessageTab",
        // Название родительского элемента.
        "parentName": "Tabs",
        // Название свойства.
        "propertyName": "tabs",
        // Конфигурационный объект свойств.
        "values": {
            // Массив дочерних элементов.
            "items": []
        }
    },
    // Добавление контейнера сообщений.
    {
        "operation": "insert",
        "name": "CallsMessageTabContainer",
        "parentName": "CallsMessageTab",
        "propertyName": "items",
        "values": {
            // Тип элемента – контейнер.
            "itemType": this.Terrasoft.ViewItemType.CONTAINER,
            // CSS-класс для контейнера.
            "classes": {
                "wrapClassName": ["calls-message-content"]
            },
            "items": []
        }
    },
    // Добавление модуля UsrCallsMessageModule.
]

```

```

{
    "operation": "insert",
    "name": "UsrCallsMessageModule",
    "parentName": "CallsMessageTab",
    "propertyName": "items",
    "values": {
        // CSS-класс для модуля вкладок.
        "classes": {
            "wrapClassName": ["calls-message-module", "message-module"]
        },
        // Тип элемента – модуль.
        "itemType": this.Terrasoft.ViewItemType.MODULE,
        // Название модуля.
        "moduleName": "UsrCallsMessagePublisherModule",
        // Привязка метода, выполняемого после отрисовки элемента.
        "afterrender": {
            "bindTo": "onMessageModuleRendered"
        },
        // Привязка метода, выполняемого после перерисовки элемента.
        "afterrerender": {
            "bindTo": "onMessageModuleRendered"
        }
    }
}
/**SCHEMA_DIFF*/
};

}
);

```

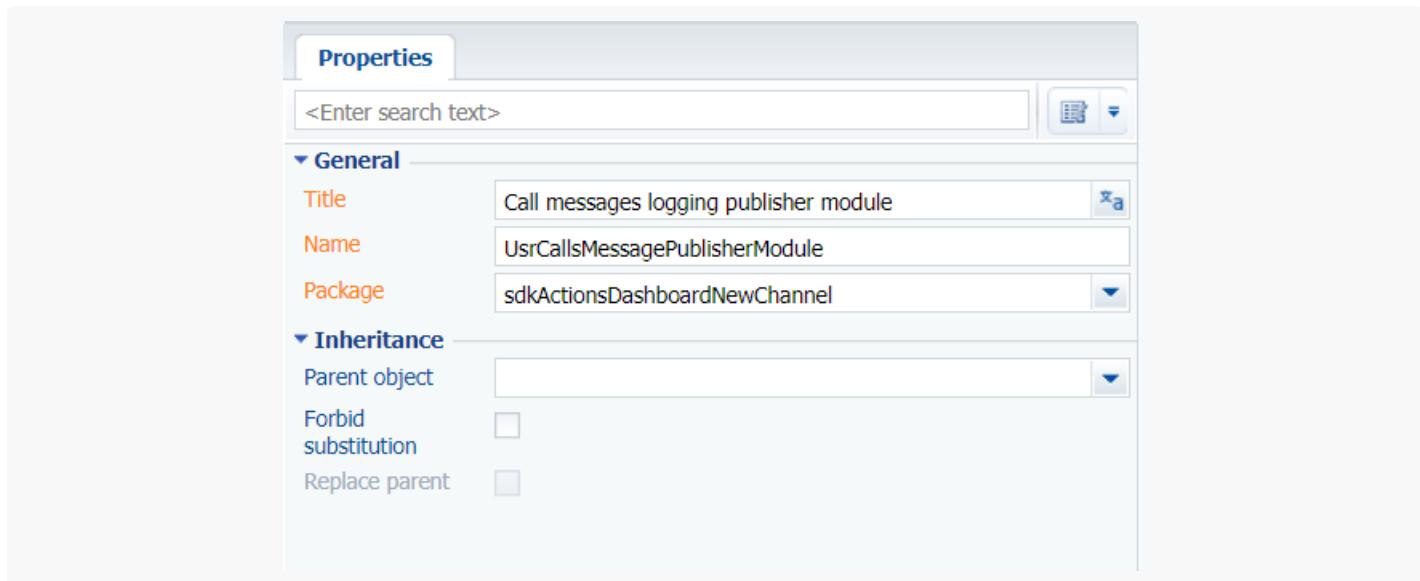
3. Создать модуль UsrCallsMessagePublisherModule

Модуль `UsrCallsMessagePublisherModule` служит контейнером для прорисовки в `SectionActionsDashboard` страницы `UsrCallsMessagePublisherPage`, в которой будет реализована логика добавляемого канала.

Для модуля установите следующие значения (рис. 4):

- [Заголовок] ([*Title*]) — "Модуль издателя сообщений логирования звонка" ("Call messages logging publisher module");
- [Название] ([*Name*]) — "UsrCallsMessagePublisherModule".

Рис. 4. — Свойства модуля



Исходный код модуля:

```
define("UsrCallsMessagePublisherModule", ["BaseMessagePublisherModule"],
    function() {
        // Определение класса.
        Ext.define("Terrasoft.configuration.UsrCallsMessagePublisherModule", {
            // Базовый класс.
            extend: "Terrasoft.BaseMessagePublisherModule",
            // Сокращенное имя класса.
            alternateClassName: "Terrasoft.UsrCallsMessagePublisherModule",
            // Инициализация страницы, которая будет отрисовываться в данном модуле.
            initSchemaName: function() {
                this.schemaName = "UsrCallsMessagePublisherPage";
            }
        });
        // Возвращает объект класса, определенного в модуле.
        return Terrasoft.UsrCallsMessagePublisherModule;
    });
});
```

4. Создать страницу UsrCallsMessagePublisherPage

Для создаваемой страницы установите в качестве родительского объекта схему `BaseMessagePublisherPage` пакета `MessagePublisher`. В качестве названия и заголовка укажите значение `"UsrCallsMessagePublisherPage"`.

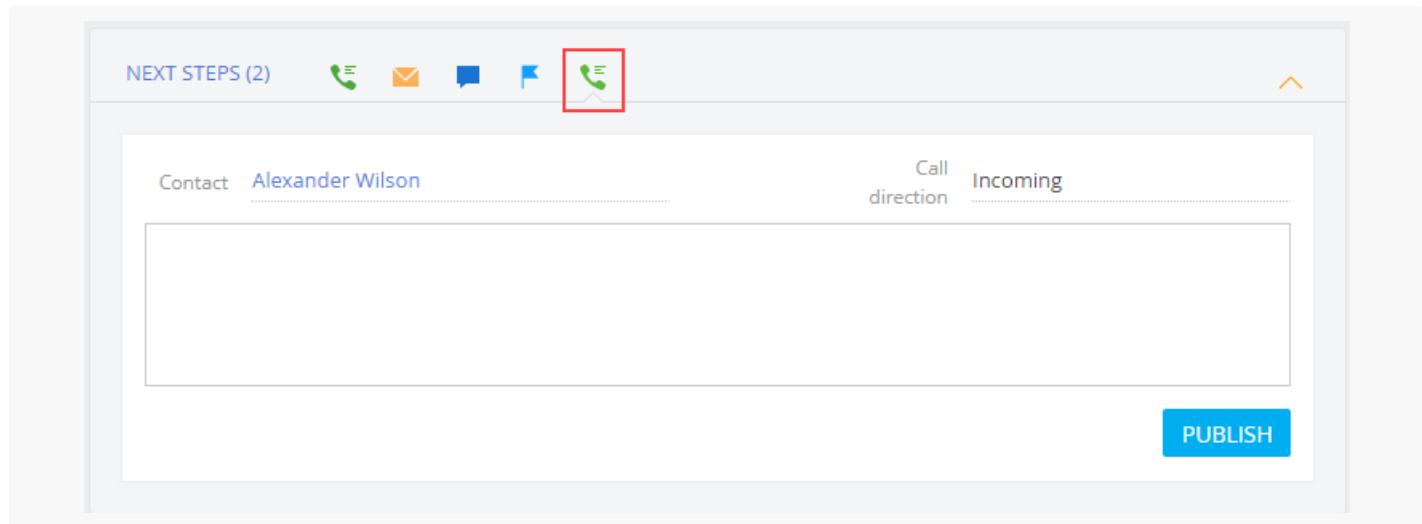
В исходном коде страницы укажите имя схемы объекта, с которым будет работать страница (в данном случае `Activity`), реализуйте логику публикации сообщения и переопределите метод `getServiceConfig`, в котором укажите имя класса из конфигурации.

```
// Задает класс, который будет работать с данной страницей.
```

```
getServiceConfig: function() {
    return {
        className: "Terrasoft.Configuration.CallsMessagePublisher"
    };
}
```

Реализация логики публикации сообщения содержит довольно большое количество методов, атрибутов и свойств. Полностью исходный код схемы `UsrCallsMessagePublisherPage` вы можете скачать по [ссылке](#). В исходном коде показана реализация рабочего канала `CallMessagePublisher`, который используется для логирования входящих и исходящих звонков. Результатом выполнения данного примера будет новый рабочий канал в `SectionActionsDashboard` (рис. 5).

Рис. 5. — Пример пользовательского канала `CallsMessagePublisherPage` в `SectionActionsDashboard` раздела [Контакты]



Добавить мультиязычные шаблоны email-сообщений

 Сложный

Creatio предоставляет возможность использовать собственную логику подбора шаблона email-сообщения по языку. На инструментальной панели действий (ActionsDashboard) записи раздела можно выбирать шаблоны email-сообщений на необходимом языке. Подбор выполняется на основании специализированных правил, которые могут быть определены в зависимости от раздела. Если специфические правила не определены, подбор ведется на основе контакта, связанного с редактируемой записью (колонка [Контакт]). Если колонки связи с контактом в объекте раздела нет, используется значение системной настройки `DefaultMessageLanguage`.

Для добавления собственной логики по подбору мультиязычных шаблонов:

1. Создайте класс или классы, унаследованные от `BaseLanguageRule` и определите правила подбора языка (один класс определяет одно правило).
2. Создайте класс, унаследованный от `BaseLanguageIterator`. В конструкторе класса определите свойство

`LanguageRules` как массив экземпляров классов, созданных на предыдущем шаге. Порядок следования соответствует приоритету правил.

3. Создайте класс-наследник от `AppEventListenerBase`, выполняющий привязку класса, определяющего правила подбора языка, к разделу.

4. В справочник [*Шаблоны email сообщений*] ([*Email Templates*]) добавьте нужные мультиязычные шаблоны.

Описание примера

В пользовательский раздел добавить логику выбора языка email-сообщения на основе колонки `UsrContact` основного объекта раздела. Для примера использовать английский и испанский языки.

Исходный код

Пакет с реализацией примера можно скачать по [ссылке](#).

Важно.

Пакет можно установить для продуктов Creatio, содержащих пакет `EmailTemplates`. После установки пакета убедитесь, что выполнены все предварительные настройки, описанные ниже.

Предварительные настройки

Для корректной работы примера:

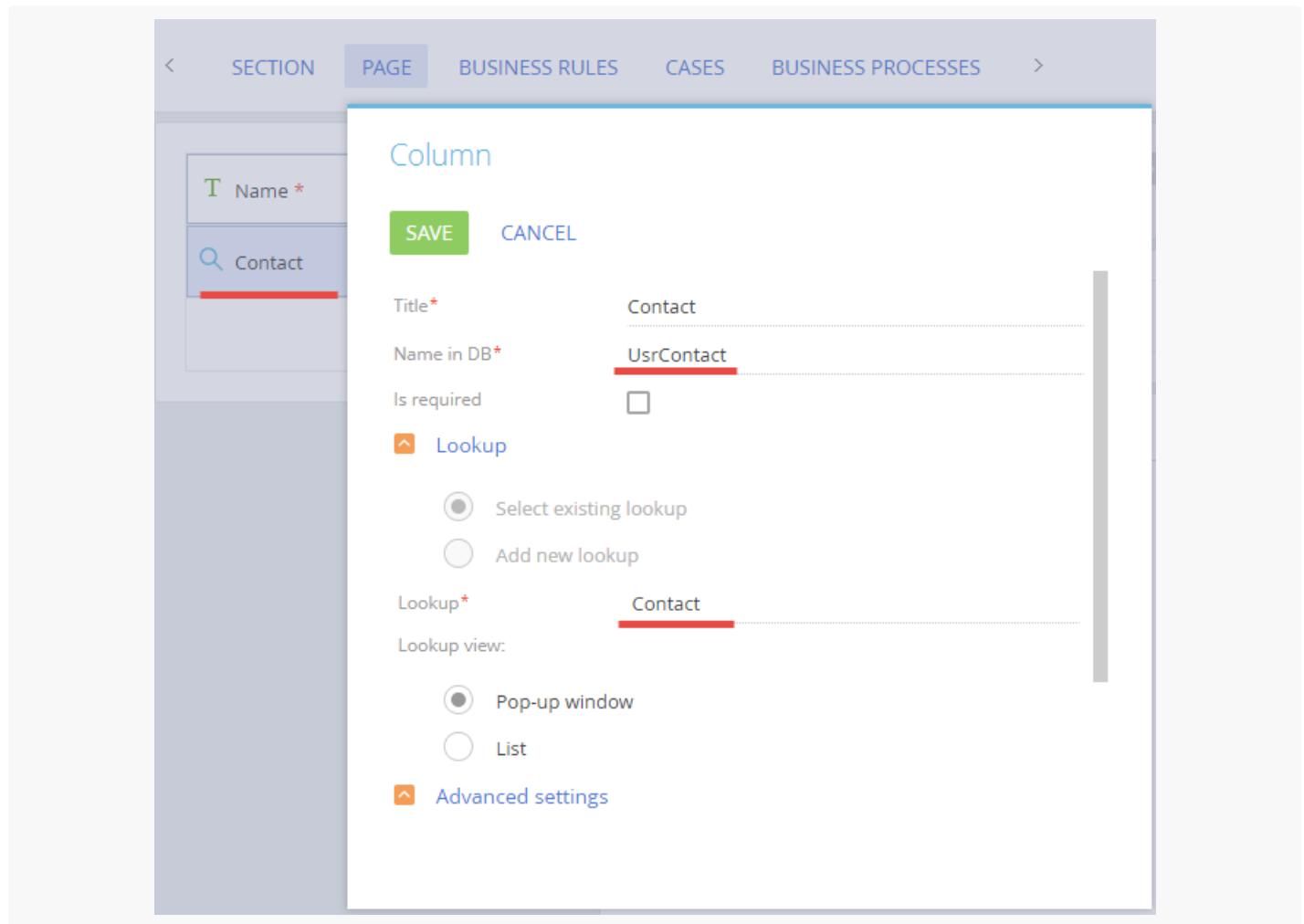
1. Убедитесь, что в справочнике [*Языки общения*] ([*Customer languages*]) используются английский и испанский языки (рис. 1).

Рис. 1. — Справочник [Языки общения]

Name	Description	Code	Is used
English (United St...)	English (United St...)	en-US	Yes
Spanish (Spain)	Español (España, ...)	es-ES	Yes

2. В мастере разделов проверьте, что на странице записи пользовательского раздела существует колонка `UsrContact`, связанная со справочником [*Контакт*] ([*Contact*]) (рис. 2).

Рис. 2. — Колонка UsrContact



Алгоритм реализации примера

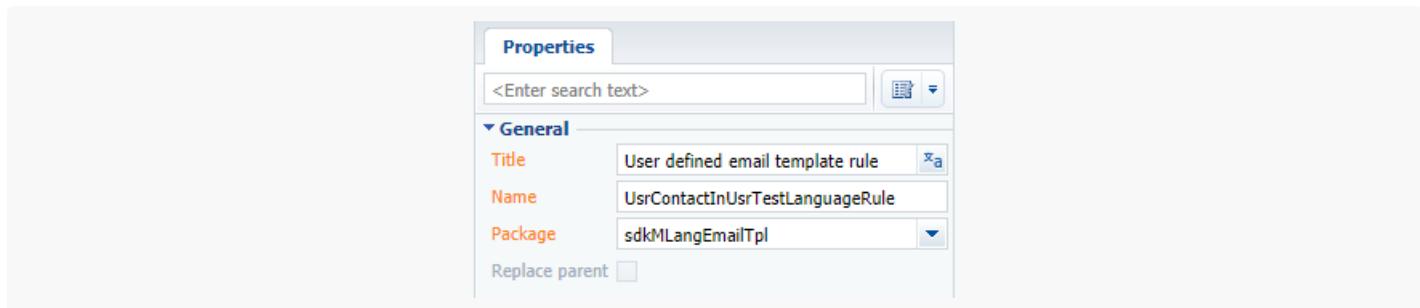
1. Добавить правило подбора языка

В пользовательском пакете создайте схему [Исходный код] (см. "[Создать схему \[Исходный код \]](#)").

Для созданной схемы укажите (рис. 3):

- [Название] ([Name]) — "UsrContactInUsrTestLanguageRule";
- [Заголовок] ([Title]) — "Пользовательское правило шаблона email-сообщений" ("User defined email template rule").

Рис. 3. — Свойства схемы [Исходный код]



Добавьте в схему следующий исходный код:

```

namespace Terrasoft.Configuration
{
    using System;
    using Terrasoft.Core;
    using Terrasoft.Core.Entities;
    public class ContactInUsrTestLanguageRule : BaseLanguageRule
    {
        public ContactInUsrTestLanguageRule (UserConnection userConnection) : base(userConnectio
        {
        }
        // Определяет идентификатор предпочтаемого языка пользователя.
        // recId – идентификатор текущей записи.
        public override Guid GetLanguageId(Guid recId)
        {
            // Создание экземпляра EntitySchemaQuery для основного объекта пользовательского раз
            var esq = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "UsrMLangEmailTp
            // Определение названия колонки языка контакта.
            var languageColumnName = esq.AddColumn("UsrContact.Language.Id").Name;
            // Получение экземпляра текущей записи.
            Entity usrRecEntity = esq.GetEntity(UserConnection, recId);
            // Получение значения идентификатора предпочтаемого языка пользователя.
            Guid languageId = usrRecEntity.GetTypedColumnValue<Guid>(languageColumnName);
            return languageId;
        }
    }
}

```

Опубликуйте схему.

2. Определить порядок правил подбора языка

В пользовательском пакете создайте схему [Исходный код] (см. "[Создать схему \[Исходный код \]](#)").

Для созданной схемы укажите (рис. 3):

- [Название] ([Name]) — "UsrTestLangelriterator";

- [Заголовок] ([Title]) — "Пользовательский итератор правил выбора языка" ("User defined language iterator").

Добавьте в схему исходный код, приведенный ниже:

```
namespace Terrasoft.Configuration
{
    using Terrasoft.Core;
    public class UsrTestLanguageIterator: BaseLanguageIterator
    {
        public UsrTestLanguageIterator(UserConnection userConnection): base(userConnection)
        {
            // Массив правил выбора языка.
            LanguageRules = new ILanguageRule[] {
                // Пользовательское правило.
                new ContactInUsrTestLanguageRule (UserConnection),
                // Правило по умолчанию.
                new DefaultLanguageRule(UserConnection),
            };
        }
    }
}
```

Вторым элементом массива является `DefaultLanguageRule`. Это правило использует для получения языка системную настройку `DefaultLanguage` и используется по умолчанию, если язык не был найден другими, более приоритетными правилами.

Выполните публикацию схемы.

3. Привязать итератор правил выбора языка к разделу

В пользовательском пакете создайте схему [Исходный код] (см. "[Создать схему \[Исходный код \]](#)").

Для созданной схемы укажите (рис. 3):

- [Название] ([Name]) — "UsrTestMLangBinder";
- [Заголовок] ([Title]) — "UsrTestMLangBinder".

Добавьте в схему следующий исходный код:

```
namespace Terrasoft.Configuration
{
    using Terrasoft.Core.Factories;
    using Terrasoft.Web.Common;
    public class UsrTestMLangBinder: AppEventListenerBase
    {
        public override void OnAppStart(AppEventArgs context)
        {
```

```

// Вызов базовой логики.
base.OnAppStart(context);
// Привязка итератора к пользовательскому разделу.
// UsrMLangEmailTpl – название основного объекта раздела.
ClassFactory.Bind<ILanguageIterator, UsrTestLanguageIterator>("UsrMLangEmailTpl");
}
}
}

```

Опубликуйте схему.

4. Добавить необходимые мультиязычные шаблоны

В справочник [Шаблоны email сообщений] ([Email Templates]) добавьте новую запись (рис. 4), в которой определите шаблоны email-сообщений на требуемых языках (рис. 5).

Рис. 4. — Новая запись в справочнике [Шаблоны email сообщений]

The screenshot shows a list of email templates. At the top, there is a header 'Email templates' and a 'Filters/folders' dropdown. Below the header, there is one visible item:

Case feedback request notification	Subject New message on case #[#Number#]
MLEmailTpptest (US, ES)	

The row for 'MLEmailTpptest (US, ES)' has a red underline underneath it, indicating it is selected or highlighted.

Рис. 5. — Добавление шаблонов на требуемых языках

The screenshot shows a software interface for managing email templates. At the top, there's a header bar with a search field 'What can I do for you?' and a 'Creatio' logo. Below the header, a blue button says 'CLOSE'. On the right, there's a 'VIEW' dropdown. A sidebar on the left has a 'Template name*' field containing 'MLEmailTpltest' and a 'Macro source' section. The main content area shows language tabs: 'ENGLISH (UNITED STATES)' and 'SPANISH (SPAIN)', with 'SPANISH (SPAIN)' currently selected. Under the tabs, there's an 'Email template' section with an 'Edit' button. Below this, a 'Subject' field contains the text 'Español'. The main body of the template is displayed in a large text area with the text 'HTML & CSS'.

В результате выполнения примера на странице записи пользовательского раздела (рис. 6) в канале инструментальной панели действий (рис. 6, 1) шаблоны email сообщений (рис. 6, 2) будут выбираться автоматически на том языке контакта (рис. 6, 3), который установлен как предпочтительный (рис. 7).

Рис. 6. — Результат выполнения примера

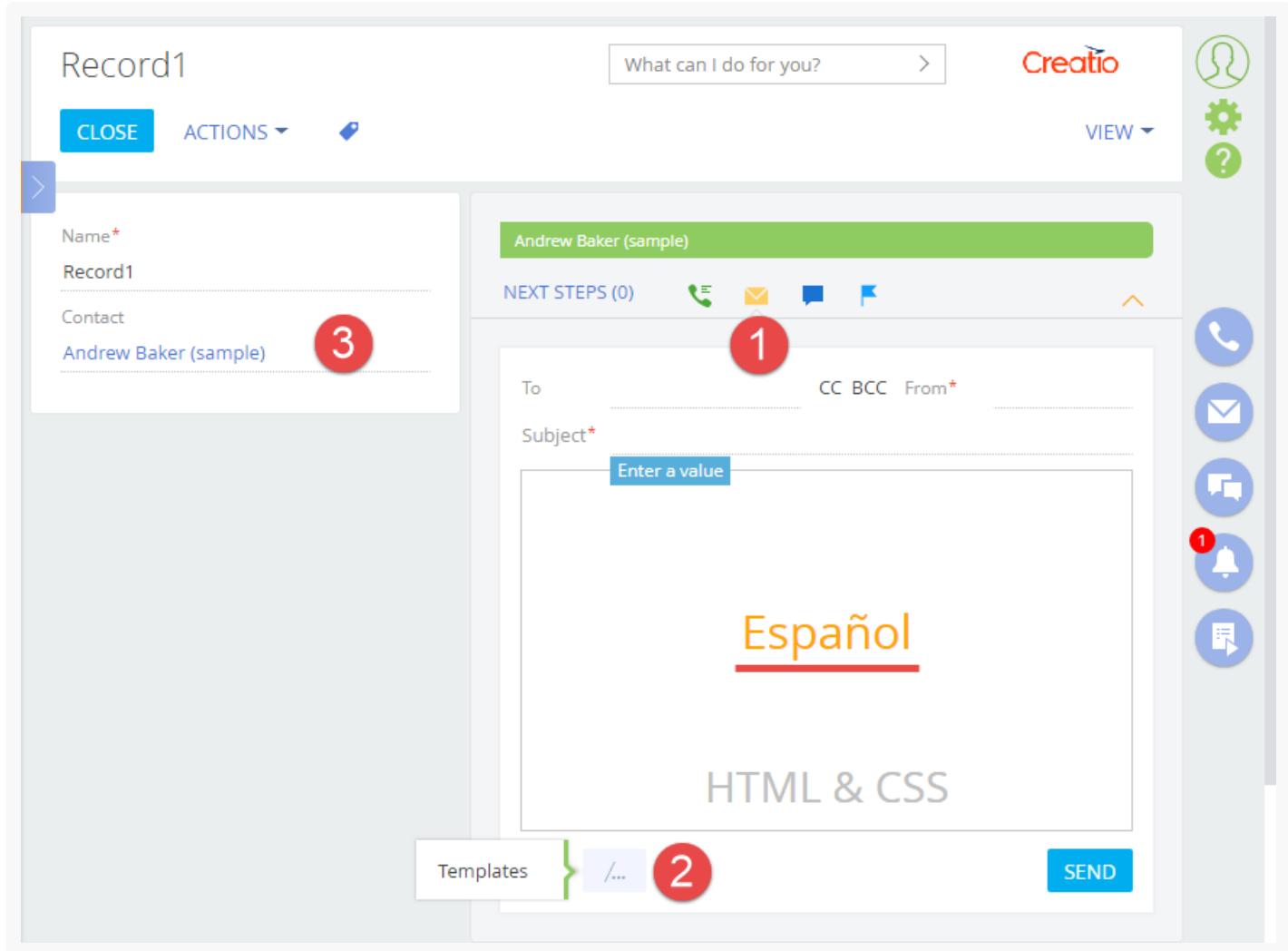


Рис. 7. — Предпочтительный язык контакта

This screenshot shows the 'CONTACT INFO' tab of a contact record. The contact information includes:

- Type: Customer
- Title: Mr.
- Owner: Supervisor
- Gender: Male
- Preferred language: Spanish (Spain)

The 'Preferred language' field is highlighted with a red border.

Создать пользовательскую страницу действия верификации

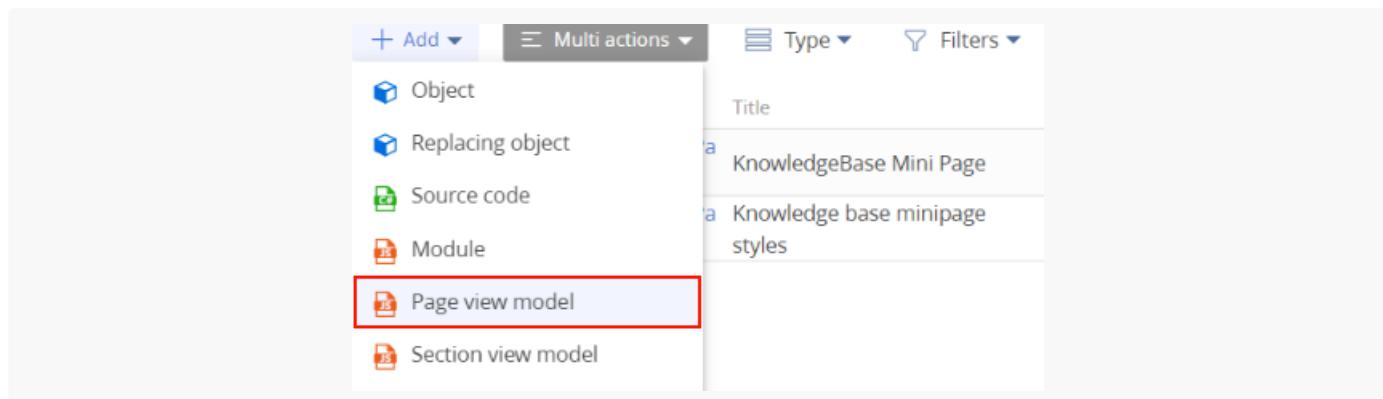
Сложный

Важно. Пример может быть реализован только в продукте Financial Services Creatio.

Пример. Создать страницу действия верификации, на которой будет скрыто поле [Комментарий]. Страница действия верификации подробно описана в статье [Панель действий](#).

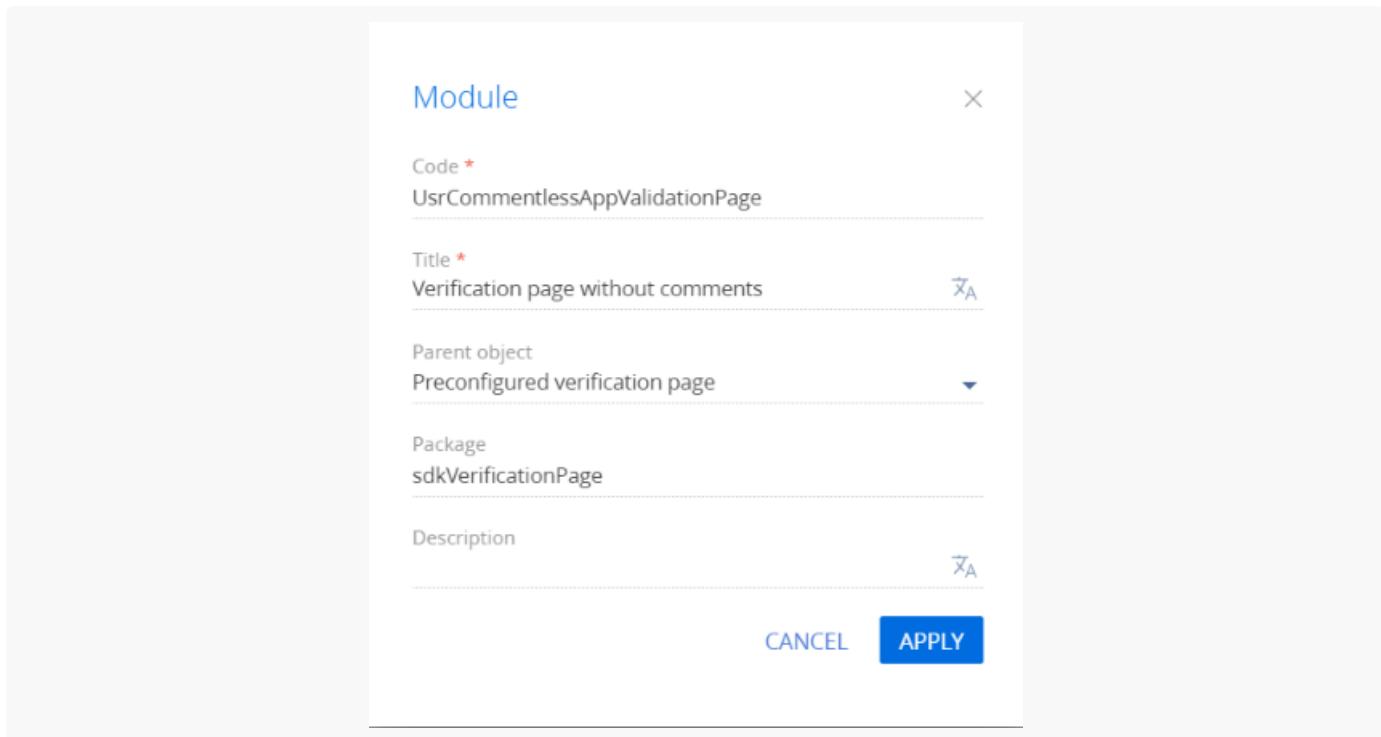
1. Создать схему страницы действия верификации

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



3. В дизайнере схем заполните свойства схемы:

- [Код] ([Code]) — "UsrCommentlessAppValidationPage".
- [Заголовок] ([Title]) — "Verification page without comments".
- [Родительский объект] ([Parent object]) — выберите "Преднастроенная пользовательская страница" ("Preconfigured verification page").



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

2. Настроить представление страницы

В дизайнере схем добавьте необходимый исходный код.

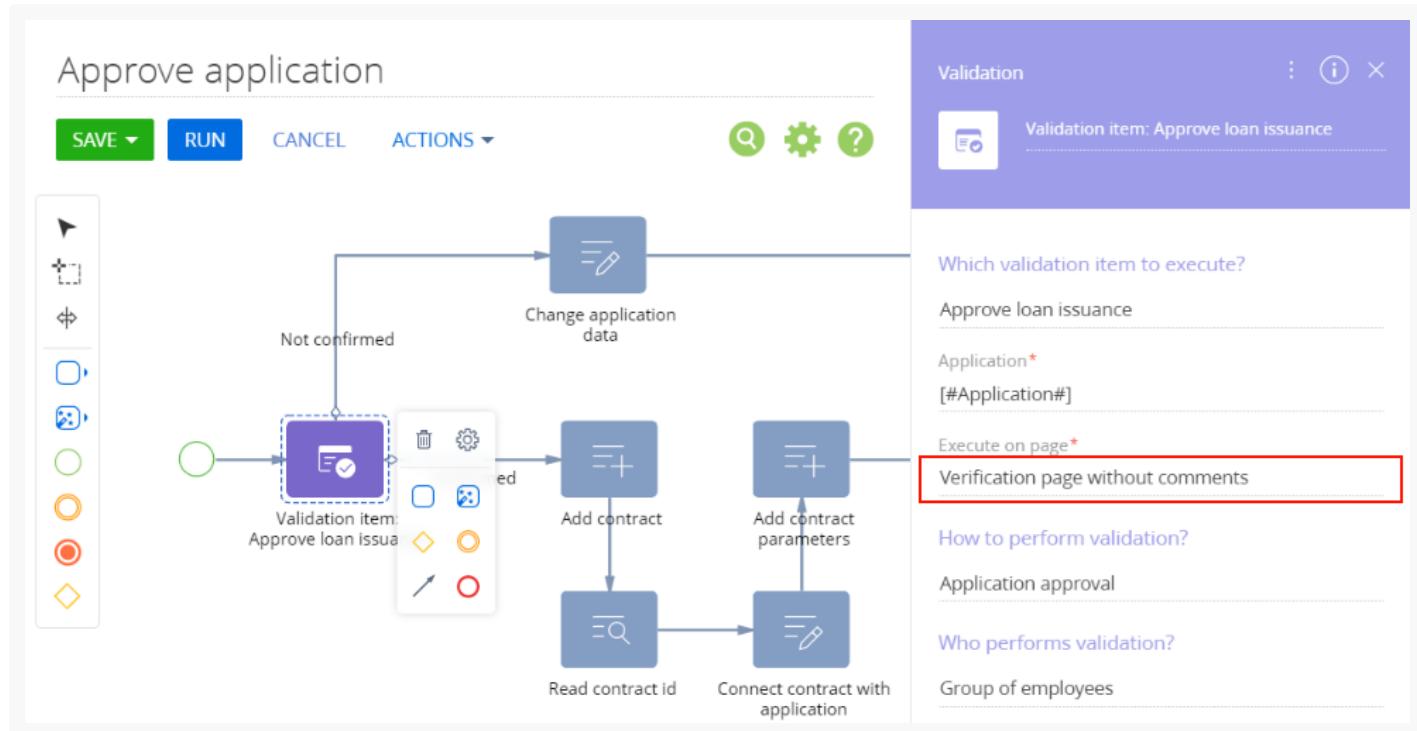
В массиве модификаций diff удалите из родительского элемента поле [*Комментарий*].

UsrCommentlessAppValidationPage.js

```
define("UsrCommentlessAppValidationPage", [], function() {
    return {
        entitySchemaName: "AppValidation",
        diff: [
            {
                "operation": "remove",
                "name": "CommentContainer"
            }
        ]
    };
});
```

2. Использовать созданную схему в бизнес-процессе

Чтобы использовать созданную схему, необходимо указать ее в поле [*Выполнить на странице*] ([*Execute on page*]) элемента [*Действие верификации*]([*Validation item*]) бизнес-процесса. Эта схема может быть использована как в новых, так и в уже существующих бизнес-процессах, например, `Aprrove application`.

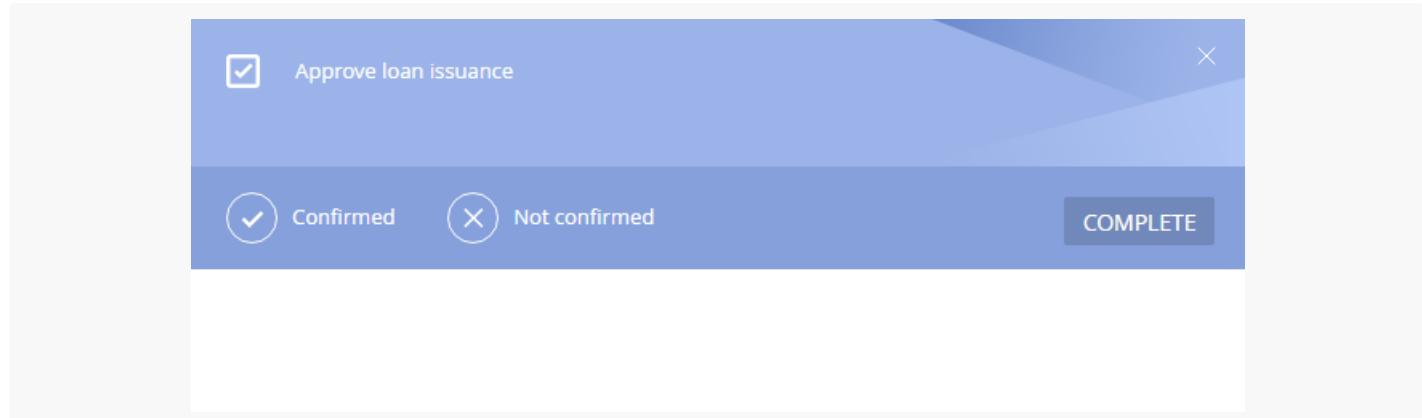


Для применения изменений бизнес-процесс необходимо сохранить.

Важно. Чтобы изменения были окончательно применены, требуется перезапуск сайта приложения в IIS.

Результат выполнения примера

После окончательного применения изменений прежняя страница верификации будет заменена пользовательской, не содержащей поля [Комментарий].



Поле

Средний

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 [Настройте 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 изменится 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 объявиите 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 объягите 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 дизайнере 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
[Константа] ([Constant])	Возможные типы колонок, для которых можно установить константу в качестве значения по умолчанию: <ul style="list-style-type: none"> [Стока] ([String]). [Число] ([Number]). [Справочник] ([Lookup]). [Логическое] ([Boolean]).
[Системная настройка] ([System setting])	Системные настройки содержатся в разделе [Системные настройки] ([System settings]), в который можно добавить пользовательскую системную настройку. Значение системной настройки устанавливается на уровне пользователя, а не на уровне приложения.
[Системная переменная] ([System variable])	Системные переменные — 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 Операции с данными (front-end).</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": {
    dataType: 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.
- 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**.
 - а. В свойстве `attributes` включите блокировку с помощью атрибута `IsModelItemsEnabled`.
 - б. 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 автонумерации 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*]).

The screenshot shows a search interface with a list of items. At the top, there are buttons for '+ Add', 'Type' (with a dropdown arrow), 'Filters' (with a dropdown arrow), 'Search' (with a magnifying glass icon), and a gear icon. Below these are four categories: 'Object' (blue cube icon), 'Replacing object' (blue cube icon with a red border), 'Source code' (green folder icon), and 'Module' (red folder icon). To the right of the categories, there are columns for 'Status' and 'Type'. Under 'Replacing object', there is one entry: 'Activity'.

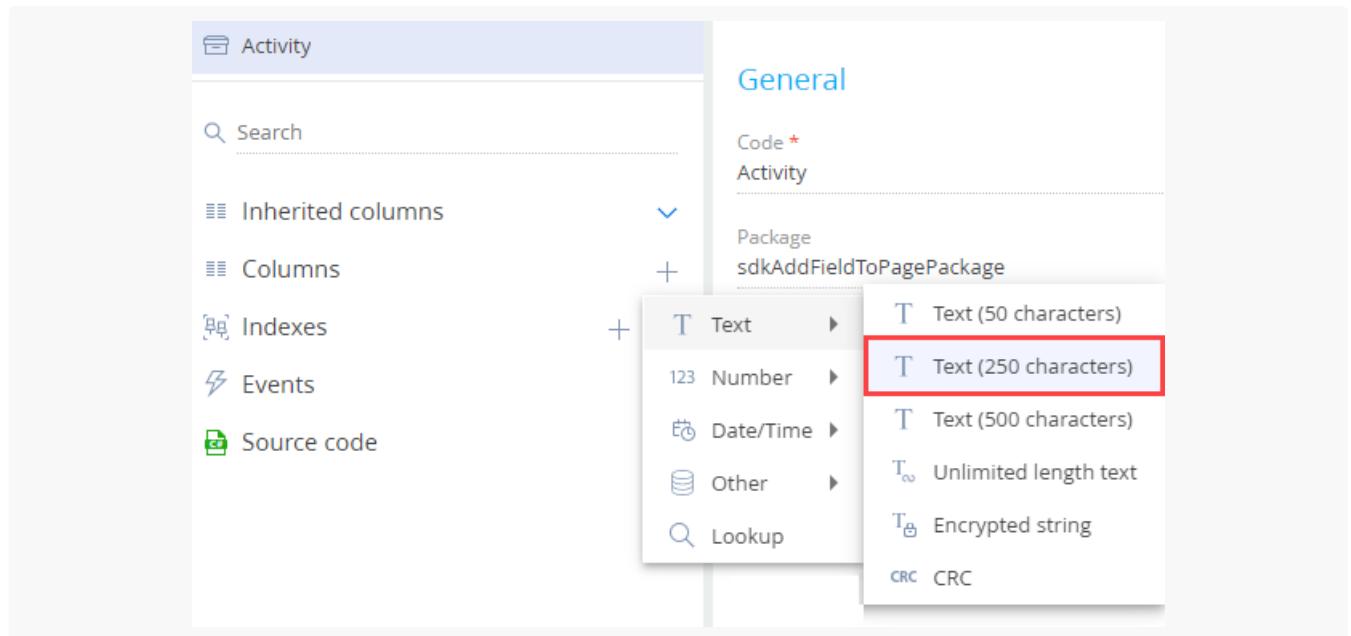
3. Заполните **свойства схемы**.

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

The screenshot shows the 'General' tab of a schema configuration. It includes fields for 'Code *' (Activity), 'Title *' (Activity), 'Package' (sdkAddFieldToPagePackage), and 'Description'. The 'Inheritance' tab is also visible, showing 'Parent object *' (Activity) and a checked 'Replace parent' checkbox.

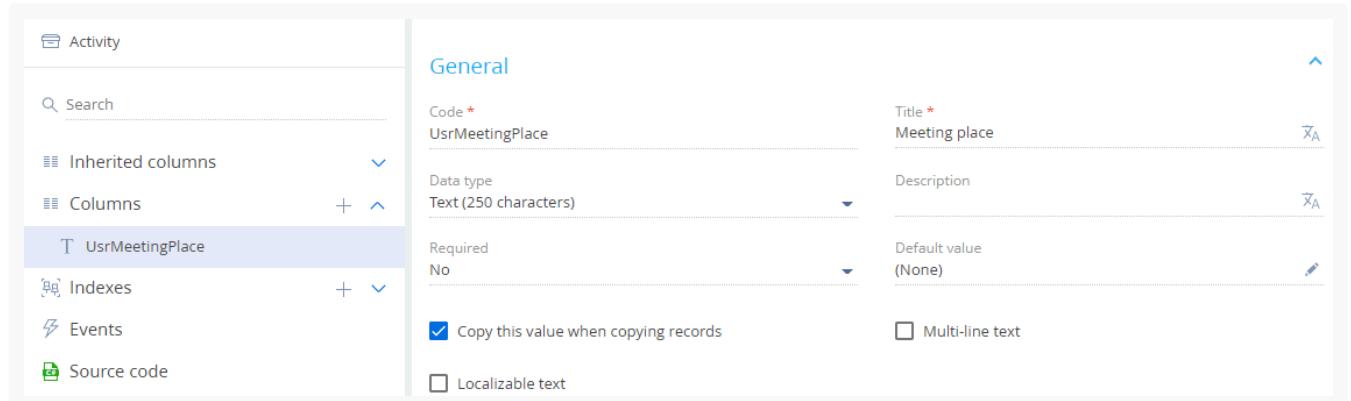
4. В схему добавьте **колонку**.

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



с. Заполните свойства добавляемой колонки.

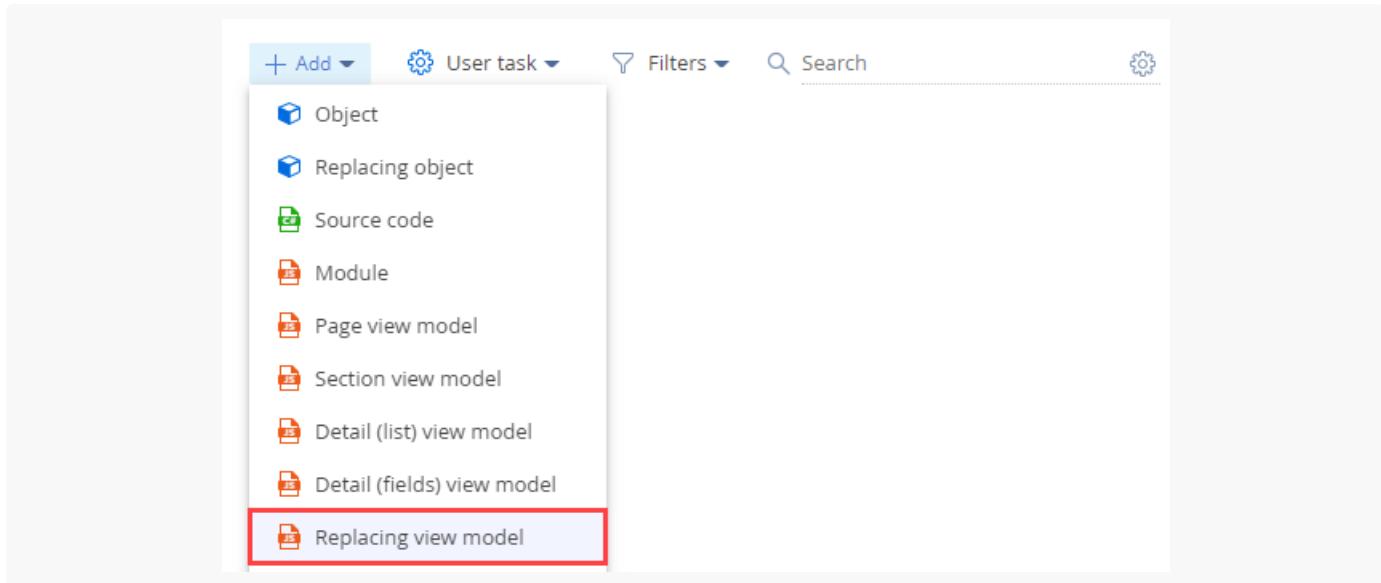
- [Код] ([Code]) — "UsrMeetingPlace".
- [Заголовок] ([Title]) — "Место встречи" ("Meeting place").



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

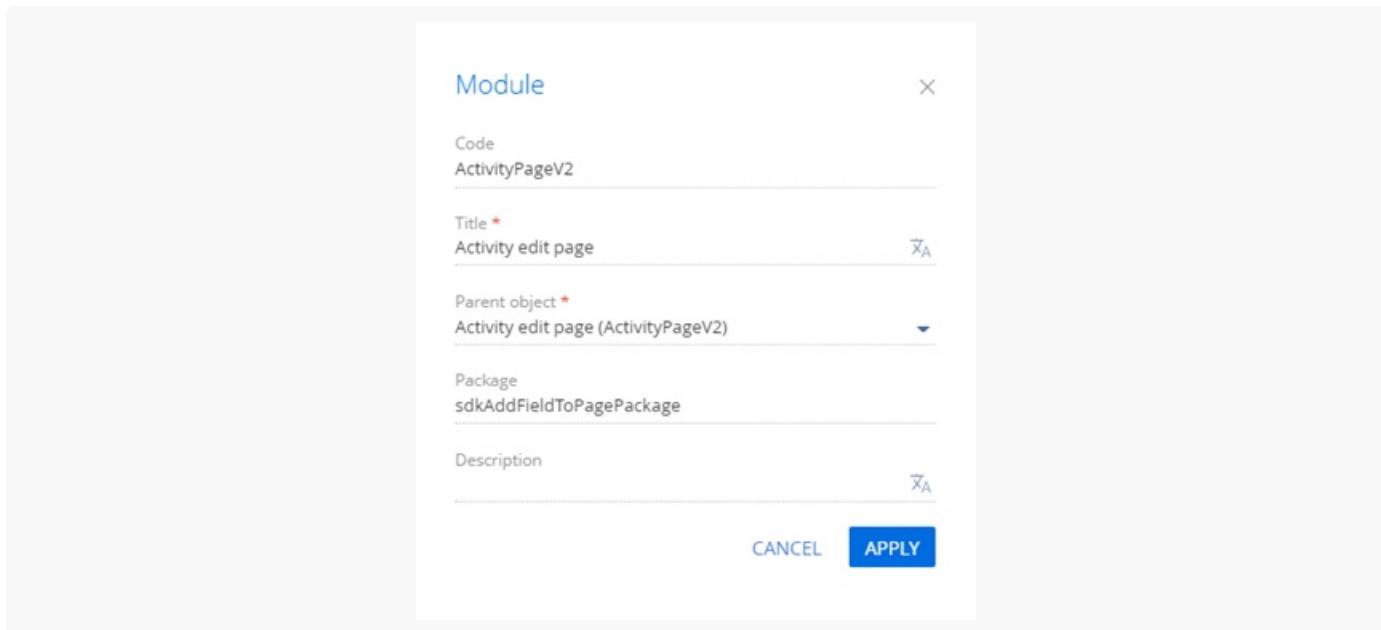
2. Создать схему замещающей модели представления страницы активности

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]).



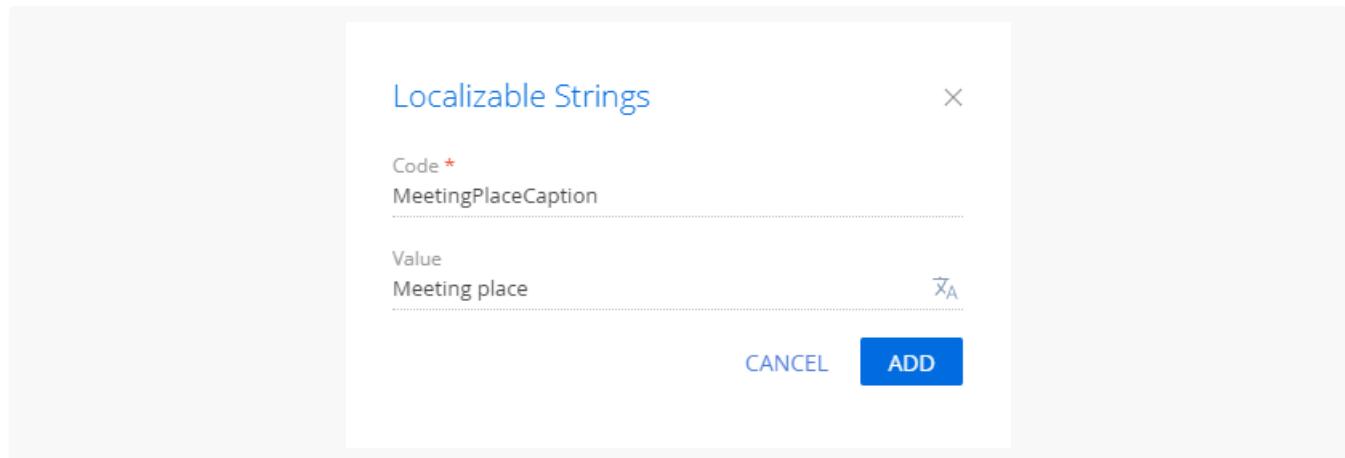
3. Заполните **свойства схемы**.

- [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. 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*/
);
});

```

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

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

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

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

Subject*	Test task		
Start*	11/15/2021	7:00 AM	
Due*	11/15/2021	7:30 AM	
Status*	Not started		
Show in calendar	<input checked="" type="checkbox"/>		
Meeting place			
Role			
Owner	Marina Kysla		
Reporter*	Marina Kysla		
Priority*	Medium		
Category*	To do		

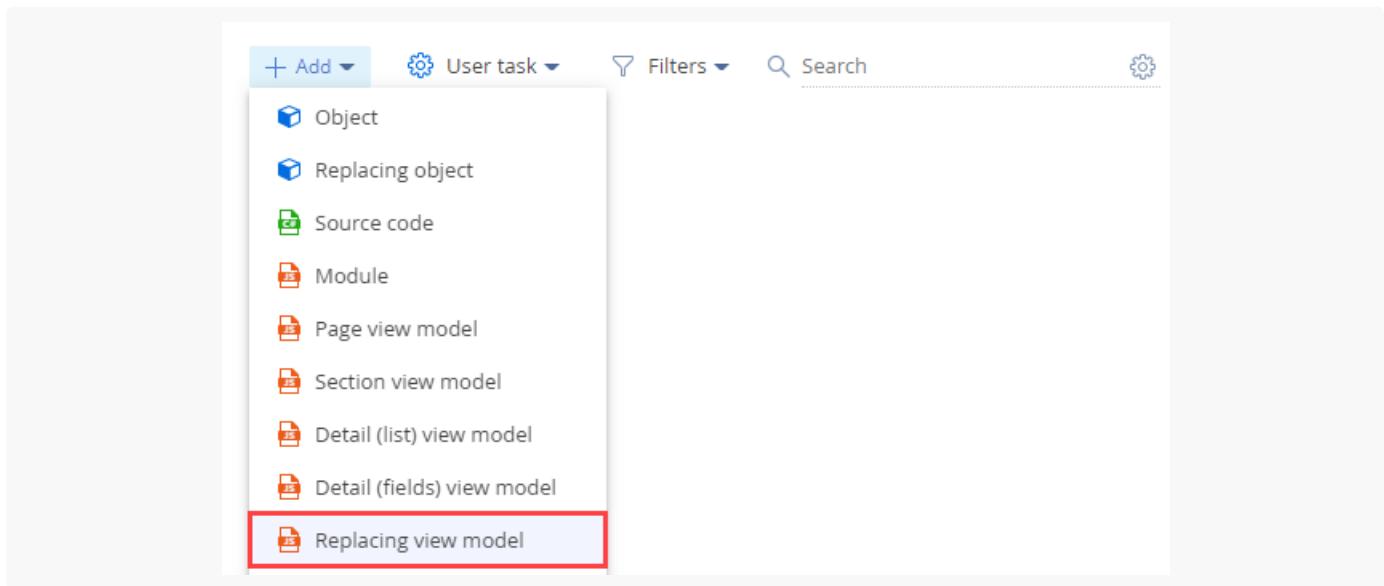
Добавить поле на страницу записи с использованием существующей колонки

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".

The screenshot shows the 'Module' configuration dialog with the following settings:

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

At the bottom are 'CANCEL' and 'APPLY' buttons.

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

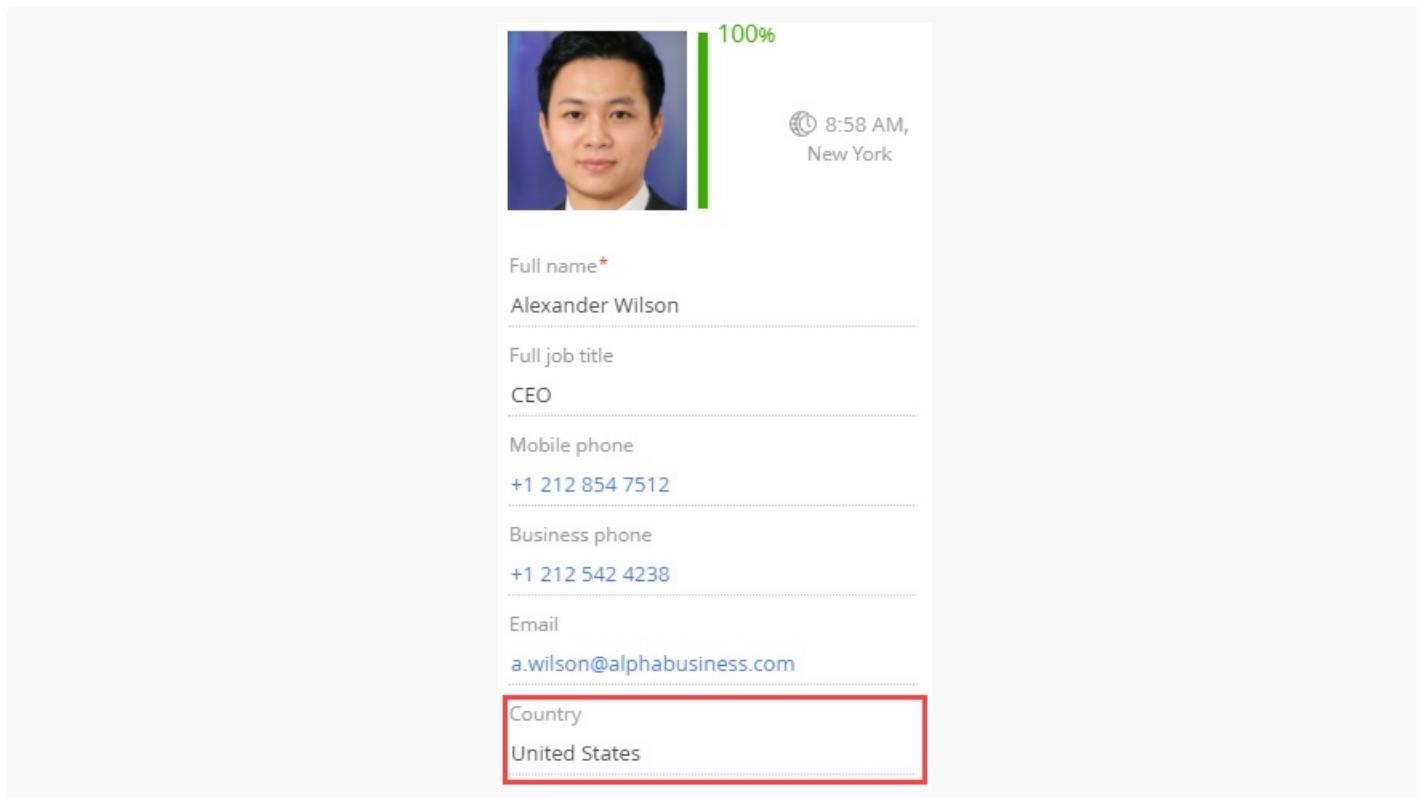
```
define("ContactPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение поля на странице записи. */
        diff: [
            /* Метаданные для добавления на страницу поля [Страна]. */
            {
                /* 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]).



The screenshot shows a contact profile with the following details:

- Profile Picture:** A placeholder image showing a person's face with a green vertical bar on the right labeled "100%".
- Timezone:** 8:58 AM, New York
- Fields:**
 - 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



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

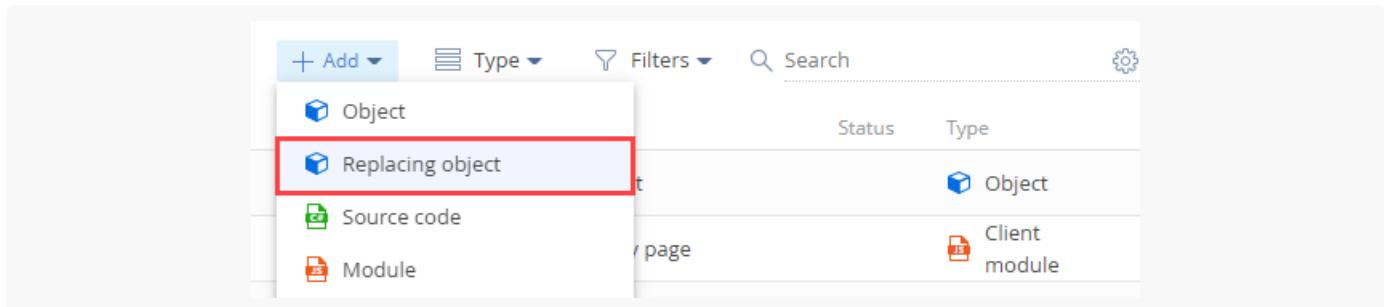


1. Создать схему замещающего объекта

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

будет добавлена схема.

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



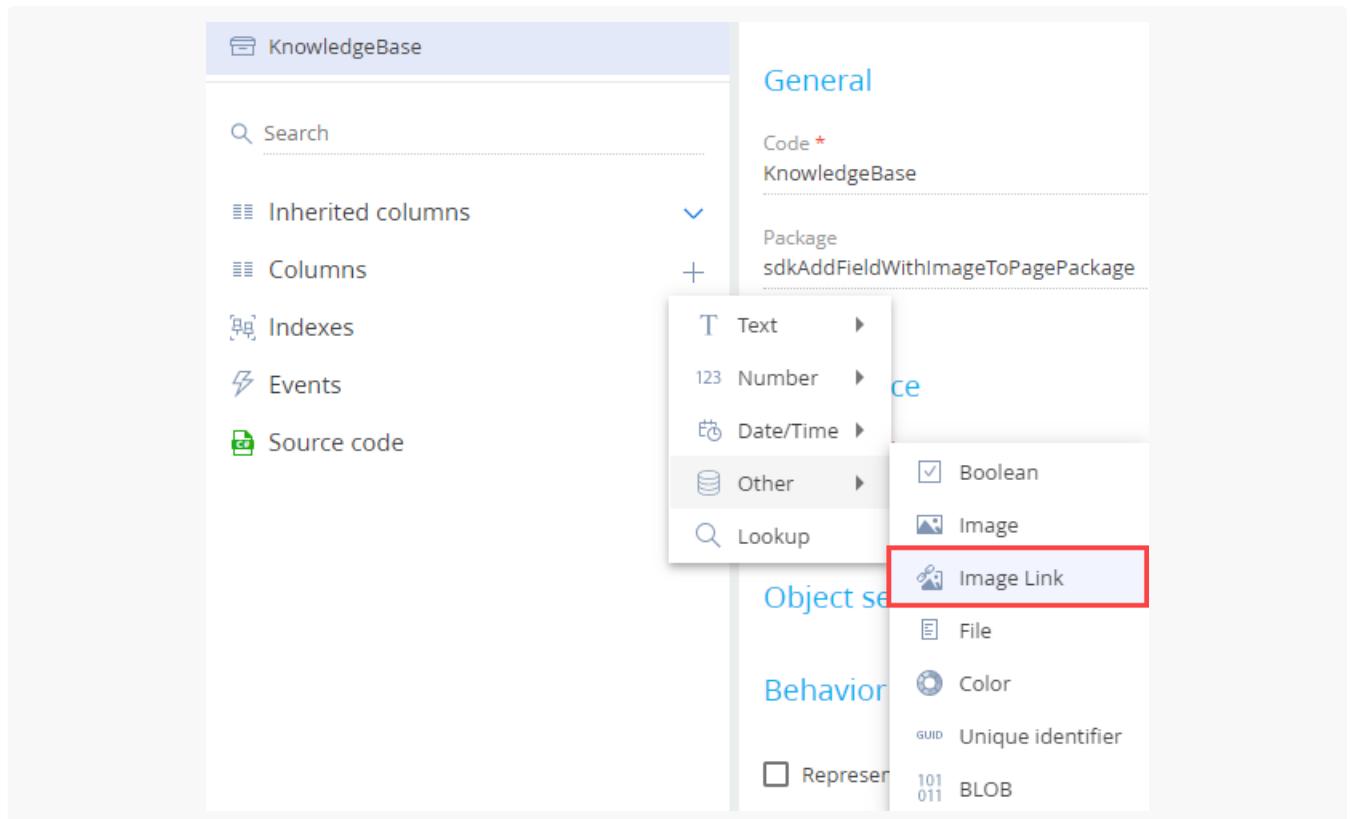
- Заполните **свойства схемы**.

- [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
<input checked="" type="checkbox"/> Replace parent	

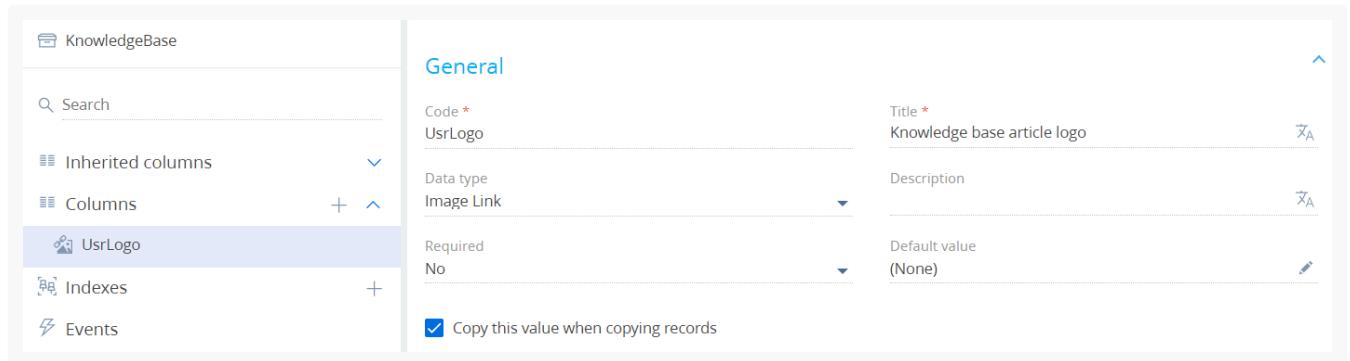
- В схему добавьте **колонку**.

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



c. Заполните **свойства добавляемой колонки**.

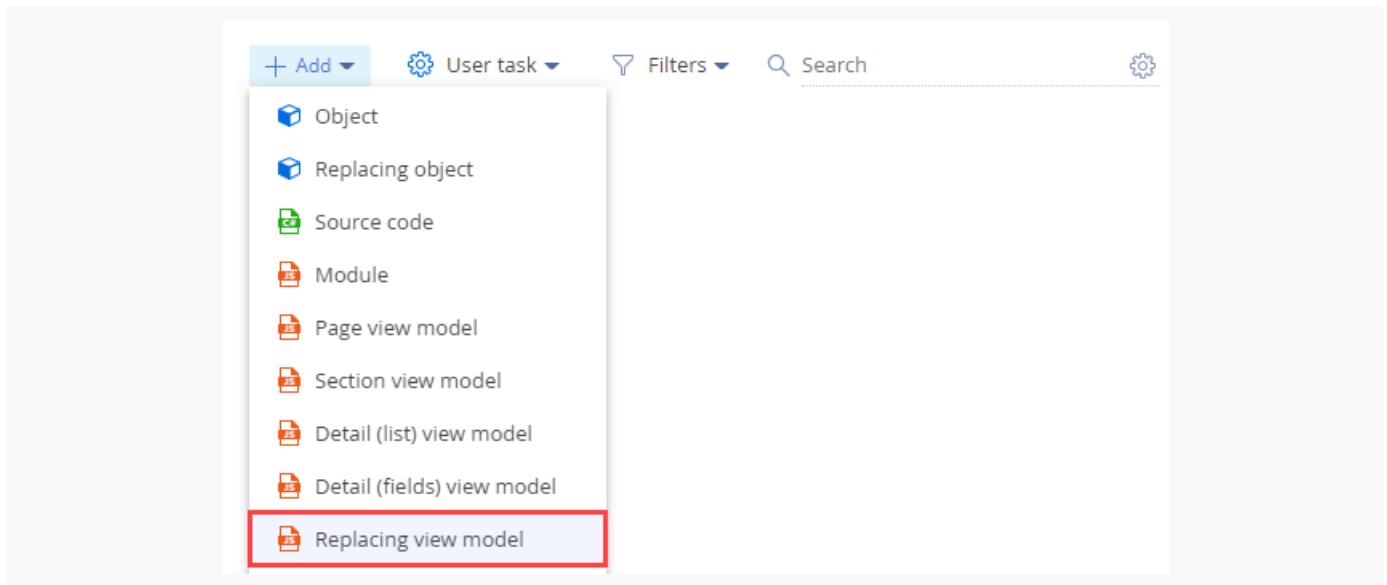
- [Код] ([Code]) — "UsrLogo".
- [Заголовок] ([Title]) — "Логотип статьи базы знаний" ("Knowledge base article logo").



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

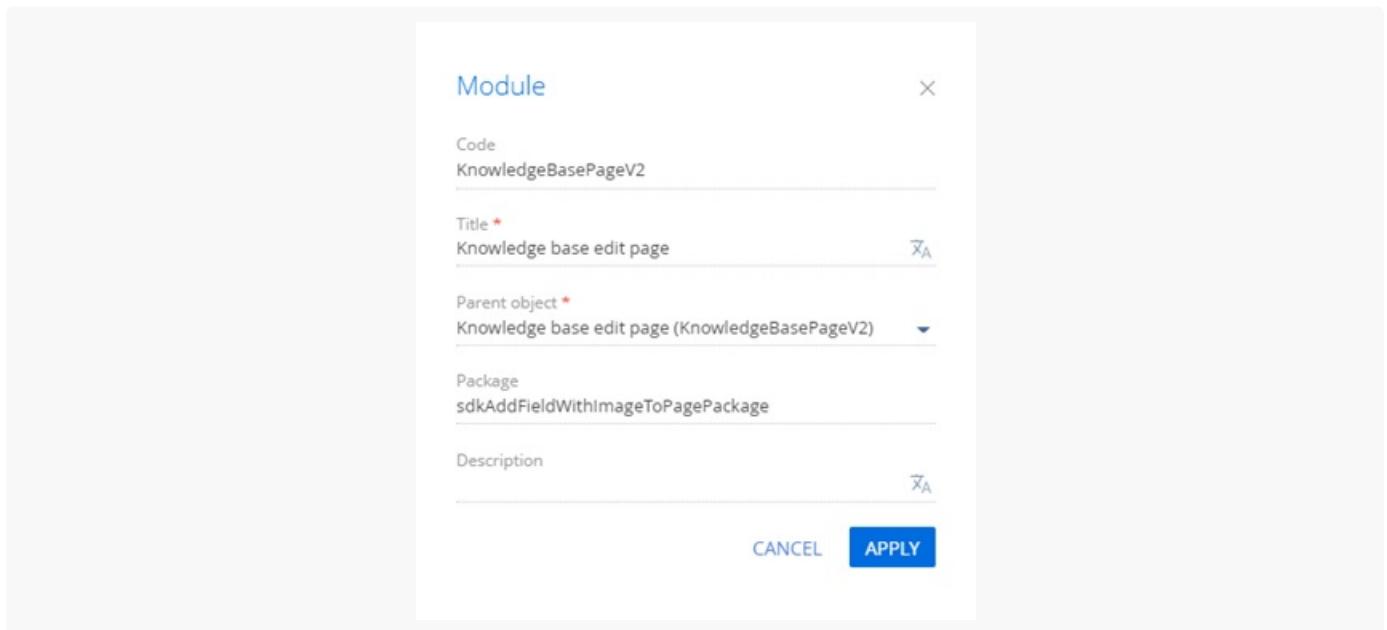
2. Создать схему замещающей модели представления страницы статьи базы знаний

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]).



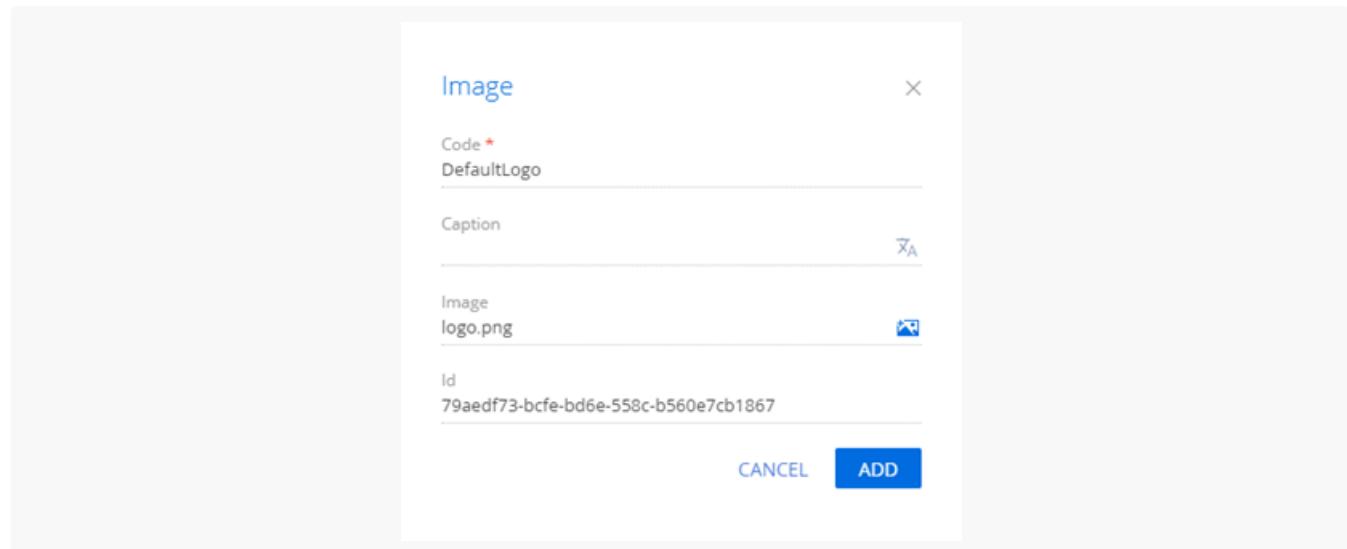
3. Заполните **свойства схемы**.

- [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.
- f. 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"], function()
    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"));
    },
    /* Обрабатывает 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 обработчик.
        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*/
};

});
```

- 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.

The screenshot shows a 'New Presentation Style' record in the Creatio application. At the top, there is a header with 'New Presentation Style', a search bar 'What can I do for you?', and the 'Creatio' logo with version '7.18.4.1532'. Below the header, there are buttons for 'SAVE', 'CANCEL', 'ACTIONS', and a 'VIEW' dropdown. The main content area contains the following fields:

- Name***: New Presentation Style
- Type***: Advertising materials
- Modified by**: Marina Kysla
- Modified on**: 11/14/2021 12:06 PM

On the left side of the content area, there is a placeholder image for the presentation style, featuring a large 'KB' logo with a cursor icon over it, and two small icons below it.

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

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

The screenshot shows a search interface with a list of items. The items are categorized by type: Object, Replacing object, Source code, and Module. The 'Replacing object' item is highlighted with a red box.

Status	Type
	Object
	Replacing object
	Source code
	Module

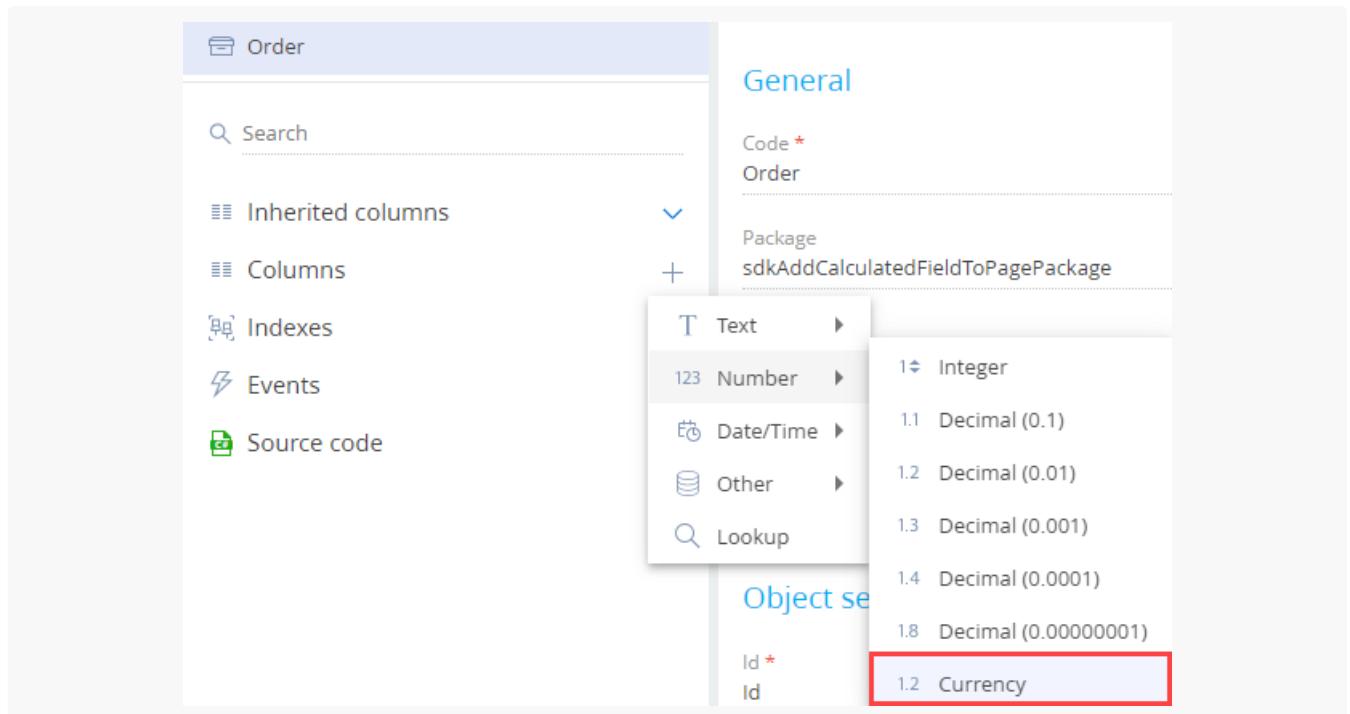
3. Заполните **свойства схемы**.

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

The screenshot shows the configuration dialog for a schema. It has two tabs: 'General' and 'Inheritance'.
General Tab:
 - Code: Order
 - Title: Order
 - Package: sdkAddCalculatedFieldToPagePackage
 - Description:
Inheritance Tab:
 - Parent object: Order
 - Replace parent:

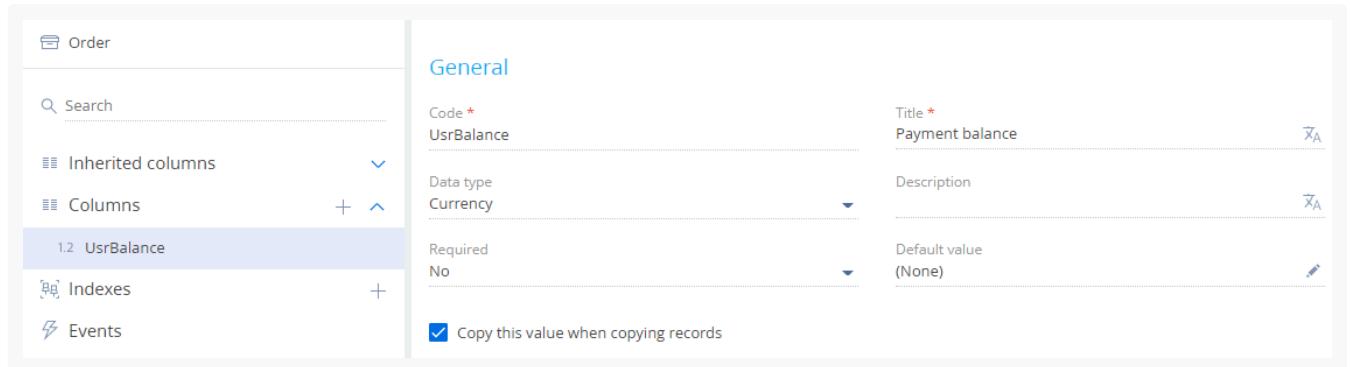
4. В схему добавьте **колонку**.

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



с. Заполните свойства добавляемой колонки.

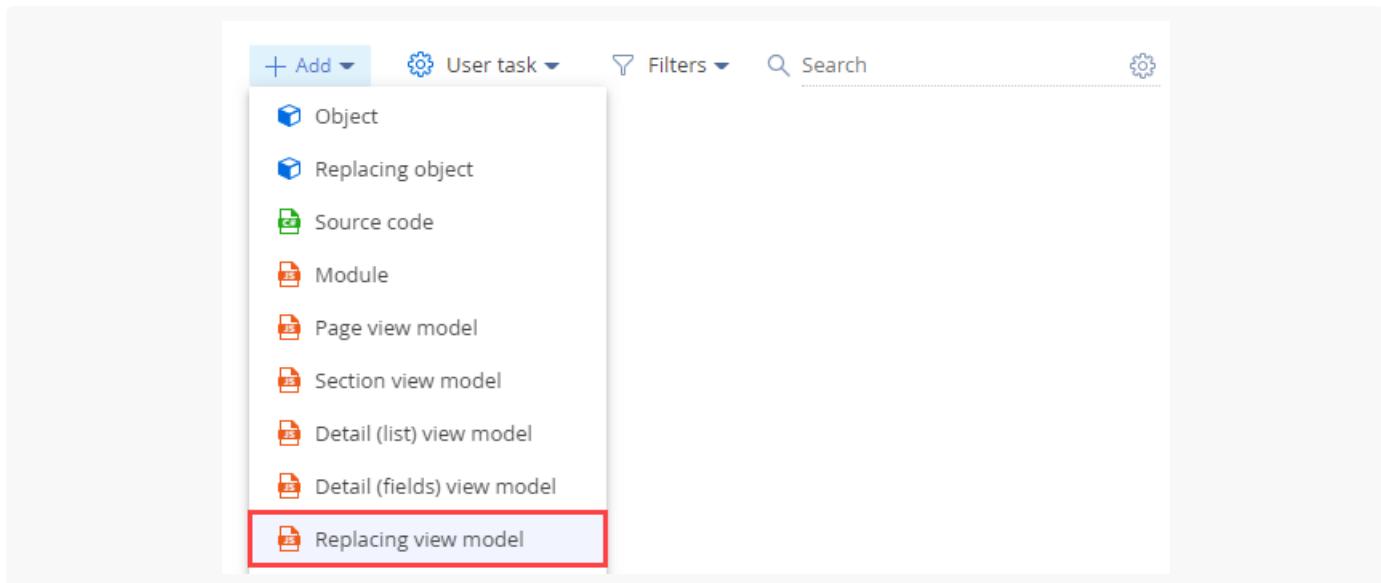
- [Код] ([Code]) — "UsrBalance".
- [Заголовок] ([Title]) — "Остаток для оплаты" ("Payment balance").



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

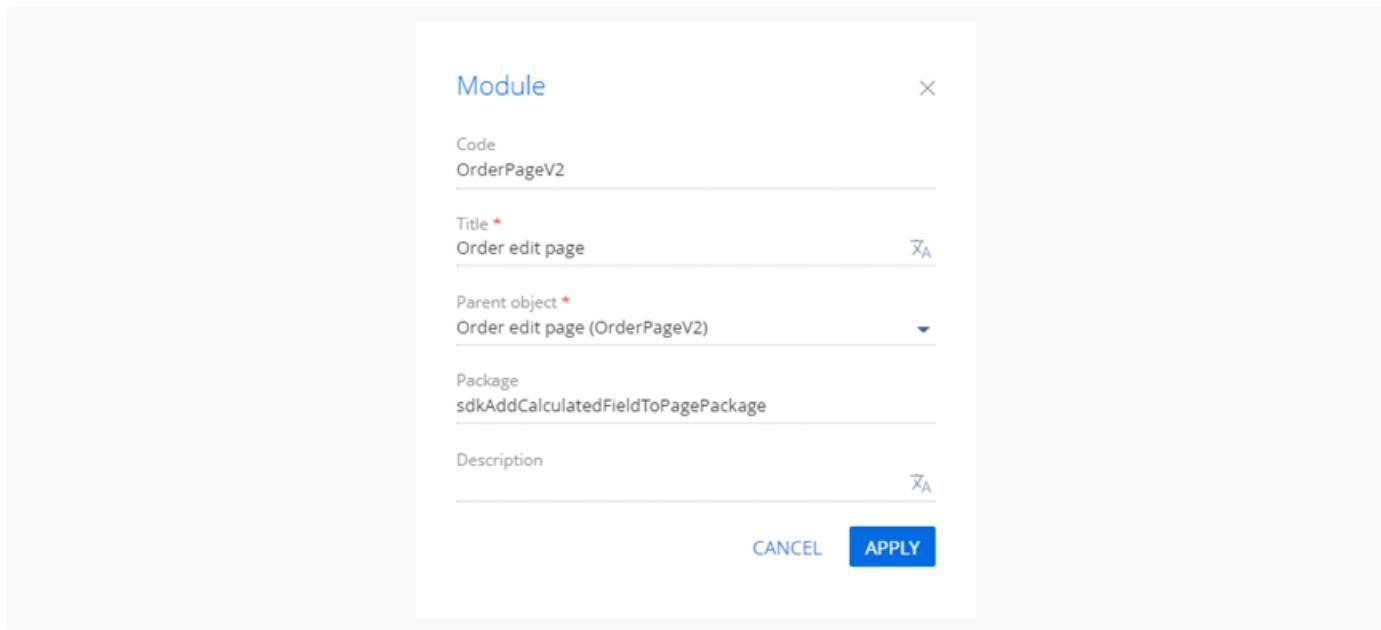
2. Создать схему замещающей модели представления страницы заказа

- [Перейдите в раздел \[Конфигурация \]](#) ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- 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.
- e. 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. */
                dataType: Terrasoft.DataType.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*/
);
});

```

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

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

- Очистите кэш браузера.
- 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])).

Customer*	Streamline Development	Total, \$	13,850.00
Status	4. Completed	Payment amount, \$	13,000.00
		Payment balance	850.00

Добавить мультивалютное поле на страницу записи

Сложный

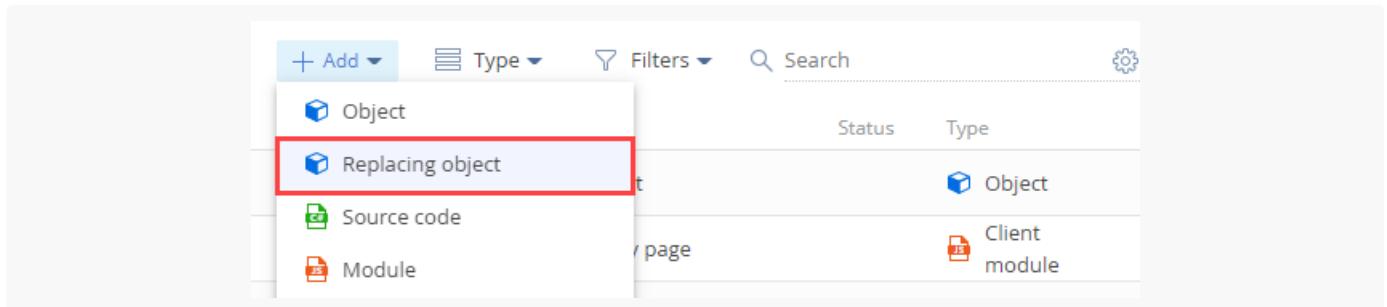
Пример. Добавить мультивалютное поле [Общая сумма] ([Amount]) на страницу проекта.

1. Создать схему замещающего объекта

- Перейдите в раздел [Конфигурация] ([Configuration]) и выберите пользовательский пакет, в который

будет добавлена схема.

- 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".

The 'General' tab displays the following fields:

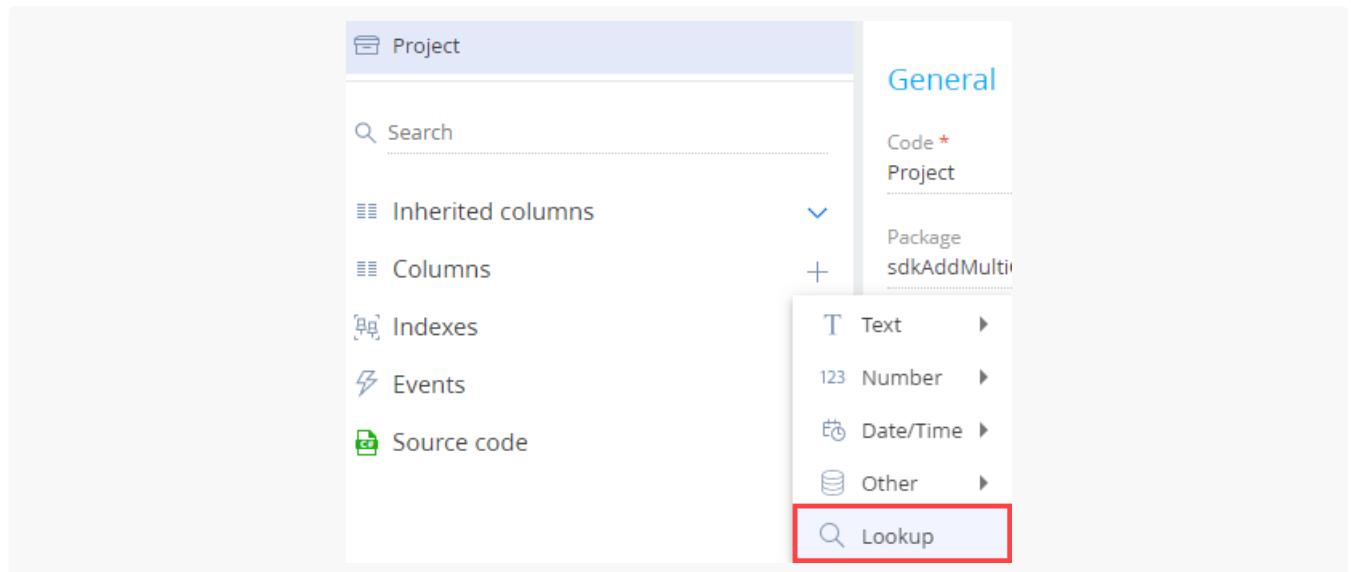
Code *	Title *
Project	Project
Package	Description
sdkAddMultiCurrencyFieldToPagePackage	(empty)

The 'Inheritance' tab displays the following field:

Parent object *	<input checked="" type="checkbox"/> Replace parent
Project	(empty)

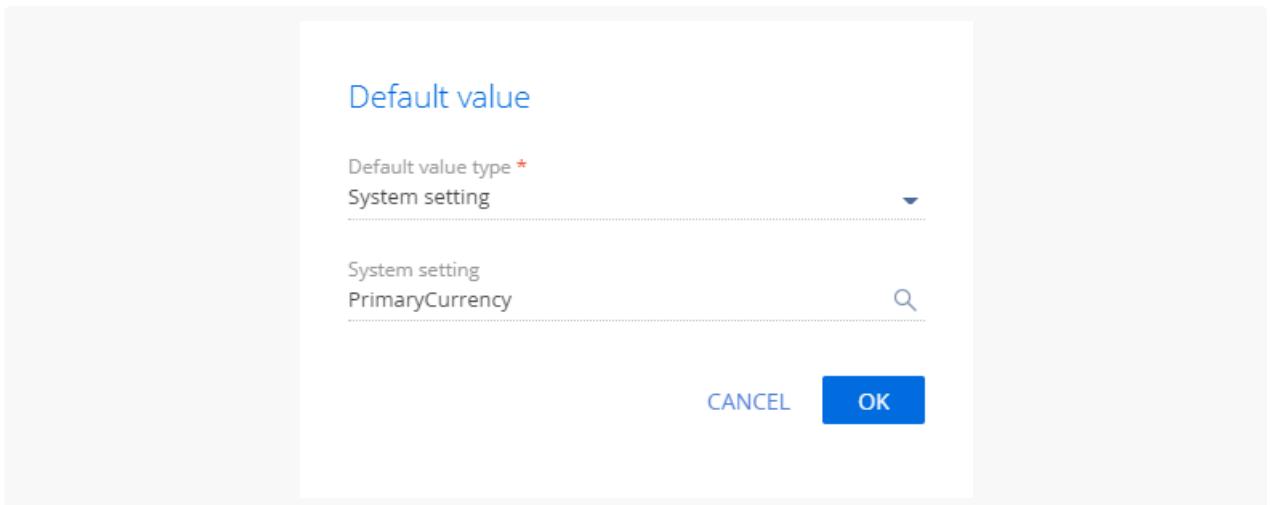
4. В схему добавьте **колонку**.

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

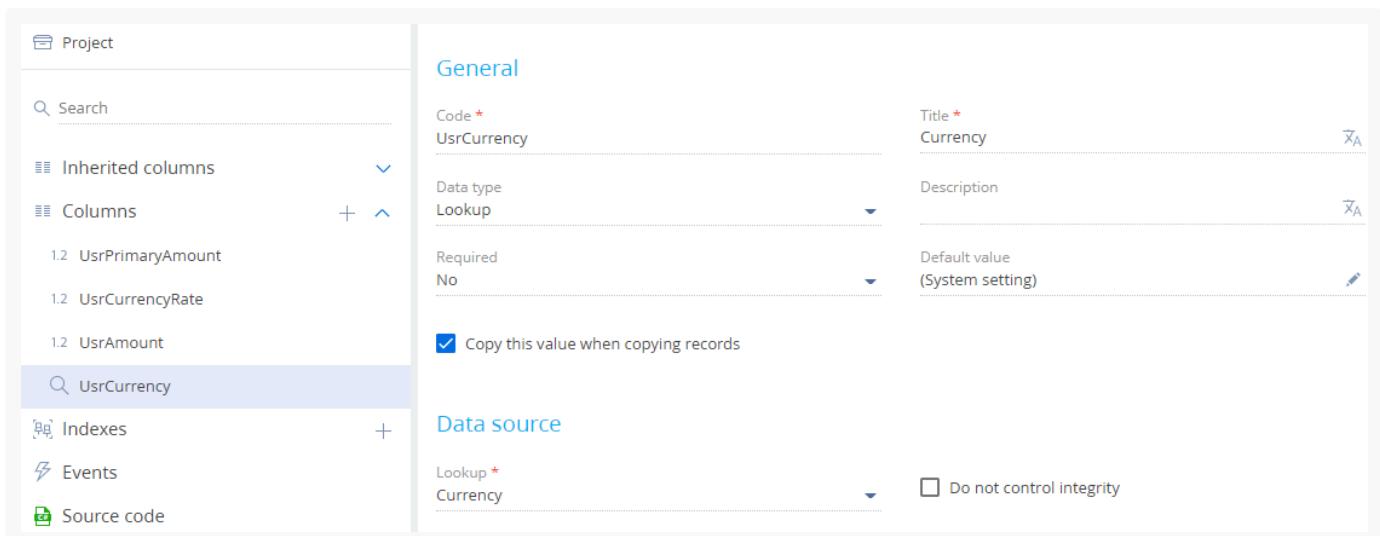


с. Заполните **свойства добавляемой колонки**.

- [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.



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

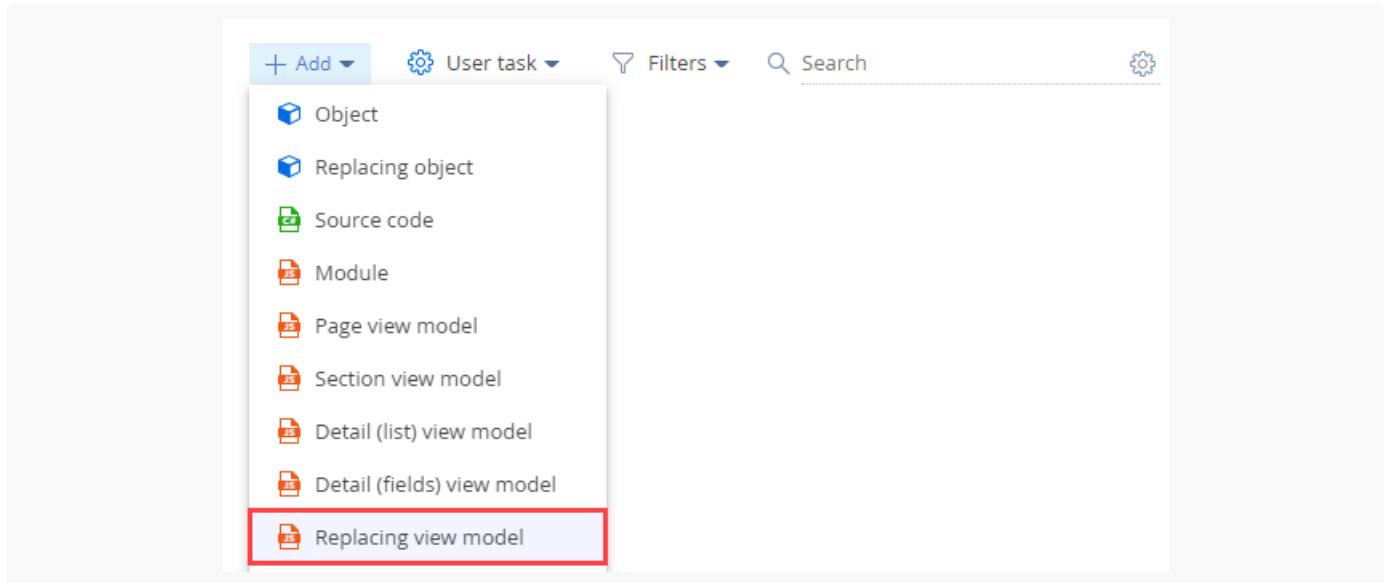
Свойства добавляемых колонок

[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]).

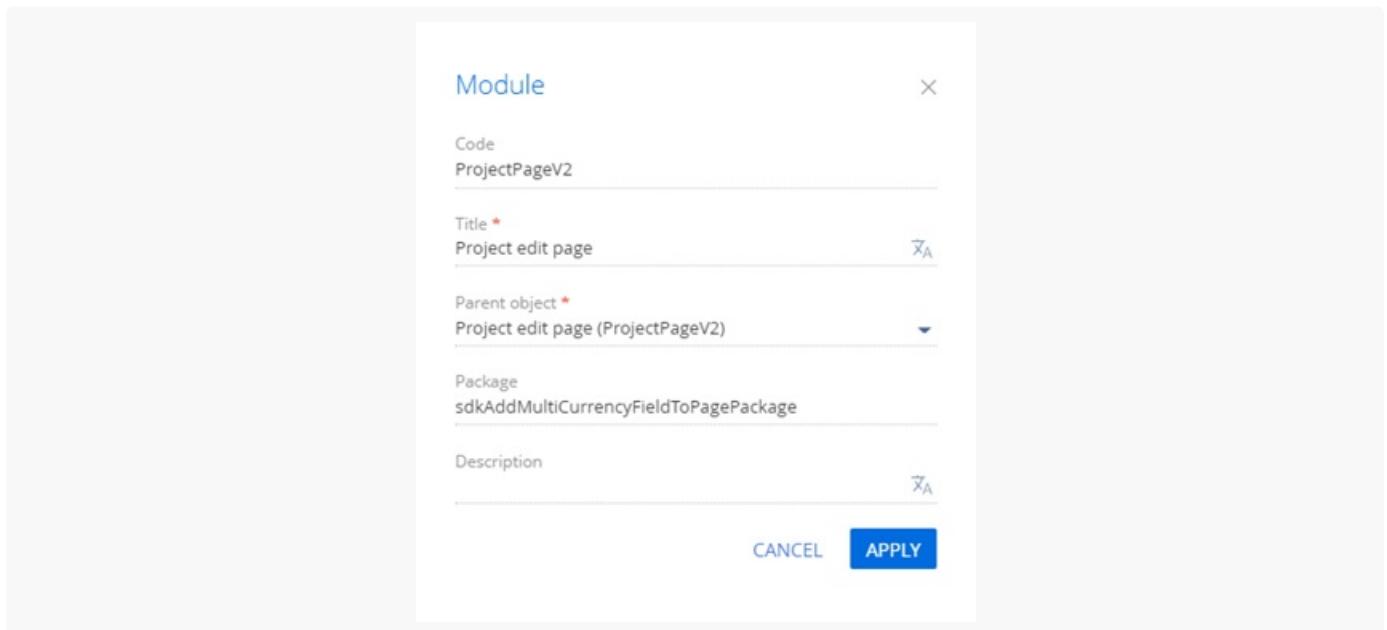
2. Создать схему замещающей модели представления страницы проекта

- Перейдите в раздел [Конфигурация] ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- 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` объявили 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 колонки */
    "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": {
    dataType: this.Terrasoft.DataValueType.COLLECTION,
    value: this.Ext.create("Terrasoft.Collection")
}

```

```

},
/* Коллекция для кнопки выбора валюты. */
"CurrencyButtonMenuList": {
    dataType: 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? > Creatio 7.18.4.1532

SAVE **CANCEL** **ACTIONS** **VIEW**

Name* Software Documentation Development

Status* In progress

Owner* Sarah M. Richards

Amount, aud ▾ 20.01

aud
€
hrn.
rub.
\$

GENERAL INFORMATION 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? > Creatio 7.18.4.1532

SAVE **CANCEL** **ACTIONS** **VIEW**

Name* Software Documentation Development

Status* In progress

Owner* Sarah M. Richards

Amount, \$ ▾ 16.00

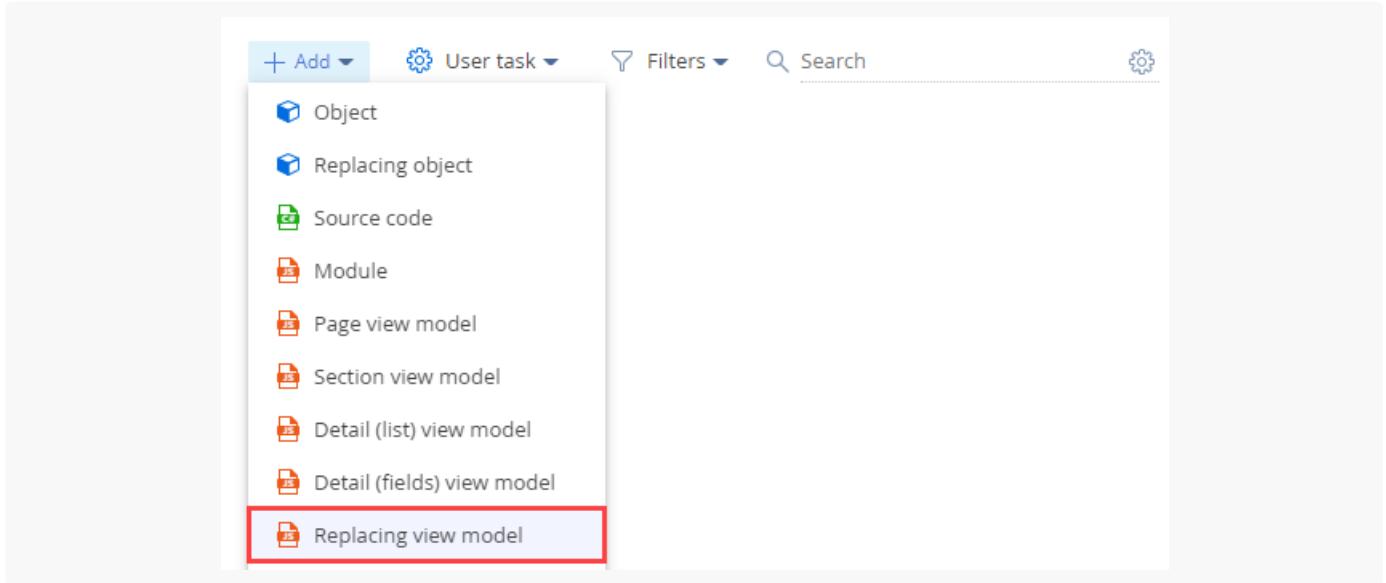
Реализовать валидацию поля типа [Дата/Время] на странице записи



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

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

The screenshot shows the 'Module' configuration dialog box. The fields are as follows:

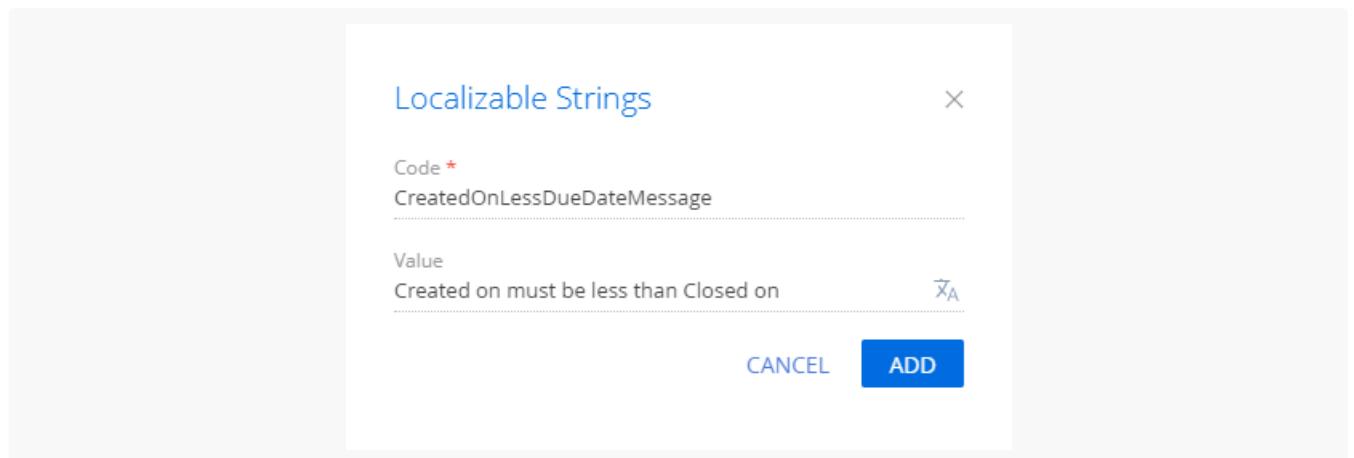
- Code:** OpportunityPageV2
- Title ***: Opportunity edit page
- Parent object ***: Opportunity edit page (OpportunityPageV2)
- Package**: sdkDateTimeFieldValidationPackage
- Description**: (empty)

At the bottom right are the **CANCEL** and **APPLY** buttons.

4. Добавьте **локализуемую строку**.

- В контекстном меню узла [Локализуемые строки] ([Localizable strings]) нажмите кнопку [+](#).
- 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 валидации */
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

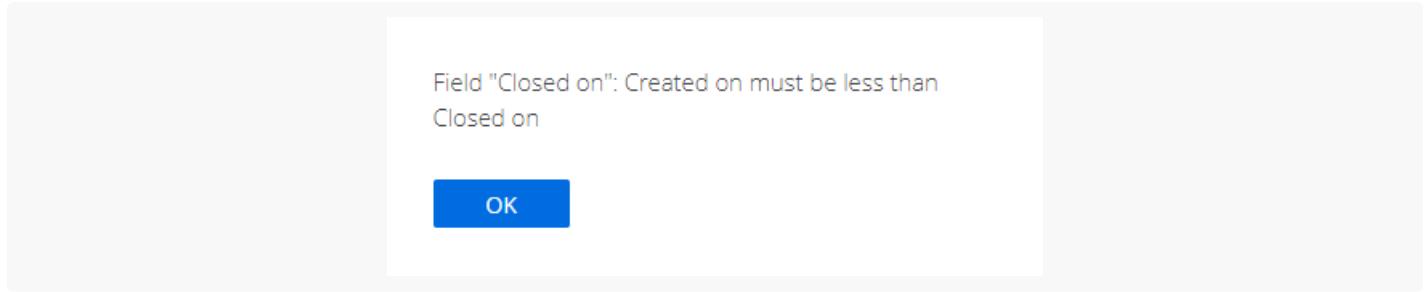
Чтобы **посмотреть результат выполнения примера**:

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.

The screenshot shows the 'OPPORTUNITY DETAILS' tab selected in the header. On the left, there's a sidebar with sections like 'Budget' (14,500.00), 'Decision maker', 'Customer need*' (Hardware), and two date fields: 'Closed on' (9/20/2021) and 'Created on' (9/24/2021). Both date fields have red borders around them, indicating validation errors. A blue tooltip message 'Created on must be less than Closed on' appears above the 'Created on' field. Other visible fields include 'Name' (019 / Sunrise Investments / Sale of Goods), 'Opportunity amount' (13,533.60), 'Probability, %' (0), 'Category' (Medium business), 'Type', 'Division', 'Owner' (Marina Kysla), and 'Source'. The 'Description' section is partially visible at the bottom.

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



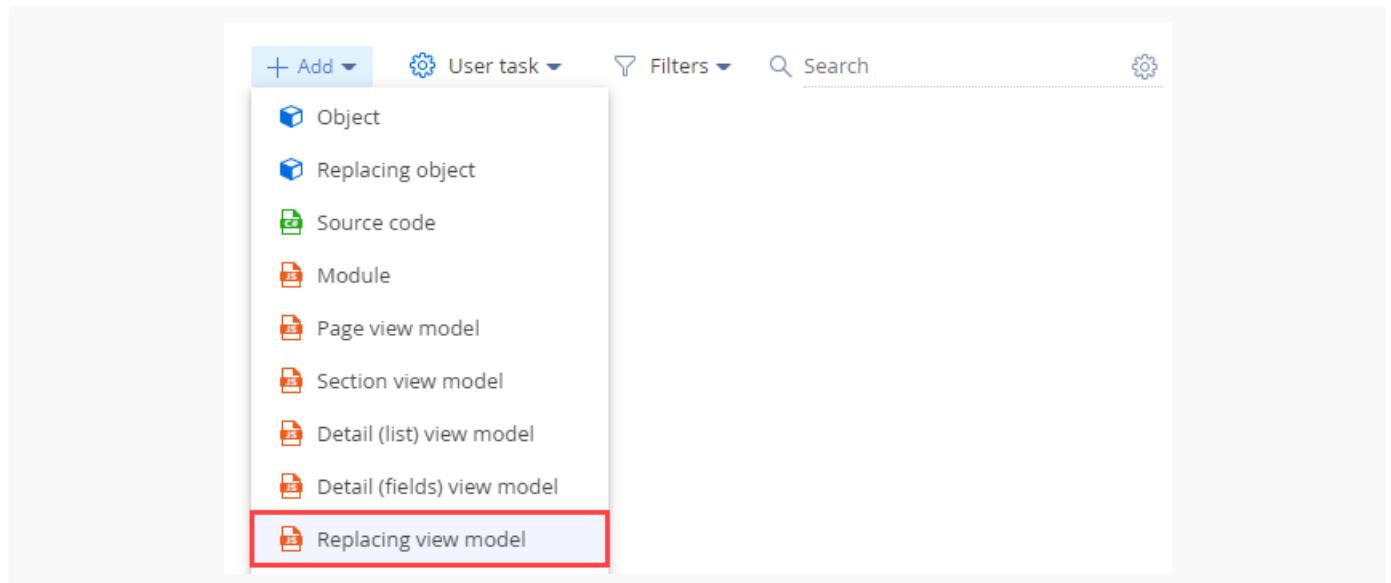
Реализовать валидацию поля типа [Строка] на странице записи

Средний

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

Создать схему замещающей модели представления страницы контакта

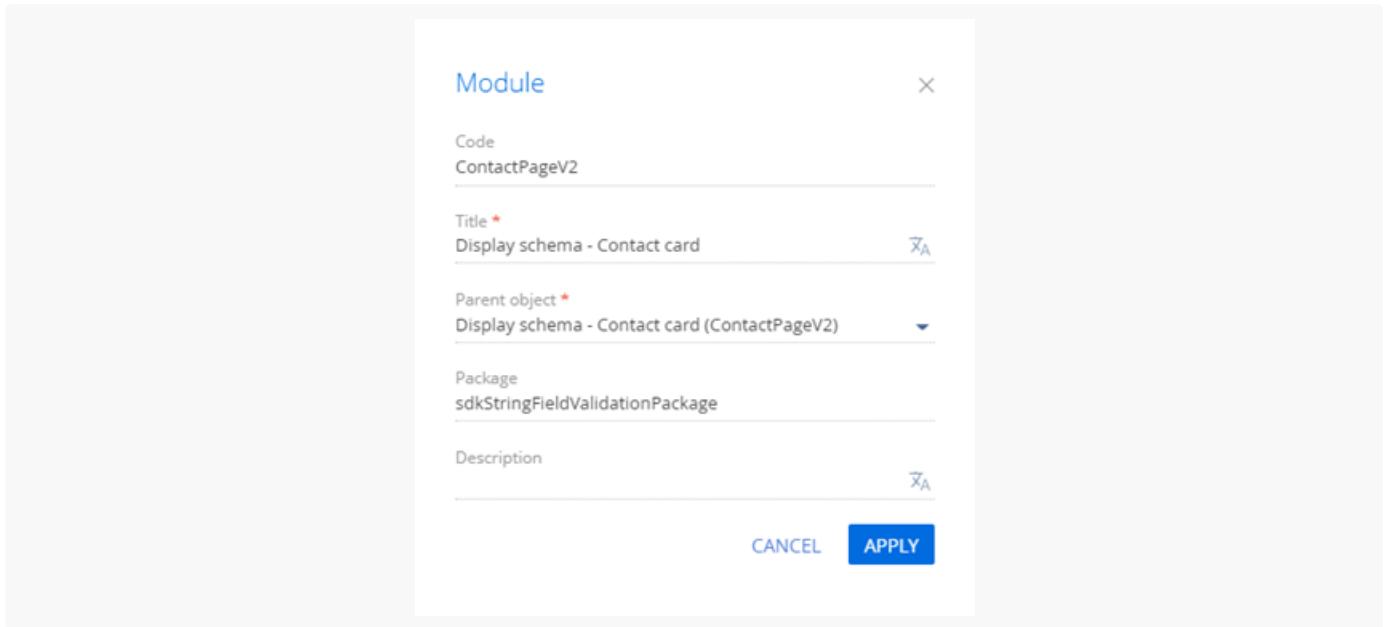
1. [Перейдите в раздел \[Конфигурация \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [Добавить] —> [Замещающая модель представления] ([*Add*] —> [*Replacing view model*]).



3. Заполните **свойства схемы**.

- [Код] ([*Code*]) — "ContactPageV2".

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

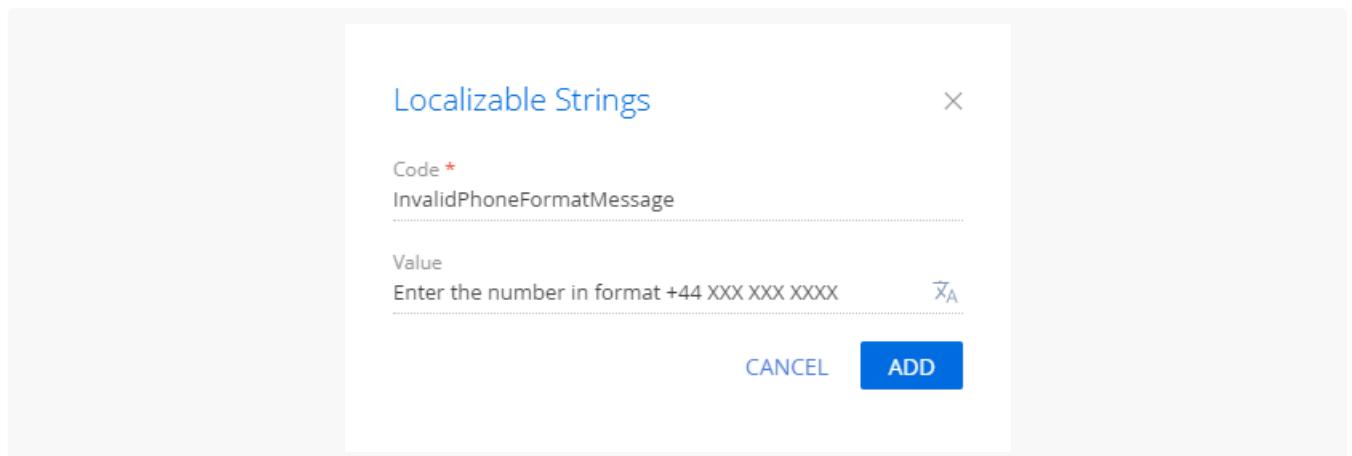


4. Добавьте локализуемую строку.

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

b. 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").



e. 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 валидации. */
            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 валидно, то значение null. */
                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? >

Creatio
7.18.4.1532

ACTIONS ▾

SAVE CANCEL

100%

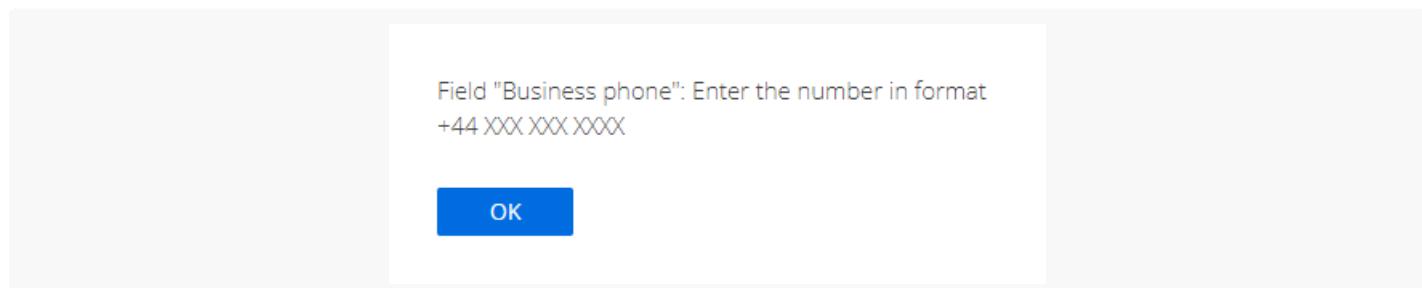
12:28 AM, New York

NEXT STEPS (8)

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

Business phone
+1 212 542 4238
Enter the number in format +44 XXX XXX XXXX
XXXX
a.wilson@alpha-business.com

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



Установить значение по умолчанию для

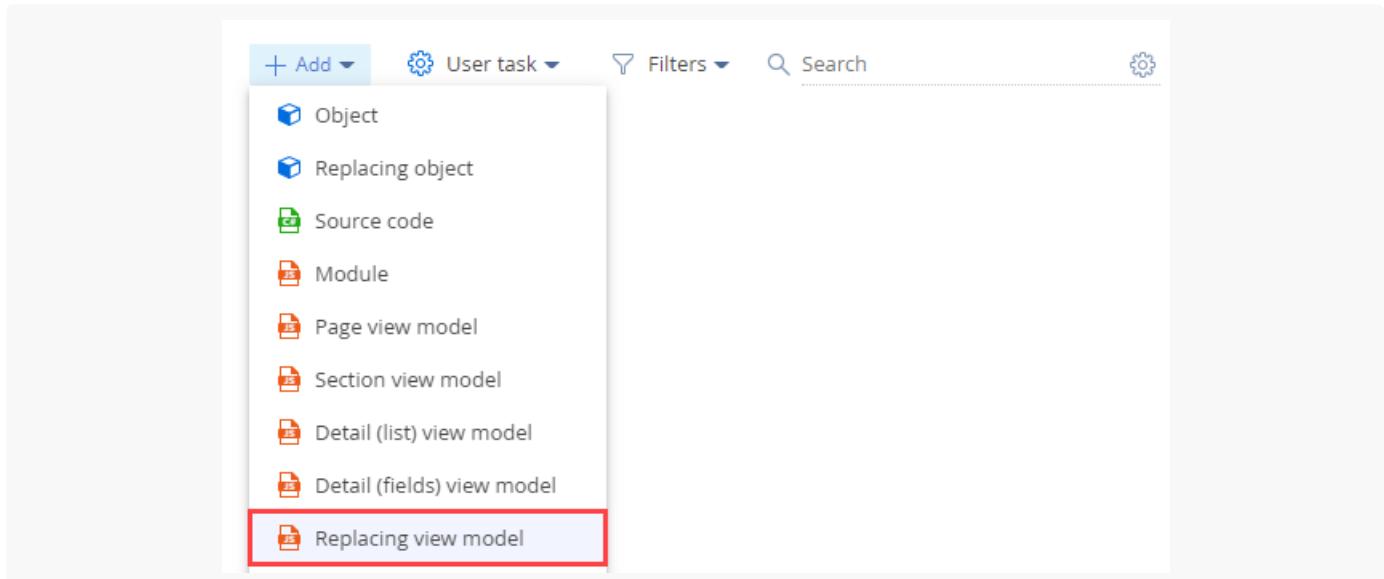
Поля на странице записи

 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]).

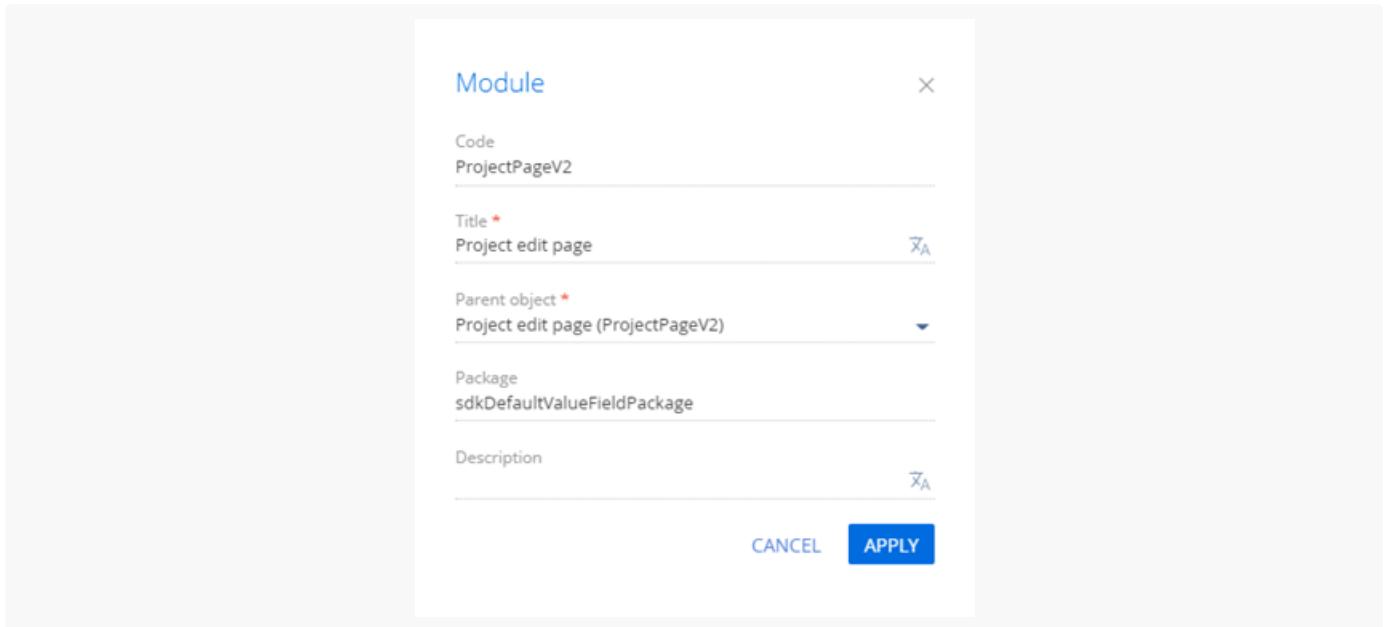
Создать схему замещающей модели представления страницы проекта

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]).

GENERAL INFORMATION		STRUCTURE	FINANCIAL INDICATORS	HISTORY	ATTACHMENTS AND NOTES	FEED
Account Completion % Calculate automatically <input type="checkbox"/> Start 11/19/2021 End		Contact Type*	Duration 0 min Deadline 11/29/2021			

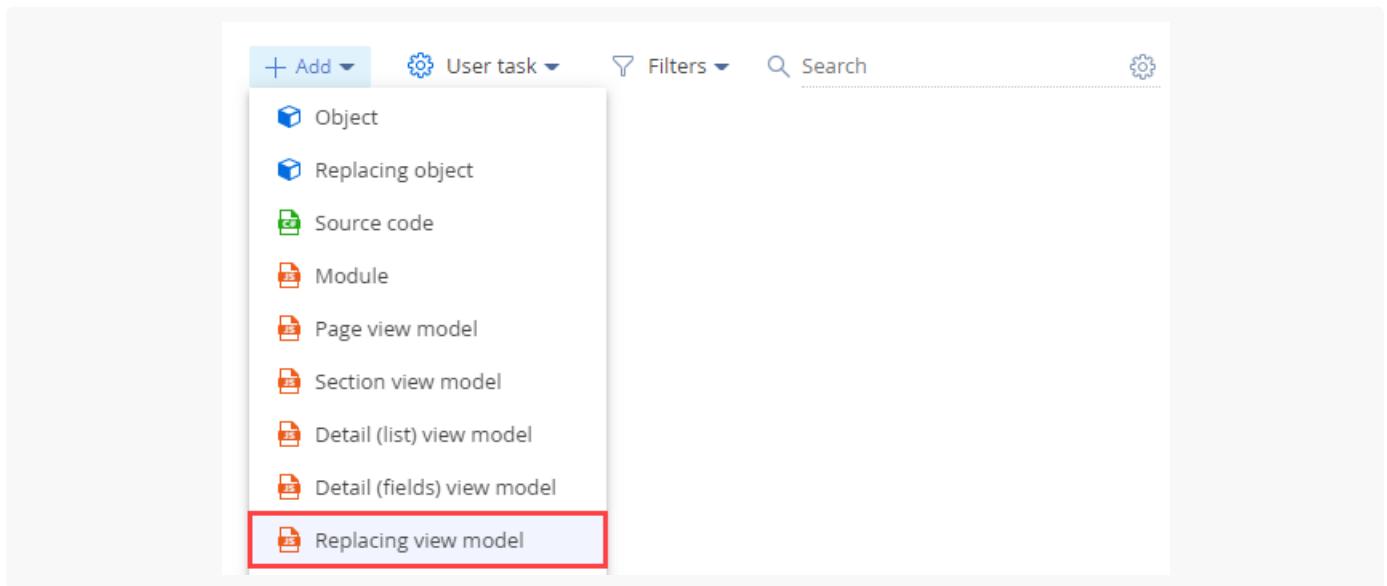
Настроить обязательность для поля на странице записи

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".

The screenshot shows the 'Module' configuration dialog with the following fields filled in:

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

At the bottom right are the 'CANCEL' and 'APPLY' buttons.

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 значение */
                        "conditions": [
                            /* Выражение левой части условия. */
                            "leftExpression": {
                                /* Тип выражения – атрибут (колонка) модели представления. */
                                "type": BusinessRuleModule.enums.ValueType.ATTRIBUTE,
                                /* Название колонки модели представления, значение которой сравнивается */
                                "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

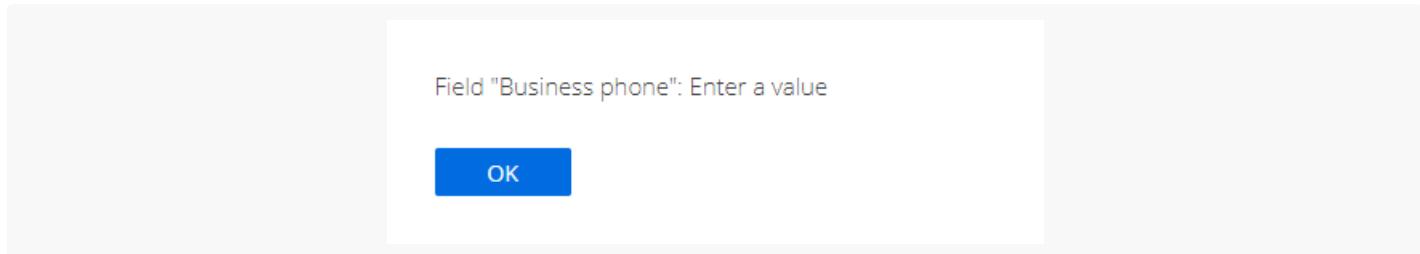
Чтобы **посмотреть результат выполнения примера**:

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

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

Field	Value
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
Type	Customer
Owner	Marina Kysla
Gender	Male
Preferred language	English (United States)
Age	49

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



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

Field	Value
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

Настроить фильтрацию значений справочного поля на странице записи

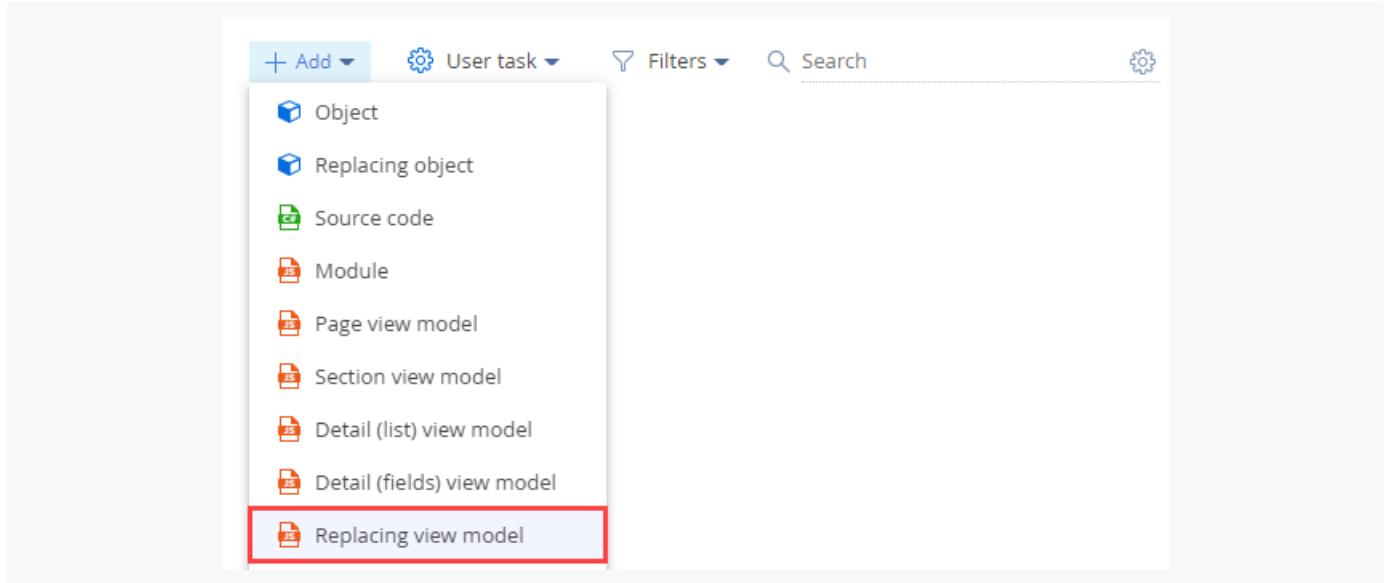


Пример. Настроить фильтрацию контактов, которые доступны для выбора при заполнении справочного поля [Ответственный] ([Owner]) страницы контрагента. В окне выбора контактов отображать:

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

Создать схему замещающей модели представления страницы контрагента

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

The dialog box has the following fields:

- Code**: AccountPageV2
- Title ***: Account edit page
- Parent object ***: Account edit page (AccountPageV2)
- Package**: sdkLookupFieldFiltrationPackage
- Description**: (empty)

At the bottom are **CANCEL** and **APPLY** buttons.

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 присоединен
                             filterGroup.add("IsUser", Terrasoft.createColumnIsNotNullFilter("*/
                            /* Добавляет фильтр "IsActive" в результирующую коллекцию фильтров.
                             Выбирает все записи из корневой схемы [Contact], к которой присоединен
                             filterGroup.add("IsActive",
                                Terrasoft.createColumnFilterWithParameter(
                                    Terrasoft.ComparisonType.EQUAL,
                                    "[SysAdminUnit>Contact].Active",
                                    true));
                            return filterGroup;
                        }
                    ]
                }
            }
        };
    });
});
```

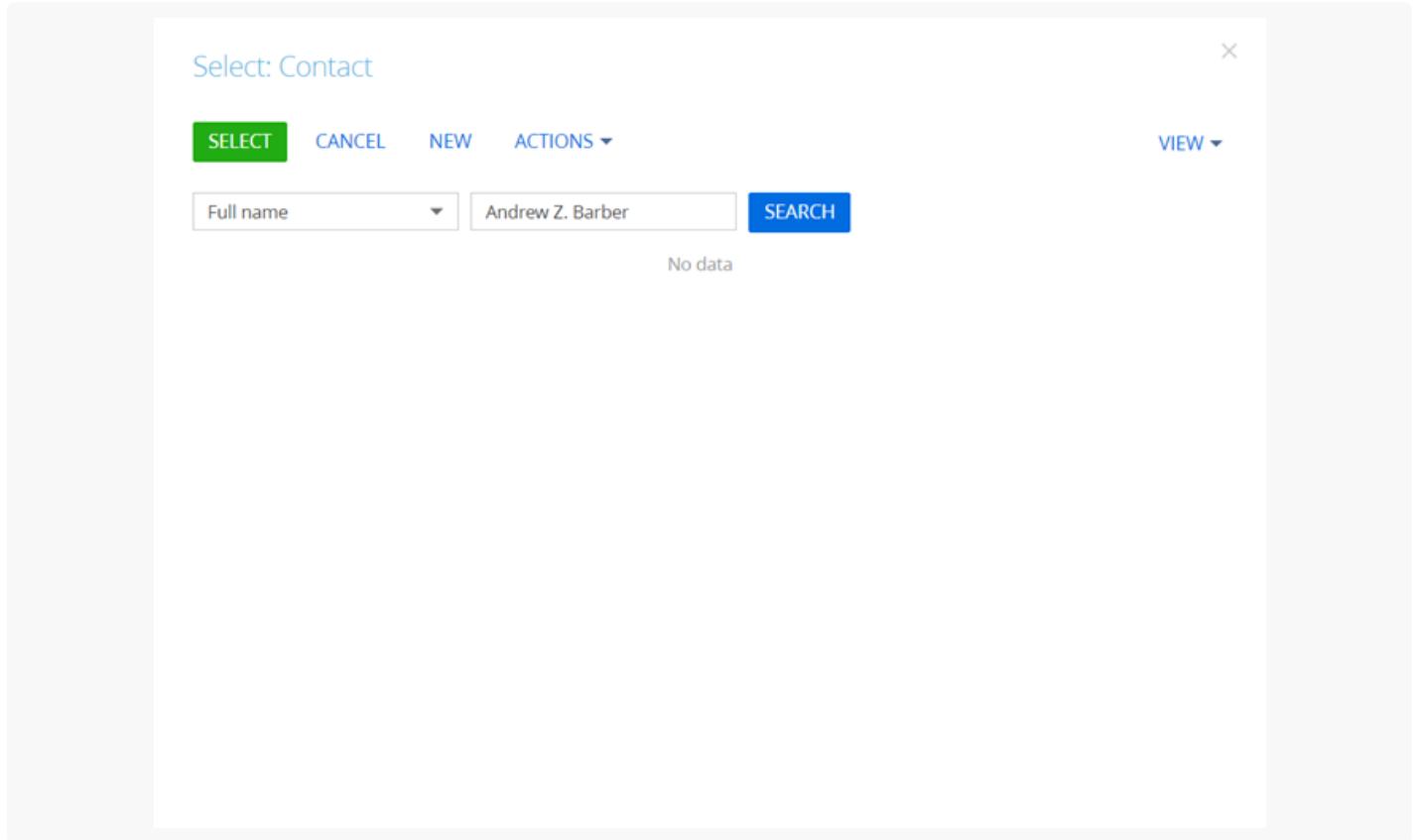
5. На панели инструментов дизайнера нажмите [Сохранить] ([Save]).

Результат выполнения примера

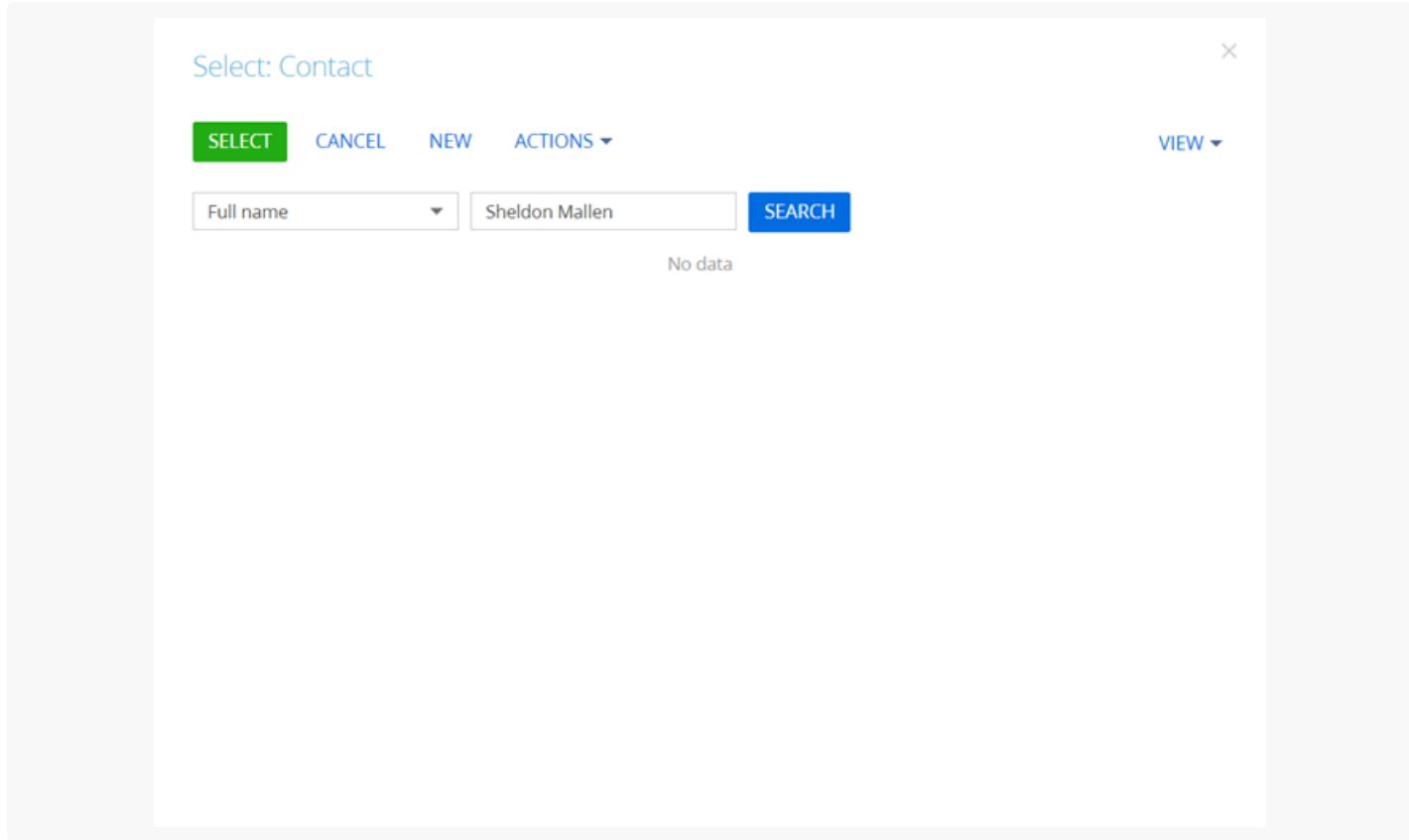
Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [Контрагенты] ([Accounts]).

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.



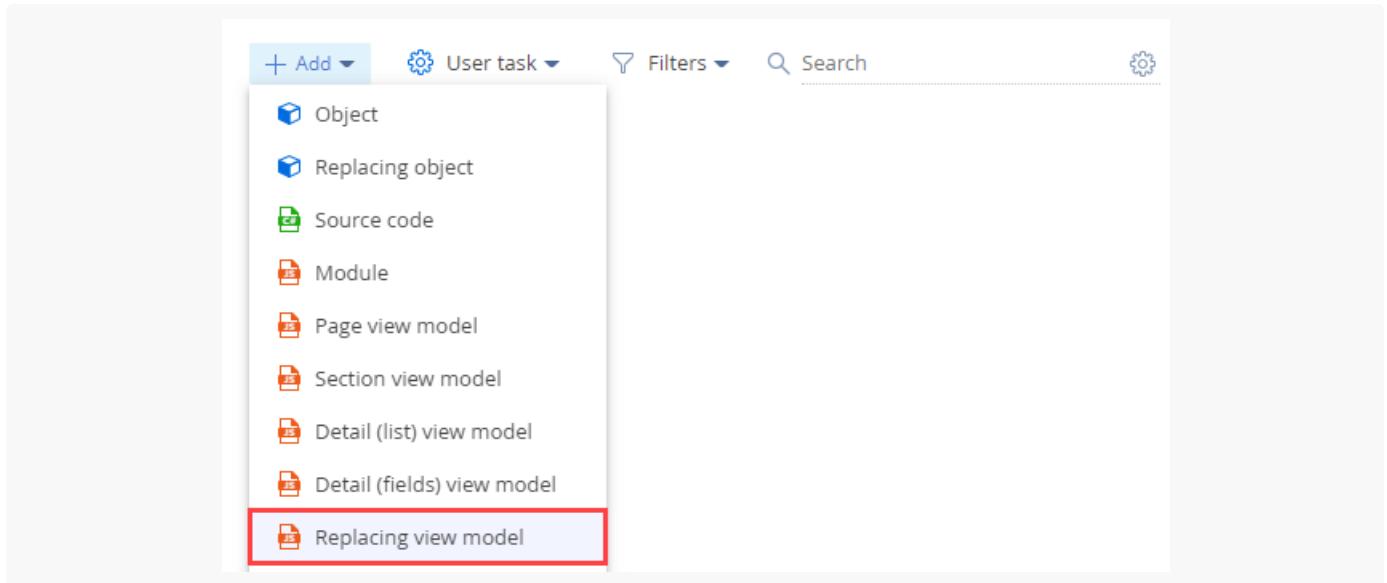
Настроить фильтрацию значений связанных справочных полей на странице записи

 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*]).

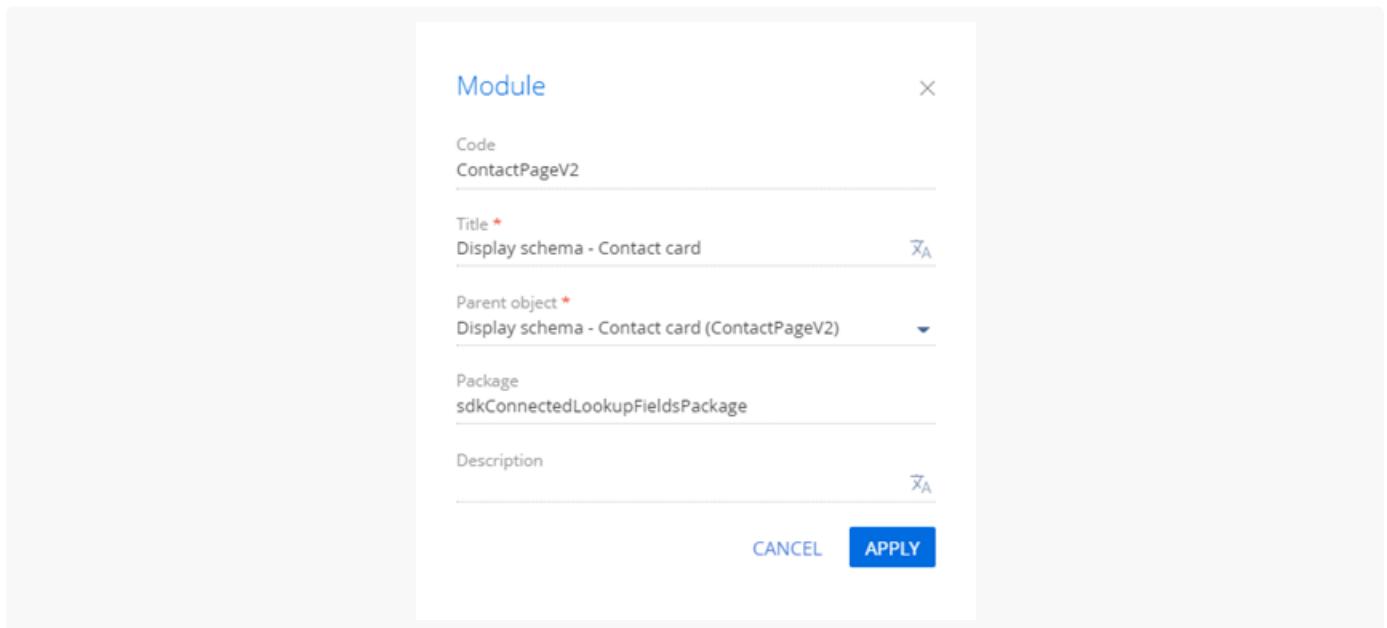
Создать схему замещающей модели представления страницы контакта

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 сравнивается */
                        "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.



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

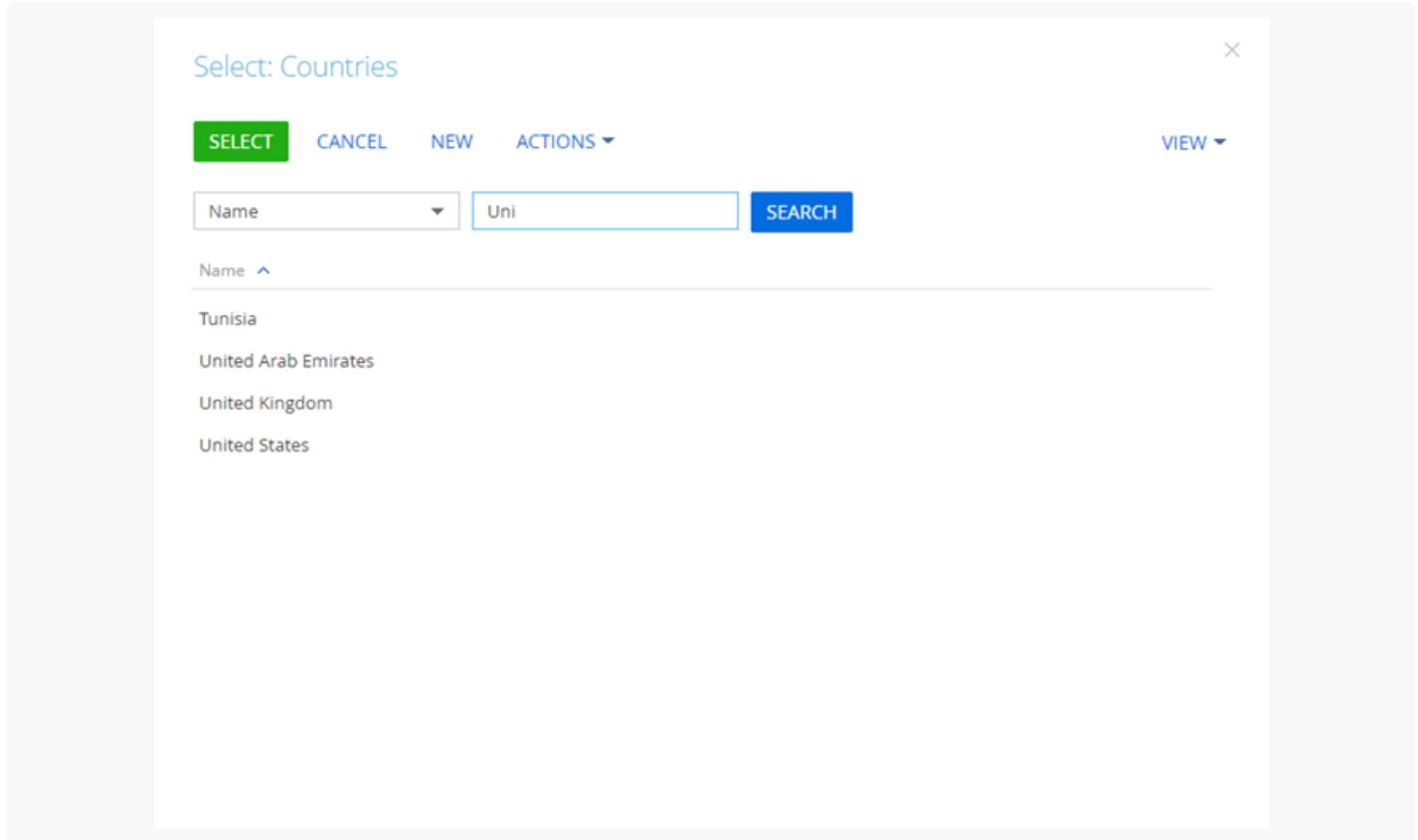
a.wilson@alphabusiness.com

Country

🔍

- United Arab Emirates
- United Kingdom
- United States
- Tunisia
- New Uni

Также фильтрация выполняется в окне выбора страны.



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.

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

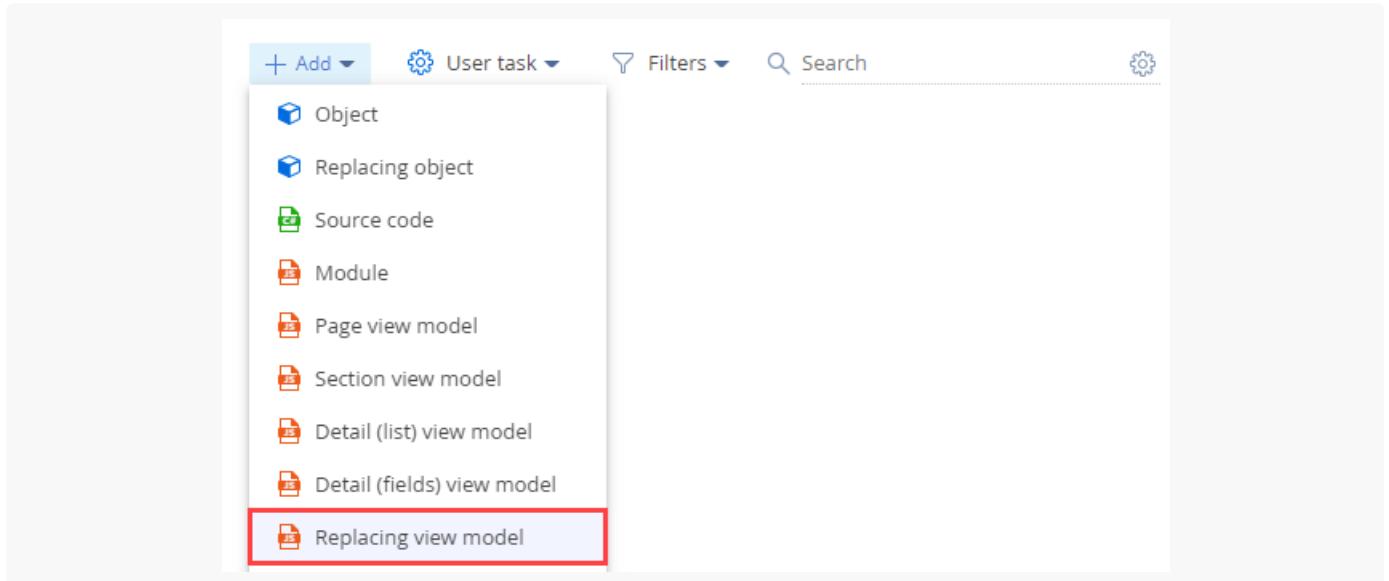
Средний

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

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

Code	ContactPageV2
Title *	Display schema - Contact card
Parent object *	Display schema - Contact card (ContactPageV2)
Package	sdkBlockFieldConditionPackage
Description	

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 значение в колонке [MobilePhone]. */
                    "conditions": [
                        /* Выражение левой части условия. */
                        "leftExpression": {
                            /* Тип выражения – атрибут (колонка) модели представления. */
                            "type": BusinessRuleModule.enums.ValueType.ATTRIBUTE,
                            /* Название колонки модели представления, значение которой сравнивается с константным значением. */
                            "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": ""
                        }
                    ]
                }
            }
        }
    }
})
```

```

    }
};

});

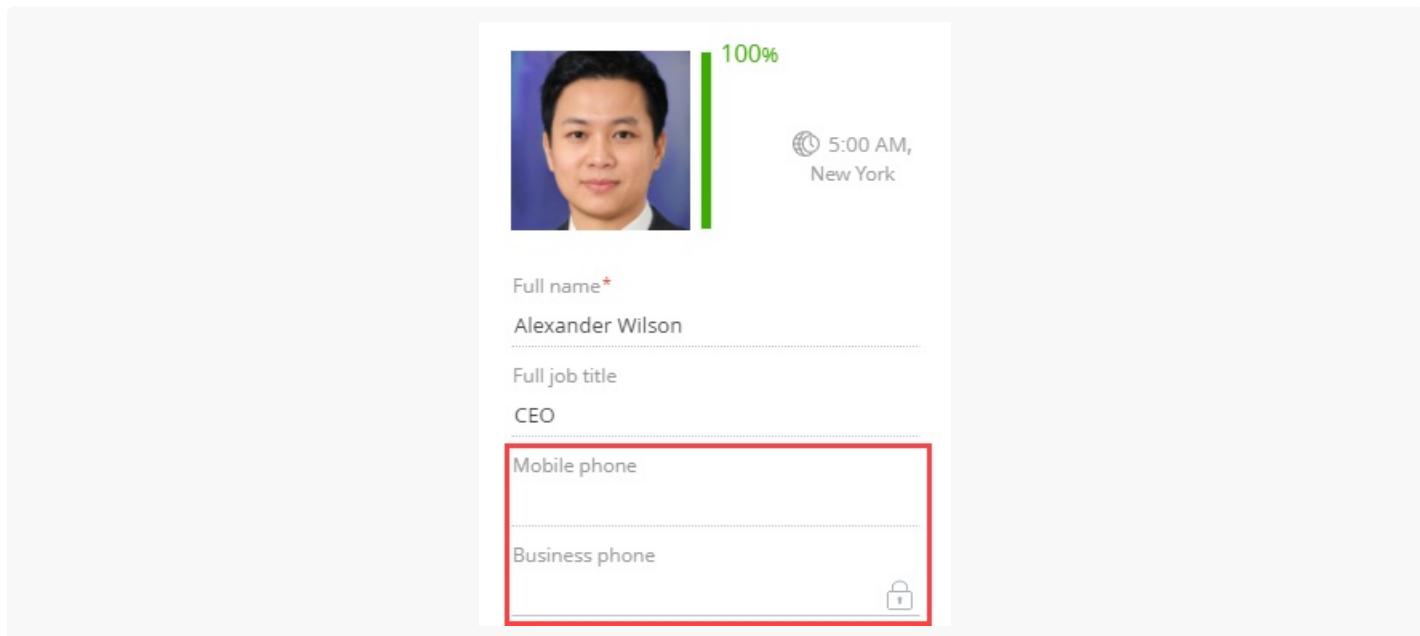
```

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]).



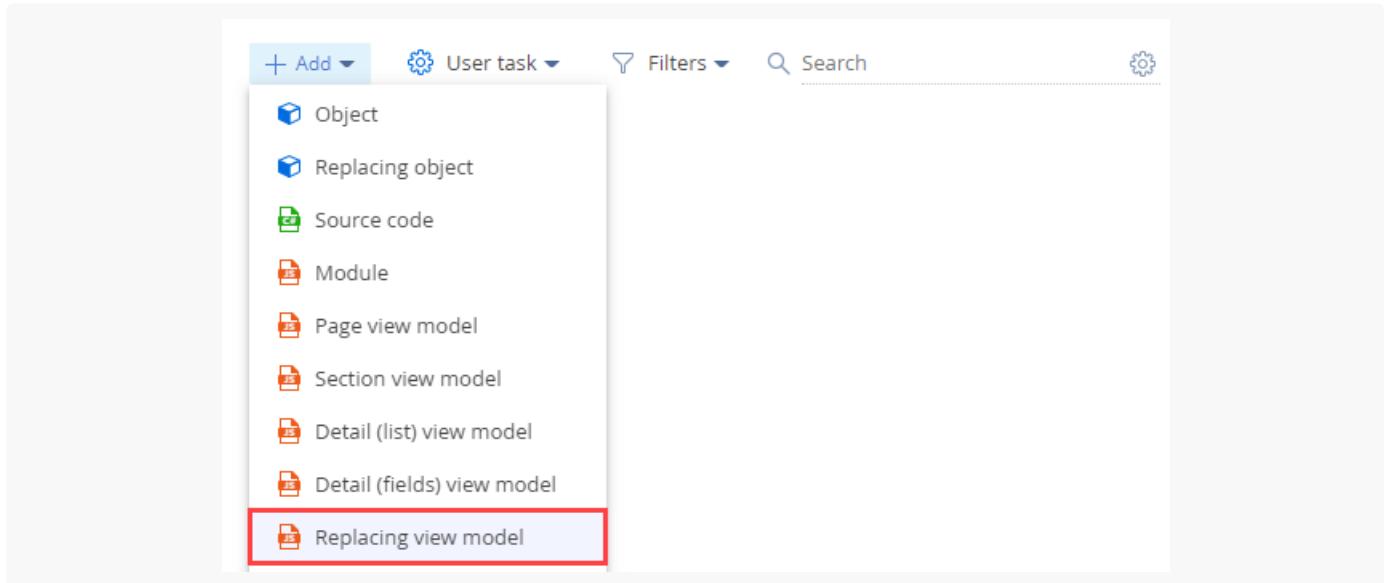
Настроить исключения блокировки полей на странице записи

Средний

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

Создать схему замещающей модели представления страницы счета

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

The screenshot shows the 'Module' configuration dialog with the following fields:

- Code: InvoicePageV2
- Title: Invoice edit page
- Parent object: Invoice edit page (InvoicePageV2)
- Package: sdkBlockFieldsPackage
- Description: (empty)

At the bottom are 'CANCEL' and 'APPLY' buttons.

- В объявлении класса модели представления в качестве зависимостей добавьте модуль `InvoiceConfigurationConstants`.
- 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 блокировать 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. */
        dataType: Terrasoft.DataValueType.BOOLEAN,
        value: true,
        /* Массив конфигурационных объектов, которые определяют зависимости атрибута
        dependencies: [
          /* Значение колонки [IsModelItemsEnabled] зависит от значения колонки [Ра
          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 "Оплачено 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	Customer details Partners, USD
Supplier Our company	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]) выбрано значение "Встреча" ("Meeting").

1. Создать схему замещающего объекта

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

The screenshot shows a list of objects. The 'Replacing object' item is highlighted with a red box. The columns are 'Status' and 'Type'. Other items include 'Object', 'Source code', and 'Module'.

Status	Type
	Object
	Replacing object
	Source code
	Module

3. Заполните **свойства схемы**.

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

The 'General' tab shows the following fields:

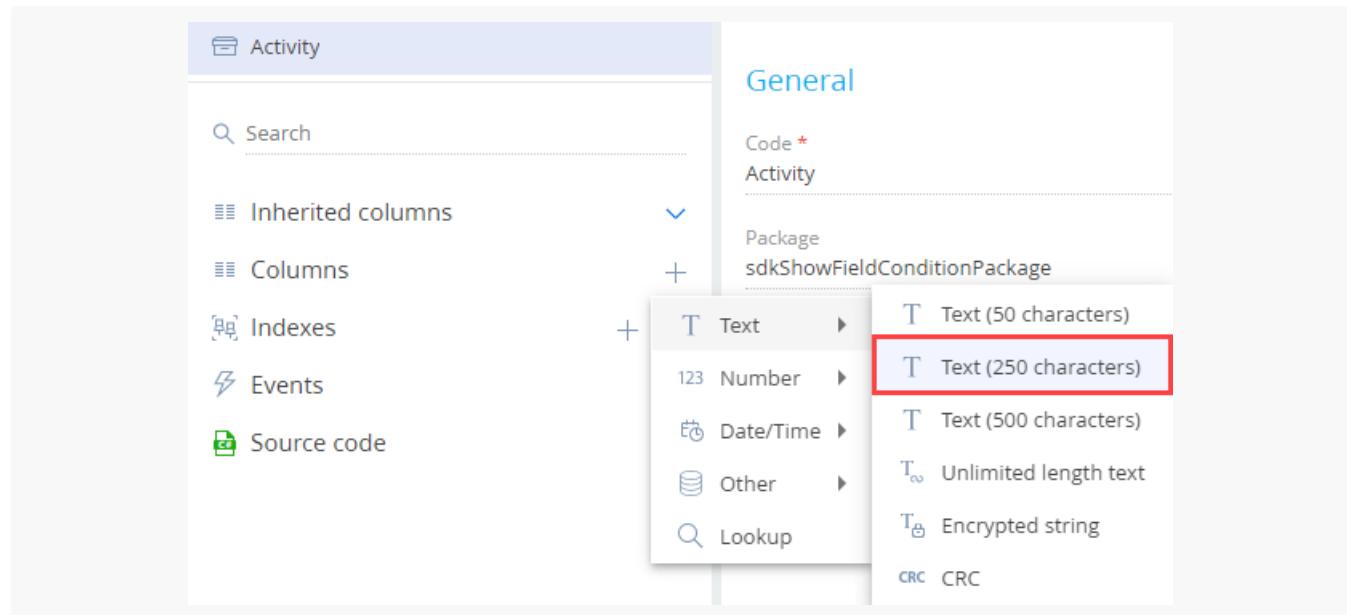
- Code ***: Activity
- Title ***: Activity
- Package**: sdkShowFieldConditionPackage
- Description**: (empty)

The 'Inheritance' tab shows:

- Parent object ***: Activity
- Replace parent

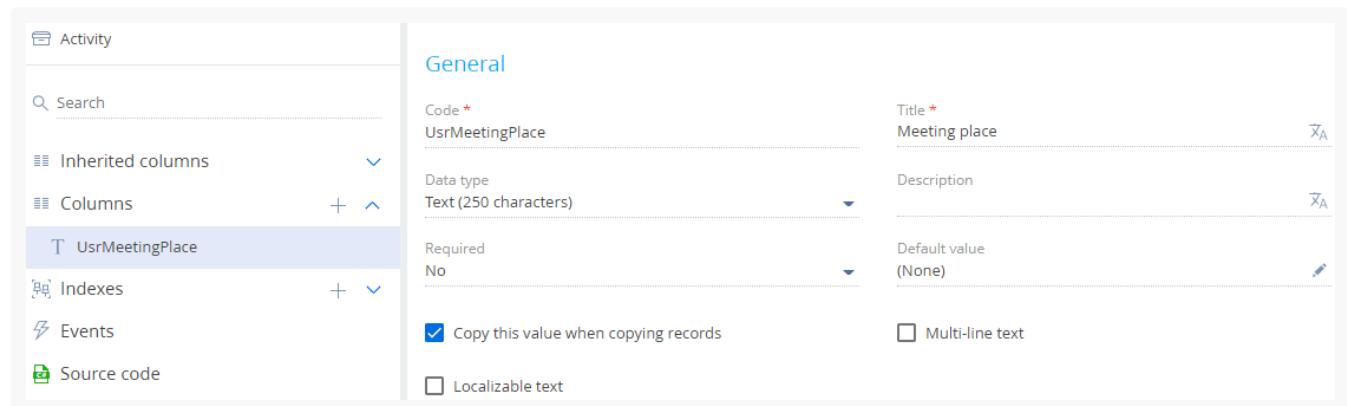
4. В схему добавьте **колонку**.

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



с. Заполните **свойства добавляемой колонки**.

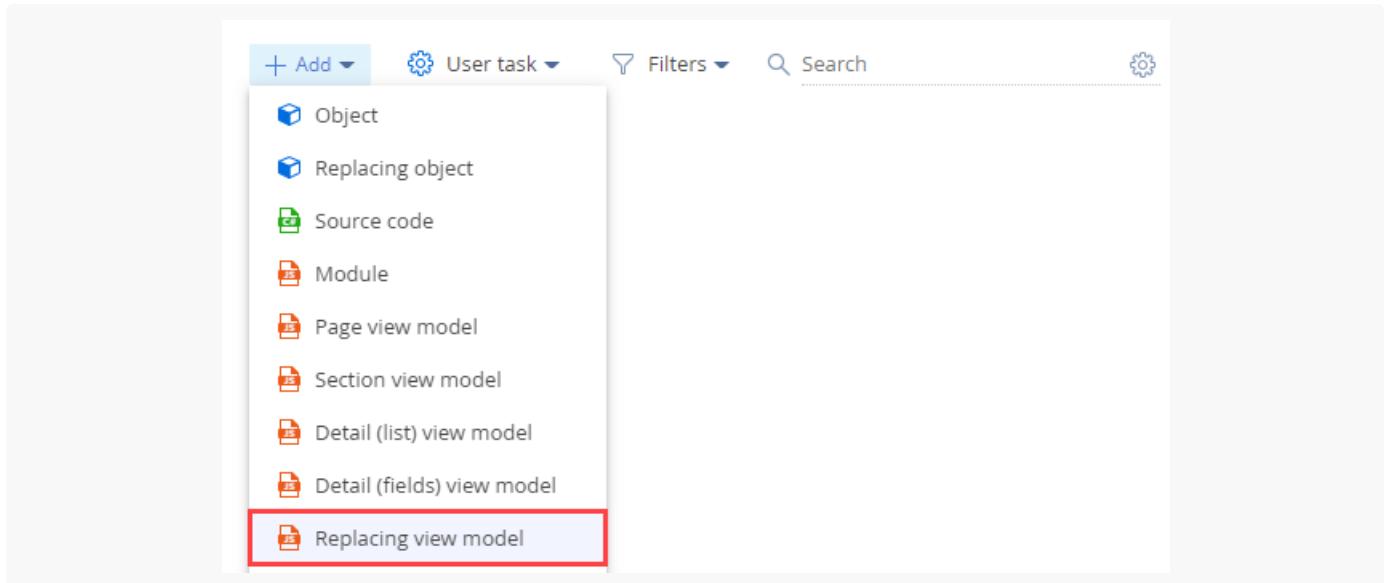
- [Код] ([Code]) — "UsrMeetingPlace".
- [Заголовок] ([Title]) — "Место встречи" ("Meeting place").



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

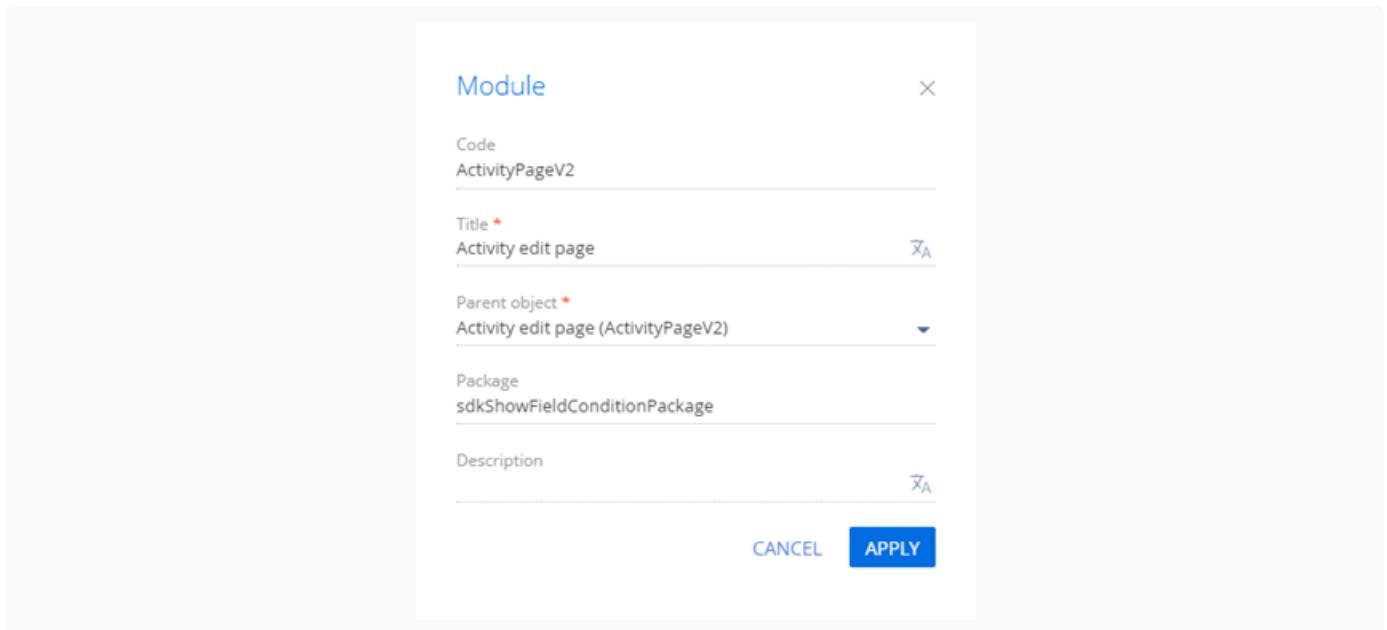
2. Создать схему замещающей модели представления страницы активности

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]).



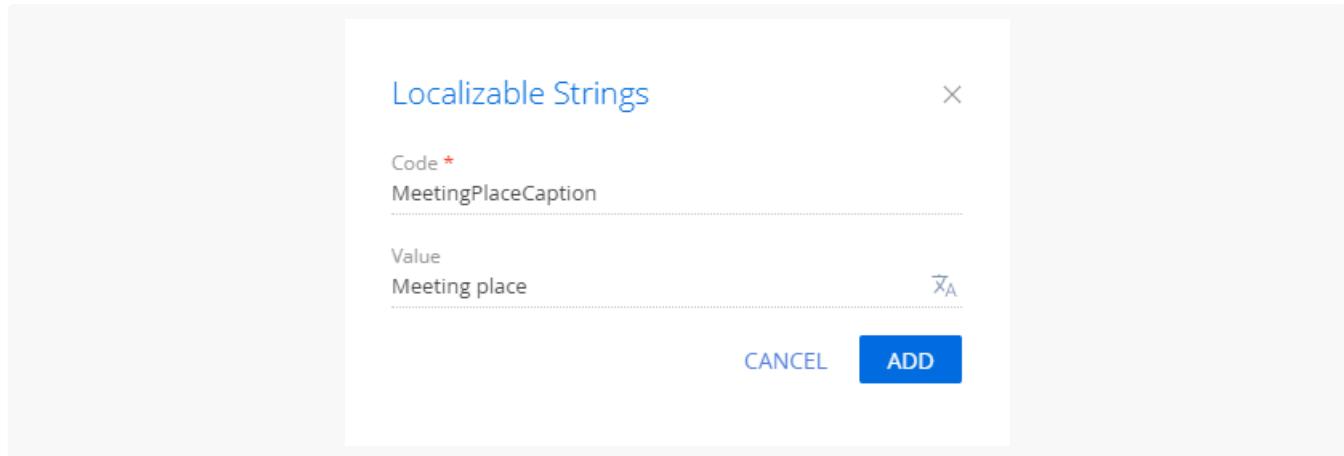
3. Заполните **свойства схемы**.

- [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").



- e. 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**.
 - a. В свойство `rules` для колонки [*UsrMeetingPlace*]:
 - a. В свойстве `ruleType` укажите значение `BINDPARAMETER`, которое задает тип бизнес-правила. Типы правил представлены перечислением `BusinessRuleModule.enums.RuleType` .
 - b. В свойстве `property` укажите значение `VISIBLE`, которое устанавливает видимость колонки. Свойства бизнес-правила `BINDPARAMETER` представлены перечислением `BusinessRuleModule.enums.Property` .
 - c. 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*]).
 - b. 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**:

- Обновите страницу раздела [Активности] ([Activities]).
- 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 a task creation form titled 'Test task'. The form includes fields for Subject, Start date, Due date, Status, Show in calendar, Role, Owner, Reporter, Priority, and Category. The 'Meeting place' field and the 'Category' field (set to 'Meeting') are highlighted with red boxes.

Field	Value
Subject*	Test task
Start*	11/23/2021
Due*	10:00 AM 11/23/2021
Status*	Not started
Show in calendar	<input checked="" type="checkbox"/>
Meeting place	
Category*	Meeting

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 ▾

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.
 - 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*]) — "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}".

The screenshot shows the 'Product code mask' configuration screen. It includes fields for Name (Product code mask), Type (Unlimited length text), Default value (ART_{0:00000}), and Description. On the right, there are settings for Code (ProductCodeMask), Cached (checked), and Save value for current user (unchecked). Buttons for SAVE and CANCEL are at the top.

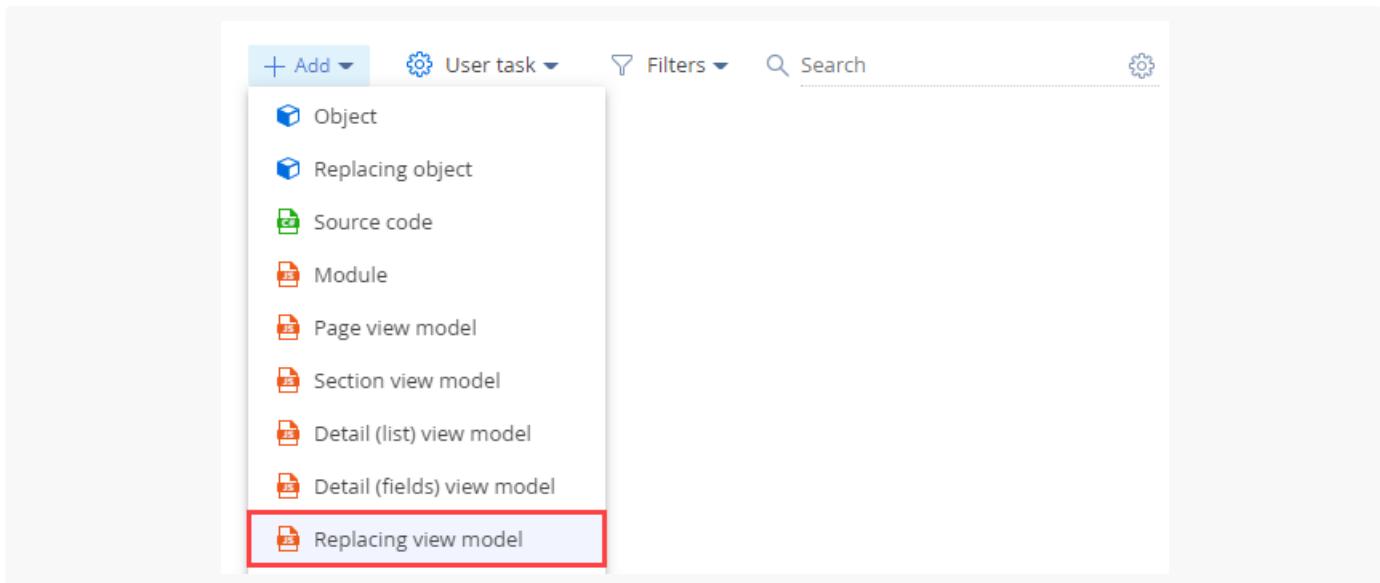
2. Создайте [системную настройку](#) с текущим кодом продукта.

- 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").

The screenshot shows the 'Product last number' configuration screen. It includes fields for Name (Product last number), Type (Integer), Default value (0), and Description. On the right, there are settings for Code (ProductLastNumber), Cached (checked), and Save value for current user (unchecked). Buttons for SAVE and CANCEL are at the top.

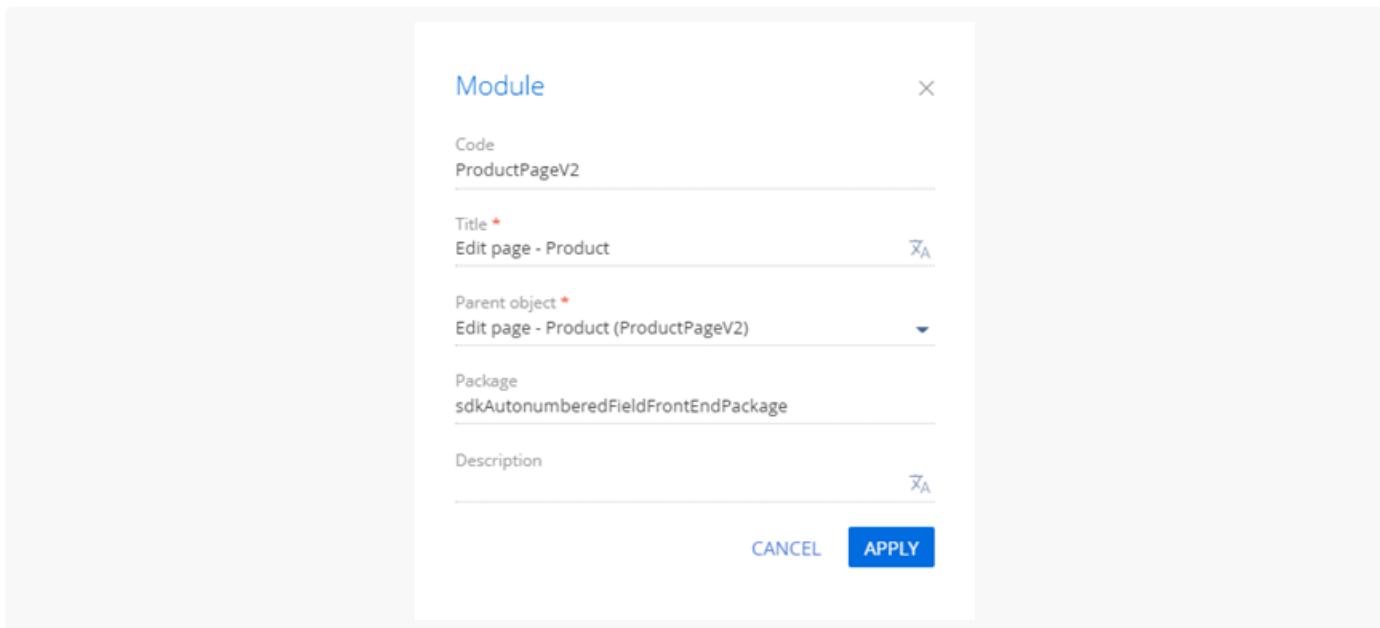
2. Создать схему замещающей модели представления страницы продукта

- Перейдите в раздел [Конфигурация] ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- 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, который генерирует автонумерацию для нового элемента. */
            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 генерирует и возвращает автонумерацию. */
                    this.getIncrementCode(function(response) {
                        /* Сгенерированный номер возвращается в колонку [Code]. */
                        this.set("Code", response);
                    });
                }
            }
        }
    };
});

```

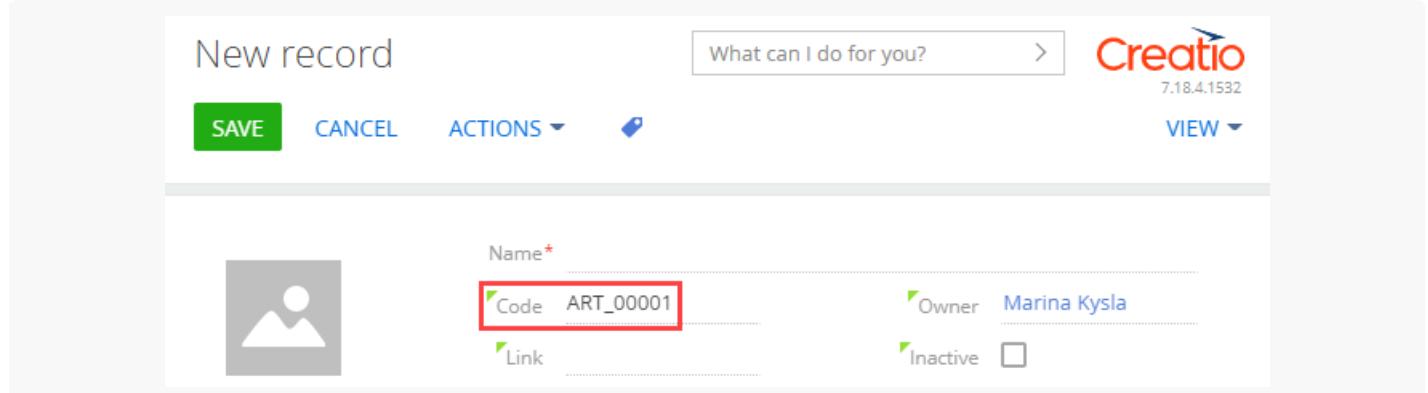
- На панели инструментов дизайнера нажмите [Сохранить] ([Save]).

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

- Очистите кэш браузера.
- Обновите страницу раздела [Продукты] ([Products]).

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



The screenshot shows the 'New record' screen for a 'Product' entity. At the top, there are buttons for 'SAVE', 'CANCEL', 'ACTIONS', and 'VIEW'. On the left, there is a placeholder image icon. The main form area contains fields for 'Name*' (with a placeholder 'Name'), 'Code' (containing 'ART_00001'), 'Owner' (set to 'Marina Kysla'), and 'Inactive' (unchecked). The 'Code' field is highlighted with a red border. The top right corner of the window displays the 'Creatio' logo and version '7.18.4.1532'.

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

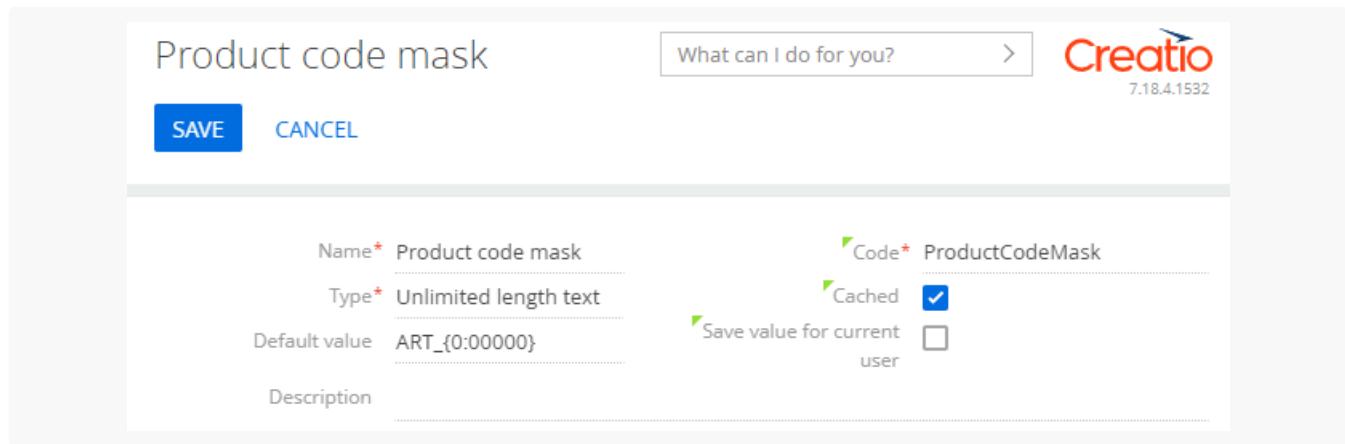
 Сложный

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 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").

The screenshot shows the 'Product last number' configuration screen. At the top, there are 'SAVE' and 'CANCEL' buttons. Below them, the field properties are listed: Name* Product last number, Type* Integer, Default value 0, Code* ProductLastNumber, Cached (checked), and Save value for current user (unchecked). The Creatio logo and version 7.18.4.1532 are visible in the top right corner.

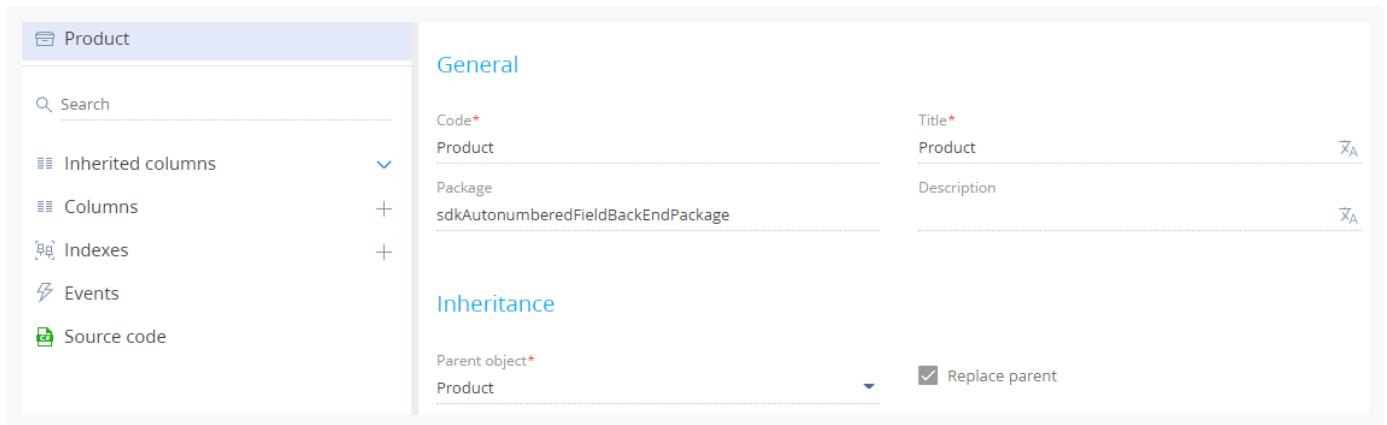
2. Создать схему замещающего объекта

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]).

The screenshot shows a list of item types: Object, Replacing object (highlighted with a red box), Source code, and Module. To the right, a table lists items with columns for Status and Type. The first item is Object, and the second is Client module.

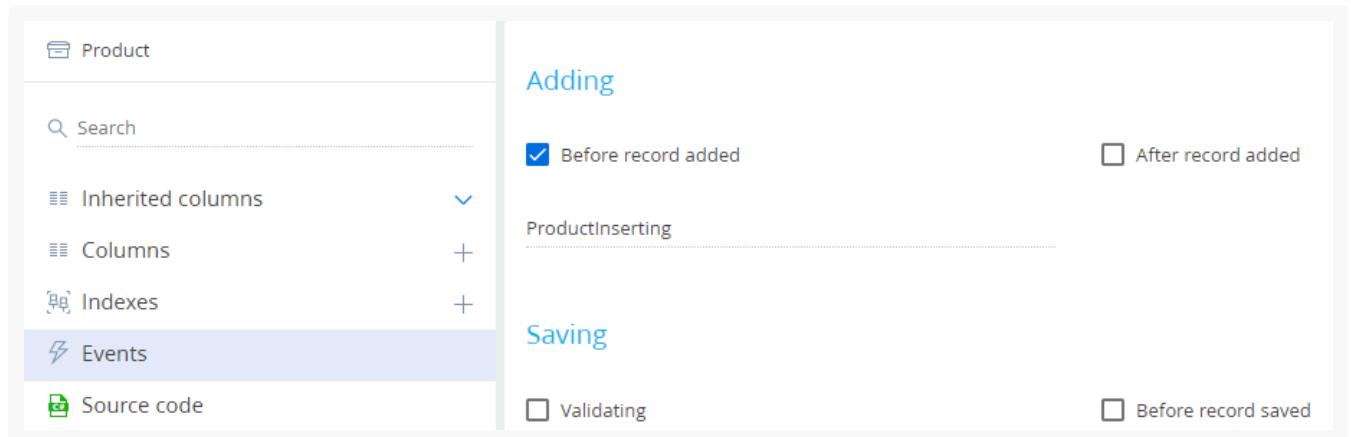
3. Заполните **свойства схемы**.

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



4. В схему добавьте **событие.**

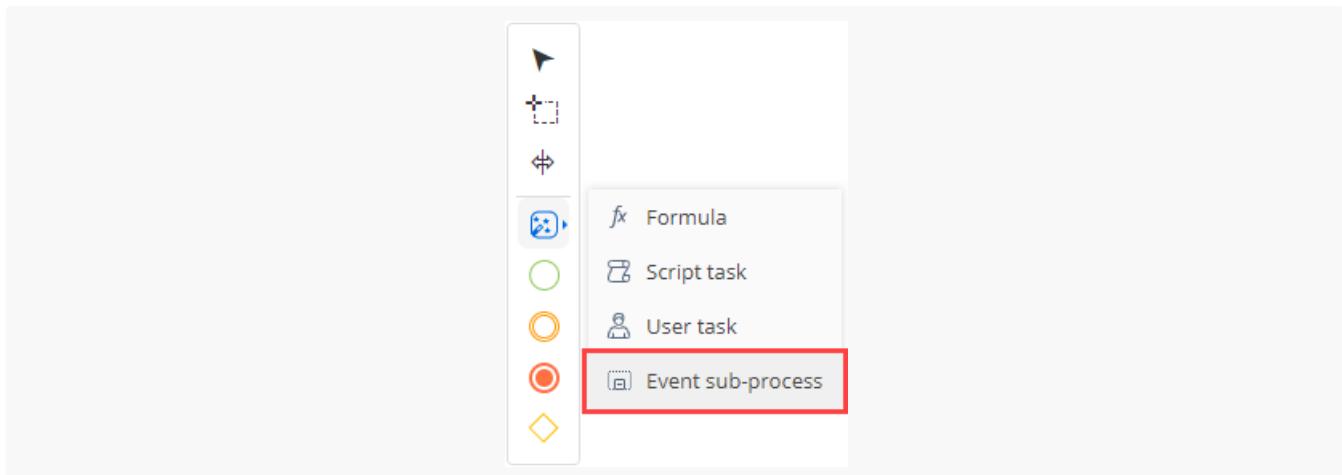
- Перейдите в узел [События] ([Events]) структуры объекта.
- В блоке [Добавление] ([Adding]) установите признак [Перед добавлением записи] ([Before record added]). Событию присвоено имя `ProductInserting`.



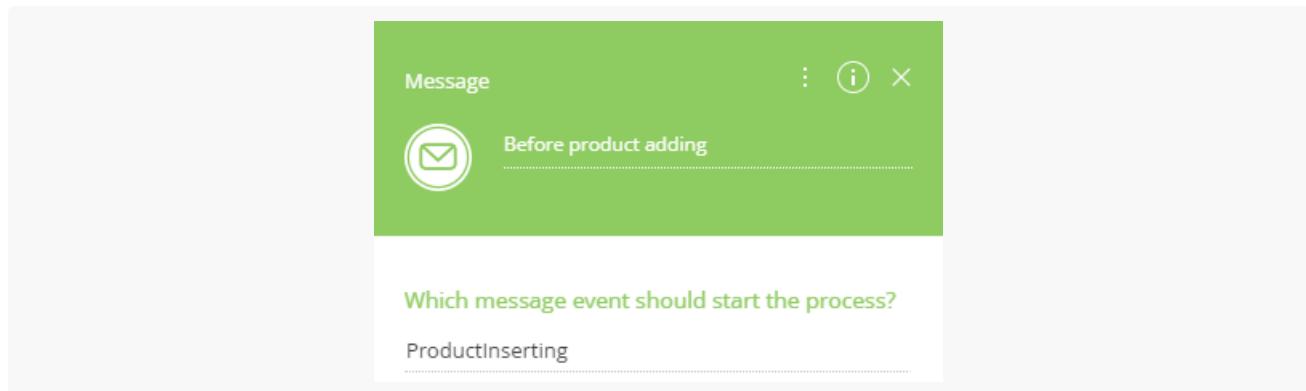
- На панели инструментов дизайнера объектов нажмите [Сохранить] ([Save]).

5. Реализуйте **событийный подпроцесс.**

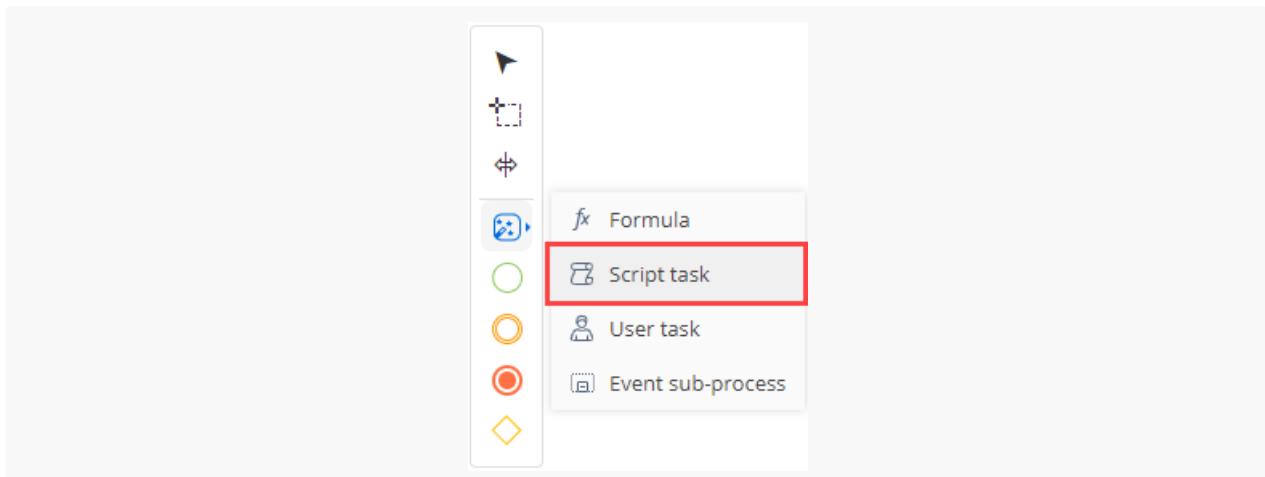
- На панели инструментов дизайнера объектов нажмите [Открыть процесс] ([Open process]).
- В области элементов дизайнера нажмите [Действия системы] ([System actions]) и разместите элемент [Событийный подпроцесс] ([Event sub-process]) в рабочей области дизайнера процессов.



- 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.



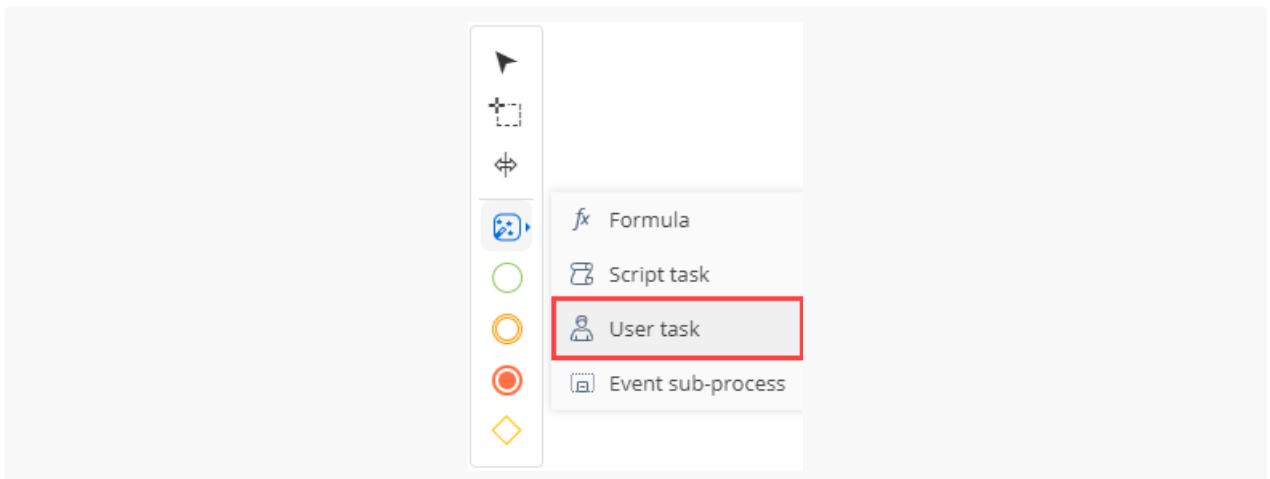
- b. Действию системы [Задание-сценарий] ([Script task]) добавьте имя "Определить схему объекта для генерации номера" ("Get entity schema to generate number").
- c. Добавьте код действия системы [Задание-сценарий] ([Script task]).

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

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

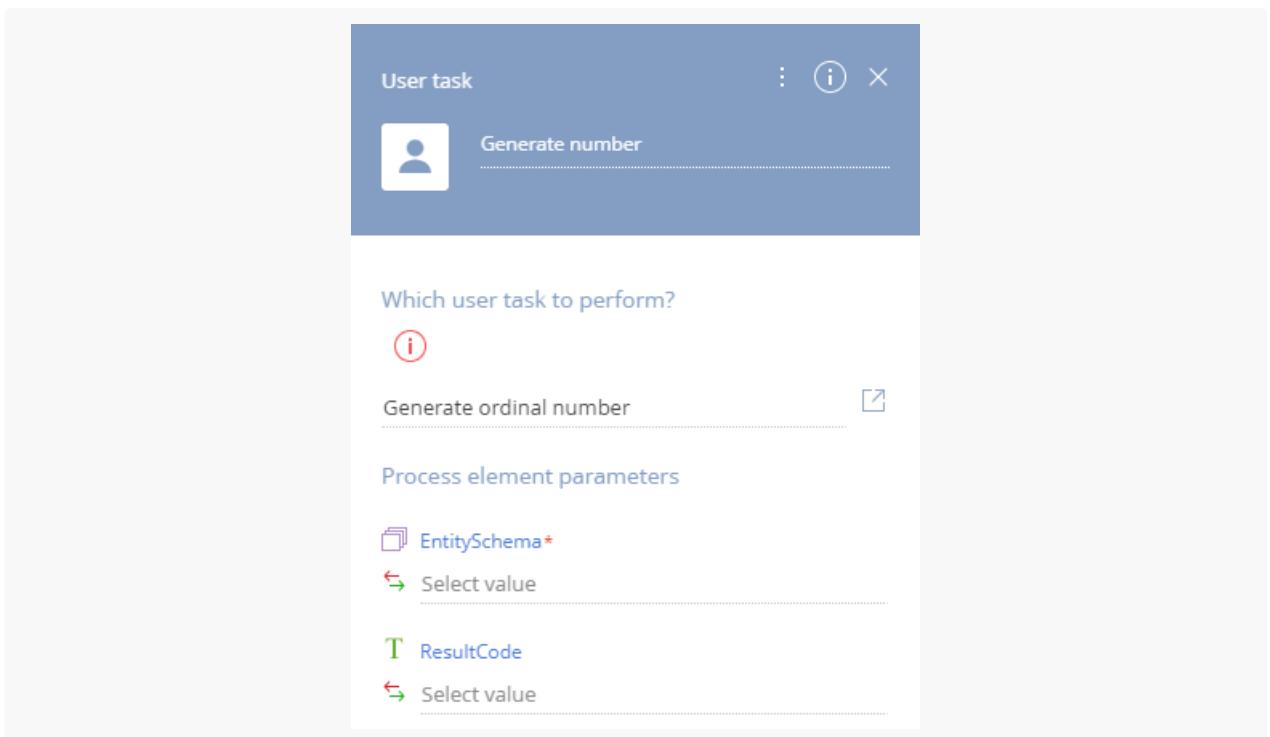
UserTask1 — код действия системы [Выполнить действие процесса] ([User task]). Выполнить генерацию номера (Generate number), настройка которого описана на [следующем шаге](#). Изменить код можно в расширенном режиме настройки действия системы [Выполнить действие процесса] ([User task]).

- 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.



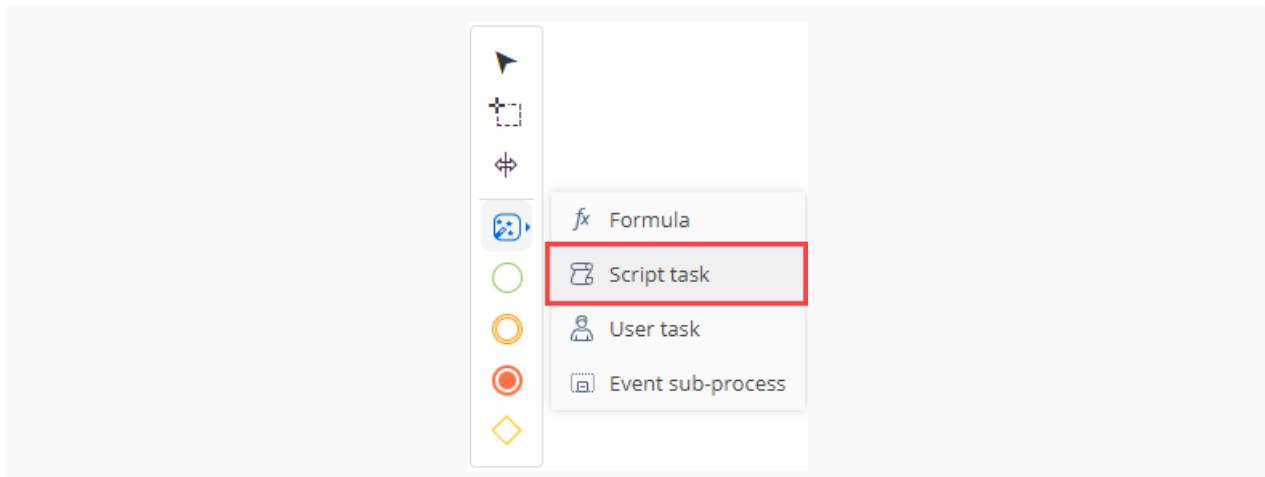
b. Заполните **свойства действия системы**.

- [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-схемарий] ([Script task]).

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

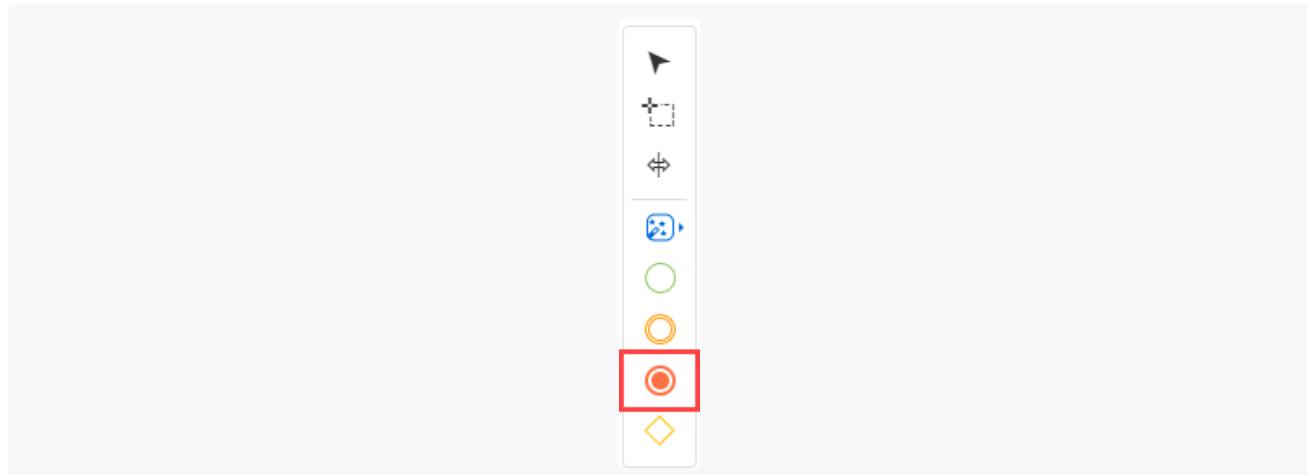


- b. Действию системы [Задание-сценарий] ([Script task]) добавьте имя "Записать полученный номер в колонку объекта" ("Save number to entity column").
- c. Добавьте код действия системы [Задание-сценарий] ([Script task]).

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. Насторойте **потоки**.
- a. Насторойте **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**.

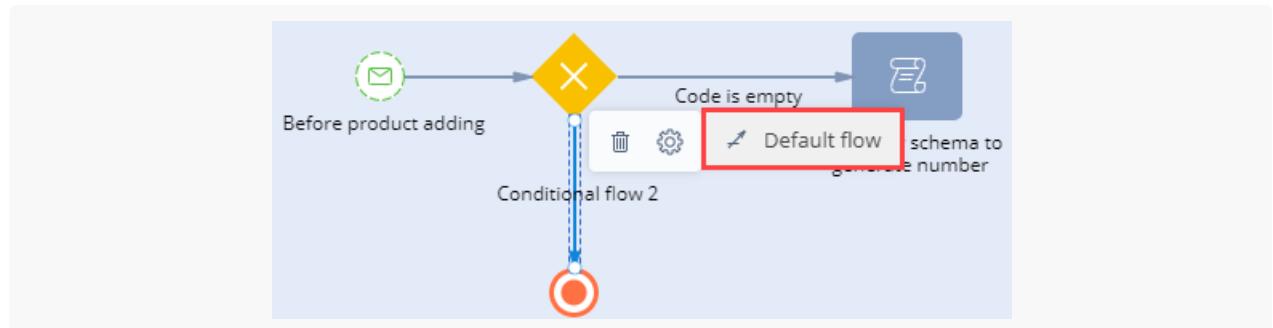
- [Заголовок] ([Title]) — "Код не заполнен" ("Code is empty").
- [Условие перехода] ([Condition to move down the flow]).
 - На панели настройки элементов в свойстве [Условие перехода] ([Condition to move down the flow]) нажмите кнопку .
 - Задайте формулу.

```
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]).

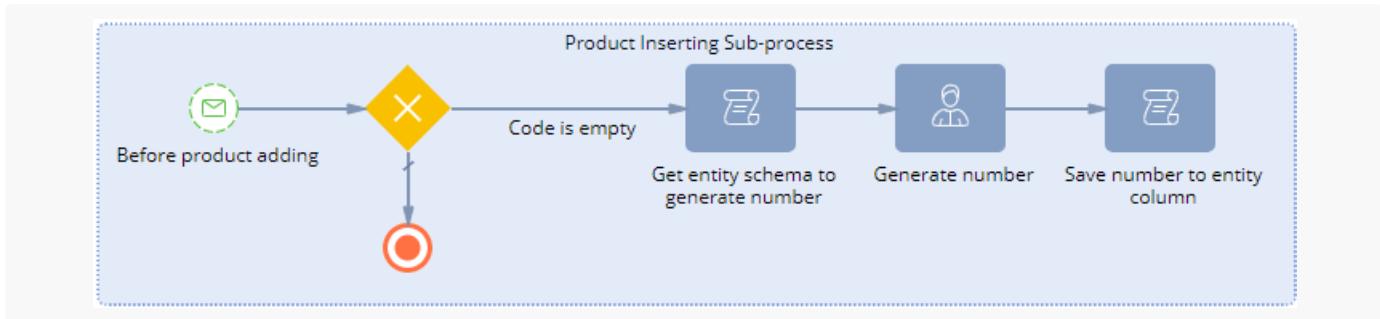
- В меню логического оператора [Исключающее "ИЛИ"] ([Exclusive gateway (OR)]) нажмите на кнопку и соедините логический оператор [Исключающее "ИЛИ"] ([Exclusive gateway (OR)]) с событием [Останов] ([Terminate]).
- Трансформируйте поток управления в поток по умолчанию. Для этого в меню потока нажмите —>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



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

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

- Очистите кэш браузера.
- Обновите страницу раздела [Продукты] ([Products]).
- 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.

	Name* Test product	Code ART_00001	Owner Marina Kysla
	Link	Inactive <input type="checkbox"/>	

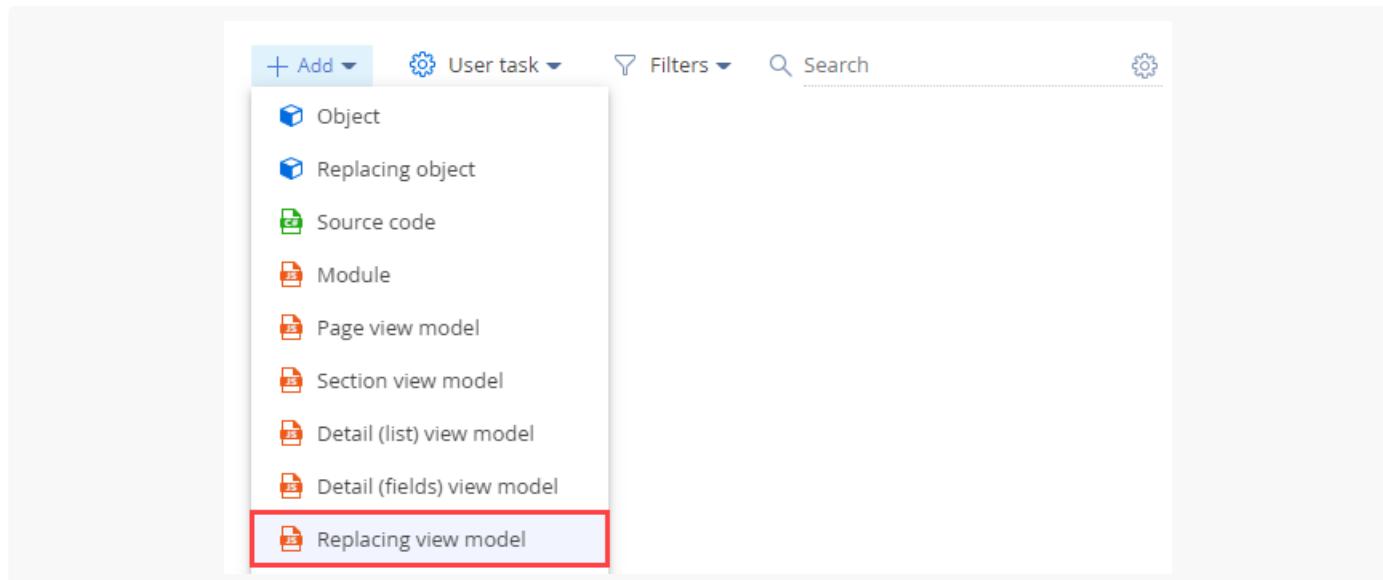
Добавить информационную кнопку к полю на странице записи

Средний

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

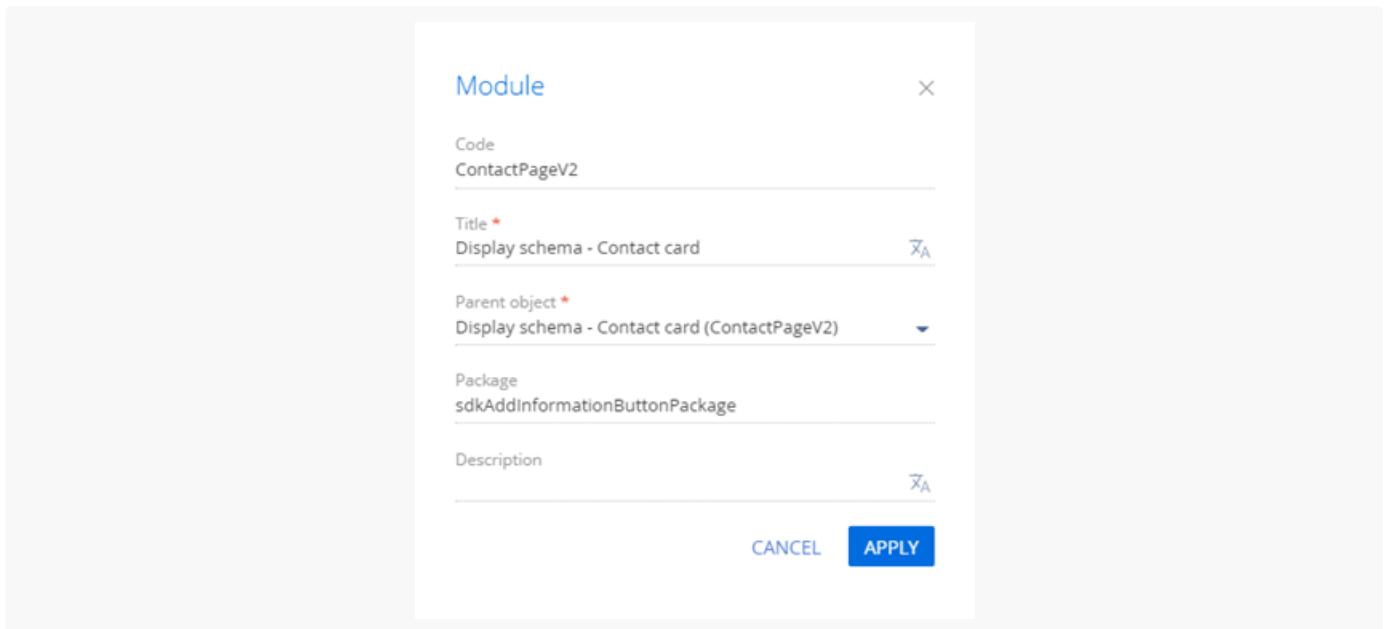
Создать схему замещающей модели представления страницы контакта

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*]).



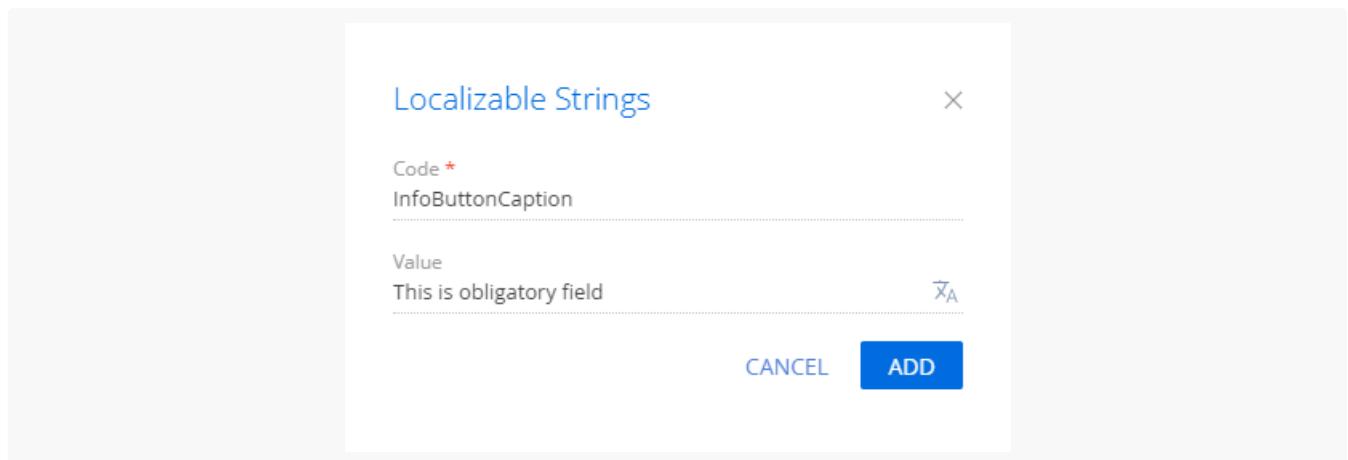
3. Заполните **свойства схемы**.

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



4. Добавьте локализуемую строку.

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



- 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 кнопка. */
            "parentName": "ProfileContainer",
            /* Информационная кнопка добавляется в коллекцию элементов родительского элемента. */
            "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*/
};

});
```

- 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

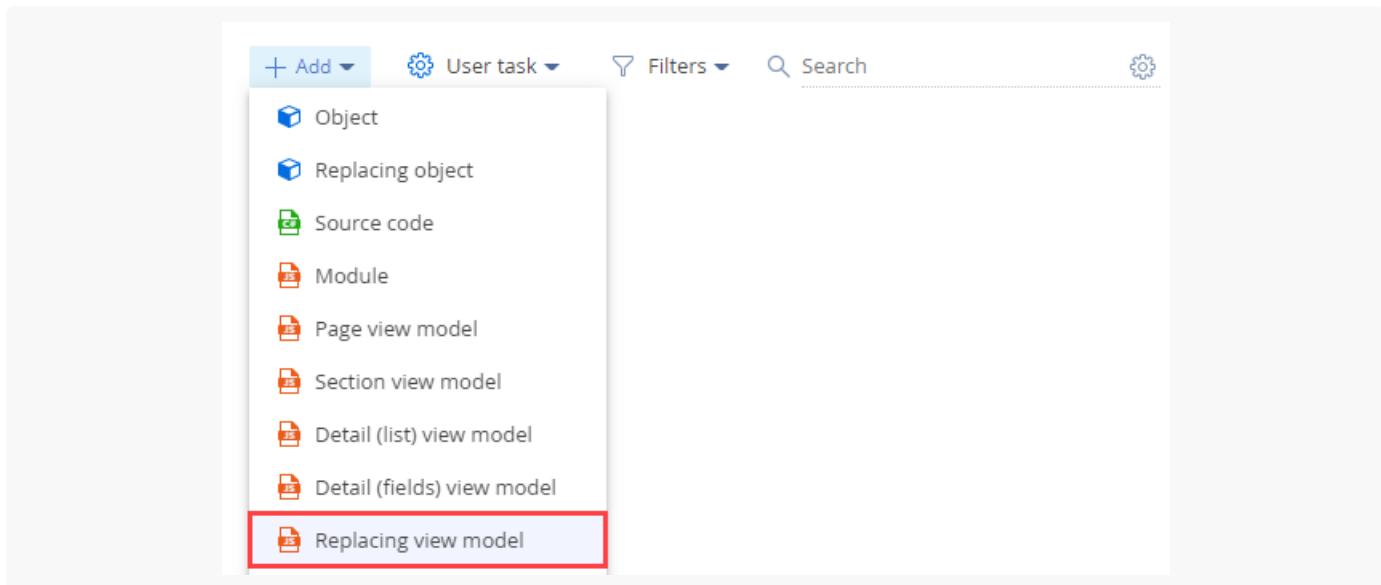
Средний

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

Создать схему замещающей модели представления страницы контакта

- 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' configuration dialog. The fields are as follows:

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

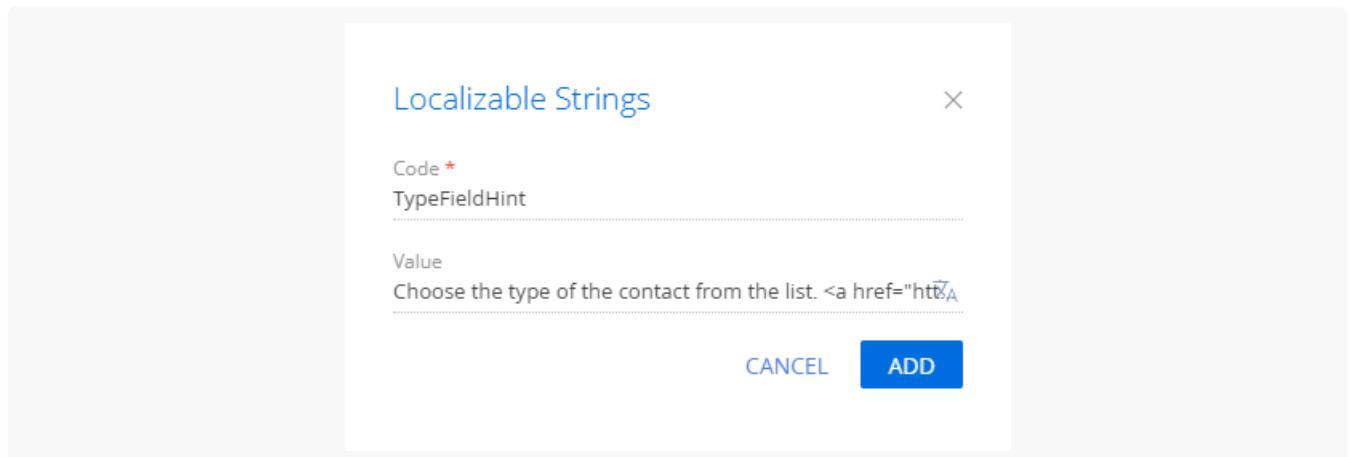
At the bottom right are 'CANCEL' and 'APPLY' buttons.

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

- а. 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. Узнать больше" ("Choose the type of the contact from the list. Read more").



- 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": {
                    /* Всплывающая подсказка для поля Type. */
                    "TypeFieldHint": {
                        /* Код локализуемой строки. */
                        "Code": "TypeFieldHint",
                        /* Значение локализуемой строки. */
                        "Value": "Choose the type of the contact from the list. <a href='https://academy.terrasoft.ua/docs/user/bazis_platformy/interfejs/stranitsy_zapisey/stranicy_zapisej' target='_blank'>Read more</a>"
                    }
                }
            }
        ]
    }
})
```

```

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

]/**SCHEMA_DIFF*/
};

});
});
```

- 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.

The screenshot shows a contact record for Andrew Z. Barber. The contact card includes fields for Full name*, Full job title, Mobile phone, and Business phone. A modal window titled "Choose the type of the contact from the list. Read more" is open, with "Customer" selected. The contact's details also show Title: Mr., Recipient's name: Barber, Age: 46, Owner: Marina Kysla, Gender: Male, and Preferred language.

Кнопка



Контейнеры кнопок

Виды контейнеров кнопок, которые реализованы в Creatio:

- **Контейнер кнопок действий** — содержит кнопки действий страницы раздела. Содержит контейнеры стандартных кнопок и контейнер кнопок с выпадающим меню.
- **Контейнер стандартных кнопок** — содержит кнопки [Сохранить] ([Save]), [Отмена] ([Cancel]), [Теги] ([Tags]) и выпадающее меню кнопки [Действия] ([Actions]).
- **Контейнер кнопок с выпадающим меню** — содержит выпадающие меню кнопок [Печать] ([Print]) и [Вид] ([View]).

Контейнеры кнопок отличаются для страницы раздела, страницы записи и страницы добавления записи.

На заметку. В приложении используются мета-имена html-контейнеров. На основании мета-имен приложение формирует фактические идентификаторы соответствующих html-элементов страницы записи.

Мета-имена контейнеров кнопок представлены в таблице ниже.

Мета-имена контейнеров кнопок

Название элемента интерфейса	Контейнеры		
	Контейнер кнопок действий	Контейнер стандартных кнопок	Контейнер кнопок с выпадающим меню
Страница раздела	SeparateModeActionButtonsContainer	SeparateModeActionButtonsLeftContainer	SeparateModeActionButtonsRightContainer
Страница записи	CombinedModeActionButtonsCardContainer	CombinedModeActionButtonsCardLeftContainer	CombinedModeActionButtonsCardRightContainer
Страница добавления записи	ActionButtonsContainer	LeftContainer	RightContainer

Контейнеры кнопок **страницы раздела** представлены на рисунке ниже.

Контейнеры кнопок **страницы записи** представлены на рисунке ниже.

Контейнеры кнопок **страницы добавления записи** представлены на рисунке ниже.

Добавить кнопку

Алгоритм, который необходимо использовать для добавления кнопки, зависит от вида кнопки, которую планируется добавить.

Виды кнопок, которые реализованы в Creatio:

- Простая кнопка.
- Кнопка выбора цвета.

Добавить простую кнопку на страницу записи или раздела

1. Создайте схему замещающей модели представления страницы записи или раздела, на которой будет размещена кнопка. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающей модели представления добавьте локализуемую строку, которая содержит название кнопки. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
3. В схеме замещающей модели представления реализуйте **логику работы кнопки**:
 - a. В свойстве `methods` реализуйте:
 - Метод-обработчика, который вызывается по нажатию на кнопку.
 - Вспомогательные методы, которые необходимы для функционирования элемента управления. Это могут быть методы, которые управляют видимостью или доступностью элемента управления.
 - d. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения кнопки на странице записи или раздела. В массиве модификаций укажите соответствующий контейнер, в котором планируется разместить кнопку.

Важно. **Совмещенный режим** — режим отображения страницы записи, у которой открыт вертикальный реестр. Чтобы **добавить кнопку на страницу записи в совмещенном режиме**, внесите изменения в схему замещающей модели представления страницы раздела и в схему замещающей модели представления страницы записи.

Добавить простую кнопку в строку записи раздела

1. Создайте схему замещающей модели представления раздела, в котором будет размещена кнопка. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающей модели представления добавьте локализуемую строку, которая содержит название кнопки. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
3. В схеме замещающей модели представления реализуйте **логику работы кнопки**:

a. В свойстве `methods` реализуйте:

- метод `onActiveRowAction()` — переопределенный базовый метод схемы `Base DataView` пакета `NUI`, который связывает кнопку с методом-обработчиком.
- Метод-обработчика, который вызывается по нажатию на кнопку.

d. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения кнопки на странице раздела.

- В свойстве `parentName` укажите контейнер `DataGrid`.
- в свойстве `propertyName` укажите коллекцию `activeRowActions`.
- Вместо свойства `itemType` укажите свойство `className`, для которого установите значение `Terrasoft.Button`.
- Укажите свойство `tag`, которое будет идентифицировать кнопку в методе `onActiveRowAction()`.

Добавить кнопку выбора цвета

1. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающего объекта добавьте колонку типа [Строка (50 символов)] ([Text (50 characters)]), которая будет хранить информацию о выбранном цвете. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
3. Создайте схему замещающей модели представления страницы записи или раздела, на которой будет размещена кнопка. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
4. В схеме замещающей модели представления реализуйте **логику работы кнопки**:

a. В свойстве `methods` реализуйте:

- Метод-обработчика, который вызывается по нажатию на кнопку.
- Вспомогательные методы, которые необходимы для функционирования элемента управления. Это могут быть методы, которые управляют видимостью или доступностью элемента управления.

d. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения кнопки на странице записи или раздела.

- В свойстве `itemType` укажите тип `COLOR_BUTTON`.
- В свойстве `value` установите привязку к добавленной колонке схемы замещающего объекта.

Добавить к кнопке всплывающую подсказку

Всплывающие подсказки — текстовые сообщения, которые предоставляют пользователю дополнительную информацию о функциональности кнопки. Всплывающая подсказка к кнопке отображается при наведении курсора на кнопку.

Чтобы **добавить к кнопке всплывающую подсказку**:

1. Создайте схему замещающей модели представления страницы записи или раздела, на которой будет размещена кнопка. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схему замещающей модели представления добавьте локализуемую строку, которая содержит название кнопки. Для этого воспользуйтесь инструкцией, которая приведена в статье [Операции с локализуемыми ресурсами](#).
3. В массиве модификаций `diff` схемы замещающей модели представления реализуйте **всплывающую подсказку**.

Способы добавления всплывающей подсказки в конфигурационный объект кнопки:

- Свойство `hint`.
- Свойство `tips`.

Чтобы добавить всплывающую подсказку к кнопке **с использованием свойства `hint`**, в свойство `values` элемента управления добавьте свойство `hint`, которое содержит текст всплывающей подсказки.

Чтобы добавить всплывающую подсказку к кнопке **с использованием свойства `tips`**:

1. В свойство `values` элемента управления добавьте свойство `tips`.
2. В массив `tips` с помощью операции `insert` добавьте конфигурационный объект подсказки.
3. В свойстве `values` конфигурационного объекта подсказки добавьте свойство `content`, которое содержит текст всплывающей подсказки.

Использование свойства `tips` для добавления всплывающей подсказки к кнопке позволяет:

- Изменить стиль отображения.
- Привязать видимость подсказки к событию модели представления.
- Добавить элементы управления и т. д.

Типы элементов, которые позволяют использовать свойство `tips`:

- `Terrasoft.ViewItemType.BUTTON`.
- `Terrasoft.ViewItemType.LABEL`.
- `Terrasoft.ViewItemType.COLOR_BUTTON`.
- `Terrasoft.ViewItemType.HYPERLINK`.
- `Terrasoft.ViewItemType.INFORMATION_BUTTON`.
- Элементы, для которых указано свойство `generator`.

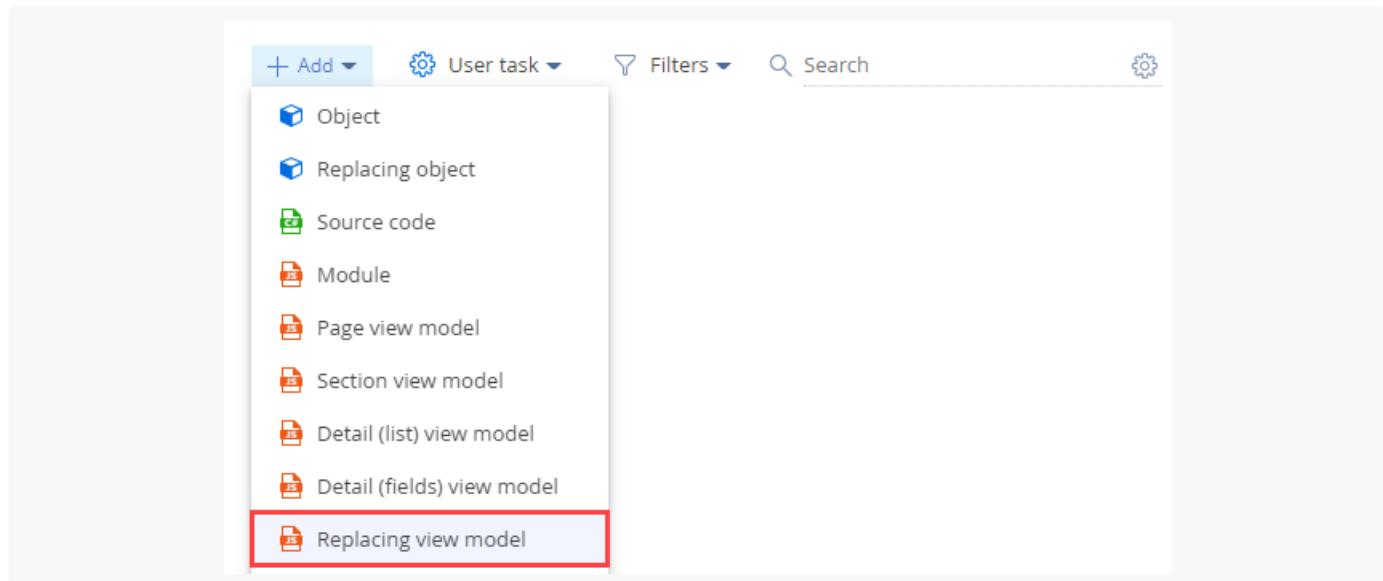
Добавить кнопку на панель инструментов раздела



Пример. Добавить кнопку на панель инструментов раздела [Контрагенты] ([Accounts]). Кнопка активна при выборе в реестре раздела контрагента, для которого указан основной контакт. При нажатии на кнопку открывается страница основного контакта активного контрагента.

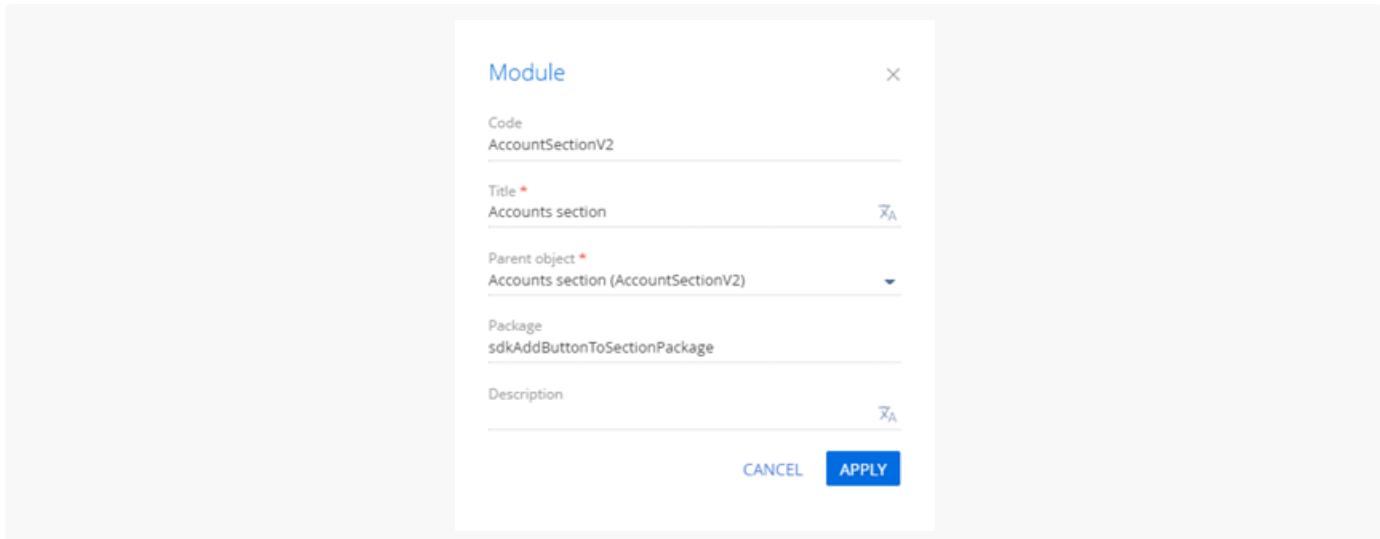
Создать схему замещающей модели представления раздела

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]) — "AccountSectionV2".
- [Заголовок] ([Title]) — "Раздел контрагенты" ("Account section").
- [Родительский объект] ([Parent object]) — выберите "AccountSectionV2".

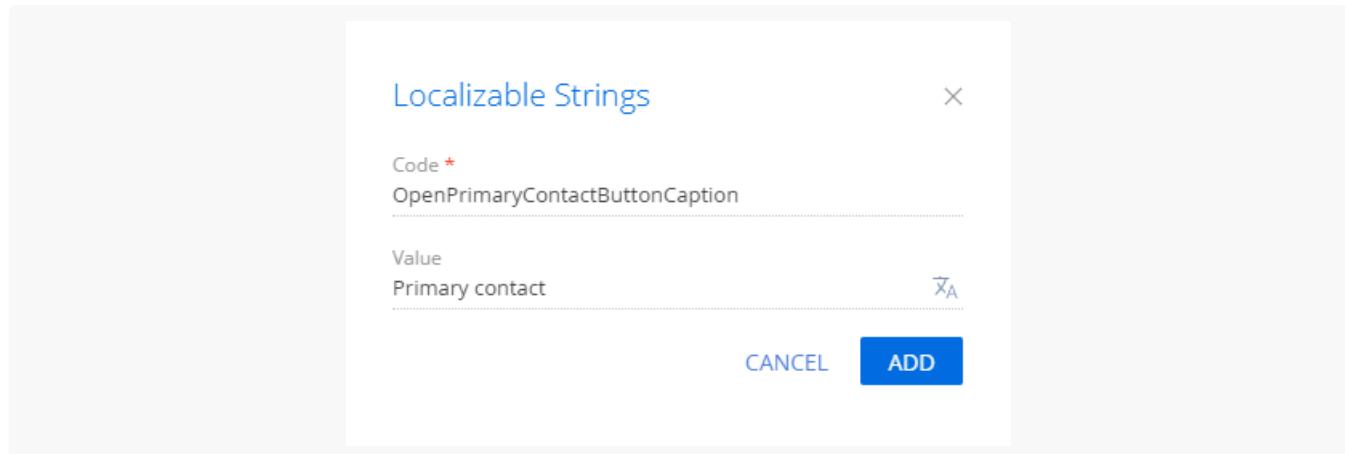


4. Добавьте локализуемую строку.

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

b. Заполните **свойства локализуемой строки**.

- [Код] ([Code]) — "OpenPrimaryContactButtonCaption".
- [Значение] ([Value]) — "Основной контакт" ("Primary contact").



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

5. Реализуйте логику работы кнопки.

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

- `isAccountPrimaryContactSet()` — проверяет заполнение поля [Основной контакт] ([Primary contact]) страницы.
- `onOpenPrimaryContactClick()` — метод-обработчик нажатия кнопки. Выполняет переход на страницу основного контакта.

d. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения кнопки на странице.

Обращение к выделенной записи выполняется через атрибут `ActiveRow` модели представления раздела. Атрибут возвращает значение первичной колонки выделенной записи. В свою очередь, значение первичной колонки выделенной записи может использоваться для получения значений, загруженных в реестр полей выбранного объекта, например, из коллекции данных списочного представления реестра раздела, которая хранится в свойстве `GridData` модели представления реестра.

Исходный код схемы замещающей модели представления страницы раздела представлен ниже.

AccountSectionV2

```
define("AccountSectionV2", [], function() {
    return {
        /* Название схемы объекта раздела. */
        entitySchemaName: "Account",
        /* Методы модели представления раздела. */
        methods: {
            /* Метод-обработчик нажатия кнопки. */
            onOpenPrimaryContactClick: function() {
                /* Получение идентификатора выбранной записи. */
                var activeRow = this.get("ActiveRow");
                if (!activeRow) {
                    return;
                }
                /* Определение идентификатора основного контакта. */
                var primaryId = this.get("GridData").get(activeRow).get("PrimaryContact").val
                if (!primaryId) {
                    return;
                }
                /* Формирование строки адреса. */
                var requestUrl = "CardModuleV2/ContactPageV2/edit/" + primaryId;
                /* Публикация сообщения о пополнении истории навигации по страницам и переход */
                this.sandbox.publish("PushHistoryState", {
                    hash: requestUrl
                });
            },
            /* Проверяет заполнение поля [Основной контакт] выбранного элемента. */
            isAccountPrimaryContactSet: function() {
                var activeRow = this.get("ActiveRow");
                if (!activeRow) {
                    return false;
                }
                var pc = this.get("GridData").get(activeRow).get("PrimaryContact");
                return (pc || pc !== "") ? true : false;
            }
        },
        /* Отображение кнопки в разделе. */
        diff: /**SCHEMA_DIFF*/[
    }
});
```

```

/* Метаданные для добавления в раздел пользовательской кнопки. */
{
    /* Выполняется операция добавления элемента на страницу. */
    "operation": "insert",
    /* Мета-имя родительского контейнера, в который добавляется кнопка. */
    "parentName": "ActionButtonsContainer",
    /* Кнопка добавляется в коллекцию элементов родительского элемента. */
    "propertyName": "items",
    /* Мета-имя добавляемой кнопки. */
    "name": "MainContactSectionButton",
    /* Свойства, передаваемые в конструктор элемента. */
    "values": {
        /* Тип добавляемого элемента – кнопка. */
        "itemType": Terrasoft.ViewItemType.BUTTON,
        /* Привязка заголовка кнопки к локализуемой строке схемы. */
        "caption": { bindTo: "Resources.Strings.OpenPrimaryContactButtonCaption" },
        /* Привязка метода-обработчика нажатия кнопки. */
        "click": { bindTo: "onOpenPrimaryContactClick" },
        /* Привязка свойства доступности кнопки. */
        "enabled": { bindTo: "isAccountPrimaryContactSet" },
        /* Настройка расположения кнопки. */
        "layout": {
            /* Номер столбца. */
            "column": 1,
            /* Номер строки. */
            "row": 6,
            /* Диапазон занимаемых столбцов. */
            "colSpan": 1
        }
    }
}
]/**SCHEMA_DIFF*/
};

});
);

```

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

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

- Очистите кэш браузера.
- Обновите страницу раздела [Контрагенты] ([Accounts]).

В результате выполнения примера на панель инструментов раздела [Контрагенты] ([Accounts]) добавлена кнопка [Основной контакт] ([Primary contact]). Кнопка активна при выборе в реестре раздела контрагента, для которого указан основной контакт.

Accounts

What can I do for you? > Creatio 7.18.3.1241

ACTIONS ▾ PRIMARY CONTACT

Filters/folders Tag

	Vertigo Systems	Web www.vertigosys.com	Primary phone +44 (20) 3427 1374	Type Customer
	Primary contact Peter Moore	Address 83 Ashton Street	City London	Country United Kingdom
	OPEN	COPY	DELETE	
	Sunrise Investments	Web www.sunrise-invest.co.uk	Primary phone +44 (15) 1437 1598	Type Customer
	Primary contact Sarah M. Richards	Address 34 King's Road	City Liverpool	Country United Kingdom

При нажатии на кнопку [Основной контакт] ([Primary contact]) открывается страница основного контакта активного контрагента.

Accounts

What can I do for you? > Creatio 7.18.3.1241

ACTIONS ▾ PRIMARY CONTACT

Filters/folders Tag

	Vertigo Systems	Web www.vertigosys.com	Primary phone +44 (20) 3427 1374	Type Customer
	Primary contact Peter Moore	Address 83 Ashton Street	City London	Country United Kingdom
	Sunrise Investments	Web www.sunrise-invest.co.uk	Primary phone +44 (15) 1437 1598	Type Customer
	Primary contact Sarah M. Richards	Address 34 King's Road	City Liverpool	Country United Kingdom
	RealWay	Web www.realway.co.uk	Primary phone +44 (20) 3425 2139	Type Customer
	Primary contact Timothy Sawyer	Address 26-a Carpenter Street	City London	Country United Kingdom
	FlashNet Consulting	Web www.flashnetconsulting.co m	Primary phone +1 206 429 1595	Type Customer
	Primary contact Tracy Wilkinson	Address 36 Village Street	City Seattle	Country United States
	Gtech	Web www.gtech.com	Primary phone +1 646 487 28 91	Type Customer
	Primary contact	Address	City	Country

Добавить кнопку на панель инструментов страницы записи в совмещенном режиме

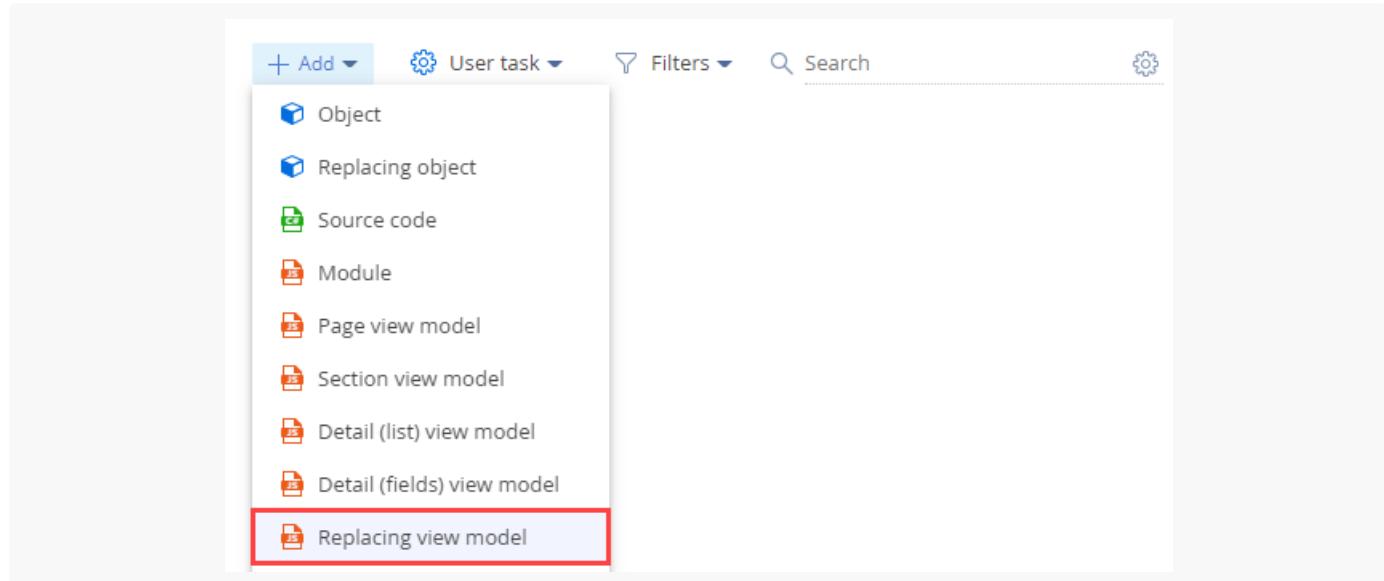
Сложный

Пример. Добавить кнопку на панель инструментов страницы контрагента, которая отображается в совмещенном режиме.

- Если для контрагента указан основной контакт, то при нажатии на кнопку открывается страница этого контакта.
- Если для контрагента не указан основной контакт, то кнопка неактивна.

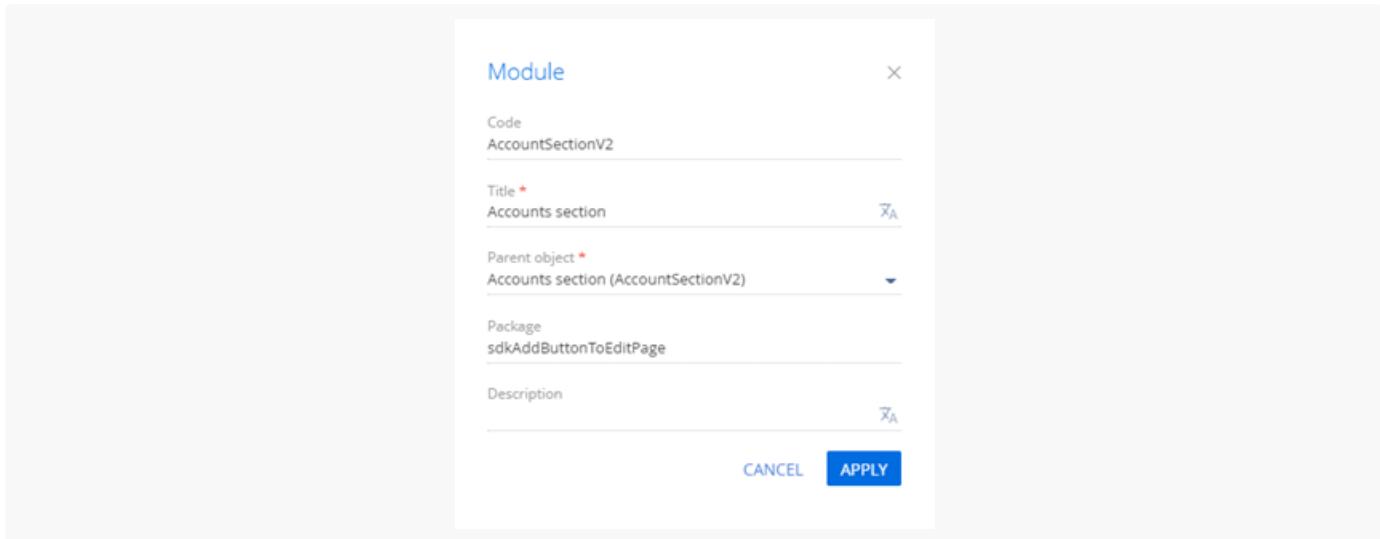
Создать схему замещающей модели представления раздела

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]).



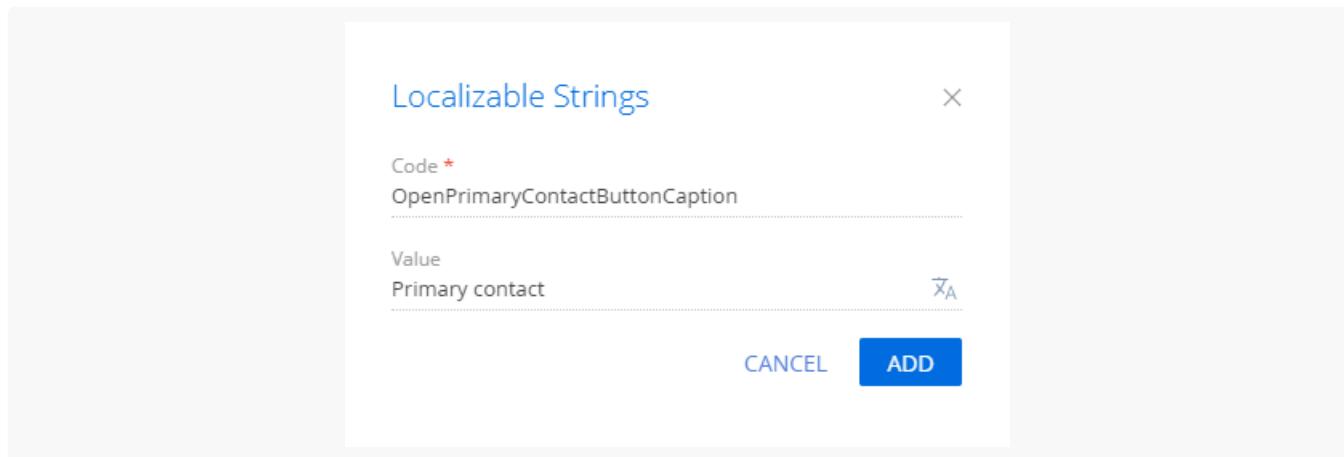
3. Заполните **свойства схемы**.

- [Код] ([Code]) — "AccountSectionV2".
- [Заголовок] ([Title]) — "Раздел контрагенты" ("Account section").
- [Родительский объект] ([Parent object]) — выберите "AccountSectionV2".



4. Добавьте локализуемую строку.

- В контекстном меню узла [Локализуемые строки] ([Localizable strings]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
 - [Код] ([Code]) — "OpenPrimaryContactButtonCaption".
 - [Значение] ([Value]) — "Основной контакт" ("Primary contact").



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

5. Реализуйте логику работы кнопки.

- В свойство `attributes` добавьте атрибут `ButtonEnabled` типа `Terrasoft.DataValueType.BOOLEAN`, который сохраняет состояние доступности кнопки.
- В свойстве `methods` реализуйте **методы**:
 - `onOpenPrimaryContactClick()` — выполняет переход на страницу основного контакта.
 - `onCardRendered()` — подписка на события загрузки данных в реестр и событие изменения активной записи реестра. Выполняется после отрисовки страницы контрагента в совмещенном режиме.
 - `isPrimaryContactExist()` — определяет наличие основного контакта для активной записи

реестра.

- `setButtonEnabled()` — устанавливает значение атрибута `ButtonEnabled` в зависимости от наличия основного контакта контрагента.
- g. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения кнопки на странице.

Исходный код схемы замещающей модели представления страницы раздела представлен ниже.

AccountSectionV2

```
define("AccountSectionV2", [], function() {
    return {
        /* Название схемы объекта раздела. */
        entitySchemaName: "Account",
        attributes: {
            /* Атрибут для хранения состояния доступности кнопки. */
            "ButtonEnabled": {
                "dataValueType": Terrasoft.DataValueType.BOOLEAN,
                "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
                "value": false
            }
        },
        /* Методы модели представления раздела. */
        methods: {
            /* Метод-обработчик нажатия кнопки. */
            onOpenPrimaryContactClick: function() {
                /* Определение активной записи вертикального реестра. */
                var activeRow = this.get("ActiveRow");
                if (activeRow) {
                    /* Определение идентификатора основного контакта. */
                    var primaryId = this.get("GridData").get(activeRow).get("PrimaryContact");
                    if (primaryId) {
                        /* Формирование строки адреса. */
                        var requestUrl = "CardModuleV2/ContactPageV2/edit/" + primaryId;
                        /* Публикация сообщения о пополнении истории навигации по страницам и */
                        this.sandbox.publish("PushHistoryState", {
                            hash: requestUrl
                        });
                    }
                }
            },
            /* Выполняется после отрисовки страницы контрагента. */
            onCardRendered: function() {
                this.callParent();
                /* Данные реестра. */
                var gridData = this.get("GridData");
                var activeRow = this.get("ActiveRow");
            }
        }
    }
},
```

```

if (activeRow)
{
    this.setButtonEnabled(activeRow, this);
}
/* После закрытия страницы основного контакта теряется активная запись. Ее нужно восстановить */
else {
    var historyState = this.sandbox.publish("GetHistoryState");
    var hash = historyState.hash;
    if (hash && hash.valuePairs)
    {
        activeRow = hash.valuePairs[0].name;
        /* Восстановление активной записи. */
        this.set("ActiveRow", activeRow);
        /* Сохранение контекста в локальную переменную. */
        var self = this;
        /* Подписка на событие полной загрузки данных в вертикальный реестр. */
        gridData.on("dataloaded", function() {
            self.setButtonEnabled(activeRow, self);
        });
    }
}
/* Подписка на событие изменения активной записи реестра. */
gridData.on("itemchanged", function() {
    this.setButtonEnabled(activeRow, this);
}, this);
},
/* Определяет наличие основного контакта для активной записи реестра. */
isPrimaryContactExist: function(id) {
    var pc = this.get("GridData").get(id).get("PrimaryContact");
    return (pc || pc !== "") ? true : false;
},
/* Устанавливает значение атрибута ButtonEnabled в зависимости от того, определен ли контакт */
setButtonEnabled: function(activeRow, context) {
    if (context.isPrimaryContactExist(activeRow)) {
        context.set("ButtonEnabled", true);
    }
    else {
        context.set("ButtonEnabled", false);
    }
},
/* Отображение кнопки на странице записи. */
diff: [
    /* Метаданные для добавления на страницу пользовательской кнопки. */
    {
        /* Выполняется операция добавления элемента на страницу. */
        "operation": "insert",
        /* Мета-имя родительского контейнера, в который добавляется кнопка. */
        "parentName": "CombinedModeActionButtonsCardLeftContainer",
    }
]
}

```

```

/* Кнопка добавляется в коллекцию элементов родительского элемента. */
"propertyName": "items",
/* Мета-имя добавляемой кнопки. */
"name": "MainContactButton",
/* Свойства, передаваемые в конструктор элемента. */
"values": {
    /* Тип добавляемого элемента – кнопка. */
    "itemType": Terrasoft.ViewItemType.BUTTON,
    /* Привязка заголовка кнопки к локализуемой строке схемы. */
    "caption": {bindTo: "Resources.Strings.OpenPrimaryContactButtonCaption"},
    /* Привязка метода-обработчика нажатия кнопки. */
    "click": {bindTo: "onOpenPrimaryContactClick"},
    /* Стиль отображения кнопки. */
    "style": Terrasoft.controls.ButtonEnums.style.GREEN,
    /* Привязка свойства доступности кнопки. */
    "enabled": {bindTo: "ButtonEnabled"}
}
},
];
);
);

```

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

Результат выполнения примера

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

В результате выполнения примера на панель инструментов раздела [Контрагенты] ([Accounts]) добавлена кнопка [Основной контакт] ([Primary contact]).

Если для контрагента указан основной контакт, то при нажатии на кнопку [Основной контакт] ([Primary contact]) открывается страница этого контакта.

The screenshot shows the 'Accounts' module in the Creatio application. On the left, there's a sidebar with various icons. The main area displays a list of accounts, with 'Vertigo Systems' selected. The account details for 'Vertigo Systems' are shown on the right, including its type (Customer), owner (Mary King), and various contact and segmentation details. A red box highlights the 'PRIMARY CONTACT' button at the top of the account info section. Below it, a modal window shows the contact information for 'Peter Moore', including his mobile phone number (+44 782 335 8812) and email (peter.moore@yahoo.com). The entire contact information block is also highlighted with a red box.

Если для контрагента не указан основной контакт, то кнопка [Основной контакт] ([Primary contact]) неактивна.

This screenshot shows the same 'Accounts' module interface as the previous one, but for the account 'Wilson & Young'. The 'PRIMARY CONTACT' button at the top of the account info section is highlighted with a red box. A modal window below it shows the contact information for 'Wilson & Young', including its name, type (Competitor), owner (Marina Kysla), and web address (www.wilson-young.gov). The contact information block is also highlighted with a red box.

Добавить кнопку на панель

инструментов страницы добавления записи

 Средний

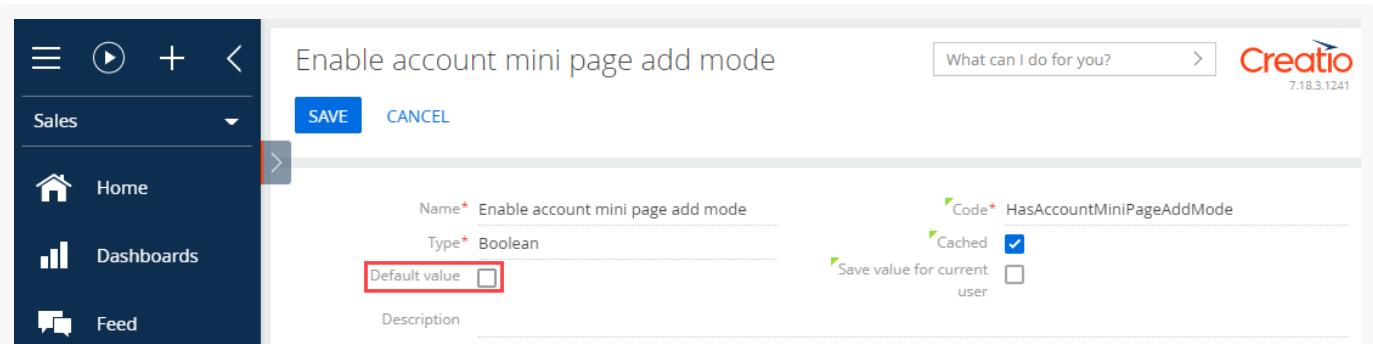
Пример. Добавить кнопку на панель инструментов страницы добавления контрагента. Кнопка активна после добавления основного контакта контрагента. При нажатии на кнопку открывается страница основного контакта этого контрагента.

1. Изменить способ добавления контрагента

По умолчанию для добавления контрагента используется мини-карточка.

Чтобы для добавления контрагента **использовать страницу добавления**:

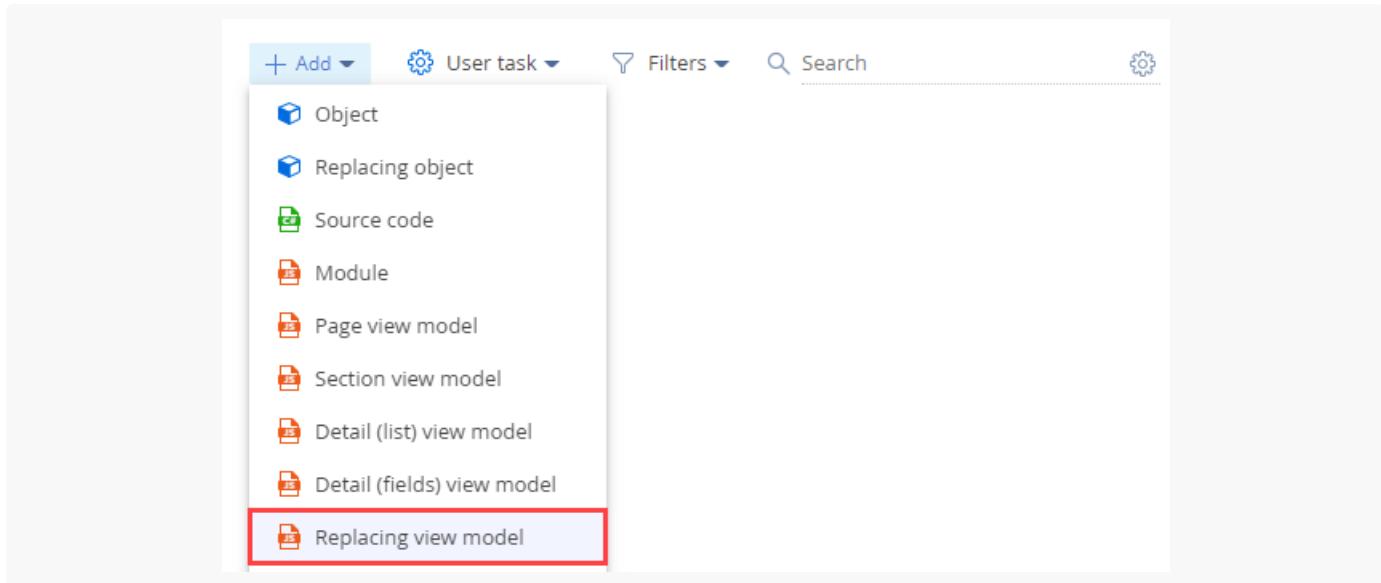
- Перейдите в дизайнер системы по кнопке .
- В блоке [Настойка системы] ([System setup]) перейдите по ссылке [Системные настройки] ([System settings]).
- Выберите настройку [Использовать миникарточку добавления контрагента] ([Enable account mini page add mode], код `HasAccountMiniPageAddMode`).
- Снимите признак [Значение по умолчанию] ([Default value]).



- Выполните повторный вход в приложение.

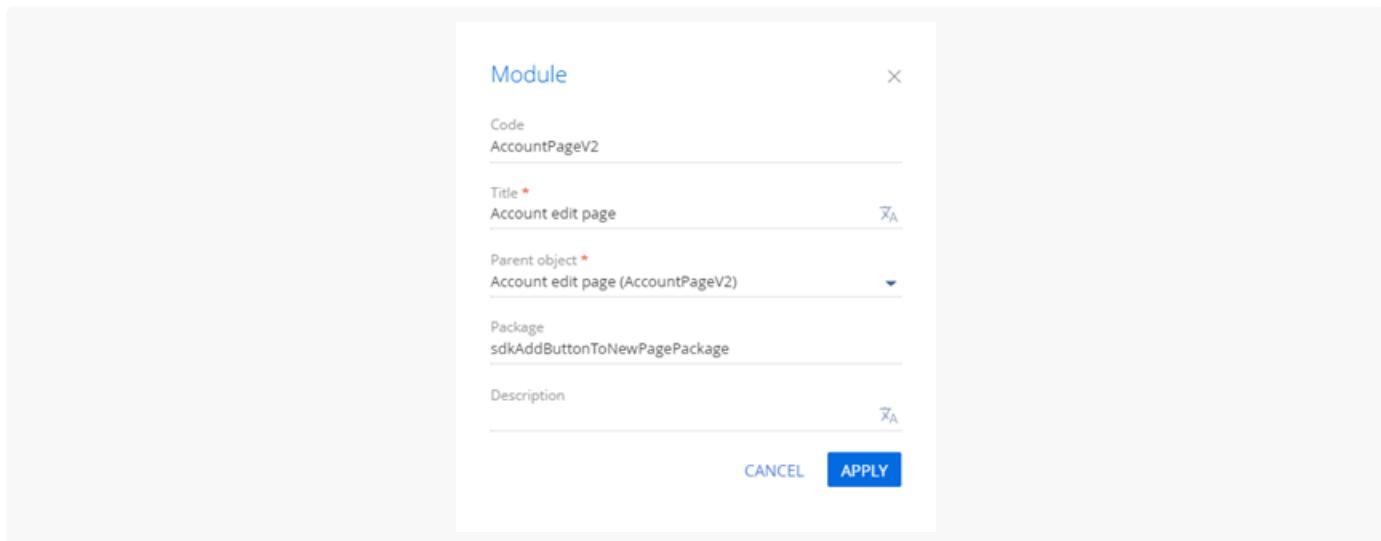
2. Создать схему замещающей модели представления страницы контрагента

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



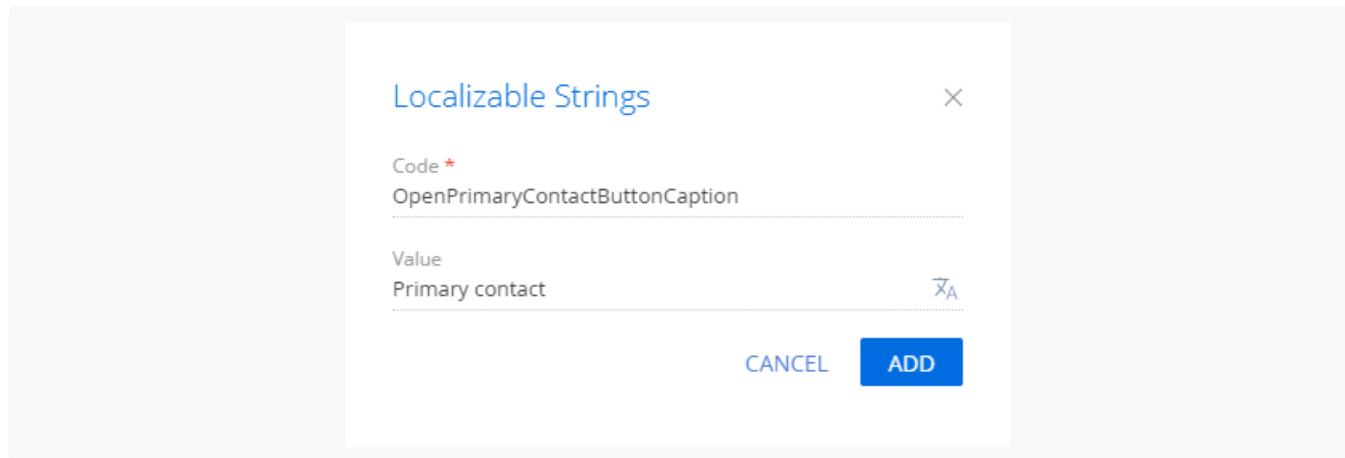
3. Заполните **свойства схемы**.

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



4. Добавьте **локализуемую строку**.

- В контекстном меню узла [Локализуемые строки] ([Localizable strings]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
 - [Код] ([Code]) — "OpenPrimaryContactButtonCaption".
 - [Значение] ([Value]) — "Основной контакт" ("Primary contact").



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

5. Реализуйте логику работы кнопки.

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

- `isAccountPrimaryContactSet()` — проверяет заполнение поля [Основной контакт] ([Primary contact]) страницы.
- `onOpenPrimaryContactClick()` — метод-обработчик нажатия кнопки. Выполняет переход на страницу основного контакта.

д. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения кнопки на странице.

Исходный код схемы замещающей модели представления страницы контрагента представлен ниже.

AccountPageV2

```
define("AccountPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Account",
        /* Методы модели представления страницы записи. */
        methods: {
            /* Проверяет заполнение поле [Основной контакт] страницы записи. */
            isAccountPrimaryContactSet: function() {
                return this.get("PrimaryContact") ? true : false;
            },
            /* Метод-обработчик нажатия кнопки. */
            onOpenPrimaryContactClick: function() {
                var primaryContactObject = this.get("PrimaryContact");
                if (primaryContactObject) {
                    /* Определение идентификатора основного контакта. */
                    var primaryContactId = primaryContactObject.value;
                    /* Формирование строки адреса. */
                    var requestUrl = "CardModuleV2/ContactPageV2/edit/" + primaryContactId;
                    // Публикация сообщения о пополнении истории навигации по страницам и пер
            }
        }
    }
});
```

```

        this.sandbox.publish("PushHistoryState", {
            hash: requestUrl
        });
    }
},
/* Отображение кнопки на странице записи. */
diff: [
    /* Метаданные для добавления на страницу пользовательской кнопки. */
    {
        /* Выполняется операция добавления элемента на страницу. */
        "operation": "insert",
        /* Мета-имя родительского контейнера, в который добавляется кнопка. */
        "parentName": "LeftContainer",
        /* Кнопка добавляется в коллекцию элементов родительского элемента. */
        "propertyName": "items",
        /* Мета-имя добавляемой кнопки. */
        "name": "PrimaryContactButton",
        /* Свойства, передаваемые в конструктор элемента. */
        "values": {
            /* Тип добавляемого элемента – кнопка. */
            "itemType": Terrasoft.ViewItemType.BUTTON,
            /* Привязка заголовка кнопки к локализуемой строке схемы. */
            "caption": {bindTo: "Resources.Strings.OpenPrimaryContactButtonCaption"},
            /* Привязка метода-обработчика нажатия кнопки. */
            "click": {bindTo: "onOpenPrimaryContactClick"},
            /* Привязка свойства доступности кнопки. */
            "enabled": {bindTo: "isAccountPrimaryContactSet"},
            /* Стиль отображения кнопки. */
            "style": Terrasoft.controls.ButtonEnums.style.BLUE
        }
    }
],
};

});

```

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

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Обновите страницу раздела [Контрагенты] ([Accounts]).
2. На панели инструментов раздела [Контрагенты] ([Accounts]) нажмите кнопку [Добавить контрагента] ([New account]).

В результате выполнения примера на панель инструментов страницы добавления контрагента добавлена неактивна кнопка [Основной контакт] ([Primary contact]).

New record

What can I do for you? >

Creatio
7.18.3.1241

VIEW ▾

SAVE CANCEL ACTIONS ▾

PRIMARY CONTACT

Primary phone

No. of employees

Business entity

Category

Annual revenue

Communication options +

Addresses

Banking details

Noteworthy events + :

No data

Configuration items + :

No data

Industry

PRIMARY CONTACT

New contact

Search

Кнопка [Основной контакт] ([Primary contact]) активируется после указания основного контакта контрагента. При нажатии на кнопку [Основной контакт] ([Primary contact]) открывается страница этого контакта.

Добавить кнопку выбора цвета на страницу записи

Средний

Пример. Добавить кнопку выбора цвета на страницу продукта.

1. Создать схему замещающего объекта

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

The screenshot shows a list of objects in a software application. The objects listed are:

Status	Type
	Object
Created	Replacing object
Page	Object
Module	Client module

3. Заполните **свойства схемы**.

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

General

Code *	Title *
Product	Product

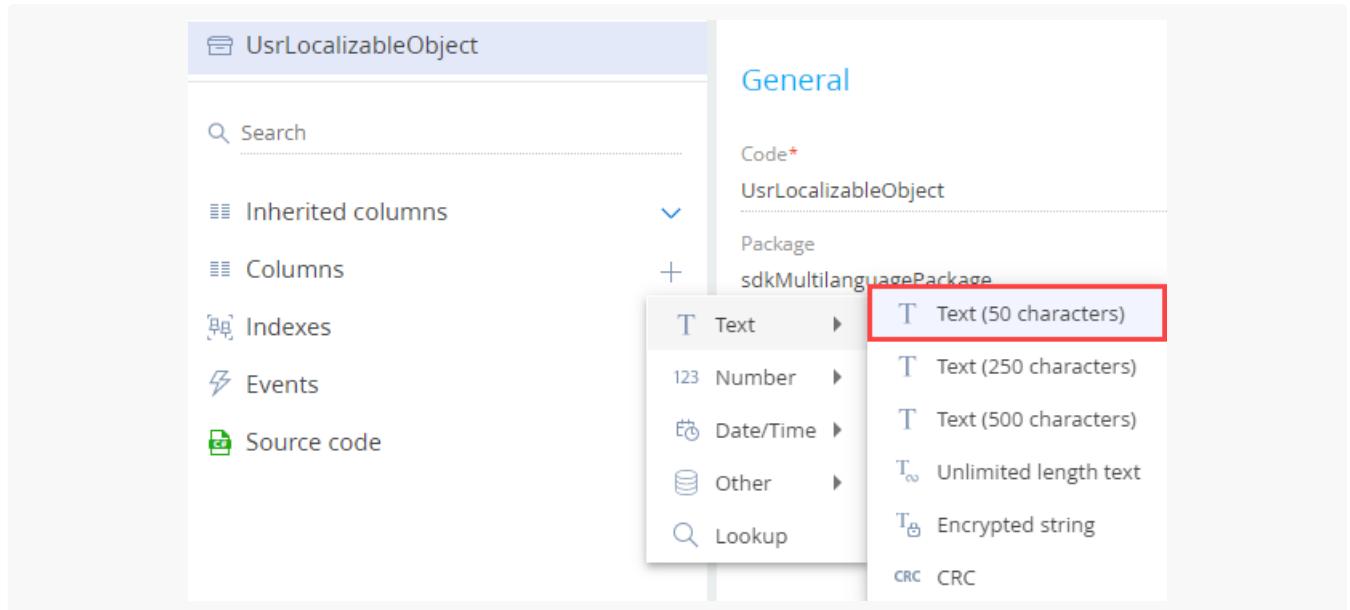
Package	Description
sdkAddColorButtonPackage	

Inheritance

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

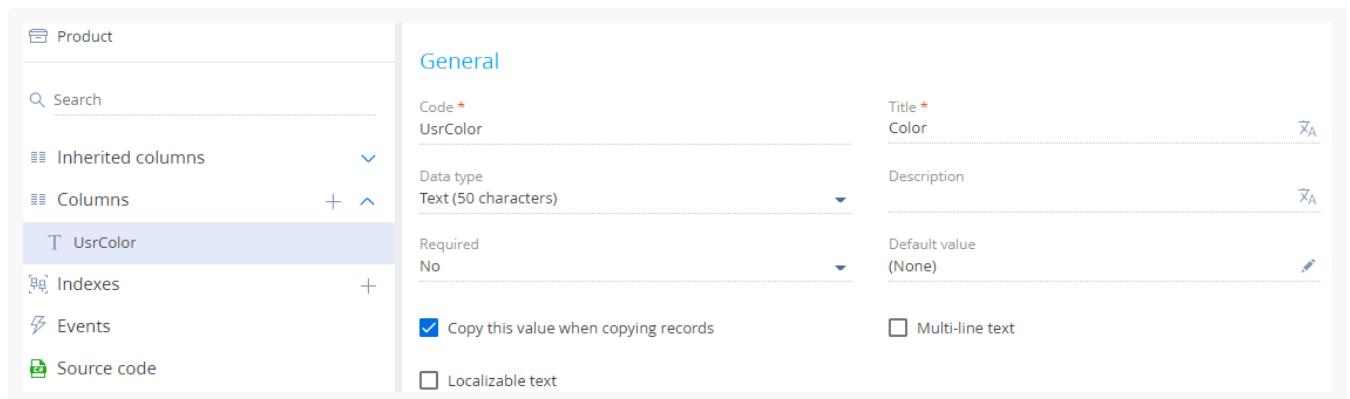
4. В схему добавьте **колонку**.

- В контекстном меню узла [Колонки] ([Columns]) структуры объекта нажмите **+**.
- В выпадающем меню нажмите [Странка] —> [Странка (50 символов)] ([Text] —> [Text (50 characters)]).



с. Заполните **свойства добавляемой колонки**.

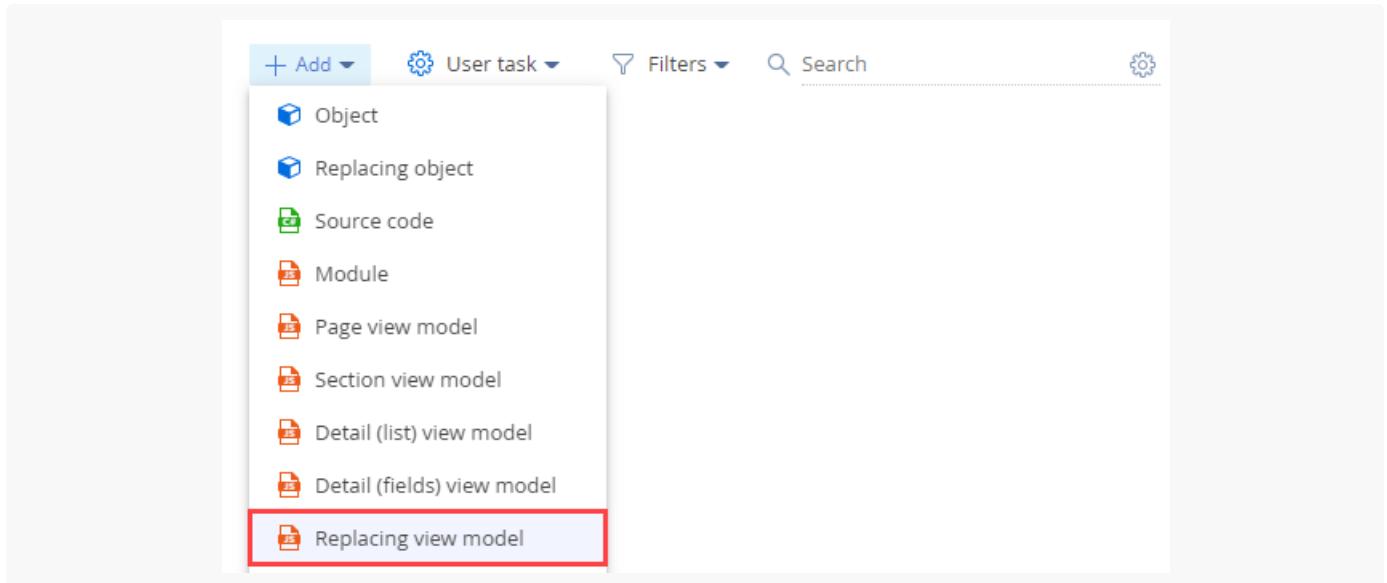
- [Код] ([Code]) — "UsrColor".
- [Заголовок] ([Title]) — "Цвет" ("Color").



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

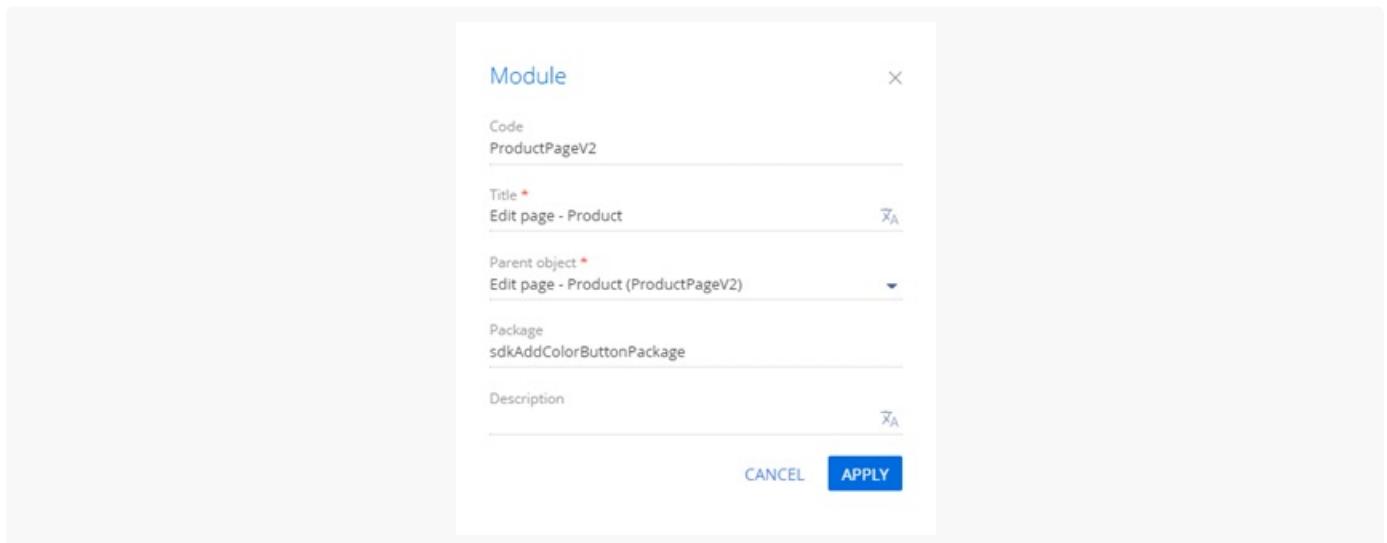
2. Создать схему замещающей модели представления страницы контрагента

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



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

Исходный код схемы замещающей модели представления страницы продукта представлен ниже.

```
ProductPageV2

define("ProductPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Product",
        /* Отображение кнопки на странице записи. */
    }
})
```

```

diff: /**SCHEMA_DIFF*/
  /* Кнопка выбора цвета. */
  {
    /* Выполняется операция добавления элемента на страницу. */
    "operation": "insert",
    /* Мета-имя родительского контейнера, в который добавляется кнопка. */
    "parentName": "ProductGeneralInfoBlock",
    /* Кнопка добавляется в коллекцию элементов родительского элемента. */
    "propertyName": "items",
    /* Мета-имя добавляемой кнопки. */
    "name": "ColorButton",
    /* Свойства, передаваемые в конструктор элемента. */
    "values": {
      /* Тип добавляемого элемента – кнопка выбора цвета. */
      "itemType": this.Terrasoft.ViewItemType.COLOR_BUTTON,
      /* Привязка значения элемента управления к колонке модели представления. */
      "value": { "bindTo": "UsrColor" },
      /* Настройка расположения кнопки. */
      "layout": {
        /* Номер столбца. */
        "column": 5,
        /* Номер строки. */
        "row": 6,
        /* Диапазон занимаемых столбцов. */
        "colSpan": 12
      }
    }
  }
];
/**SCHEMA_DIFF*/
);
});

```

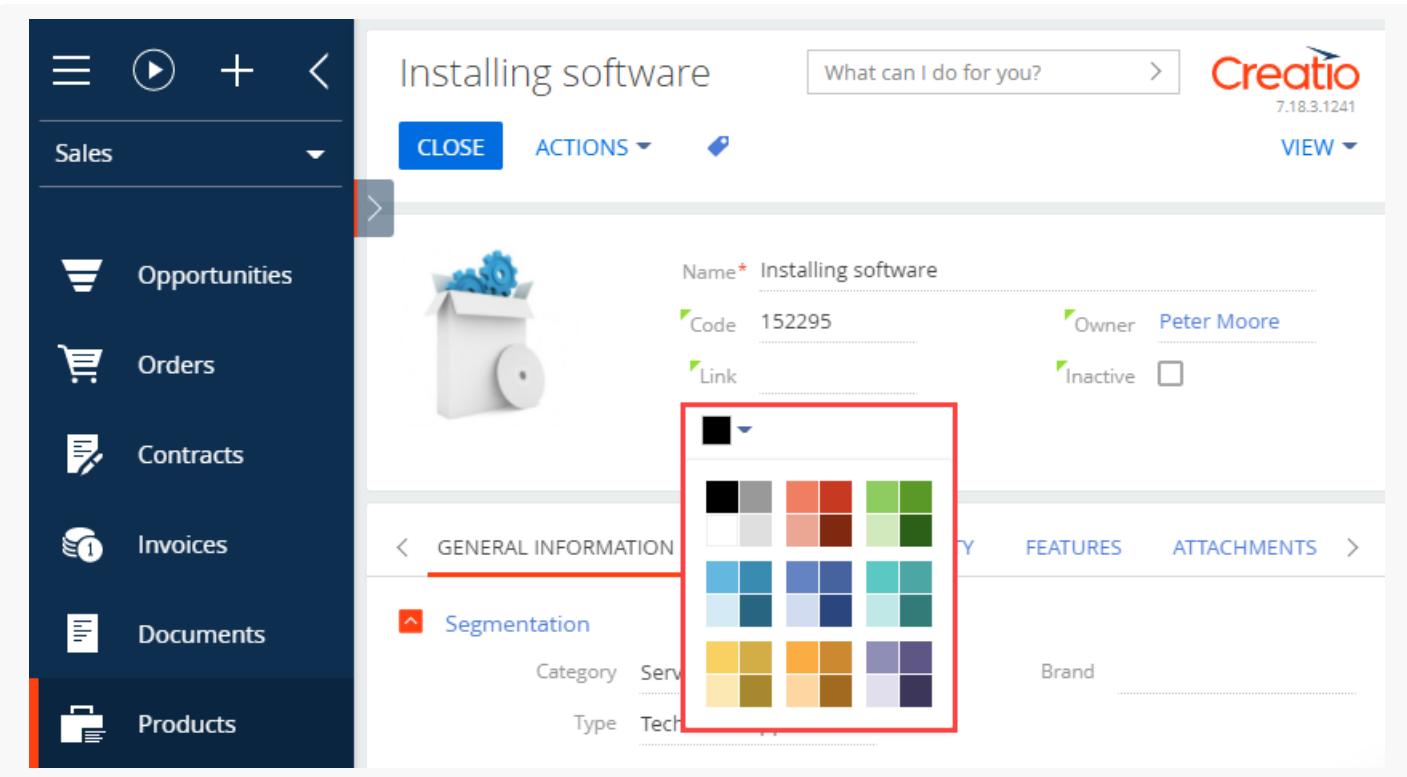
- На панели инструментов дизайнера нажмите [Сохранить] ([Save]).

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

- Обновите страницу раздела [Продукты] ([Products]).
- Откройте страницу продукта.

В результате выполнения примера на страницу продукта добавлена кнопка выбора цвета.



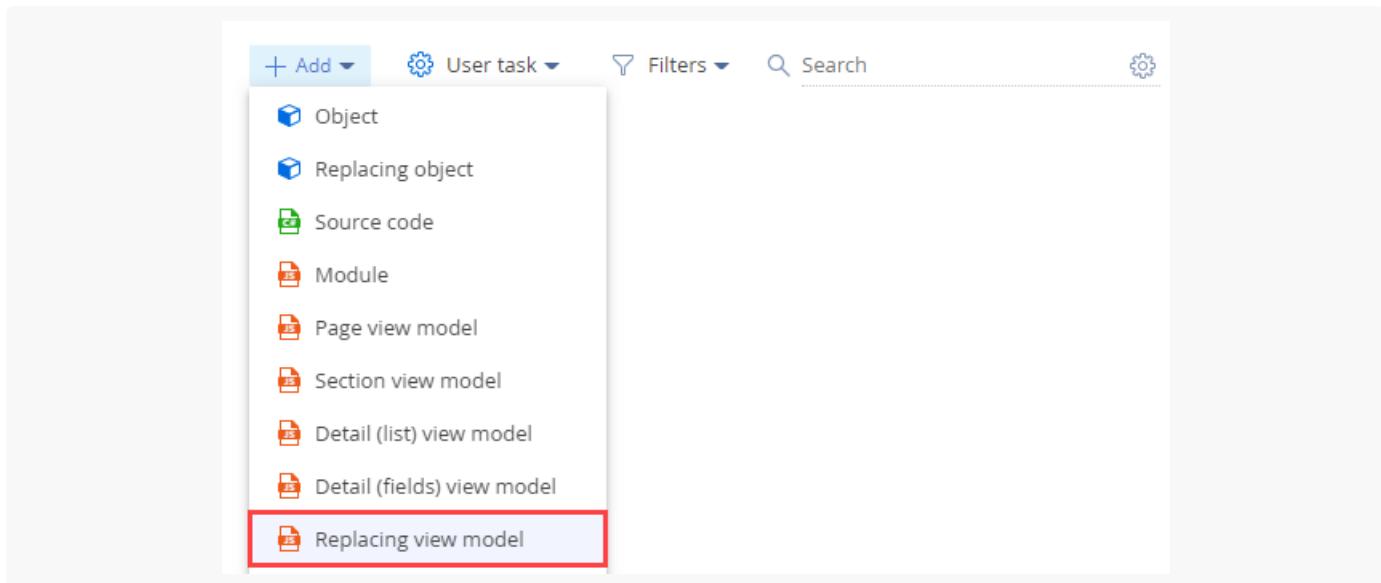
Добавить к кнопке всплывающую подсказку

Сложный

Пример. Добавить всплывающую подсказку к кнопке [Сохранить] ([Save]) страницы контакта.

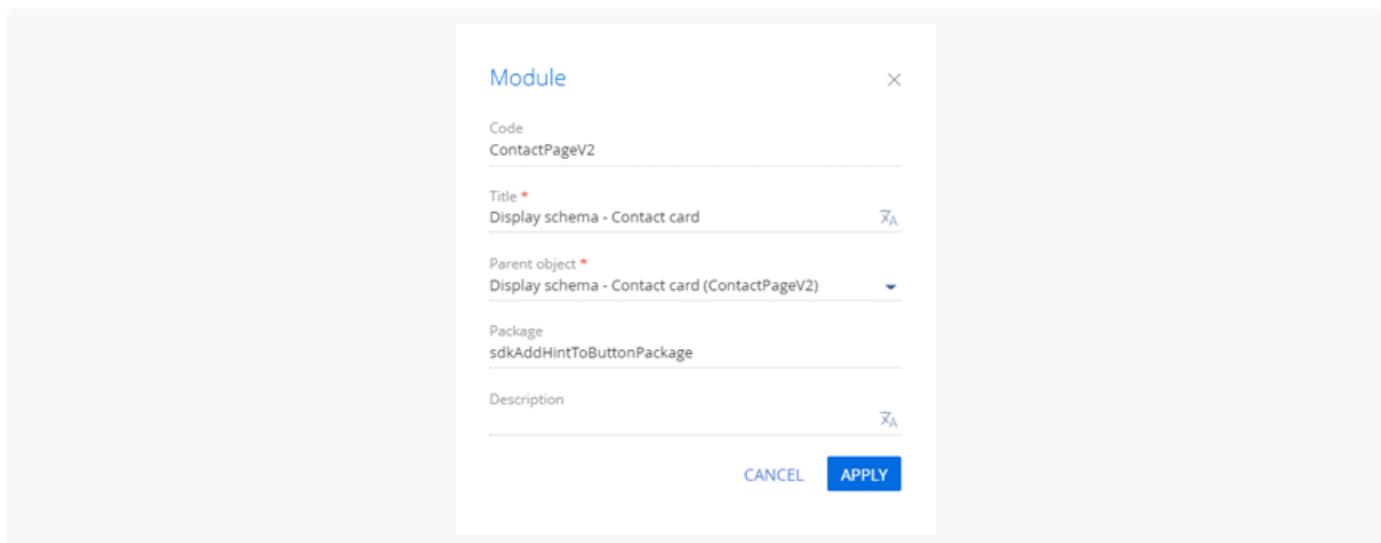
Создать схему замещающей модели представления страницы контакта

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]).



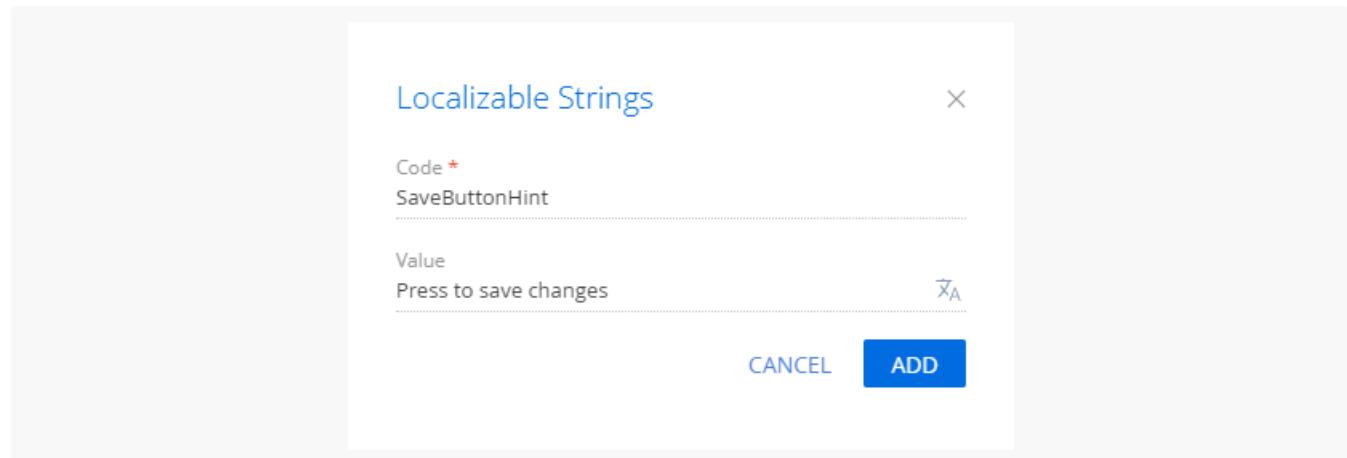
3. Заполните **свойства схемы**.

- [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.

- В контекстном меню узла [Локализуемые строки] ([Localizable strings]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
 - [Код] ([Code]) — "SaveButtonHint".
 - [Значение] ([Value]) — "Нажмите, чтобы сохранить изменения" ("Press to save changes").



- е. Для добавления локализуемой строки нажмите [Добавить] ([Add]).
5. Настройте **всплывающую подсказку к кнопке** [Сохранить] ([Save]) страницы контакта. Для этого в массив модификаций `diff` добавьте конфигурационный объект кнопки на странице. Исходный код схемы замещающей модели представления страницы контакта представлен ниже. Используются разные способы добавления всплывающей подсказки в конфигурационный объект кнопки.

ContactPageV2 (вариант 1)

```
define("ContactPageV2", [], function () {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение всплывающей подсказки. */
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для добавления к кнопке всплывающей подсказки. */
            {
                /* Выполняется операция изменения существующего элемента. */
                "operation": "merge",
                /* Мета-имя родительского контейнера, в котором изменяется кнопка. */
                "parentName": "LeftContainer",
                /* Кнопка изменяется в коллекции элементов родительского элемента. */
                "propertyName": "items",
                /* Мета-имя изменяемой кнопки. */
                "name": "SaveButton",
                /* Свойства, передаваемые в конструктор элемента. */
                "values": {
                    /* Всплывающая подсказка для кнопки. */
                    "hint": { "bindTo": "Resources.Strings.SaveButtonHint" }
                }
            }
        ]/**SCHEMA_DIFF*/
    };
});
```

ContactPageV2 (вариант 2)

```

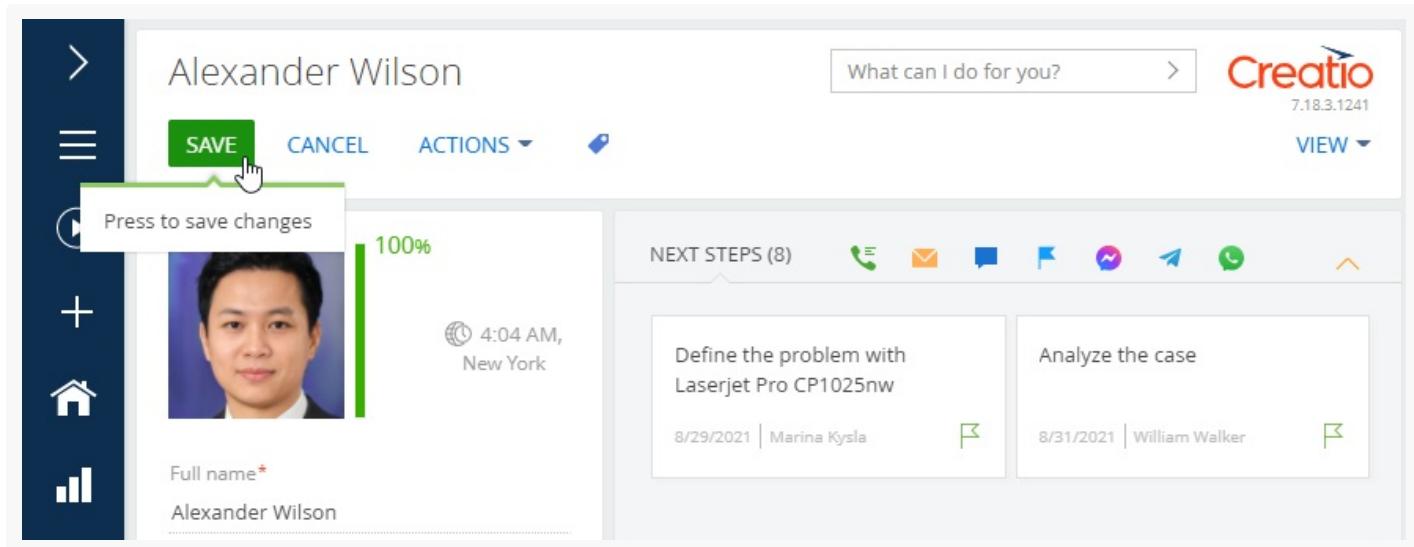
define("ContactPageV2", [], function () {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение всплывающей подсказки. */
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для добавления к кнопке всплывающей подсказки. */
            {
                /* Выполняется операция изменения существующего элемента. */
                "operation": "merge",
                /* Мета-имя родительского контейнера, в котором изменяется кнопка. */
                "parentName": "LeftContainer",
                /* Кнопка изменяется в коллекции элементов родительского элемента. */
                "propertyName": "items",
                /* Мета-имя изменяемой кнопки. */
                "name": "SaveButton",
                /* Свойства, передаваемые в конструктор элемента. */
                "values": {
                    /* Массив подсказок для кнопки. */
                    "tips": []
                }
            },
            /* Конфигурационный объект простой подсказки. */
            {
                /* Выполняется операция добавления элемента. */
                "operation": "insert",
                /* Мета-имя добавляемой кнопки. */
                "parentName": "SaveButton",
                /* Подсказка добавляется в коллекцию элементов родительского элемента. */
                "propertyName": "tips",
                /* Мета-имя добавляемой подсказки. */
                "name": "CustomShowedTip",
                /* Свойства, передаваемые в конструктор элемента. */
                "values": {
                    /* Текст подсказки. */
                    "content": {"bindTo": "Resources.Strings.SaveButtonHint"}
                    /* Здесь можно дополнительно настроить другие параметры отображения и раб
                }
            },
            /**SCHEMA_DIFF*/
        ];
    };
});

```

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

Результат выполнения примера

В результате выполнения примера к кнопке [Сохранить] ([Save]) страницы контакта добавлена всплывающая подсказка.



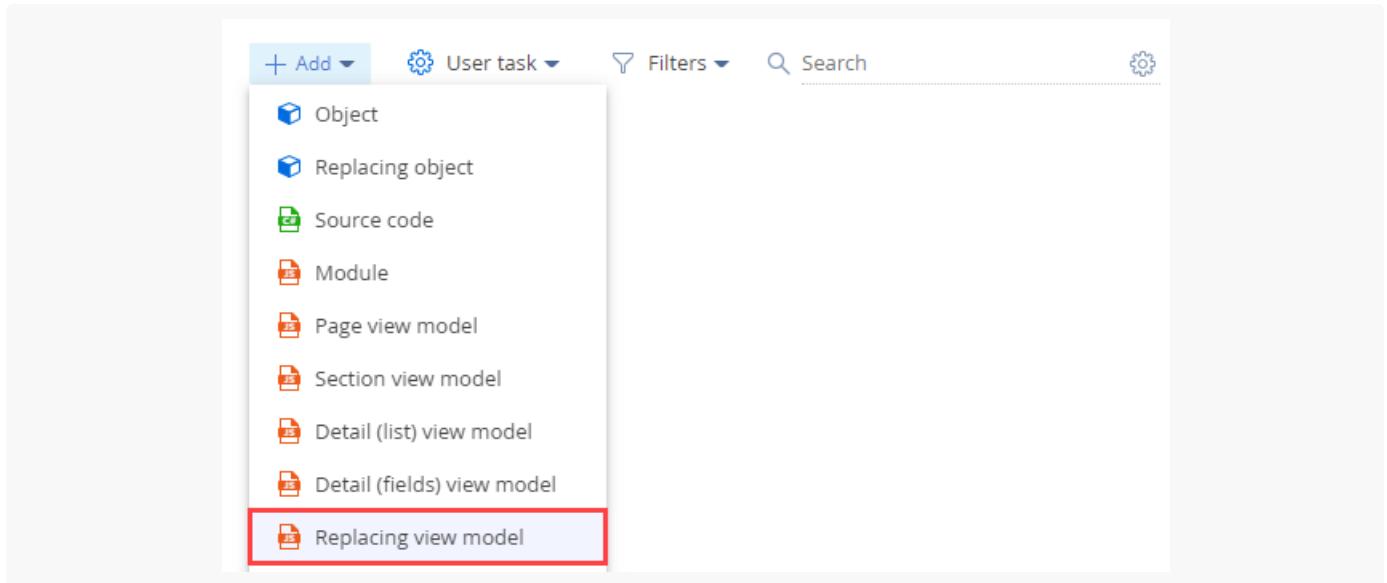
Добавить кнопку в строку записи раздела

Средний

Пример. Добавить кнопку [Показать возраст] ([Show age]) в строку активной записи раздела [Контакты] ([Contacts]). При нажатии на кнопку во всплывающем окне будет отображаться возраст выбранного контакта.

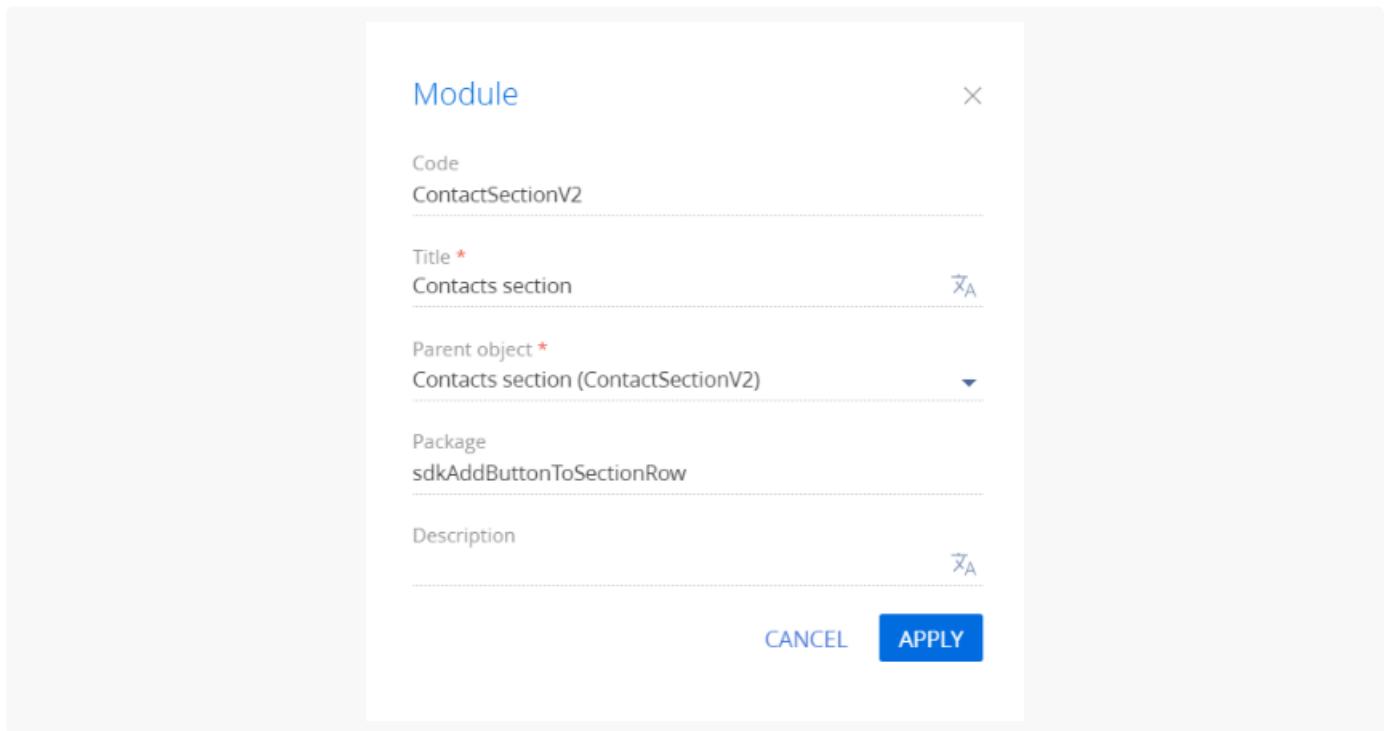
Создать схему замещающей модели представления раздела

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



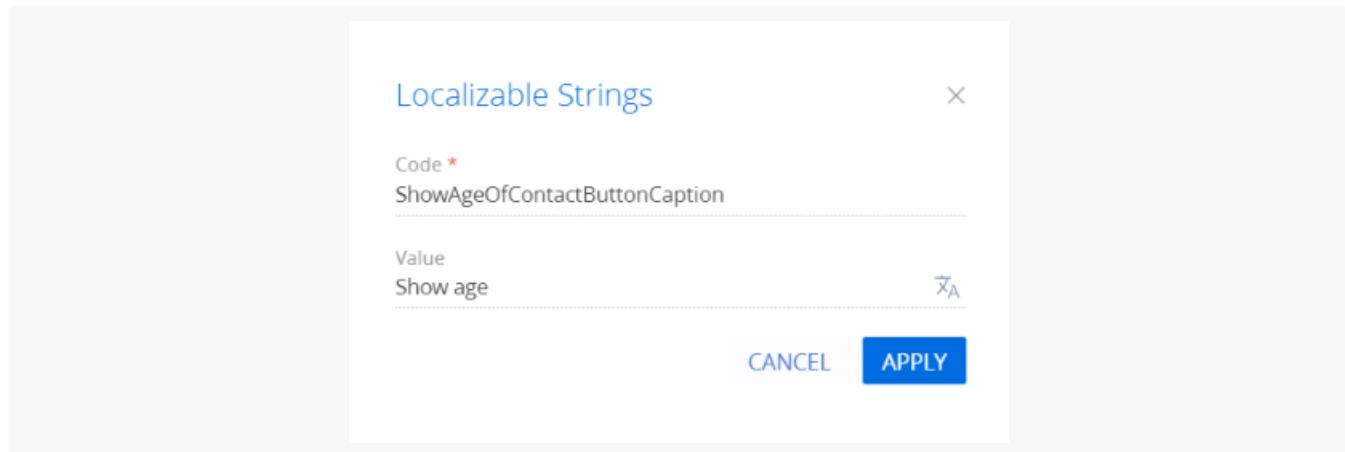
3. Заполните **свойства схемы**.

- [Код] ([Code]) — "ContactSectionV2".
- [Заголовок] ([Title]) — "Раздел контакты" ("Contact section").
- [Родительский объект] ([Parent object]) — выберите "ContactSectionV2".



4. Добавьте **локализуемую строку**.

- В контекстном меню узла [Локализуемые строки] ([Localizable strings]) нажмите кнопку **+**.
- Заполните **свойства локализуемой строки**.
 - [Код] ([Code]) — "ShowAgeOfContactButtonCaption".
 - [Значение] ([Value]) — "Показать возраст" ("Show age").



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

5. Реализуйте логику работы кнопки.

а. В свойстве `methods` реализуйте методы:

- `onActiveRowAction()` — присваивает метод-обработчик кнопке, расположенной в строке активной записи раздела.
- `onShowAgeButtonClicked()` — метод-обработчик нажатия кнопки. Возвращает возраст контакта во всплывающем окне.

д. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения кнопки на странице.

Обращение к выделенной записи выполняется через атрибут `ActiveRow` модели представления раздела. Атрибут возвращает значение первичной колонки выделенной записи. В свою очередь, значение первичной колонки выделенной записи может использоваться для получения значений, загруженных в реестр полей выбранного объекта.

Исходный код схемы замещающей модели представления страницы раздела представлен ниже.

ContactSectionV2

```
define("ContactSectionV2", ["ContactSectionV2Resources"], function (resources) {
    return {
        /* Название схемы объекта раздела. */
        entitySchemaName: "Contact",
        /* Методы модели представления раздела. */
        methods: {
            onActiveRowAction: function (buttonTag) {
                switch (buttonTag) {
                    case "showAgeButton":
                        this.onShowAgeButtonClicked();
                        break;
                    default:
                        this.callParent(arguments);
                        break;
                }
            }
        }
    }
});
```

```

        }
    },
    /* Метод-обработчик нажатия кнопки. */
    onShowAgeButtonClicked: function () {
        var message = "";
        var activeRow = this.getActiveRow();
        var recordId = activeRow.get("Id");
        /* Создаем экземпляр класса Terrasoft.EntitySchemaQuery с корневой схемой Con
        var esq = this.Ext.create("Terrasoft.EntitySchemaQuery", {
            rootSchemaName: "Contact"
        });
        /* Добавление колонки с возрастом. */
        esq.addColumn("Age", "Age");
        /* Получение записи из выборки по Id объекта. */
        esq.getEntity(recordId, function(result) {
            if (!result.success) {
                this.showInformationDialog("Error");
                return;
            }
            message += "Age of contact is " + result.entity.get("Age");
            this.showInformationDialog(message);
        }, this);
    }
},
/* Отображение кнопки в разделе. */
diff: /**SCHEMA_DIFF*/[
    /* Метаданные для добавления в раздел пользовательской кнопки. */
    {
        /* Выполняется операция добавления элемента на страницу. */
        "operation": "insert",
        /* Мета-имя добавляемой кнопки. */
        "name": "DataGridActiveRowShowAgeButton",
        /* Мета-имя родительского контейнера, в который добавляется кнопка. */
        "parentName": "DataGrid",
        /* Кнопка добавляется в коллекцию элементов родительского элемента. */
        "propertyName": "activeRowActions",
        /* Свойства, передаваемые в конструктор элемента. */
        "values": {
            "className": "Terrasoft.Button",
            "style": Terrasoft.controls.ButtonEnums.style.GREEN,
            /* Привязка заголовка кнопки к локализуемой строке схемы. */
            "caption": resources.localizableStrings.ShowAgeOfContactButtonCaption,
            "tag": "showAgeButton"
        }
    }
]/**SCHEMA_DIFF*/
};

});

```

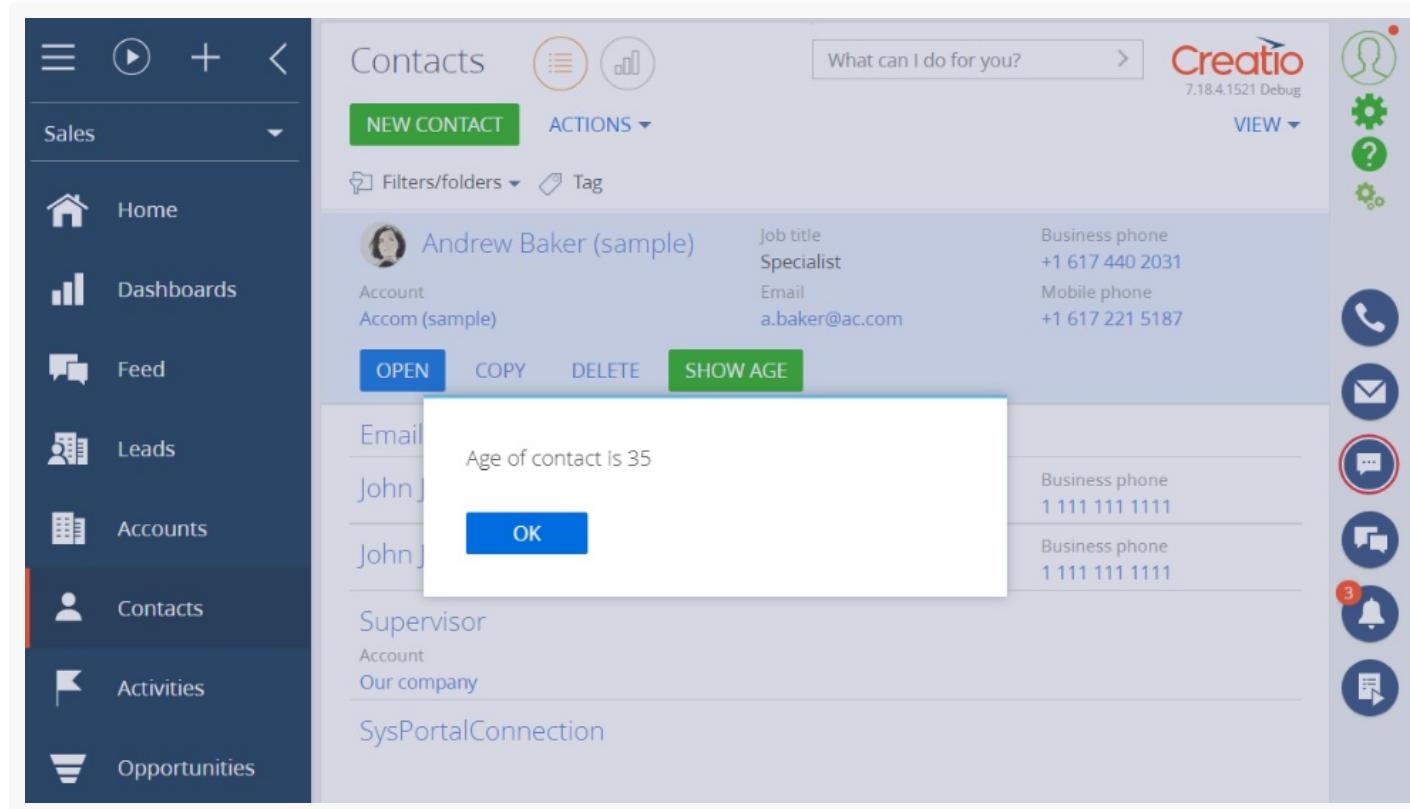
6. На панели инструментов дизайнера нажмите [Сохранить] ([Save]).

Результат выполнения примера

Чтобы посмотреть результат выполнения примера:

1. Очистите кэш браузера.
2. Обновите страницу раздела [Контакты] ([Contacts]).

В результате выполнения примера в строку активной записи раздела [Контакты] ([Contacts]) добавлена кнопка [Показать возраст] ([Show Age]). При нажатии на кнопку [Показать возраст] ([Show Age]) отображается всплывающее окно с информацией о возрасте контакта.



Свойство diff объекта кнопки js

Легкий

Назначение массива модификаций `diff` объекта кнопки — настройка расположения кнопки на странице записи путем использования конфигурационного объекта. Настройка выполняется в схеме модели представления страницы записи. Описание свойств схемы содержится в статье [Клиентская схема](#).

Свойства

operation

Операция с кнопкой.

Возможные значения

<code>set</code>	Значение кнопки устанавливается значением параметра <code>values</code> .
<code>merge</code>	Значения из родительских, замещаемых и замещающих схем сливаются вместе, при этом свойства из значения параметра <code>values</code> последнего наследника имеют приоритет.
<code>remove</code>	Кнопка удаляется из схемы.
<code>move</code>	Кнопка перемещается в другой родительский элемент.
<code>insert</code>	Кнопка добавляется в схему.

parentName

Мета-имя родительского элемента управления, в который помещается кнопка. Если это функциональная кнопка, то в качестве родительских контейнеров могут выступать `LeftContainer` и `RightContainer`.

propertyName

Имя параметра родительского элемента. Для кнопки указывается значение `items`.

name

Мета-имя добавляемой кнопки.

values

Конфигурационный объект с настройками дополнительных свойств кнопки.

Свойства конфигурационного объекта

itemType	Тип элемента. Задается значением перечисления <code>Terrasoft.ViewItemType</code> . Для кнопки используется значение <code>BUTTON</code> .
caption	Заголовок кнопки. Рекомендуется задавать значения заголовков через привязку к локализируемой строке схемы.
click	Привязка метода-обработчика кнопки.
layout	Объект настроек расположения кнопки в сетке.
enabled	Регулирует доступность (активность) кнопки.
visible	Регулирует видимость кнопки.
style	Стиль компонента. Задается значением перечисления <code>Terrasoft.controls.ButtonEnums.style</code> . Возможные значения (<code>Terrasoft.controls.ButtonEnums.style</code>)
DEFAULT	Стиль по умолчанию.
GREEN	Цвет кнопки — зеленый.
RED	Цвет кнопки — красный.
BLUE	Цвет кнопки — синий.
GREY	Цвет кнопки — серый.
TRANSPARENT	Прозрачная кнопка. Значение из предыдущих версий Creatio.

Хронология



Начиная с версии 7.12.0 для быстрого анализа истории работы с клиентами, продажами, обращениями и т.п. используется вкладка [Хронология]. Эта вкладка по умолчанию доступна в разделах [Контакты], [Контрагенты], [Лиды], [Продажи], [Обращения].

Таблицы базы данных

Для настройки хронологии в базе данных предусмотрены следующие таблицы:

- TimelinePageSetting — для настройки разделов и их плиток.
- TimelineTileSetting — для настройки всех преднастроенных и пользовательских плиток хронологии.
- SysTimelineTileSettingLcz — для локализации имен плиток.

Основные колонки таблицы TimelinePageSetting

Колонка	Описание
Id	Идентификатор записи.
Key	Ключ — название схемы страницы раздела. Например, AccountPageV2, ContactPageV2 и т. д.
Data	Настройки хронологии для раздела в формате JSON.

Основные колонки таблицы TimelineTileSetting

Колонка	Описание
Id	Идентификатор записи.
Name	Заголовок плитки, который будет отображаться в меню фильтра. Должен быть во множественном числе, например "Задачи" ("Tasks"). Локализация осуществляется с помощью таблицы SysTimelineTileSettingLcz. Если данное поле не будет указано, тогда заголовок плитки будет взят из названия сущности или типа.
Data	Настройки хронологии для раздела в формате JSON.
Image	Иконка плитки, которая будет отображаться в меню фильтра и слева от плитки во вкладке [Хронология].

Параметры конфигурации плитки хронологии в формате JSON

Колонка	Описание	Обязательность	Пример

<code>entityConfigKey</code>	Ключ плитки. Должен совпадать с Id в таблице <code>TimelineTileSetting</code> соответствующей преднастроенной плитки, которую следует отображать для данной сущности.	Да	706f803d-6a30-4bcd-
<code>entitySchemaName</code>	Название схемы объекта сущности.	Да	Activity
<code>referenceColumnName</code>	Название колонки объекта, по которой будет происходить отбор записей.	Да	Account
<code>masterRecordColumnName</code>	Название колонки родительской записи, по которой будет происходить отбор записей.	Да	Id
<code>typeColumnName</code>	Название колонки типа.	Нет	Type
<code>typeColumnValue</code>	Значение колонки типа.	Указывается только при указании <code>typeColumnName</code>	fbe0acdc-cfc0-df11-
<code>viewModelClassName</code>	Название класса модели представления преднастроенной плитки.	Нет. Если значение отсутствует, то будет применен базовый класс <code>BaseTimelineItemViewModel</code>	Terrasoft.Activity`
<code>viewClassName</code>	Название класса представления преднастроенной плитки.	Нет. Если значение отсутствует, то будет применен базовый класс <code>BaseTimelineItemView</code>	Terrasoft.Activity`
<code>orderColumn</code>	Колонка для сортировки.	Да	StartDate
<code>authorColumnName</code>	Колонка для	Да	Owner

	автора.		
<code>captionColumnName</code>	Колонка для заголовка.	Да, если не указана колонка <code>messageColumnName</code>	<code>Title</code>
<code>messageColumnName</code>	Колонка для информационного сообщения.	Да, если не указана колонка <code>captionColumnName</code>	<code>DetailedResult</code>
<code>caption</code>	Заголовок плитки, который будет отображаться в меню фильтра. Должен быть во множественном числе, например "Задачи" ("Tasks"). Используется для задания заголовка плитки, отличного от указанного в поле <code>Name</code> настройки соответствующей плитки в <code>TimelinePageSetting</code> .	Нет	<code>MyActivity</code>
<code>columns</code>	Массив настроек дополнительных колонок для плитки.	Нет	
<code>columnName</code>	Путь к колонке в объекте сущности.	Да	<code>Result</code>
<code>columnAlias</code>	Псевдоним колонки в представлении модели плитки.	Да	<code>ResultMessage</code>
<code>isSearchEnabled</code>	Указывает на возможность текстового поиска по значению в колонке (только для текстовых	Нет	<code>true</code>

колонок).

Добавление вкладки [Хронология] в раздел

Для [добавления вкладки \[Хронология \]](#) на страницу раздела и отображения в нем записей определенных плиток, необходимо:

1. Создать новую запись в таблице `TimelinePageSetting`.
2. Заполнить соответствующие колонки. В колонке `Key` необходимо указать название схемы страницы раздела. Например, если необходимо добавить вкладку в раздел [Контрагенты], то значением колонки `Key` будет "AccountPageV2". Колонка `Data` содержит конфигурацию плиток хронологии, отображаемых на вкладке в указанном разделе, в формате JSON.

Важно. Вкладка [Хронология] не будет отображаться на странице записи раздела, если отсутствует конфигурация плиток в колонке `Data` или если же есть ошибки (например, синтаксические ошибки) в конфигурации.

Добавить базовую хронологию в раздел



Сложный

Пример. Добавить плитку [Договор] ([*Contract*]) на страницу раздела [Заказы] ([*Orders*]). Отсортировать записи по колонке `[StartDate]`.

Данные для полей плитки:

- Заголовок — колонка `[Number]`.
- Автор — колонка `[Owner]`.
- Сообщение — колонка `[Notes]`.

1. Создать SQL-сценарий

1. [Перейдите в раздел \[Конфигурация \]](#) ([*Configuration*]) и выберите пользовательский [пакет](#), в который будет добавлен SQL-сценарий.
2. На панели инструментов реестра раздела нажмите [Добавить] —> [SQL-сценарий] ([*Add*] —> [*SQL script*]).

The screenshot shows a list of object types on the left side of a dialog window. The types listed are: Object, Replacing object, Source code, Module, Page view model, Section view model, Detail (list) view model, Detail (fields) view model, Replacing view model, Business process, Rest service, Soap service, User task, and SQL script. The 'SQL script' option is highlighted with a red box.

3. Заполните **свойства схемы**.

- [Код] ([Code]) — "usrAddTimelineScript".
- Тип СУБД (DBMS type) — выберите необходимый тип СУБД, например "MSSql".
- Тип установки (Installation type) — выберите "AfterPackage".

The screenshot shows the configuration dialog for a SQL script. The fields are as follows:

- Code ***: usrAddTimelineScript
- DBMS type ***: MSSql
- Installation type ***: AfterPackage
- Package**: sdkBaseTimelineAdding

At the bottom right of the dialog are two buttons: **CANCEL** and **APPLY**.

4. Добавьте код SQL-схемария.

В коде укажите значения колонок:

- [Key] — "OrderPageV2".
- [Data] — JSON-объект с конфигурацией данных плитки.

В примере используется базовая плитка `Orders`. Для нее в таблице `[TimelineTileSettings]` базы данных уже существует запись с идентификатором "0ef5bd15-f3d3-4673-8af7-f2e61bc44cf0".

usrTimelineScript

MSSql

```
INSERT INTO TimelinePageSetting ([Key], [Data]) VALUES ('OrderPageV2', convert(VARBINARY(MAX)
{
    "entityConfigKey": "0ef5bd15-f3d3-4673-8af7-f2e61bc44cf0",
    "entitySchemaName": "Contract",
    "referenceColumnName": "Order",
    "orderColumnName": "StartDate",
    "authorColumnName": "Owner",
    "captionColumnName": "Number",
    "messageColumnName": "Notes",
    "caption": "My Contracts",
    "masterRecordColumnName": "Id"
})
])
```

PostgreSql

```
INSERT INTO "TimelinePageSetting" ("Key", "Data") VALUES ('OrderPageV2', cast ('[{"entityConf
```

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

2. Установить данные SQL-сценария

В контекстном меню созданного SQL-сценария нажмите [Установить] ([Install]) для выполнения скрипта в базе данных.

Скриншот интерфейса ПО, показывающий список объектов. Контекстное меню для элемента 'usrTimelineScript *' включает в себя опцию 'Install', которая выделена красным квадратом.

Результат выполнения примера

В результате выполнения примера в разделе [Заказы] ([Orders]) отображается вкладка [Хронология] ([Timeline]) с базовой плиткой.

Скриншот страницы деталей заказа 'ORD-1 (sample)'. Вкладка 'TIMELINE' выбрана. Ниже показано время и событие: '201 (sample)' с суммой '\$ 3,492.00' и заметкой 'Supervisor Mo 10/26/2020 12:00 AM'.

Создать хронологию, связанную с пользовательским разделом

Сложный

Пример. На вкладке [Хронология] ([Timeline]) страницы контрагента отобразить плитки,

связанные с пользовательским разделом [Книги] ([Books]). Плитки должны содержать:

- Иконка.
- Название.
- Автор.
- Дата добавления записи о книге.
- Стоимость.
- ISBN номер.
- Краткое описание книги.

Важно. Для реализации примера используйте on-site приложение.

1. Создать раздел [Книги] (Books))

Чтобы **создать раздел** [Книги] ([Books]), установите пакет примера [Привязать данные к пакету](#).

На заметку. Вы можете создать раздел самостоятельно, используя мастер разделов.

После установки пакета в рабочем месте [Продажи] ([Sales]) доступен раздел [Книги] ([Books]).

Name	Author	Publisher	ISBN	Price
JavaScript: The Definitive Guide: Activate Your Web Pages	David Flanagan	Apress	978-0596805524	33.89
Pro C# 7: With .NET and .NET Core	Andrew Troelsen	Apress	978-1484230176	56.99

На вкладке [Books] страницы контрагента появится деталь, которая отображает связанные записи раздела [Книги] ([Books]).

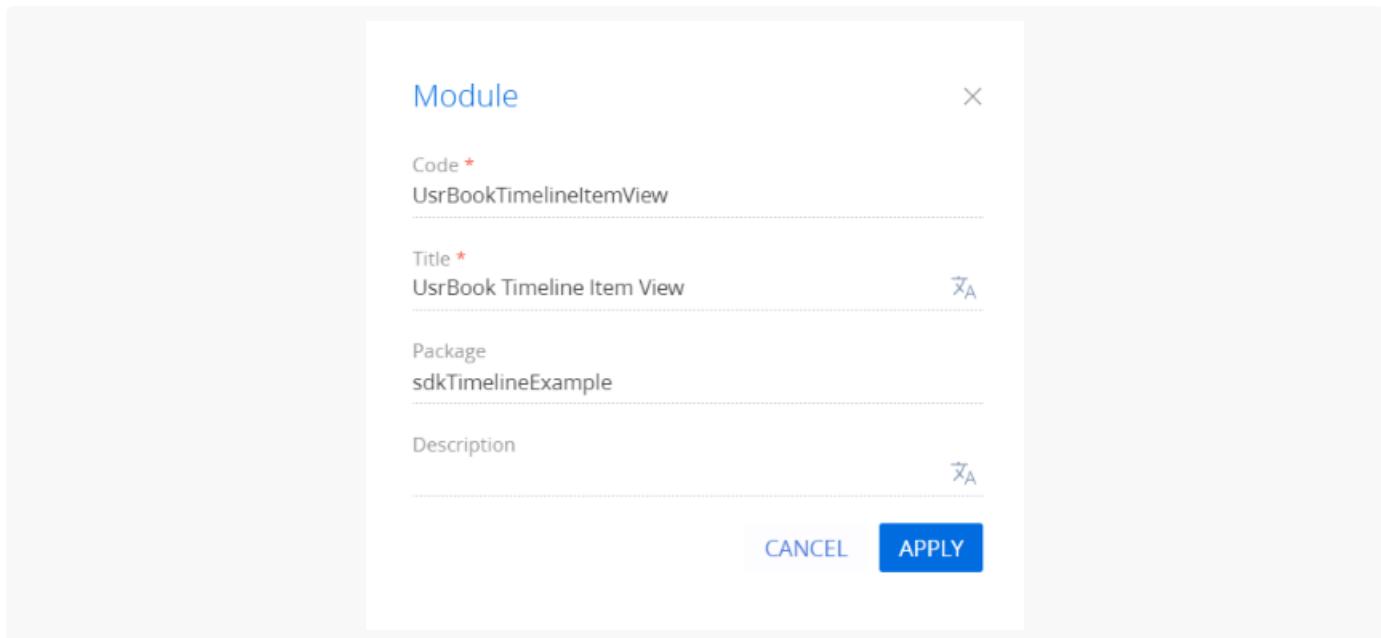
Name	ISBN	Author	Publisher
JavaScript: The Definitive Guide: Activate Your Web Pages	978-0596805524	David Flanagan	Apress
Pro C# 7: With .NET and .NET Core	978-1484230176	Andrew Troelsen	Apress

2. Создать модуль представления плитки

- Перейдите в раздел [Конфигурация] ([Configuration]) и выберите пользовательский пакет, в который будет добавлена схема. Установить зависимость от пакета Timeline.
- На панели инструментов реестра раздела нажмите [Добавить] —> [Модуль] ([Add] —> [Module]).

3. Заполните **свойства схемы**.

- [Код] ([Code]) — "UsrBookTimelineItemView".
- [Заголовок] ([Title]) — "Представление элемента хронологии UsrBook" ("UsrBook Timeline Item View").



4. Добавьте логику отображения плитки. Для этого реализуйте методы:

- `getUsrISBNViewConfig` — возвращает конфигурацию дополнительного поля [*UsrISBN*] плитки.
- `getUsrPriceViewConfig` — возвращает конфигурацию дополнительного поля [*UsrPrice*] плитки.
- `getBodyViewConfig` — переопределенный метод, который возвращает общую конфигурацию плитки.

Исходный код схемы модуля представления плитки представлен ниже.

UsrBookTimelineItemView

```
/* Определение модуля и его зависимостей.*/
define("UsrBookTimelineItemView", ["UsrBookTimelineItemViewResources", "BaseTimelineItemView"]
    /* Определение класса представления плитки.*/
    Ext.define("Terrasoft.configuration.UsrBookTimelineItemView", {
        extend: "Terrasoft.BaseTimelineItemView",
        alternateClassName: "Terrasoft.UsrBookTimelineItemView",
        /* Метод, возвращающий конфигурацию дополнительного поля [UsrISBN] плитки.*/
        getUsrISBNViewConfig: function() {
            return {
                /* Название поля.*/
                "name": "UsrISBN",
                /* Тип поля – метка.*/
                "itemType": Terrasoft.ViewItemType.LABEL,
                /* Заголовок.*/
                "caption": {
                    "bindTo": "UsrISBN"
                },
                /* Видимость.*/
                "visible": {
                    /* Привязка к колонке связанной с плиткой сущности.*/
                    "bindTo": "UsrISBN",

```

```

        /* Настройка видимости.*/
        "bindConfig": {
            /* Поле видимо, если значение в колонке не пустое.*/
            "converter": "checkIsEmpty"
        }
    },
    /* CSS-стили поля.*/
    "classes": {
        "labelClass": ["timeline-text-light"]
    }
};

},
/* Метод, возвращающий конфигурацию дополнительного поля [UsrPrice] плитки.*/
getUsrPriceViewConfig: function() {
    return {
        "name": "UsrPrice",
        "itemType": Terrasoft.ViewItemType.LABEL,
        "caption": {
            "bindTo": "UsrPrice"
        },
        "visible": {
            "bindTo": "UsrPrice",
            "bindConfig": {
                "converter": "checkIsEmpty"
            }
        },
        "classes": {
            "labelClass": ["timeline-item-subject-label"]
        }
    };
},
/* Переопределенный метод, возвращающий общую конфигурацию плитки.*/
getBodyViewConfig: function() {
    /* Получение стандартных настроек.*/
    var bodyConfig = this.callParent(arguments);
    /* Добавление конфигураций дополнительных полей.*/
    bodyConfig.items.unshift(this.getUsrISBNViewConfig());
    bodyConfig.items.unshift(this.getUsrPriceViewConfig());
    return bodyConfig;
}
});
});
});

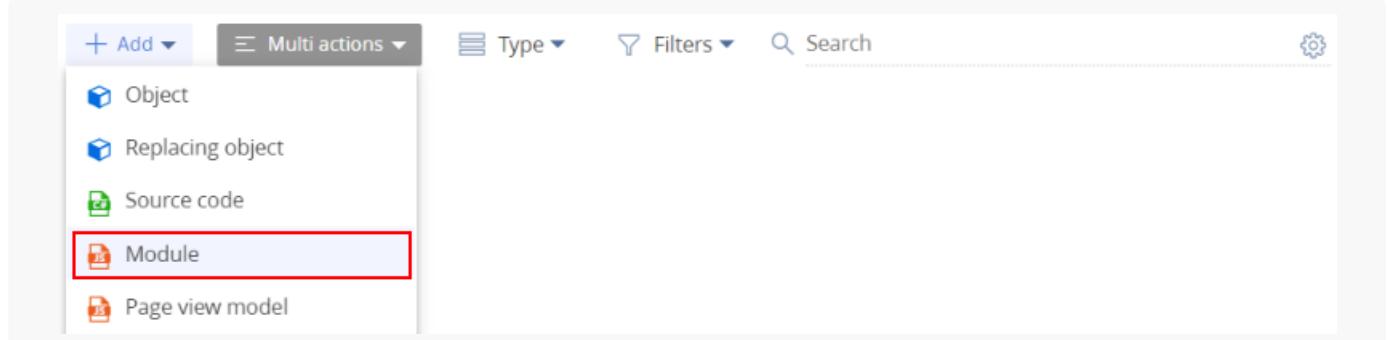
```

Здесь определяется конфигурация дополнительно отображаемых на плитке полей [UsrISBN] и [UsrPrice]. Стандартная конфигурация определена в модуле `BaseTimelineItemView`.

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

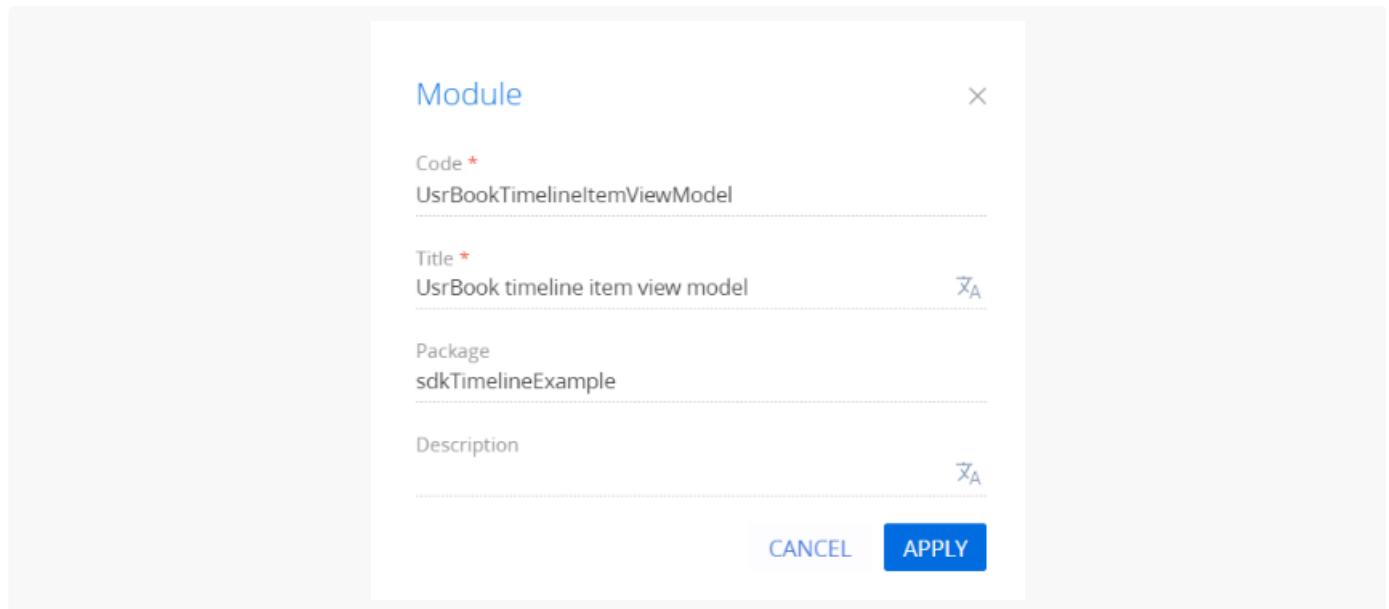
3. Создать модуль модели представления плитки

- Перейдите в раздел [Конфигурация] ([Configuration]) и выберите пользовательский пакет, в который будет добавлена схема. Установить зависимость от пакета `Timeline`.
- На панели инструментов реестра раздела нажмите [Добавить] —> [Модуль] ([Add] —> [Module]).



3. Заполните **свойства схемы**.

- [Код] ([Code]) — "UsrBookTimelineItemViewModel".
- [Заголовок] ([Title]) — "Модель представления элемента хронологии UsrBook" ("UsrBook timeline item view model").



4. В объявлении класса модуля в качестве зависимостей добавьте модули

`UsrBookTimelineItemViewModelResources` И `BaseTimelineItemViewModel`.

Исходный код модуля модели представления плитки представлен ниже.

UsrBookTimelineItemViewModel

```
define("UsrBookTimelineItemViewModel", ["UsrBookTimelineItemViewModelResources", "BaseTimelineItemViewModel",
    function() {
        Ext.define("Terrasoft.configuration.UsrBookTimelineItemViewModel", {
```

```

        alternateClassName: "Terrasoft.UsrBookTimelineItemViewModel",
        extend: "Terrasoft.BaseTimelineItemViewModel"
    });
});

```

Здесь определяется класс `Terrasoft.configuration.UsrBookTimelineItemViewModel`. Поскольку этот класс определен, как наследник `Terrasoft.BaseTimelineItemViewModel`, то это позволяет использовать функциональность базового класса.

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

4. Настроить отображение плитки

Настройка [свойств плиток хронологии](#) выполняется в таблице [TimelineTileSetting] базы данных.

Чтобы **настроить отображение плитки**:

- Создайте новую запись в таблице [TimelineTileSetting]. Для этого выполните SQL-запрос.

SQL-запрос

```

INSERT INTO TimelineTileSetting (CreatedOn, CreatedById, ModifiedOn, ModifiedById, Name, Data
VALUES (GETUTCDATE(), NULL, GETUTCDATE(), NULL, 'UsrBooks', NULL, NULL);

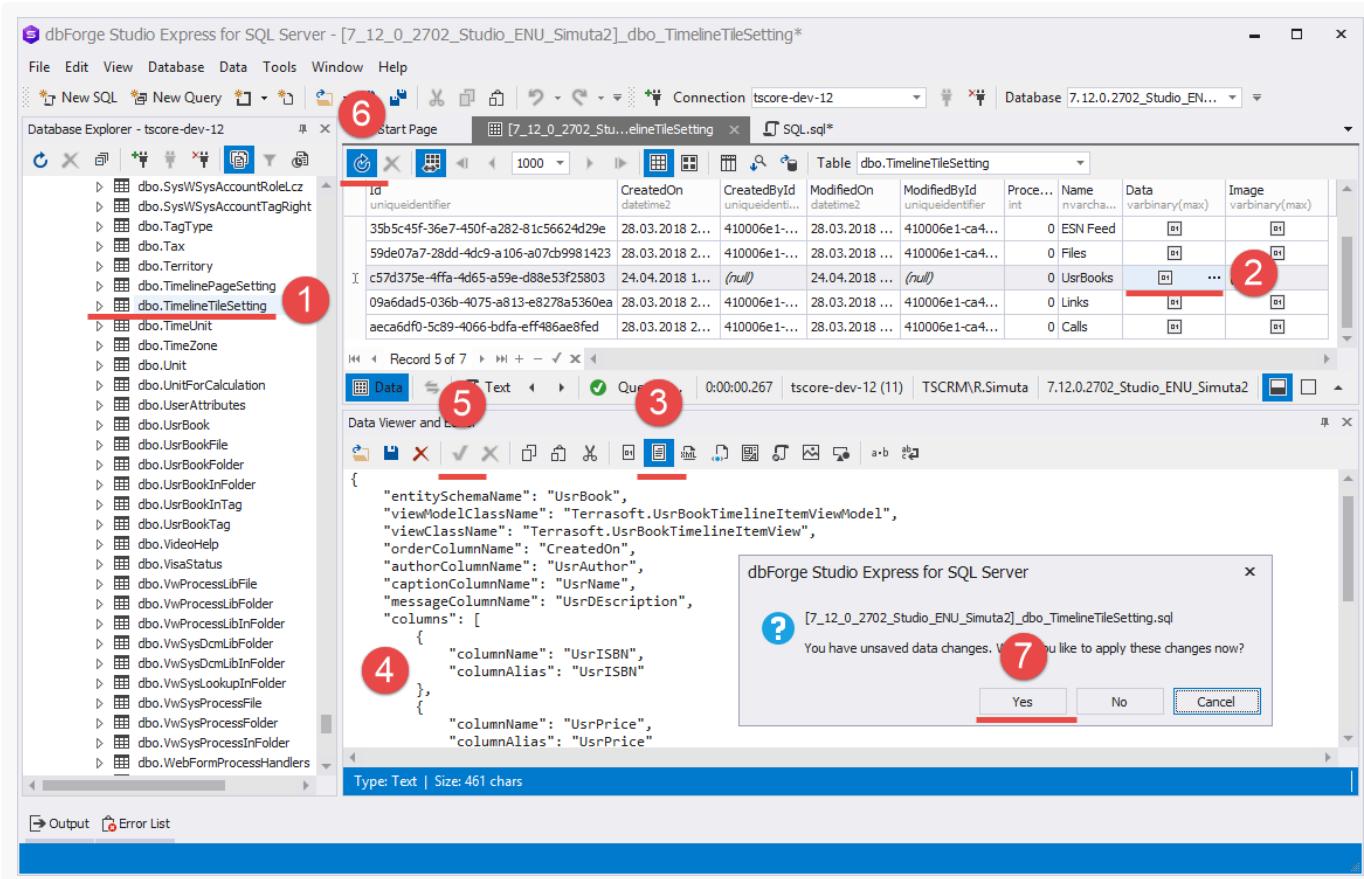
```

- Добавьте значение в колонки `[Data]` и `[Image]`.

Поскольку данные в колонках `[Data]` и `[Image]` хранятся в формате `varbinary(max)`, то редактировать их удобнее всего с помощью специализированных редакторов, например, dbForge Studio Express for SQL Server.

Чтобы **добавить значение в колонки `[Data]` и `[Image]`** с помощью dbForge Studio Express for SQL Server:

- Выберите необходимую таблицу (1).
- Выберите необходимую колонку записи и кликните по кнопке редактирования (2).
- В редакторе данных перейдите в режим текстового отображения данных(3).
- Добавьте необходимые данные (4).
- В редакторе данных нажмите на кнопку применения изменений (5).
- Нажмите на кнопку обновления данных (6).
- В появившемся диалоговом окне согласитесь с применением изменений (7).



Важно. Это способ подходит только для сред разработки, которые развернуты on-site.

Изменения вносятся непосредственно в базу данных, они не привязаны ни к одному пакету. При установке пакета со схемами представления и модели представления плитки в другое приложение изменения в базу данных внесены не будут. Для корректного переноса разработанной функциональности следует привязать SQL-скрипты, которые вносят соответствующие изменения в базу данных при установке пакета.

Добавьте в колонку [Data] конфигурационный объект.

Данные колонки [Data]

```
{
    "entitySchemaName": "UsrBook",
    "viewModelClassName": "Terrasoft.UsrBookTimelineItemViewModel",
    "viewClassName": "Terrasoft.UsrBookTimelineItemView",
    "orderColumnName": "CreatedOn",
    "authorColumnName": "UsrAuthor",
    "captionColumnName": "UsrName",
    "messageColumnName": "UsrDEscription",
    "columns": [
        {
            "columnName": "UsrISBN",
            "columnAlias": "UsrISBN"
        },
        {
            "columnName": "UsrPrice",
            "columnAlias": "UsrPrice"
        }
    ]
}
```

```

        "columnAlias": "UsrISBN"
    },
    {
        "columnName": "UsrPrice",
        "columnAlias": "UsrPrice"
    }
]
}

```

Здесь, кроме основных полей, унаследованных от базовой плитки, указывается также массив дополнительных полей, отображение которых сконфигурировано в модуле представления `UserBookTimelineItemView`.

Для отображения иконки, соответствующей иконке раздела, добавьте в колонку [`Image`] данные в SVG-формате.

Данные колонки [`Image`]

```

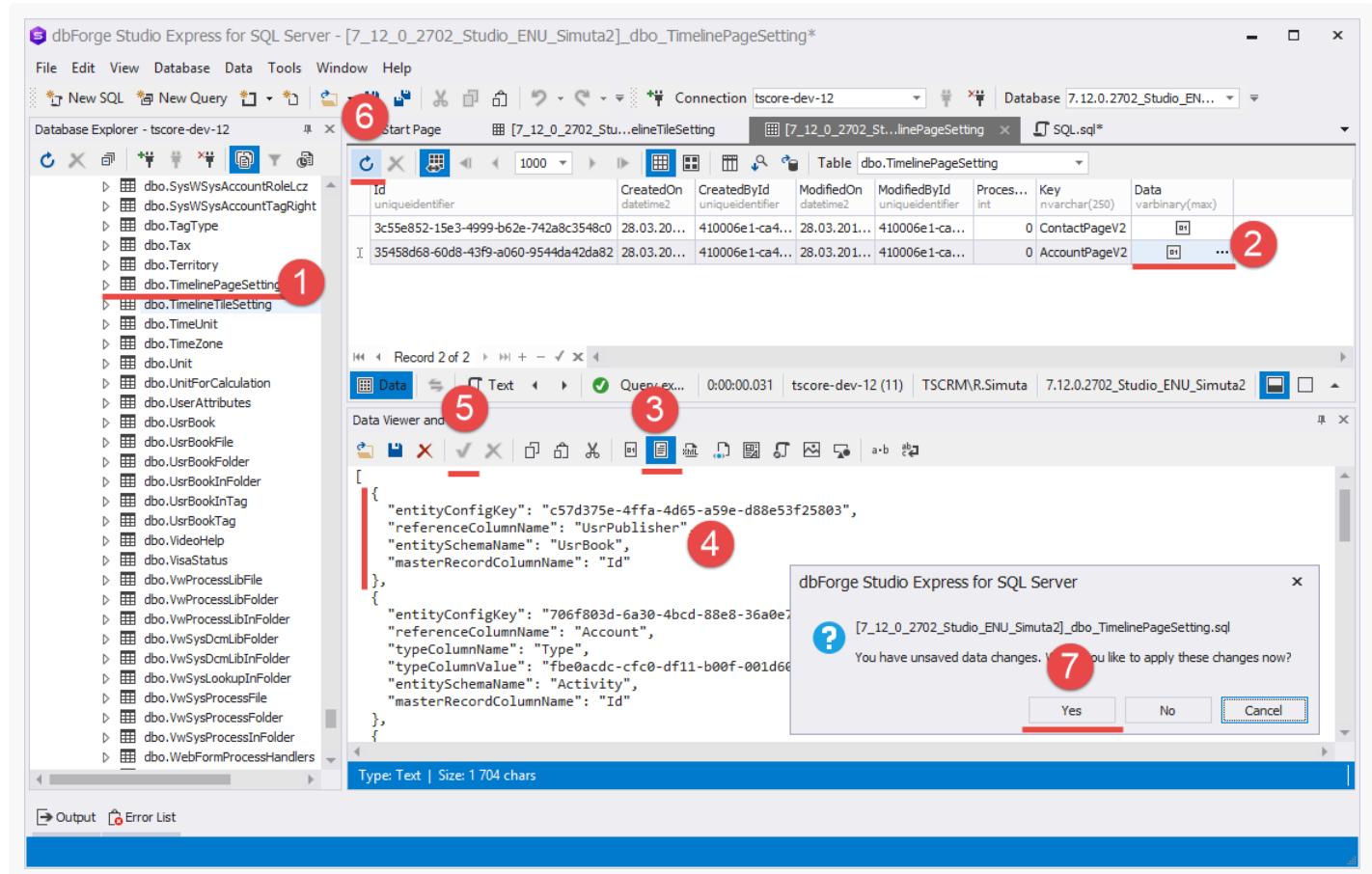
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 52 52" enable-background="new 0 0 52 52"
<path d="M46.072,31.384c-0.011-0.026-0.025-0.048-0.039-0.073c-0.036-0.064-0.077-0.125-0.123-0
c-0.018-0.022-0.034-0.044-0.053-0.064c-0.034-0.036-0.068-0.07-0.105-0.104c-0.062-0.055-0.
c-1.958-1.307-7.465-4.978-9.424-6.284c-0.388-0.258-0.703-0.845-0.703-1.312V3.938c0-0.401-
c-0.322-0.239-0.739-0.311-1.122-0.193L15.015,8.254c-0.446,0.136-1.154,0.097-1.583-0.0861-
c-0.428-0.184-0.414-0.442,0.031-0.578115.213-4.646c0.668-0.204,1.045-0.911,0.841-1.58s-0.
C7.454,5.982,7.429,5.994,7.403,6.005C7.338,6.031,7.276,6.062,7.217,6.097C7.205,6.104,7.19
c-0.015,0.01-0.026,0.025-0.041,0.035C7.081,6.191,7.03,6.236,6.982,6.284c-0.02,0.021-0.041
C6.864,6.412,6.813,6.485,6.77,6.562C6.716,6.659,6.683,6.748,6.658,6.838C6.651,6.864,6.648
C6.628,6.985,6.619,7.054,6.616,7.125C6.615,7.142,6.61,7.156,6.61,7.173V29.85c0,0.466-0.03
c-0.109,0.058-0.18,0.101-0.246,0.15c-0.025,0.018-0.046,0.037-0.069,0.058c-0.056,0.049-0.1
c-0.015,0.019-0.032,0.035-0.046,0.056c-0.057,0.079-0.105,0.164-0.142,0.257c-0.006,0.015-0
c-0.029,0.077-0.049,0.158-0.062,0.241c-0.002,0.015-0.009,0.027-0.01,0.042c-0.002,0.018,0.
c-0.003,0.031-0.009,0.062-0.009,0.094V7.312c0,0.393,0.182,0.762,0.493,1.002I14.766,11.391
c0.212,0,0.424-0.053,0.616-0.16123.203-12.938c0.401-0.224,0.649-0.646,0.649-1.105v-5.766c
C46.145,31.555,46.113,31.468,46.072,31.384z M15.4,11.625c0-0.466,0.361-0.953,0.807-1.089
c0.446-0.136,0.807,0.132,0.807,0.598v14.63c0,0.467-0.314,0.635-0.702,0.3761-1.127-0.752c-
1-13.059,5.805c-0.426,0.189-0.771-0.034-0.771-0.501C15.4,25.943,15.4,11.625,15.4,11.625z
c0.425-0.189,1.085-0.134,1.473,0.125I11.43,7.62c0.388,0.259,0.368,0.644-0.045,0.861-18.40
c-0.412,0.216-1.047,0.163-1.418-0.1211-11.789-9.001c-0.371-0.283-0.326-0.665,0.1-0.854L28
c0-0.466,0.348-0.695,0.776-0.512I2.174,0.929c0.429,0.183,0.776,0.708,0.776,1.175v2.158c-1
L9.142,9.932L9.142,9.932z M9.142,13.152c0.931,0.671,2.22,1.323,3.727,1.372v7.633c-1.57-0.
C9.142,20.548,9.142,13.152,9.142,13.152z M9.142,21.627c0.931,0.671,2.22,1.323,3.727,1.372
l-2.163,0.876c-0.432,0.175-0.782-0.061-0.782-0.527V21.627z M43.666,36.101c0,0.467-0.33,1.
c-0.407,0.228-1.036,0.18-1.405-0.104L8.897,39.127c-0.369-0.284-0.668-0.893-0.668-1.358v-2
l12.764,9.748c0.225,0.171,0.496,0.26,0.768,0.26c0.201,0,0.403-0.048,0.588-0.146I19.899-10
c0.413-0.217,0.747-0.015,0.747,0.452V36.101z" style="fill:#6c91de;"/>
<path d="M33.81,34.064c0.072,0.049,0.155,0.073,0.239,0.073c0.072,0,0.145-0.018,0.209-0.05514.
c0.126-0.072,0.207-0.204,0.212-0.349c0.006-0.146-0.063-0.283-0.183-0.365l-9.011-6.192c-0.
1-5.157,2.123c-0.143,0.059-0.243,0.191-0.259,0.346c-0.017,0.154,0.053,0.304,0.181,0.392L3

```

```
18.269,5.6821-3.692,2.111-8.803-6.052L29.492,25.426z" style="fill:#6c91de;"/>
</svg>
```

5. Изменить привязку плитки

Для раздела [Контрагенты] ([Accounts]) в таблице [TimelineTileSetting] уже существует запись с настройкой плиток, связанных с другими разделами. Это запись, которая содержит значение "AccountPageV2" в колонке [Key].



Важно. Поскольку в хронологии страницы раздела [Контрагенты] ([Accounts]) используются несколько плиток, то в колонке [Data] хранится массив конфигурационных объектов, которые подключают соответствующую плитку.

Используя приведенную на шаге 4 последовательность, измените массив конфигурационных объектов, добавив в него новую запись.

Добавление нового объекта в массив [Data]

```
[  
{
```

```

"entityConfigKey": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
"referenceColumnName": "UsrPublisher",
"entitySchemaName": "UsrBook",
"masterRecordColumnName": "Id"
},
...
]

```

Значение свойства `"entityConfigKey"` — идентификатор (колонка `[Id]`) записи таблицы `[TimelineTileSettings]`, которая создана на шаге 4.

Важно. Будьте предельно осторожны при изменении значения в колонке `[Data]`. Внесение некорректных изменений может нарушить работу существующих плиток хронологии в разделе.

Результат выполнения примера

В результате выполнения примера на вкладке `[Хронология]` (`[Timeline]`) страницы контрагента отображаются плитки, которые связаны с пользовательским разделом `[Книги]` (`[Books]`). Эти плитки содержат все поля, приведенные в условиях примера.

The screenshot shows the Creatio application interface with the following details:

- Header:** "Apress" (Entity name), "What can I do for you?", "Creatio 7.18.5.1468 Debug", "VIEW".
- Left Panel (Entity Details):**
 - Name:** Apress (marked as required with an asterisk).
 - Type:** Web.
 - Owner:** Supervisor.
 - Primary phone:** (empty field).
 - Category:** (empty field).
 - Industry:** (empty field).
 - A progress bar indicates 10% completion.
 - An "Enrich data" button is present.
- Top Bar:** "NEXT STEPS (0)" with various action icons (phone, email, message, etc.).
- Timeline Section:**
 - A message bubble icon says: "You don't have any tasks yet. Press F above to add a task".
 - The tab bar includes: ACCOUNT INFO, BOOKS, MAINTENANCE, **TIMELINE** (highlighted in red), CONTACTS AND STRUCTURE, CONNECTED TO, HISTORY, ATTACH >.
 - The timeline displays two items under "December 2021":
 - Pro C# 7: With .NET and .NET Core** by Andrew Troelsen (Fr 12/3/2021 3:10 AM). Description: Dive in and discover why Pro C# has been a favorite of C# developers worldwide for over 15 years.
 - JavaScript: The Definitive Guide: Activate Your Web Pages** by David Flanagan (Fr 12/3/2021 3:10 AM). Description: Since 1996, JavaScript: The Definitive Guide has been the bible for JavaScript programmers — a programmer's guide and comprehensive reference to the core language and to the client-side JavaScript APIs defined by web browsers.

Профиль связанной сущности



Профиль связанной сущности — элемент управления, который по умолчанию представляет собой информационный блок, наполняемый при загрузке страницы записи информацией о связанной сущности. В приложении элемент используется как профиль связанной записи на странице записи раздела. Например, при открытии страницы контакта в контейнере левой части страницы записи (`LeftModulesContainer`) отображается профиль контакта и информация о связанном с ним контрагенте. Контрагент связанный с контактом по колонке [*Account*] объекта [*Contact*]. Профиль записи и профиль связанной записи подробно описаны в статье [Страницы записей](#).

Составляющие, которые реализуют функциональность профиля связанной сущности:

- Класс `Profile`.
- `BaseProfileSchema` — базовая схема для создания профиля связанной сущности. Позволяет отобразить любой набор полей по связанной сущности, а также любое количество модулей. Является родительским классом для класса `Profile`. Все схемы профилей связанных сущностей должны наследовать схему `BaseProfileSchema`.
- `BaseMultipleProfileSchema` — базовая схема для создания профиля связанной сущности, который содержит в себе несколько профилей и свободно между ними переключается, пользуясь логикой выбора значений из справочников. Основное **отличие** от базового профиля — возможность встраивать другие профили в текущий профиль. При этом встроенные профили могут взаимодействовать друг с другом посредством сообщений. Профили `BaseMultipleProfileSchema` должны наследовать базовую схему `BaseRelatedProfileSchema`, которая реализует профили, зависящие или встроенные в другие профили.

Добавить профиль связанной сущности

1. Создайте схему модели представления профиля связанной сущности. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схеме модели представления настройте **профиль связанной сущности**.
 - a. В свойство `mixins` добавьте миксин `ProfileSchemaMixin`.
 - b. В массив модификаций `diff` добавьте конфигурационный объект с настройками связанного пользовательского профиля.
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 **добавьте связанный пользовательский профиль на страницу записи**.
 - a. В свойство `modules` добавьте модуль связанного пользовательского профиля контрагента. В свойство `masterColumnName` свойства `viewModelConfig` добавьте название колонки, по которой выполняется связь связанного профиля с основной схемой страницы записи. Опираясь на значение этой колонки, класс `Profile` загружает данные.
 - b. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения связанного пользовательского профиля контрагента.

По значению колонки, которая указана в свойстве `masterColumnName`, класс `Profile` загружает данные. На

этапе инициализации объекта профиля класс `Profile`:

- Отправляет сообщение `GetColumnInfo` для получения дополнительной информации о колонке, по которой он связан (фильтры, заголовок и т. д.).
- Запрашивает значение колонки, по которой он связан.
 - Если **связанный профиль не пустой** (т. е. в поле связи выбрана запись), то по этой записи инициализируются данные.
 - Если **связанный профиль пустой** (т. е. в поле связи не выбрана запись), то на месте связанного профиля контрагента отображаются:
 - Название поля, по которому выполняется связь. Например, на странице контакта отображается связанный профиль контрагента, который связан с контактом по полю [Контрагент] ([Account]).
 - [Добавить контрагент ([New account]) — создать новую запись в справочнике поля связи.]
 - [Выбрать] ([Search]) — выбрать существующую запись из списка доступных.

Профиль контакта, у которого присутствует связанный профиль контрагента, представлен на рисунке ниже.

The screenshot shows a CRM application interface for a contact named Caleb Jones. On the left, there's a sidebar with basic contact information: Full name (Caleb Jones), Full job title (Managing Director), Mobile phone (+44 782 223 4967), Business phone (3010), and Email (c.jones@yahoo.co.uk). A red box highlights a section where the 'Account' field is populated with 'Our company'. The main content area shows the contact's details under 'CONTACT INFO': Type (Employee), Title (Mr.), Recipient's name (Jones), Age (42), Owner (John Best), Gender (Male), Preferred language, and Communication options (Business phone 3010, Mobile phone +44 782 223 4967, Extension phone 105). Below this, there's an 'Addresses' section with one entry: Home address (153 Sunshine Street, London, United Kingdom).

Профиль контакта, у которого отсутствует связанный профиль контрагента, представлен на рисунке ниже.

Full name*
Christine Nelson

Full job title
Accountant

Mobile phone
[+44 \(788\) 247 1010](#)

Business phone
[+44 \(20\) 3488 6553](#)

Email
christine@novcorp.co.uk

ACCOUNT

- New account
- Search

NEXT STEPS (2)

Call back to the client. Inform about case resolution
11/9/2021 | Megan Lewis

Specify contract terms
11/17/2021 | John Best

CONTACT INFO CONNECTED TO MAINTENANCE TIMELINE ENGAGEMENT WEBSITE EVENTS

Type	Customer	Owner	John Best
Title	Ms.	Gender	Female
Recipient's name	Nelson	Preferred language	
Age	39		

Communication options + ☎️ 📧

Mobile phone [+44 \(788\) 247 1010](#) Business phone [+44 \(20\) 3488 6553](#)

Email christine@noveltycorpo... Email christine@novcorp.co.uk

Addreses + :

Address type	Primary	Address	City	Country	ZIP/postal...
Actual	Yes	1537 Green Lane	London	United Kingdom	

При очистке поля или изменении значения в объекте профиля выполняется переинициализация данных. При выборе существующей связанной сущности на справочник накладывается бизнес-логика, которая определена в странице записи и связана с атрибутом, который ссылается на эту сущность. Т. е. сохраняется фильтрация, настройки колонок запроса и т. д. Удаление атрибута из схемы страницы записи приведет к удалению бизнес-логики.

Добавить пользовательский профиль связанной сущности на страницу записи



Пример. Добавить связанный пользовательский профиль контрагента на страницу контакта.

1. Создать схему модели представления связанного профиля контрагента

- Перейдите в раздел [Конфигурация] ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [Добавить] —> [Модель представления страницы] ([Add] —> [Page view model]).

The screenshot shows a list of objects in the Terrasoft application. The 'Multi actions' dropdown menu is open, and the 'Page view model' option is highlighted with a red box.

	Status	Type	Object	Modified on
play schema - contact card		Client module		12/9/2021, 6:01:24 AM
CountProfileSchema		Client module		12/9/2021, 6:26:24 AM

3. Заполните **свойства схемы**.

- [Код] ([Code]) — "UsrAccountProfileSchema".
- [Заголовок] ([Title]) — "Профиль контрагента" ("Account profile").
- [Родительский объект] ([Parent object]) — выберите "AccountProfileSchema".

The screenshot shows the 'Module' configuration dialog. The fields are filled as follows:

- Code: UsrAccountProfileSchema
- Title: Account profile
- Parent object: Account profile (AccountProfileSchema)
- Package: sdkBuiltInProfilePackage

The 'APPLY' button is highlighted.

4. В объявлении класса модели представления в качестве зависимостей добавьте миксин `ProfileSchemaMixin`.

5. Настройте **связанный пользовательский профиль контрагента**.

- В свойство `mixins` добавьте миксин `ProfileSchemaMixin`.
- В массив модификаций `diff` добавьте конфигурационный объект с настройками связанного пользовательского профиля контрагента.

Исходный код схемы модели представления связанного профиля контрагента представлен ниже.

```
UsrAccountProfileSchema

/* В качестве зависимостей укажите миксин ProfileSchemaMixin. */
define("UsrAccountProfileSchema", ["ProfileSchemaMixin"], function () {
```

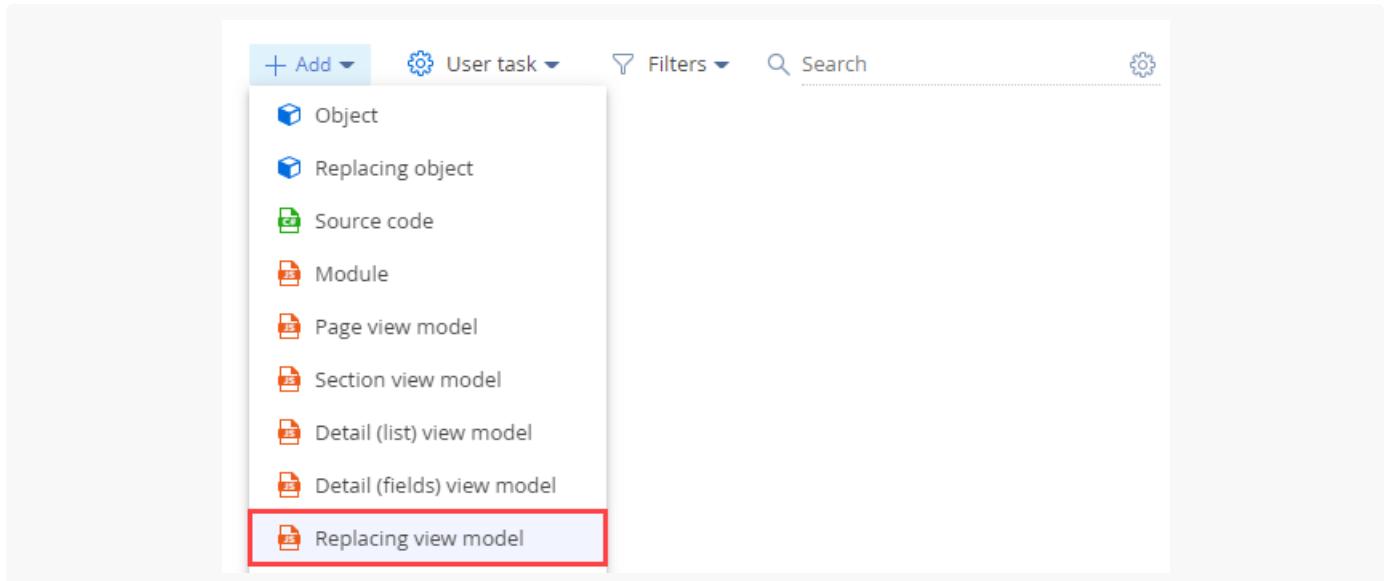
```

return {
    /* Название схемы объекта. */
    entitySchemaName: "Account",
    /* Миксины. */
    mixins: {
        /* Миксин, который содержит функции для получения иконок и картинок профиля. */
        ProfileSchemaMixin: "Terrasoft.ProfileSchemaMixin"
    },
    /* Массив модификаций diff. */
    diff: /**SCHEMA_DIFF*/[
        {
            /* Операция добавления. */
            "operation": "insert",
            /* Имя сущности. */
            "name": "Contact",
            /* Имя родительского элемента, в который выполняется вставка. */
            "parentName": "ProfileContentContainer",
            /* Свойство элемента родителя, с которым выполняется операция. */
            "propertyName": "items",
            /* Значение добавляемого элемента. */
            "values": {
                /* Привязка к значению свойства Account объекта Contact. */
                "bindTo": "Account",
                /* Конфигурация разметки. Позиционирование элемента. */
                "layout": {
                    "column": 3,
                    "row": 10,
                    "colSpan": 19
                }
            }
        }
    ],
    /* Другие конфигурационные объекты массива модификаций. */
    ]/**SCHEMA_DIFF*/
];
});
);
});
```

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

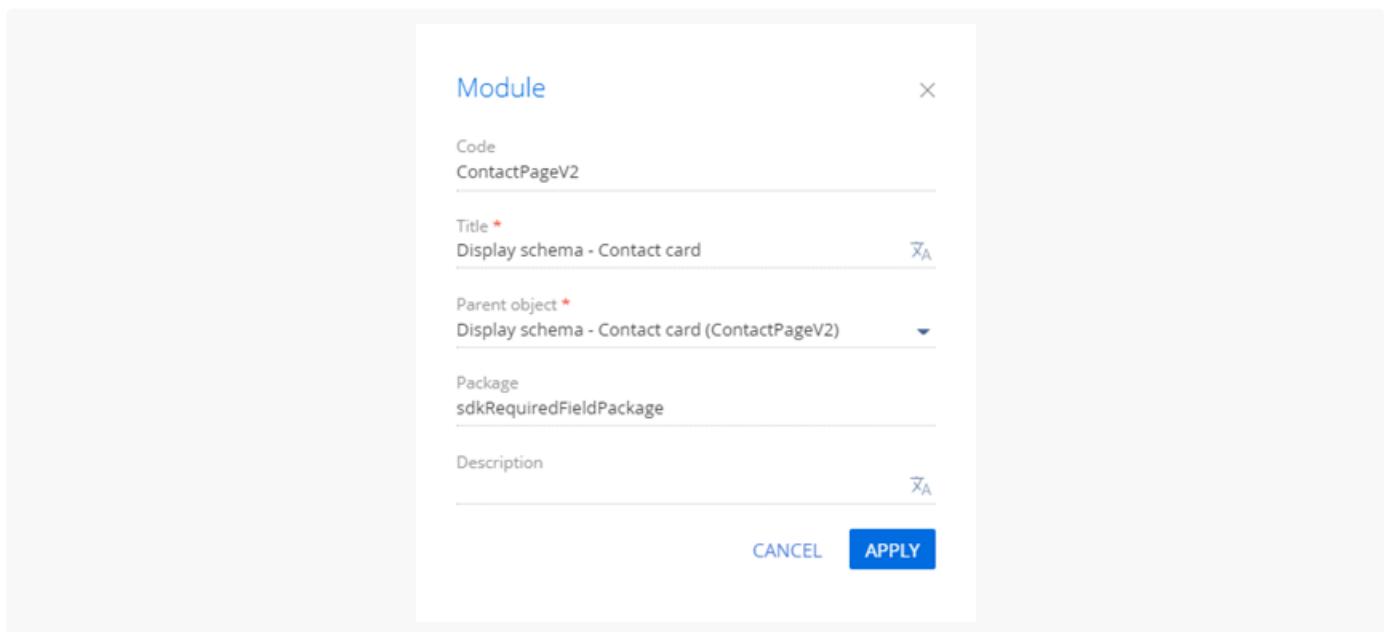
2. Создать схему замещающей модели представления страницы контакта

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

`BaseFiltersGenerateModule`, `BusinessRuleModule`, `ContactPageV2Resources`, `ConfigurationConstants`,
`ContactCareer`, `DuplicatesSearchUtilitiesV2`, `UsrAccountProfileSchema`.

5. Добавьте **связанный пользовательский профиль контрагента на страницу контакта**.

- В свойство `modules` добавьте модуль связанного пользовательского профиля контрагента.
- В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения связанного пользовательского профиля контрагента.

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

ContactPageV2

```
/* Определение схемы страницы записи и ее зависимостей. */
define("ContactPageV2", ["BaseFiltersGenerateModule", "BusinessRuleModule", "ContactPageV2Res
    return {
        entitySchemaName: "Contact",
        /* Модули. */
        modules: /**SCHEMA_MODULES*{
            /* Модуль профиля контрагента. */
            "AccountProfile1": {
                /* Конфигурация профиля. */
                "config": {
                    /* Название схемы. */
                    "schemaName": "UsrAccountProfileSchema",
                    /* Признак, который сообщает об инициализации конфигурации схемы. */
                    "isSchemaConfigInitialized": true,
                    /* Признак, который сообщает, что не используется HistoryState. */
                    "useHistoryState": false,
                    /* Параметры профиля. */
                    "parameters": {
                        /* Конфигурация модели представления. */
                        "viewModelConfig": {
                            /* Название колонки связанной сущности. */
                            masterColumnName: "Account"
                        }
                    }
                }
            }
        }
    }
}/**SCHEMA_MODULES*/,
/* Массив модификаций. */
diff: /**SCHEMA_DIFF*/[
    {
        /* Операция добавления. */
        "operation": "insert",
        /* Имя родительского элемента, в который выполняется вставка. */
        "parentName": "LeftModulesContainer",
        /* Свойство элемента родителя, с которым выполняется операция. */
        "propertyName": "items",
        /* Имя сущности. */
        "name": "AccountProfile1",
        /* Значение добавляемого элемента. */
        "values": {
            /* Тип элемента – модуль. */
            "itemType": Terrasoft.ViewItemType.MODULE
        }
    }
]
```

```
    ]/**SCHEMA_DIFF*/
};

});
```

- 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 страницу контакта добавлен связанный пользовательский профиль контрагента.

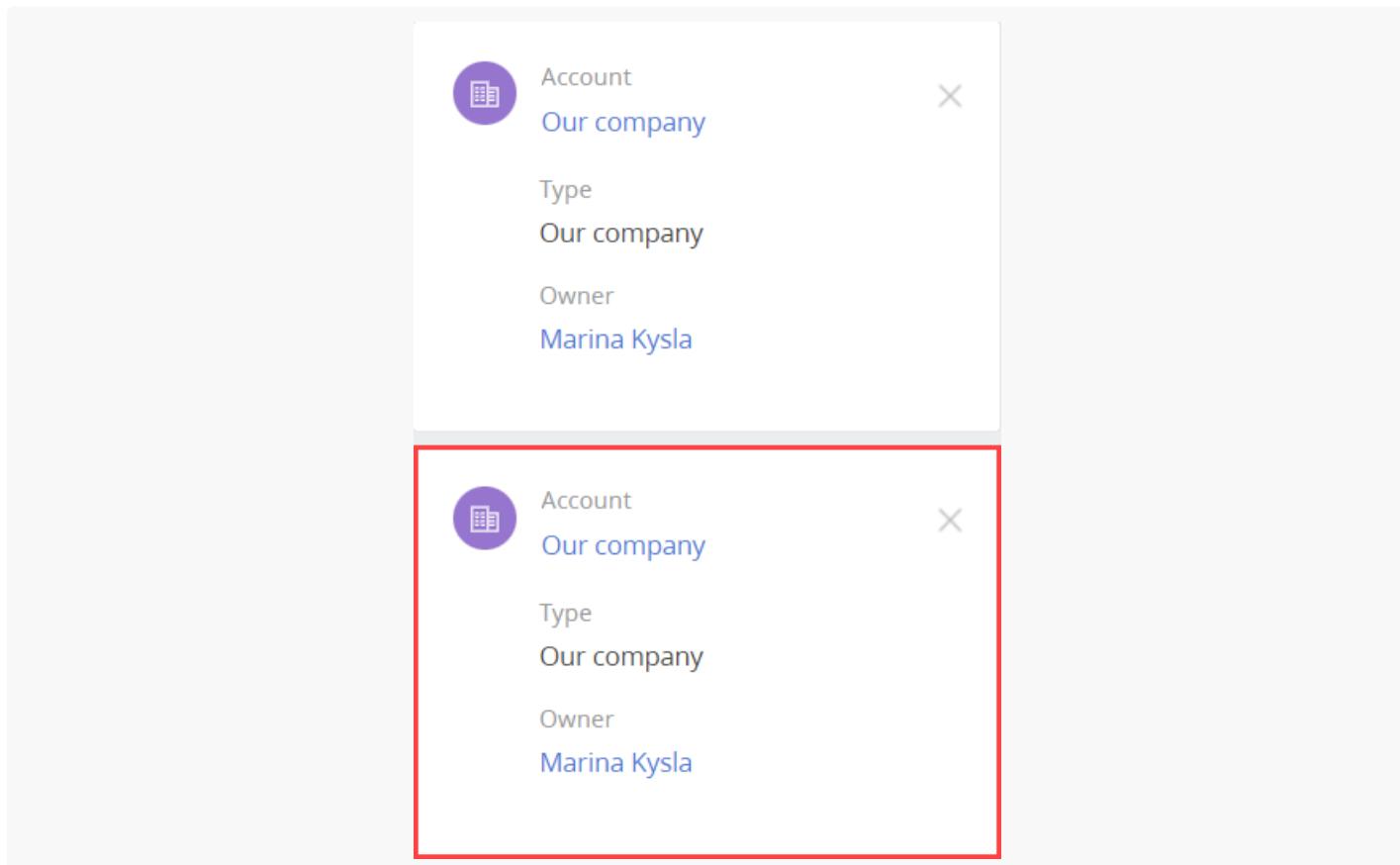


Схема BaseProfileSchema js

Средний

`BaseProfileSchema` — базовая схема для создания профиля связанной сущности. Позволяет отобразить любой набор полей по связанной сущности, а также любое количество модулей. Реализована в пакете `NUI`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Все схемы профилей связанных сущностей должны наследовать схему `BaseProfileSchema`.

Атрибуты

`MasterColumnValue` `GUID`

Значение главной колонки.

`MasterColumnInfo` `CUSTOM_OBJECT`

Информация о главной колонке.

`DataItemMarkerTpl` `TEXT`

Шаблон маркера элемента данных.

Сообщения

Сообщения базовой схемы

Название	Режим	Направление	Описание
<code>Entity Initialized</code>	Широковещательное	Подписка	Событие инициализации основной сущности.
<code>GetEntityColumn Changes</code>	Широковещательное	Подписка	Обрабатывает изменения колонки сущности.
<code>GetColumnsValues</code>	Адресное	Публикация	Возвращает запрошенные значения колонки.
<code>GetLookupQuery Filters</code>	Адресное	Публикация	Получает справочник фильтров запроса.
<code>GetColumnInfo</code>	Адресное	Публикация	Возвращает информацию колонки.
<code>UpdateCard Property</code>	Адресное	Публикация	Изменяет значение карточки модели.
<code>OpenCard</code>	Адресное	Публикация	Открывает карточку.
<code>CardModule Response</code>	Адресное	Подписка	Ответное сообщение карточки модуля.

Режимы сообщений представлены перечислением `Terrasoft.core.enums.MessageMode`, а направления сообщений — перечислением `Terrasoft.core.enums.MessageDirectionType`. Перечисление `MessageMode` описано в [Библиотеке JS классов](#). Перечисление `MessageDirectionType` описано в [Библиотеке JS классов](#).

Методы

`getMiniPageConfig(options)`

Возвращает конфигурацию мини-карточки.

Параметры

<code>{Object} options</code>	Свойства мини-карточки.
-------------------------------	-------------------------

`getLookupConfig(config)`

Возвращает справочник конфигурации открытого справочника.

Параметры

<code>{Object} config</code>	Конфигурация открытой справочной страницы.
------------------------------	--

`getVisibleBlankSlate()`

Возвращает тег видимости контейнера `BlankSlateContainer`.

`getVisibleContent()`

Возвращает тег видимости контейнера `ProfileContentContainer`.

`getClearButtonHint()`

Возвращает всплывающую подсказку к кнопке [*Очистить*] ([*Clear*]).

`getUpdateCardPropertyConfig(response)`

Получает свойства конфигурации обновления карточки при ее сохранении.

Параметры

<code>{Object} response</code>	Ответ сервера.
--------------------------------	----------------

Схема BaseMultipleProfileSchema



Средний

`BaseMultipleProfileSchema` — базовая схема для создания профиля связанной сущности, который содержит в себе несколько профилей и свободно между ними переключается, пользуясь логикой выбора значений из справочников. Реализована в пакете `NUI`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Профили `BaseMultipleProfileSchema` должны наследовать базовую схему `BaseRelatedProfileSchema`, которая реализует профили, зависимые или встроенные в другие профили.

Атрибуты

`EditColumnName` STRING

Название колонки свойства карточки.

`MasterColumnNames` CUSTOM_OBJECT

Массив имен основных колонок модулей.

Сообщения

Сообщения базовой схемы

Название	Режим	Направление	Описание
<code>ProfileEntityColumnChanges</code>	Широковещательное	Публикация	Обрабатывает изменения колонки сущности профиля.
<code>GetProfileEntityColumnChanges</code>	Адресное	Подписка	Возвращает запрошенные значения колонки.
<code>ProfileOpenCard</code>	Адресное	Подписка	Отправляет запрос открытой карточки с конфигурацией.

Режимы сообщений представлены перечислением `Terrasoft.core.enums.MessageMode`, а направления сообщений — перечислением `Terrasoft.core.enums.MessageDirectionType`. Перечисление `MessageMode` описано в [Библиотеке JS классов](#). Перечисление `MessageDirectionType` описано в [Библиотеке JS классов](#).

Схема BaseRelatedProfileSchema



`BaseRelatedProfileSchema` — базовая схема, которая реализует профили, зависимые или встроенные в другие профили. Реализована в пакете `NUI`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Является родительской схемой для базовой

схемы `BaseMultipleProfileSchema` создания профиля связанной сущности, который содержит в себе несколько профилей и свободно между ними переключается, пользуясь логикой выбора значений из справочников.

Сообщения

Сообщения базовой схемы

Название	Режим	Направление	Описание
<code>ProfileEntityColumnChanges</code>	Широковещательное	Подписка	Обрабатывает изменения колонки сущности профиля.
<code>GetProfileEntityColumnChanges</code>	Адресное	Публикация	Отправляет запрошенные значения колонки.
<code>ProfileOpenCard</code>	Адресное	Публикация	Отправляет запрос открытой карточки с конфигурацией.

Режимы сообщений представлены перечислением `Terrasoft.core.enums.MessageMode`, а направления сообщений — перечислением `Terrasoft.core.enums.MessageDirectionType`. Перечисление `MessageMode` описано в [Библиотеке JS классов](#). Перечисление `MessageDirectionType` описано в [Библиотеке JS классов](#).

HTML-элемент iframe



Средний

HTML-элемент iframe — элемент интерфейса, который используется для отображения сторонней веб-страницы внутри страницы, в которой он размещен.

Назначение элемента `iframe` — внедрение стороннего веб-приложения в `Creatio` для обеспечения удобного просмотра сторонних веб-ресурсов (страниц, видео и т. п.) непосредственно из `Creatio`.

В HTML-коде страницы элемент `iframe` реализуется с помощью тегов `<iframe>`. URL отображаемой страницы устанавливается с помощью атрибута `src`.

Важно. Необходимо помнить, что не все сайты разрешают загрузку своих страниц в элемент `iframe`.

Для реализации элемента в front-end ядре `Creatio` реализован компонент `Terrasoft.controls.IframeControl`. Компонент `Terrasoft.controls.IframeControl` описан в [Библиотеке JS классов](#).

Назначение компонента `Terrasoft.controls.IframeControl` — отображение пользовательской HTML-разметки в `Creatio`.

Пример использования компонента `Terrasoft.controls.IframeControl` — страница шаблонов Email-

сообщений справочника [Шаблоны email-сообщений] ([*Email message templates*]).

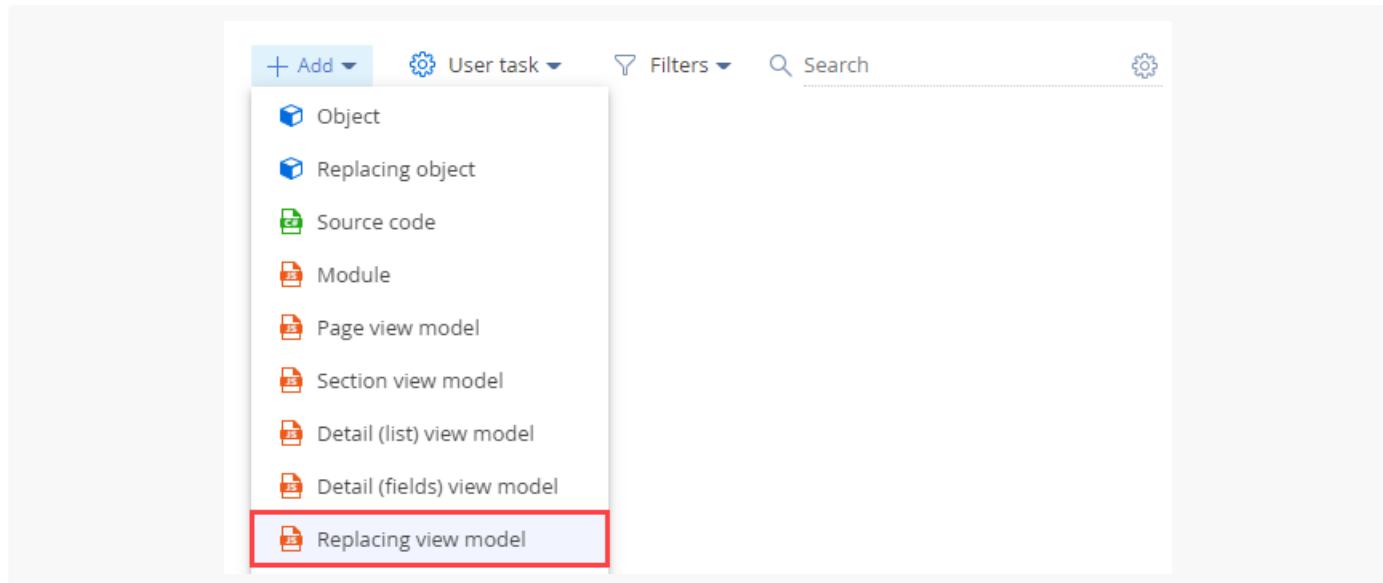
Добавить HTML-элемент iframe

 Средний

Пример. На странице записи в разделе [Контрагенты] ([*Accounts*]) создать вкладку [WEB], на которой отображается сайт, указанный в поле [Web].

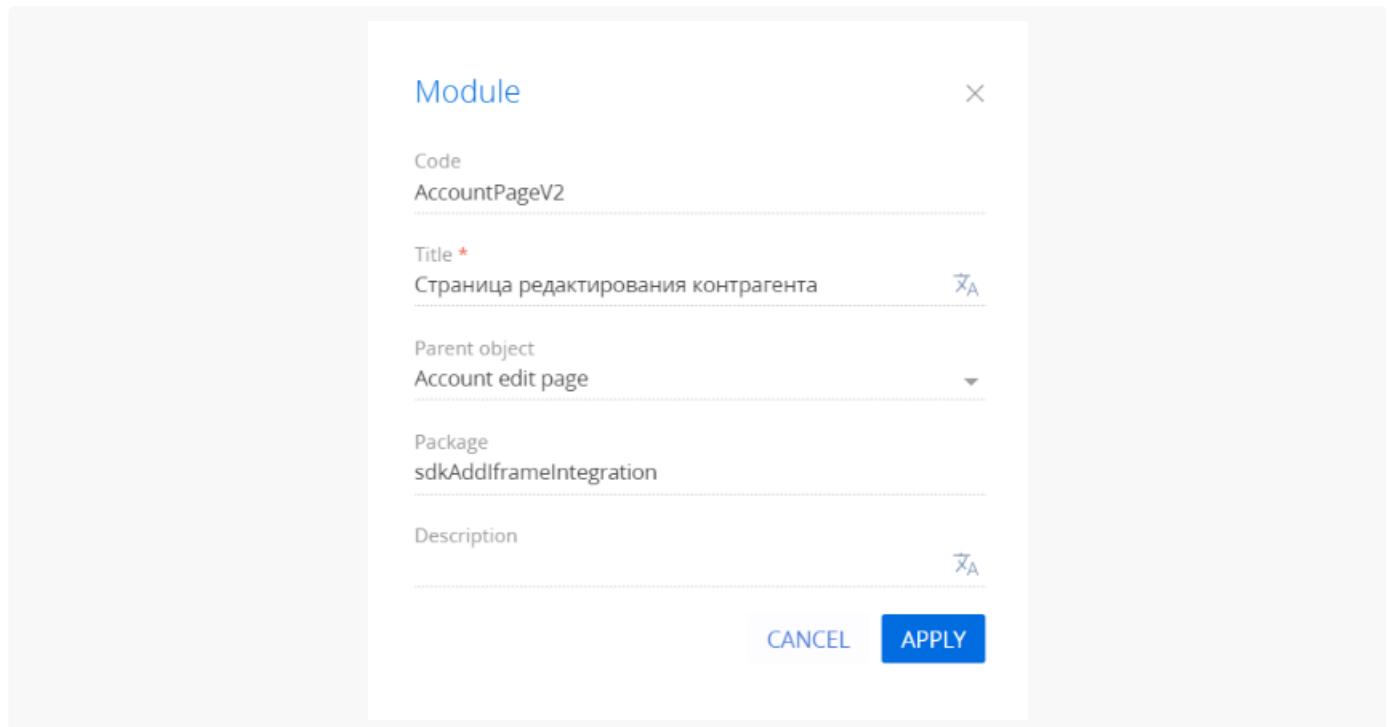
Создать схему замещающей модели представления страницы записи

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 "Account edit page".



4. Реализуйте **добавление HTML-компонента iframe**.

a. В массив модификаций `diff` добавьте **конфигурационные объекты**:

- `WebTab` — вкладка WEB.
- `UsrIframe` — компонент для отображения `Terrasoft.controls.IframeControl`.

d. В свойстве `methods` реализуйте метод `getSource()` для привязки данных колонки `Web` к свойству `src` компонента.

Исходный код схемы замещающей модели представления раздела представлен ниже.

AccountPageV2

```
define("AccountPageV2", [], function() {
    return {
        entitySchemaName: "Account",
        diff: /**SCHEMA_DIFF*/[
            /* Добавление вкладки [WEB].*/
            {
                "operation": "insert",
                "name": "WebTab",
                "values": {
                    "caption": "WEB",
                    "items": []
                },
                "parentName": "Tabs",
                "propertyName": "tabs",
                "index": 1
            }
        ]
    }
});
```

```

},
/* Добавление компонента IFrameControl.*/
{
    "operation": "insert",
    "name": "UsrIframe",
    "parentName": "WebTab",
    "propertyName": "items",
    "values": {
        "itemType": Terrasoft.ViewItemType.IFRAMECONTROL,
        "src": {
            "bindTo": "getSource"
        }
    }
}
]/**SCHEMA_DIFF*/,
methods: {
    /* Используется для привязки данных.*/
    getSource: function() {
        return this.get("Web");
    }
}
);
});

```

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

Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

- Очистите кэш браузера.
- Обновите страницу записи раздела [Контрагенты] ([Accounts]).

В результате выполнения примера на странице записи раздела отобразится вкладка [WEB], на которой отображается содержимое веб-страницы, URL которой задан в поле [Web]. Если поле [Web] не содержит значения, то отображается пустая вкладка.

Our company

CLOSE ACTIONS ▾

NEXT STEPS (0)

Enrich data

75%

Name* Our company

Type Our company

Owner

Web <https://www.makeup.com/>

Primary phone

Category

Industry

What can I do for you? >

Creatio 7.18.5.1468 Debug

VIEW ▾

NEXT STEPS (0)

You don't have any tasks yet
Press above to add a task

ACCOUNT INFO WEB MAINTENANCE TIMELINE BOOKS CONTACTS AND STRUCTURE CONNECTED TO HISTORY >




LIPS

Try This Viral TikTok Hack for Perfectly Lined Lips Every Time