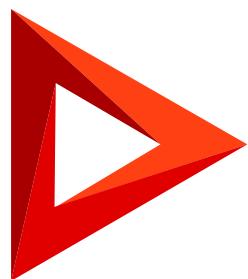


# Элементы интерфейса

Версия 8.0



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

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

# Содержание

<b>API для работы с чатами</b>	13
Интеграция с мессенджером	13
Прием сообщений	13
Сервисы для работы с чатами в Creatio	14
Добавить новый провайдер канала	14
<b>Добавить новый провайдер канала</b>	14
1. Добавить новый провайдер в Creatio	15
2. Реализовать хранение данных нового канала	15
3. Зарегистрировать новый канал в базе данных	15
4. Создать веб-сервис для приема сообщений	16
5. Реализовать конвертацию входящего сообщения в универсальный формат Creatio	19
6. Реализовать получение данных профиля пользователя	21
7. Реализовать загрузку вложений	24
8. Реализовать отправку сообщений	26
9. Выполнить связывание интерфейсов	28
<b>Добавить механизм маршрутизации чатов</b>	30
1. Добавить новое правило маршрутизации чатов	31
2. Создать класс, который реализует интерфейс IOperatorRoutingRule	32
3. Связать интерфейс и код правила из справочника	35
4. Перезапустите приложение в IIS	37
<b>Механизм Feature Toggle</b>	38
Хранить сведения о функциональности	38
Подключить дополнительную функциональность	39
Реализовать пользовательскую функциональность	42
<b>Класс FeatureUtilities</b>	45
Методы	45
Перечисление FeatureState	48
<b>Страница записи</b>	48
Контейнеры страницы записи	49
Создать страницу записи	50
Настроить страницу записи	51
Добавить пользовательское действие на страницу записи	52
<b>Добавить действие на страницу записи</b>	53
1. Создать схему замещающей модели представления страницы заказа	53
2. Создать схему замещающей модели представления раздела	57
Результат выполнения примера	59

<b>Схема BasePageV2</b>	60
Сообщения	60
Атрибуты	62
Методы	64
<b>Схема BaseEntityPage</b>	67
Сообщения	67
Атрибуты	67
Миксины	68
Методы	68
<b>Главное меню</b>	70
Способы вызова и структура главного меню	70
Контейнеры главного меню	71
<b>Интеграция с каналами чатов</b>	72
Интеграция с каналом Facebook Messenger	72
Интеграция с каналом WhatsApp	73
Интеграция с каналом Telegram	74
<b>Панель действий</b>	75
Добавить панель действий на страницу	76
Добавить новый канал на панель действий	76
Добавить пользовательское действие верификации	76
<b>Добавить панель действий</b>	78
Описание примера	78
Исходный код	78
Алгоритм реализации примера	78
<b>Добавить новый канал на панель действий</b>	82
Описание примера	82
Исходный код	82
Алгоритм выполнения примера	82
<b>Добавить мультиязычные шаблоны email-сообщений</b>	89
Описание примера	90
Исходный код	90
Предварительные настройки	90
Алгоритм реализации примера	91
<b>Создать пользовательскую страницу действия верификации</b>	96
1. Создать схему страницы действия верификации	97
2. Настроить представление страницы	98
2. Использовать созданную схему в бизнес-процессе	98
Результат выполнения примера	99
<b>Раздел</b>	99

Контейнеры раздела	100
Структура раздела	101
<b>Создать раздел</b>	110
Добавить пользовательскую колонку в реестр раздела	111
Настроить условия отображения реестра раздела	112
Добавить действие для записей раздела	113
Настроить блок быстрых фильтров раздела	114
Удалить раздел	115
<b>Добавить в раздел действие для единичной записи</b>	116
Создать схему замещающей модели представления раздела	116
Результат выполнения примера	119
<b>Добавить в раздел действие для нескольких записей</b>	121
Создать схему замещающей модели представления раздела	121
Результат выполнения примера	125
<b>Настроить обработку действия для нескольких записей раздела</b>	126
Создать схему замещающей модели представления раздела	126
Результат выполнения примера	130
<b>Настроить обработку действия с использованием справочника для нескольких записей раздела</b>	131
Создать схему замещающей модели представления раздела	132
Результат выполнения примера	135
<b>Настроить условия отображения реестра раздела</b>	137
Создать схему замещающей модели представления раздела	137
Результат выполнения примера	139
<b>Настроить блок быстрых фильтров раздела</b>	140
Создать схему замещающей модели представления раздела	140
Результат выполнения примера	144
<b>Схема BaseSectionV2</b>	144
Сообщения	144
Атрибуты	145
Методы	146
<b>Деактивация записей объектов</b>	146
Использование в дизайнере объекта	146
Использование в программном коде	147
<b>Поле</b>	148
Типы полей и операций с полями	148
Добавить поле	148
Реализовать валидацию поля	153
Установить для поля значение по умолчанию	153
Настроить обязательность для поля	155

Настроить фильтрацию значений справочного поля	155
Настроить условия блокировки поля	157
Настроить исключения блокировки поля	159
Настроить условия отображения поля	160
Добавить автонумерацию к полю	161
Добавить информационную кнопку к полю	162
Добавить всплывающую подсказку к полю	162
Вычислить разницу дат в полях	162
<b>Добавить поле на страницу записи с использованием новой колонки</b>	163
1. Создать схему замещающего объекта	163
2. Создать схему замещающей модели представления страницы активности	164
Результат выполнения примера	167
<b>Добавить поле на страницу записи с использованием существующей колонки</b>	167
Создать схему замещающей модели представления страницы контакта	168
Результат выполнения примера	169
<b>Добавить поле с изображением на страницу записи</b>	170
1. Создать схему замещающего объекта	170
2. Создать схему замещающей модели представления страницы статьи базы знаний	172
Результат выполнения примера	178
<b>Добавить вычисляемое поле на страницу записи</b>	178
1. Создать схему замещающего объекта	178
2. Создать схему замещающей модели представления страницы заказа	180
Результат выполнения примера	184
<b>Добавить мультивалютное поле на страницу записи</b>	184
1. Создать схему замещающего объекта	184
2. Создать схему замещающей модели представления страницы проекта	187
Результат выполнения примера	192
<b>Реализовать валидацию поля типа [Дата/Время] на странице записи</b>	193
Создать схему замещающей модели представления страницы продажи	193
Результат выполнения примера	196
<b>Реализовать валидацию поля типа [Строка] на странице записи</b>	197
Создать схему замещающей модели представления страницы контакта	197
Результат выполнения примера	200
<b>Установить значение по умолчанию для поля на странице записи</b>	200
Создать схему замещающей модели представления страницы проекта	201
Результат выполнения примера	203
<b>Настроить обязательность для поля на странице записи</b>	203
Создать схему замещающей модели представления страницы контакта	204
Результат выполнения примера	206

<b>Настроить фильтрацию значений справочного поля на странице записи</b>	207
Создать схему замещающей модели представления страницы контрагента	207
Результат выполнения примера	210
<b>Настроить фильтрацию значений связанных справочных полей на странице записи</b>	211
Создать схему замещающей модели представления страницы контакта	211
Результат выполнения примера	215
<b>Настроить условия блокировки поля на странице записи</b>	220
Создать схему замещающей модели представления страницы контакта	220
Результат выполнения примера	223
<b>Настроить исключения блокировки полей на странице записи</b>	223
Создать схему замещающей модели представления страницы счета	223
Результат выполнения примера	226
<b>Настроить условия отображения поля на странице записи</b>	227
1. Создать схему замещающего объекта	227
2. Создать схему замещающей модели представления страницы активности	229
Результат выполнения примера	233
<b>Добавить автонумерацию к полю на странице добавления записи (front-end)</b>	234
1. Создать системные настройки	234
2. Создать схему замещающей модели представления страницы продукта	235
Результат выполнения примера	237
<b>Добавить автонумерацию к полю на странице добавления записи (back-end)</b>	238
1. Создать системные настройки	238
2. Создать схему замещающего объекта	239
Результат выполнения примера	246
<b>Добавить информационную кнопку к полю на странице записи</b>	246
Создать схему замещающей модели представления страницы контакта	247
Результат выполнения примера	250
<b>Добавить всплывающую подсказку к полю на странице записи</b>	250
Создать схему замещающей модели представления страницы контакта	250
Результат выполнения примера	253
<b>Кнопка</b>	253
Контейнеры кнопок	254
Добавить кнопку	256
Добавить к кнопке всплывающую подсказку	257
<b>Добавить кнопку на панель инструментов раздела</b>	258
Создать схему замещающей модели представления раздела	259
Результат выполнения примера	262
<b>Добавить кнопку на панель инструментов страницы записи в совмещенном режиме</b>	263
Создать схему замещающей модели представления раздела	264

Результат выполнения примера	268
<b>Добавить кнопку на панель инструментов страницы добавления записи</b>	270
1. Изменить способ добавления контрагента	270
2. Создать схему замещающей модели представления страницы контрагента	270
Результат выполнения примера	273
<b>Добавить кнопку выбора цвета на страницу записи</b>	275
1. Создать схему замещающего объекта	275
2. Создать схему замещающей модели представления страницы контрагента	277
Результат выполнения примера	279
<b>Добавить к кнопке всплывающую подсказку</b>	280
Создать схему замещающей модели представления страницы контакта	280
Результат выполнения примера	284
<b>Добавить кнопку в строку записи раздела</b>	284
Создать схему замещающей модели представления раздела	284
Результат выполнения примера	288
<b>Свойство diff объекта кнопки</b>	288
Свойства	288
<b>Деталь</b>	291
Структура и типы деталей	291
Реализовать деталь	291
Реализовать множественное добавление записей на деталь	312
Удалить деталь	313
<b>Настроить деталь с полями</b>	313
1. Создать пользовательскую деталь	314
2. Настроить пользовательскую деталь	317
3. Добавить деталь на страницу записи раздела	319
4. Добавить пользовательские стили детали	320
5. Добавить валидацию к полю детали	323
6. Сделать виртуальными поля детали	325
<b>Добавить редактируемый реестр в деталь</b>	326
1. Создать схему замещающего объекта	326
2. Создать схему замещающей модели представления раздела	328
Результат выполнения примера	330
<b>Скрыть пункты меню детали с реестром</b>	331
Создать схему замещающей модели представления реестра детали	331
Результат выполнения примера	333
<b>Реализовать множественное добавление записей на деталь</b>	333
1. Создать схему замещающей модели представления детали	334
2. Реализуйте бизнес-логику детали	335

Результат выполнения примера	337
<b>Реализовать деталь типа [Файлы и ссылки]</b>	338
1. Создать пользовательский раздел	339
2. Создать пользовательскую деталь	340
3. Настроить пользовательскую деталь	341
4. Добавить деталь в раздел	342
5. Добавить пользовательские стили детали	343
Результат выполнения примера	347
<b>Схема BaseDetailV2</b>	347
Сообщения	348
Атрибуты	348
Методы	350
Массив модификаций	351
<b>Схема BaseGridDetailV2</b>	352
Сообщения	352
Миксины	353
Атрибуты	353
Методы	355
Массив модификаций	358
<b>Миксин GridUtilitiesV2</b>	359
Методы	360
<b>Модуль ConfigurationGrid</b>	362
Методы	363
<b>Модуль ConfigurationGridGenerator</b>	364
Методы	364
<b>Модуль ConfigurationGridUtilities</b>	364
Свойства	365
Методы	365
<b>Схема BasePageV2</b>	367
Сообщения	367
Атрибуты	369
Методы	370
<b>Схема BaseFieldsDetail</b>	373
Сообщения	373
<b>Схема FileDetailV2</b>	374
Атрибуты	374
Методы	374
<b>Мини-карточка</b>	375
Схема модели представления мини-карточки	375

Операции с мини-карточками	376
<b>Создать пользовательскую мини-карточку</b>	377
1. Создать схему модели представления мини-карточки	377
2. Отобразить поля основного объекта	378
3. Добавить функциональную кнопку в мини-карточку	380
4. Выполнить стилизацию мини-карточки	383
5. Зарегистрировать мини-карточку в базе данных	387
6. Добавить системную настройку	388
Результат выполнения примера	389
<b>Создать мини-карточку добавления</b>	389
1. Создать схему модели представления мини-карточки	389
2. Отобразить поля основного объекта	390
3. Зарегистрировать мини-карточку в базе данных	392
4. Добавить системную настройку	393
Результат выполнения примера	393
<b>Добавить мини-карточку к произвольному модулю</b>	394
1. Создать схему модуля	395
2. Создать представление и модель представления модуля	396
3. Добавить стили модуля	398
4. Создать контейнер отображения представления	398
Результат выполнения примера	401
<b>Хронология</b>	401
Таблицы базы данных	401
Добавление вкладки Хронология в раздел	404
<b>Добавить базовую хронологию в раздел</b>	405
1. Создать SQL-сценарий	405
2. Установить данные SQL-сценария	407
Результат выполнения примера	408
<b>Создать хронологию, связанную с пользовательским разделом</b>	408
1. Создать раздел Книги (Books))	409
2. Создать модуль представления плитки	410
3. Создать модуль модели представления плитки	413
4. Настроить отображение плитки	414
5. Изменить привязку плитки	417
Результат выполнения примера	418
<b>Модальное окно</b>	418
<b>Реализовать модальное окно</b>	419
1. Создать действие процесса	420
2. Добавить параметры действия процесса	421

3. Реализовать действие процесса на back-end стороне	424
4. Реализовать обработку действия процесса на front-end стороне	426
5. Создать замещающую модель представления контейнера	429
6. Создать бизнес-процесс отображения модального окна	430
Результат выполнения примера	436
<b>Профиль связанной сущности</b>	437
Добавить профиль связанной сущности	438
<b>Добавить пользовательский профиль связанной сущности на страницу записи</b>	440
1. Создать схему модели представления связанного профиля контрагента	440
2. Создать схему замещающей модели представления страницы контакта	442
Результат выполнения примера	445
<b>Схема BaseProfileSchema</b>	445
Атрибуты	446
Сообщения	446
Методы	447
<b>Схема BaseMultipleProfileSchema</b>	447
Атрибуты	448
Сообщения	448
<b>Схема BaseRelatedProfileSchema</b>	448
Сообщения	449
<b>Коммуникационная панель</b>	449
Структура коммуникационной панели	449
Контейнеры коммуникационной панели	450
<b>Создать обращение по сообщению из внутренней ленты другого обращения</b>	451
Создать схему замещающей модели представления страницы обращения	451
Результат выполнения примера	454
<b>Скрыть область ленты в едином окне</b>	454
Создать схему замещающей модели представления раздела	455
Результат выполнения примера	457
<b>Отобразить часовой пояс контакта на вкладку коммуникационной панели</b>	457
1. Создать схему замещающей модели представления страницы найденного абонента	457
2. Добавить стили отображения часового пояса	461
Результат выполнения примера	465
<b>HTML-элемент iframe</b>	465
<b>Добавить HTML-элемент iframe</b>	466
Создать схему замещающей модели представления страницы записи	466
Результат выполнения примера	469
<b>Командная строка</b>	469
Контейнер командной строки	469

Выполнить команду из командной строки	470
<b>Дашборды</b>	<b>472</b>
Структура данных для хранения информации по итогам	472
Реализация функциональности в режиме просмотра итогов	473
Реализация функциональности в режиме настройки итогов	474
Базовые классы, реализующие функциональность дашборда	475
Виды дашбордов	476
<b>Изменить расчеты в воронке продаж</b>	<b>482</b>
Последовательность действий для изменения расчетов в воронке:	483
Пример изменения отображения расчетов в воронке продаж в срезе "по количеству"	483
<b>Подключить дополнительную фильтрацию к воронке</b>	<b>487</b>
Описание примера	488
Исходный код	488
Алгоритм реализации примера	488
<b>Добавить пользовательский дашборд</b>	<b>490</b>
1. Создать модуль показателя валюты	491
2. Добавить дашборд на панель итогов и указать его параметры	495
Результат выполнения примера	497
<b>Схема BaseWidgetDesigner</b>	<b>497</b>
Методы	498
<b>Перечисление DashboardEnums</b>	<b>498</b>
Свойства	498

# API для работы с чатами



Средний

Базовая функциональность работы с чатами находится в предустановленном пакете `OmnichannelMessaging`.

Описание настройки интеграции с мессенджерами содержится в блоке статей [Настройка чатов](#).

## Интеграция с мессенджером

**Библиотеки**, в которых находится основная часть функциональности интеграции с мессенджером:

- `OmnichannelMessaging.dll`.
- `OmnichannelProviders.dll`.

Возможности интеграции с мессенджером предоставляет класс `MessageManager`.

Основные **методы** класса `MessageManager`:

- `Receive` — принимает входящие сообщения. Для сообщения используется универсальный формат (`UnifiedMessage` — унифицированный класс сообщения) или строка, которая в дальнейшем будет конвертирована в универсальный формат. При необходимости использования конвертации метод использует класс, который реализует интерфейс `IIIncomeMessageWorker`. Для каждого мессенджера используется собственная реализация данного интерфейса. Также выполняется сохранение сообщения в системе.
- `Send` — отправляет исходящие сообщения. Система подбирает подходящий класс-отправитель сообщения. Класс должен реализовывать интерфейс `IOutcomeMessageWorker`. Также выполняется сохранение сообщения в системе.
- `Register` — добавляет канал при настройке интеграции с Facebook Messenger. Метод использует класс, который реализует интерфейс `IMessengerRegistrationWorker`. Используется для мессенджеров, которые требуют выполнения дополнительных действий при регистрации, например, получение токена.
- `GetMessagesByChatId` — получает сообщения, которые относятся к чату. Для сообщения используется формат `IEnumerable<UnifiedMessage>`. Чат передается в параметрах метода.

## Прием сообщений

Специфичные для мессенджеров **сервисы** приема сообщений:

- `FacebookOmnichannelMessagingService` — сервис приема сообщений от Facebook Messenger.
- `TelegramOmnichannelMessagingService` — сервис приема сообщений от Telegram.

Для обоих сервисов базовым является сервис `OmnichannelMessagingService`, который содержит набор общих для мессенджеров методов.

`InternalReceive` — основной **метод** сервиса `OmnichannelMessagingService`. Принимает сообщения.

## Сервисы для работы с чатами в Creatio

**Сервисы** для работы с чатами в Creatio:

- `OmnichannelChatService` — сервис для работы с чатами, который позволяет получить историю, чаты оператора и т. д.
- `OmnichannelOperatorService` — сервис операторов, который позволяет получить и изменить статус.

Основные **методы** сервиса `OmnichannelChatService`:

- `AcceptChat` — закрепляет чат за текущим пользователем.
- `GetUnreadChatsCount` — получает количество непрочитанных чатов.
- `MarkChatAsRead` — помечает все сообщения в указанном чате как прочитанные.
- `GetConversation` — получает сообщения чата для отображения в коммуникационной панели.
- `CloseActiveChat` — закрывает указанный чат.
- `GetChats` — получает все чаты для указанного оператора.
- `GetChatActions` — получает список действий, доступных для очереди указанного чата.
- `GetUnreadMessagesCount` — получает количество непрочитанных сообщений во всех чатах оператора.

## Добавить новый провайдер канала

1. Добавьте новый провайдер в справочник [ *Провайдер канала* ] ([ *Channel provider* ]).
2. Реализуйте хранение данных нового канала в базе данных.
3. Зарегистрируйте новый канал в базе данных.
4. Создайте веб-сервис для приема сообщений.
5. Реализуйте конвертацию входящего сообщения в универсальный формат Creatio.
6. Реализуйте получение данных профиля пользователя в классе, который реализует интерфейс `IProfileDataProvider`.
7. Реализуйте загрузку вложений.
8. Реализуйте отправку сообщений в классе, который реализует интерфейс `IOutcomeMessageWorker`.
9. Выполните связывание интерфейсов.

## Добавить новый провайдер канала



Сложный

**Пример.** В on-site приложении Creatio создать новый провайдер чата `Test`. В качестве зависимости пользовательского [пакета](#) указать пакет `OmnichannelMessaging`.

## 1. Добавить новый провайдер в Creatio

- Перейдите в дизайнер системы по кнопке .
- В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Справочники] ([Lookups]).
- Используя фильтр в верхней части страницы, найдите справочник [Провайдер канала] ([Channel provider]).
- Откройте наполнение справочника и добавьте значение `Test`.

## 2. Реализовать хранение данных нового канала

В базе данных приложения создайте таблицу `[TestMsgSettings]` (имя таблицы следует формировать по правилу `[<ИмяПровайдера>MsgSettings]`).

Структура таблицы зависит от конкретного мессенджера и должна содержать данные, необходимые системе для выполнения отправки и приема сообщений. Как правило мессенджеры для отправки используют авторизационный токен.

### Пример скрипта, который создает таблицу `[TestMsgSettings]`

MSSQL

```
IF OBJECT_ID('TestMsgSettings') IS NULL
BEGIN
CREATE TABLE TestMsgSettings(
    Id uniqueidentifier NOT NULL DEFAULT (newid()),
    Token nvarchar(250) NOT NULL DEFAULT (''),
    UserName nvarchar(250) NOT NULL DEFAULT (''),
    CONSTRAINT PK_TestMsgSettings_Id PRIMARY KEY CLUSTERED (Id)
)
END
```

Postgres

```
CREATE TABLE IF NOT EXISTS public."TestMsgSettings" (
    "Id" uuid NOT NULL DEFAULT uuid_generate_v4(),
    "Token" character varying(250) COLLATE pg_catalog."default" NOT NULL DEFAULT ''::character v
    "UserName" character varying(250) COLLATE pg_catalog."default" NOT NULL DEFAULT ''::characte
    CONSTRAINT "PK_TestMsgSettings_Id" PRIMARY KEY ("Id")
);
```

## 3. Зарегистрировать новый канал в базе данных

Для регистрации нового канала добавьте запись в таблицу [Channel].

Описание полей таблицы [Channel]

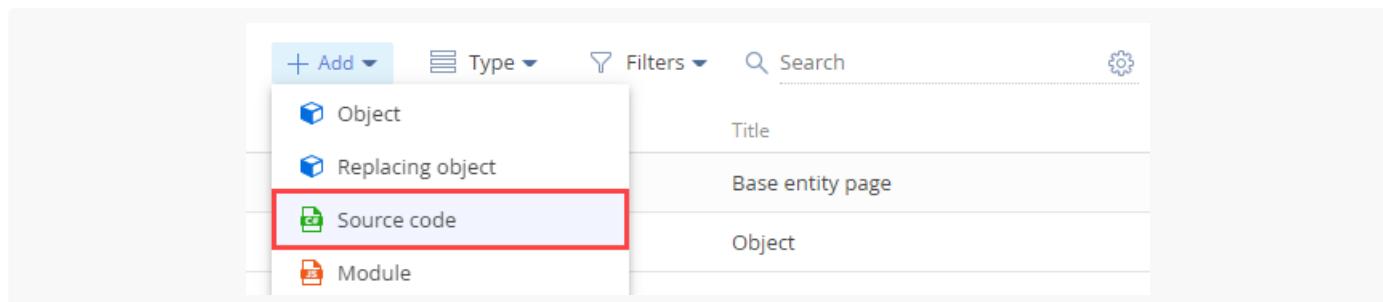
Поле	Описание
[Name]	Название канала.
[ProviderId]	Идентификатор добавленного провайдера.
[MsgSettingsId]	Идентификатор записи в таблице [TestMsgSettings].
[Source]	Идентификатор канала внутри мессенджера, например идентификатор страницы на Facebook или идентификатор клиента в Telegram. Позволяет по сообщению от мессенджера определить получателя.

## 4. Создать веб-сервис для приема сообщений

В примере реализуем вариант, при котором мессенджер присыпает входящие сообщения на указанный endpoint. Созданный веб-сервис должен быть анонимным и доступным извне. Наследовать веб-сервис следует от базового веб-сервиса `OmnichannelMessagingService`, поскольку он содержит набор методов, позволяющих сохранить сообщение, создать чат, контакт и т. д.

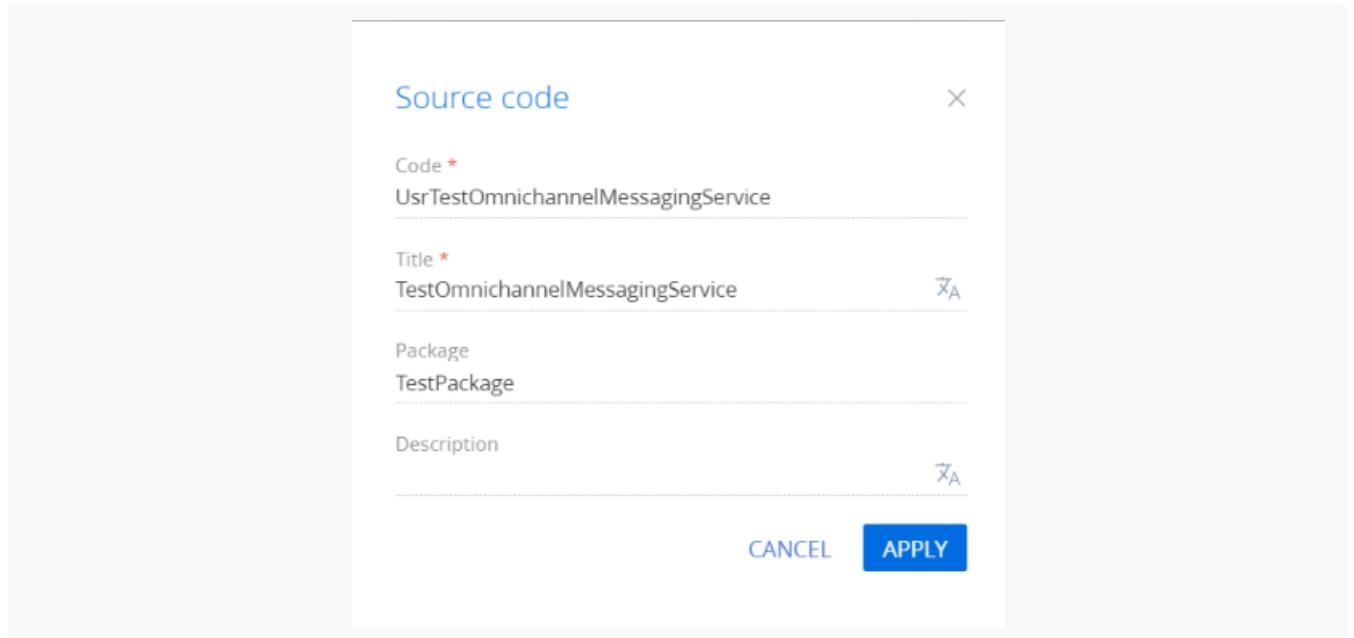
Чтобы **создать веб-сервис**:

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



3. В дизайнере схем заполните **свойства схемы**:

- [ Код ] ([ Code ]) — "UsrTestOmnichannelMessagingService".
- [ Заголовок ] ([ Title ]) — "TestOmnichannelMessagingService".



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

4. В дизайнере схем добавьте исходный код.

```
TestOmnichannelMessagingService

namespace Terrasoft.Configuration.Omnichannel.Messaging
{
    using System;
    using System.ServiceModel;
    using System.ServiceModel.Activation;
    using System.ServiceModel.Web;
    using Terrasoft.Common;
    using Terrasoft.Core;
    using Terrasoft.Core.DB;
    using Terrasoft.Web.Common.ServiceRouting;

    #region Class: TestOmnichannelMessagingService

    /// <summary>
    /// The service that sends and receives messages from the messaging integration API.
    /// </summary>
    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.R
    [DefaultServiceRoute]
    public class TestOmnichannelMessagingService : OmnichannelMessagingService
    {

        #region Constructors: Public

        public TestOmnichannelMessagingService() : base() {
```

```

}

/// <summary>
/// Initialize a new instance of <see cref="TestOmnichannelMessagingService"/>.
/// </summary>
public TestOmnichannelMessagingService(UserConnection userConnection) : base(userConn
}

#endregion

#region Methods: Private

private void GetChannelAndQueueBySource(MessagingMessage message) {
    Select channelSelect = new Select(UserConnection)
        .Top(1).Column("Id")
        .Column("ChatQueueId")
        .From("Channel")
        .Where("Source").IsEqual(Column.Parameter(message.Recipient))
        .And("IsActive").IsEqual(Column.Parameter(true)) as Select;
    channelSelect.ExecuteReader(reader => {
        message.ChannelId = reader.GetColumnValue<Guid>("Id").ToString();
        message.ChannelQueueId = reader.GetColumnValue<Guid>("ChatQueueId");
    });
}

#endregion

#region Methods: Public

/// <summary>
/// Receive messages from the integration API.
/// </summary>
/// <param name="message">Test provider message.</param>
[OperationContract]
[WebInvoke(UriTemplate = "receive", Method = "POST", RequestFormat = WebMessageFormat
    ResponseFormat = WebMessageFormat.Json)]
public void ReceiveMessage(TestIncomingMessage message) {
    MessagingMessage messagingMessage = new MessagingMessage(TestIncomingMessageConve
        GetChannelAndQueueBySource(messagingMessage);
        InternalReceive(messagingMessage);
}

#endregion

}

#endregion

}

```

В примере мессенджер присыпает входящее сообщение на endpoint `receive`.

В методе `ReceiveMessage` выполняется:

- Конвертация сообщения (метод `Convert` класса `TestIncomingMessageConverter`, который будет создан на следующем шаге).
- Идентификация канала по полю `Source` (метод `GetChannelAndQueueBySource`).
- Вызов метода `InternalReceive`, который выполнит следующие действия:
  - Создаст новый чат, если открытый чат с этим клиентом не будет найден, или добавит сообщение к существующему.
  - Создаст контакт, если это первое общение с клиентом, и выполнит заполнение его контактных данных, или привяжет чат к существующему контакту.

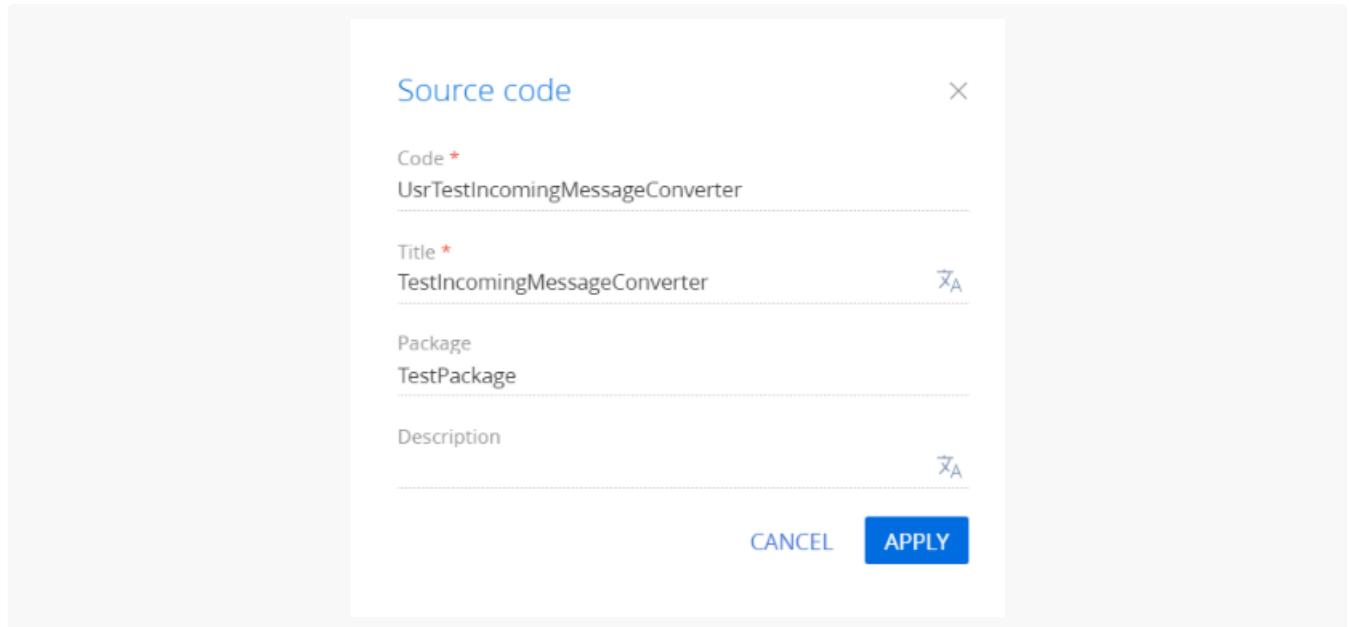
5. Сохраните и опубликуйте схему.

## 5. Реализовать конвертацию входящего сообщения в универсальный формат Creatio

После приема сообщения от мессенджера необходимо выполнить конвертацию входящего сообщения в универсальный формат Creatio (класс `MessagingMessage`). В примере эта задача выполняется классом `TestIncomingMessageConverter`, который конвертирует входящее сообщение с типом `TestIncomingMessage` (сообщение в формате мессенджера) в формат Creatio.

Чтобы **создать класс-конвертер**:

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский **пакет**, в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Исходный код ] ([ Add ] —> [ Source code ]).
3. В дизайнере схем заполните **свойства схемы**:
  - [ Код ] ([ Code ]) — "UsrTestIncomingMessageConverter".
  - [ Заголовок ] ([ Title ]) — "TestIncomingMessageConverter".



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

4. В дизайнере схем добавьте исходный код.

```
TestIncomingMessageConverter

namespace Terrasoft.Configuration.Omnichannel.Messaging
{
    using OmnichannelProviders.Domain.Entities;
    using System;
    using System.Collections.Generic;

    public class TestIncomingMessage : UnifiedMessage
    {
    }

    public static class TestIncomingMessageConverter
    {
        #region Methods: Public

        public static MessagingMessage Convert(TestIncomingMessage message) {
            var messageType = MessageType.Text;
            var messageId = Guid.NewGuid();
            var result = new MessagingMessage {
                Id = messageId,
                Message = message.Message,
                /* Получатель сообщения – идентификатор страницы, либо клиента, добавленный в */
                Recipient = message.Recipient,
                /* Идентификатор отправителя сообщения внутри мессенджера. Будет связан с кон */
                Sender = message.Sender,
                Timestamp = message.Timestamp,
                /* Идентификатор канала внутри системы, поле MsgSettingsid из объекта. */
            };
        }
    }
}
```

```

        ChannelId = message.ChannelId,
        MessageDirection = MessageDirection.Incoming,
        MessageType = messageType,
        /* Указывает на источник канала (сторонние разработчики). */
        Source = ChannelType.ThirdParty,
        /* Имя провайдера. В дальнейшем будет использоваться как идентификатор провайдера */
        ChannelName = "Test"
    };
    if (messageType != MessageType.Text) {
        result.Attachments = new List<MessageAttachment>();
        foreach (var attachment in message.Attachments){
            result.Attachments.Add(new MessageAttachment {
                MessageId = messageId,
                UploadUrl = attachment.UploadUrl,
                FileType = attachment.FileType
            });
        }
    }
    return result;
}

#endregion

}
}

```

Блок `Attachments` заполняется в зависимости от формата доступа к файлам, который предоставляет мессенджер. В примере это ссылка для загрузки. Возможен вариант с передачей в сообщении `FileId` для последующего скачивания файла.

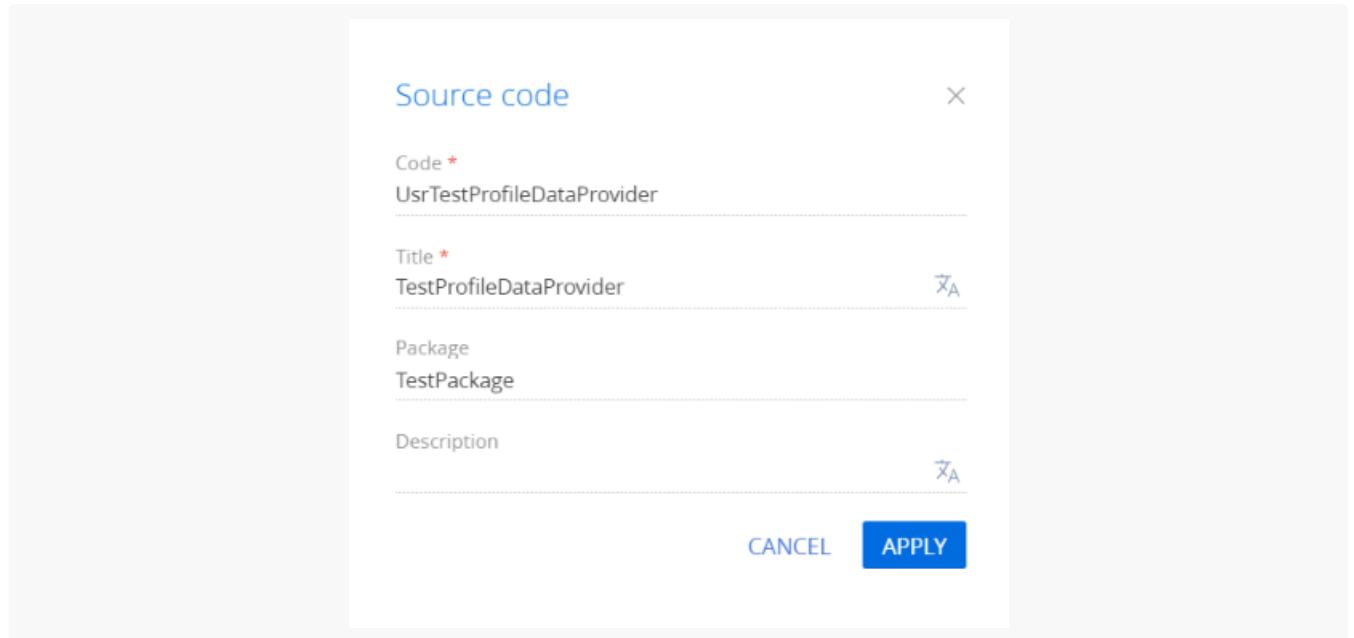
- Сохраните и опубликуйте схему.

## 6. Реализовать получение данных профиля пользователя

Мессенджеры предоставляют API для получения данных клиентов, которые отправляют сообщения. Чтобы получить эти данные необходимо создать класс, реализующий интерфейс `IProfileDataProvider`.

Чтобы **создать класс** для получения данных профиля пользователя:

- [Перейдите в раздел \[ Конфигурация \]](#) ([ *Configuration* ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [ *Добавить* ] —> [ *Исходный код* ] ([ *Add* ] —> [ *Source code* ]).
- В дизайнере схем заполните **свойства схемы**:
  - [ *Код* ] ([ *Code* ]) — "UsrTestProfileDataProvider".
  - [ *Заголовок* ] ([ *Title* ]) — "TestProfileDataProvider".



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

4. В дизайнере схем добавьте исходный код.

```

TestProfileDataProvider

namespace Terrasoft.Configuration.Omnichannel.Messaging
{
    using Newtonsoft.Json;
    using OmnichannelProviders.Domain.Entities;
    using OmnichannelProviders.Interfaces;
    using System.IO;
    using System.Net;
    using Terrasoft.Core;

    public class TestProfileData {
        public string first_name { get; set; }
        public string last_name { get; set; }
    }

    #region Class: TestProfileDataProvider

    /// <summary>
    /// Retrieve the profile data from the Test provider.
    /// </summary>
    public class TestProfileDataProvider : IProfileDataProvider
    {
        #region Properties: Private

        private readonly string _testProviderApiUrl = "https://graph.test.com/";
    }
}

```

```

#endregion

#region Constructors: Public

/// <summary>
/// Initialize a new instance of <see cref="FacebookProfileDataProvider"/>
/// </summary>
/// <param name="userConnection">User connection.</param>
public TestProfileDataProvider(UserConnection userConnection) {
}

#endregion

#region Methods: Public

/// <summary>
/// Retrieve the profile data from Facebook.
/// <param name="profileId">The Facebook profile ID.</param>
/// <param name="channelId">The channel from which to send the request.</param>
/// <returns>The contact ID.</returns>
/// </summary>
public ProfileData GetProfileDataByProfileId(string profileId, string channelId) {
    var requestUrl = string.Concat(_testProviderApiUrl, profileId);
    WebRequest request = WebRequest.Create(requestUrl);
    try {
        using (var response = request.GetResponse()) {
            using (Stream stream = response.GetResponseStream()) {
                using (StreamReader sr = new StreamReader(stream)) {
                    var testProfile = JsonConvert.DeserializeObject<TestProfileData>(
                        sr.ReadToEnd());
                    return new ProfileData {
                        FirstName = testProfile.first_name,
                        LastName = testProfile.last_name,
                    };
                }
            }
        }
    }
    catch {
        return new ProfileData();
    }
}

#endregion

}

#endregion
}

```

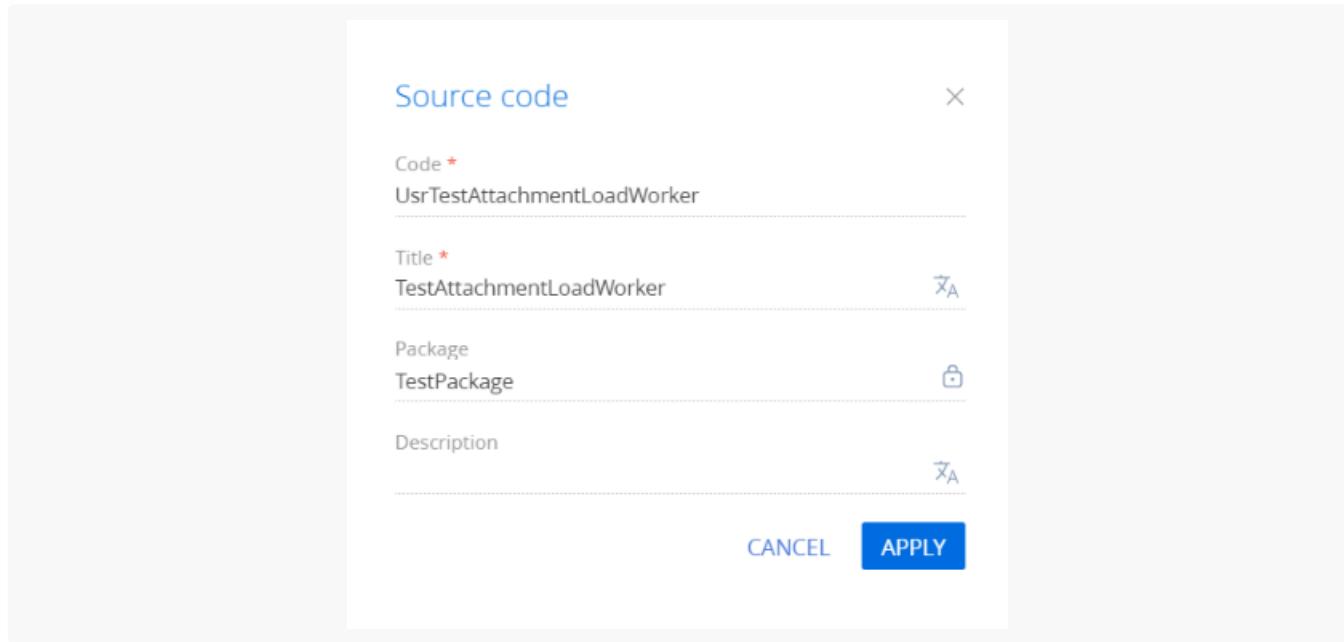
В примере класс отправляет запрос на URL `https://graph.test.com/` для получения данных в формате `TestProfileData` (предполагаемый формат мессенджера) и конвертирует полученный ответ во внутренний формат `ProfileData`. При создании контакта к нему будут добавлены полученные данные (например, имя, фамилия, фото). Если запрос не будет успешным или данные будут некорректными, то контакт будет создан с именем [ <Новый контакт><Имя канала>-<идентификатор клиента в мессенджере> ].

- Сохраните и опубликуйте схему.

## 7. Реализовать загрузку вложений

Для загрузки вложений создайте класс, реализующий интерфейс `IAttachmentsLoadWorker`:

- [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Исходный код ] ([ Add ] —> [ Source code ]).
- В дизайнере схем заполните **свойства схемы**:
  - [ Код ] ([ Code ]) — "UsrTestAttachmentLoadWorker".
  - [ Заголовок ] ([ Title ]) — "TestAttachmentLoadWorker".



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

- В дизайнере схем добавьте исходный код.



```
{
    using OmnichannelProviders;
    using OmnichannelProviders.Domain.Entities;
    using OmnichannelProviders.MessageConverters;
    using Terrasoft.Core;

    #region Class: TestAttachmentLoadWorker

    /// <summary>
    /// The class that loads attachments from the Test provider.
    /// </summary>
    public class TestAttachmentLoadWorker : IAttachmentsLoadWorker
    {

        #region Properties: Protected

        protected UserConnection UserConnection;
        protected AttachmentsDownloader AttachmentsDownloader;

        #endregion

        #region Constructors: Public

        /// <summary>
        /// Initialize a new instance of the <see cref="TestAttachmentLoadWorker" /> class.
        /// <param name="userConnection">Instance of the <see cref="UserConnection"/> class.</param>
        /// </summary>
        public TestAttachmentLoadWorker(UserConnection userConnection) {
            UserConnection = userConnection;
            AttachmentsDownloader = new AttachmentsDownloader(userConnection);
        }

        #endregion

        #region Methods: Public

        /// <summary>
        /// Load the attachments.
        /// </summary>
        /// <param name="incomeAttachment">The attachment from the messenger.</param>
        /// <param name="message">The source message.</param>
        public void Load(MessageAttachment incomeAttachment, UnifiedMessage message) {
            incomeAttachment.FileName = FileUtilities.GetFileNameFromUrl(incomeAttachment.UpL
            incomeAttachment.FileId = AttachmentsDownloader.Load(incomeAttachment);
        }

        #endregion
    }
}
```

```
#endregion
}
```

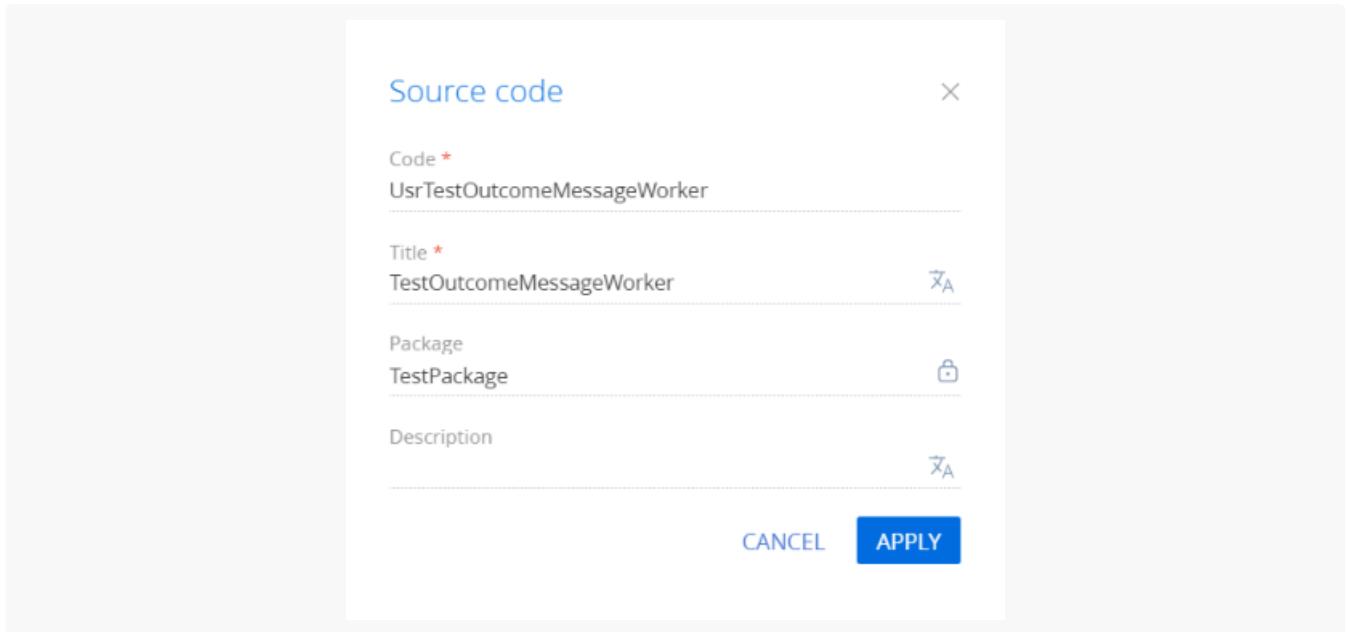
Для загрузки вложений используется внутренний класс `AttachmentsDownloader`, который выполняет загрузку и сохранение файла по ссылке. Сохранение файла происходит в таблицу `[OmnichannelMessageFile]`.

- Сохраните и опубликуйте схему.

## 8. Реализовать отправку сообщений

Для отправки сообщений **создайте класс**, реализующий интерфейс `IOutcomeMessageWorker`:

- Перейдите в раздел [ Конфигурация ] ([ Configuration ]) и выберите пользовательский пакет, в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Исходный код ] ([ Add ] —> [ Source code ]).
- В дизайнере схем заполните **свойства схемы**:
  - [ Код ] ([ Code ]) — "UsrTestOutcomeMessageWorker".
  - [ Заголовок ] ([ Title ]) — "TestOutcomeMessageWorker".



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

- В дизайнере схем добавьте исходный код.

```
TestOutcomeMessageWorker

namespace Terrasoft.Configuration.Omnichannel.Messaging
{
```

```

using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;
using OmnichannelProviders.Application.Http;
using OmnichannelProviders.Domain.Entities;
using OmnichannelProviders.MessageWorkers;
using Terrasoft.Core;

#region Class: TestOutcomeMessageWorker

/// <summary>
/// The class that sends messages to the Test provider.
/// </summary>
public class TestOutcomeMessageWorker : IOutcomeMessageWorker
{
    #region Properties: Protected

    protected UserConnection UserConnection;
    private readonly string _testProviderApiUrl = "https://graph.test.com/";
    #endregion

    #region Constructors: Public

    /// <summary>
    /// Initialize a new instance of the <see cref="TestOutcomeMessageWorker" /> class.
    /// <param name="userConnection">Instance of the <see cref="UserConnection"/> class.</param>
    /// </summary>
    public TestOutcomeMessageWorker(UserConnection userConnection) {
        UserConnection = userConnection;
    }

    #endregion

    #region Methods: Public

    /// <summary>
    /// Send the message to Test provider.
    /// </summary>
    /// <param name="message">The UnifiedMessage message.</param>
    public string SendMessage(UnifiedMessage unifiedMessage) {
        var serializerSettings = new JsonSerializerSettings();
        serializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
        var json = JsonConvert.SerializeObject(unifiedMessage, serializerSettings);
        var requestUrl = string.Concat(_testProviderApiUrl, json);
        var result = new HttpRequestSender().PostAsync(requestUrl, json).Result;
        return result;
    }
}

```

```

    #endregion
}

#endregion
}

```

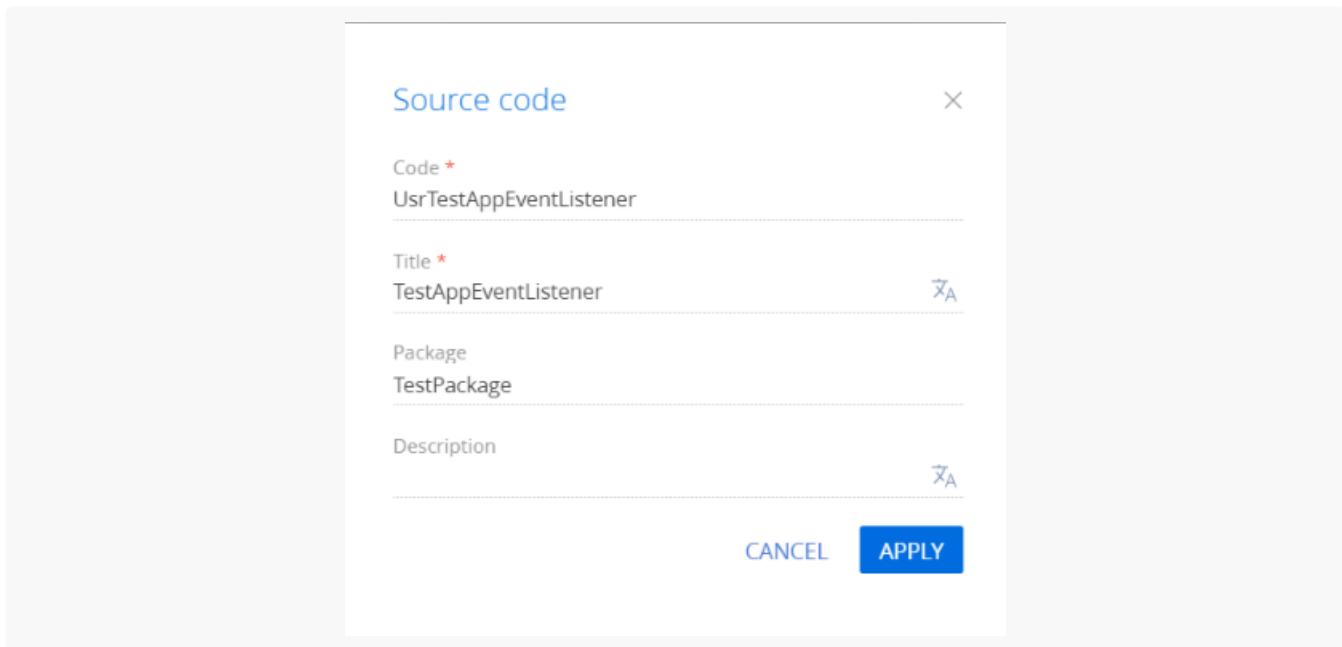
Класс `TestOutcomeMessageWorker` переводит сообщение в формат мессенджера и выполняет его отправку, используя API мессенджера. Для выполнения отправки может понадобится токен, который следует хранить в таблице `[TestMsgSettings]`. Доступ к таблице можно получить через переданный в конструкторе `UserConnection`. В примере выполняется отправка сообщения на URL `https://graph.test.com/`, используя внутренний класс `HttpRequestSender`.

- Сохраните и опубликуйте схему.

## 9. Выполнить связывание интерфейсов

Чтобы выполнить связывание интерфейсов **создайте класс** наследник `AppEventListenerBase`:

- Перейдите в раздел [ Конфигурация ] ([ *Configuration* ]) и выберите пользовательский пакет, в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Исходный код ] ([ *Add* ] —> [ *Source code* ]).
- В дизайнере схем заполните **свойства схемы**:
  - [ Код ] ([ *Code* ]) — "UsrTestAppEventListener".
  - [ Заголовок ] ([ *Title* ]) — "TestAppEventListener".



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

- В дизайнере схем добавьте исходный код.

**TestAppEventListener**

```

namespace Terrasoft.Configuration.Omnichannel.Messaging
{
    using Common;
    using Core;
    using OmnichannelProviders;
    using OmnichannelProviders.Interfaces;
    using OmnichannelProviders.MessageWorkers;
    using Terrasoft.Core.Factories;
    using Web.Common;

    #region Class : TestAppEventListener

    /// <summary>
    /// The class that runs prerequisites for OmnichannelMessaging on application start.
    /// </summary>
    public class TestAppEventListener : AppEventListenerBase
    {

        #region Fields : Protected

        protected UserConnection UserConnection {
            get;
            private set;
        }

        #endregion

        #region Methods : Protected

        /// <summary>
        /// Retrieve the user connection from the application event scope.
        /// </summary>
        /// <param name="context">The application event scope.</param>
        /// <returns>User connection.</returns>
        protected UserConnection GetUserConnection(AppEventArgs context) {
            var appConnection = context.Application["AppConnection"] as AppConnection;
            if (appConnection == null) {
                throw new ArgumentNullException("AppConnection");
            }
            return appConnection.SystemUserConnection;
        }

        protected void BindInterfaces() {
            ClassFactory.Bind<IAttachmentsLoadWorker, TestAttachmentLoadWorker>("Test");
            ClassFactory.Bind<IProfileDataProvider, TestProfileDataProvider>("Test");
            ClassFactory.Bind<IOutcomeMessageWorker, TestOutcomeMessageWorker>("Test");
        }
    }
}

```

```

}

#endregion

#region Methods : Public

/// <summary>
/// Handle the application start.
/// </summary>
/// <param name="context">The application event scope.</param>
public override void OnAppStart(AppEventArgs context) {
    base.OnAppStart(context);
    UserConnection = GetUserConnection(context);
    BindInterfaces();
}

#endregion

}

#endregion

}

```

Созданные классы связываются по тегу "Test". Этот же тег должен быть указан при конвертации сообщения (шаг 4) в поле `ChannelName`. Таким образом система определяет какие файлы необходимо использовать для получения данных профиля, загрузки вложений и отправки сообщений.

5. Сохраните и опубликуйте схему.

## Добавить механизм маршрутизации чатов

 Сложный

Для [настройки маршрутизации чатов](#) необходимо в выпадающем списке настройки очереди выбрать правило маршрутизации. Таким образом определяется механизм распределения чатов на операторов.

All agents

APPLY CANCEL

Name \*

All agents

Routing rule \*

To all agents

To all agents

To an available agent

Chat completion timeout, minutes

2

Supervisor

Рассмотрим пример добавления пользовательского механизма распределения.

**Пример.** Добавить пользовательский механизм распределения чатов. Системному пользователю (Supervisor) предоставить доступ к новым чатам. Оператору предоставить доступ к уже назначенным чатам.

## 1. Добавить новое правило маршрутизации чатов

- Перейдите в дизайнер системы по кнопке
- В блоке [ Настойка системы ] ([ System setup ]) перейдите по ссылке [ Справочники ] ([ Lookups ]).
- Используя фильтр в верхней части страницы, найдите справочник "Правила маршрутизации чатов в очереди" ("Rules for routing chat queue operators").
- Откройте наполнение справочника и добавьте **новое правило**:
  - [ Название ] ([ Name ]) — "Test rule".
  - [ Код ] ([ Code ]) — "TestRule".

Name	Description	Code
Test rule		TestRule
To an available agent	Distributes chats to free operators	ForFree
To all agents	Distributes chats to all operators	ForEveryone

## 2. Создать класс, который реализует интерфейс IOperatorRoutingRule

На этом шаге можно создать класс-наследник класса `BaseOperatorRoutingRule` или новый класс, реализующий интерфейс `IOperatorRoutingRule`.

Класс `BaseOperatorRoutingRule` содержит в себе абстрактные методы `PickUpFreeQueueOperators` и `GetChatOperator`, которые необходимо реализовать.

### Абстрактные методы класса `BaseOperatorRoutingRule`

```

/// <summary>
/// Pick up and return queue agent IDs.
/// </summary>
/// <param name="chatId"><see cref="OmniChat"/> The agent ID.</param>
/// <param name="queueId"><see cref="ChatQueue"/> The instance ID.</param>
/// <returns>The queue agent IDs.</returns>
protected abstract List<Guid> PickUpFreeQueueOperators(Guid chatId, Guid queueId);

/// <summary>
/// Returns the <see cref="OmniChat"/> agent ID.
/// </summary>
/// <param name="chatId"><see cref="OmniChat"/> The agent ID.</param>
/// <returns>The chat agent.</returns>
protected abstract Guid GetChatOperator(Guid chatId);

```

При этом в класс `BaseOperatorRoutingRule` уже заложена реализация интерфейса `IOperatorRoutingRule`, с использованием логики, приведенной ниже.

### Реализация интерфейса `IOperatorRoutingRule`

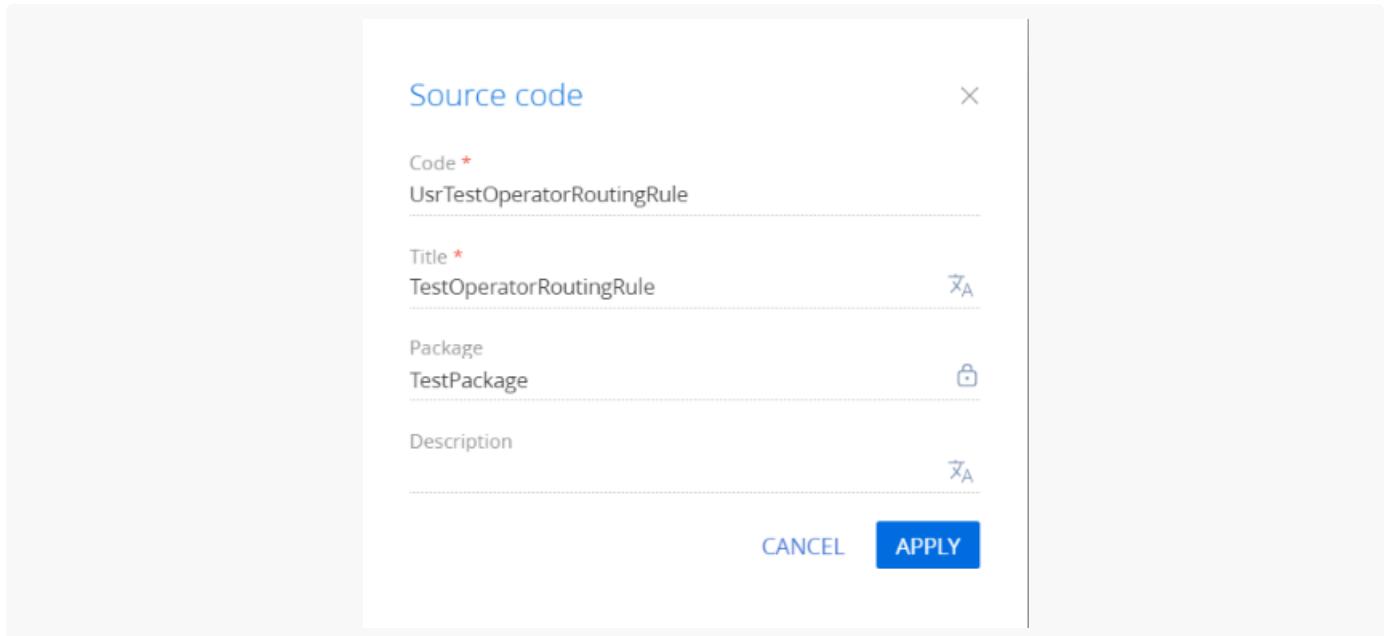
```
public List<Guid> GetOperatorIds(string chatId, Guid queueId) {
    var parsedChatId = Guid.Parse(chatId);
    var chatOperator = GetChatOperator(parsedChatId);
    return chatOperator.IsEmpty() ? new List<Guid> {
        chatOperator
    }: PickUpFreeQueueOperators(parsedChatId, queueId);
}
```

Если чату уже **назначен оператор** (в колонке `[OperatorId]` указан идентификатор пользователя), то необходимо выбрать этого оператора. Если чату **не назначен оператор**, то необходимо, используя метод `PickUpFreeQueueOperators` получить оператора или операторов в виде `List<Guid>`. В данном случае `Guid` это идентификаторы операторов из таблицы `[SysAdminUnit]`, которые в итоге и получат уведомления о новом чате, а также доступ к чату через коммуникационную панель.

Создадим класс `TestOperatorRoutingRule` реализующий простейшую логику маршрутизации чата на системного пользователя (Supervisor).

Чтобы **создать класс**:

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Исходный код ] ([ Add ] —> [ Source code ]).
3. В дизайнере схем заполните **свойства схемы**:
  - [ Код ] ([ Code ]) — "UsrTestOperatorRoutingRule".
  - [ Заголовок ] ([ Title ]) — "TestOperatorRoutingRule".



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

4. В дизайнере схем добавьте исходный код.

```

TestOperatorRoutingRule

namespace Terrasoft.Configuration.Omnichannel.Messaging {
    using System;
    using System.Collections.Generic;
    using Terrasoft.Core;
    using Terrasoft.Core.DB;

    #region Class: ForEveryoneOperatorRoutingRule

    /// <summary>
    /// Retrieve chat agents.
    /// </summary>
    public class TestOperatorRoutingRule: BaseOperatorRoutingRule {

        #region Constructors: Public

        /// <summary>
        /// Initialize a new instance of <see cref="TestOperatorRoutingRule"/>.
        /// </summary>
        /// <param name="userConnection"><see cref="UserConnection"/> the instance.</param>
        public TestOperatorRoutingRule(UserConnection userConnection) : base(userConnection) {}

        #endregion

        #region Methods: Private
    }
}

```

```

#endregion

#region Methods: Protected

/// <inheritdoc cref="BaseOperatorRoutingRule.PickUpFreeQueueOperators(Guid, Guid)">
protected override List < Guid > PickUpFreeQueueOperators(Guid chatId, Guid queueId) {
    return new List < Guid > {
        Guid.Parse("7F3B869F-34F3-4F20-AB4D-7480A5FDF647")
    };
}

/// <inheritdoc cref="BaseOperatorRoutingRule.GetChatOperator(Guid)">
protected override Guid GetChatOperator(Guid chatId) {
    Guid operatorId = (new Select(UserConnection).Column("OperatorId").From("OmniChat",
        as Select).ExecuteScalar < Guid > ());
    return operatorId;
}

#endregion
}

#endregion
}

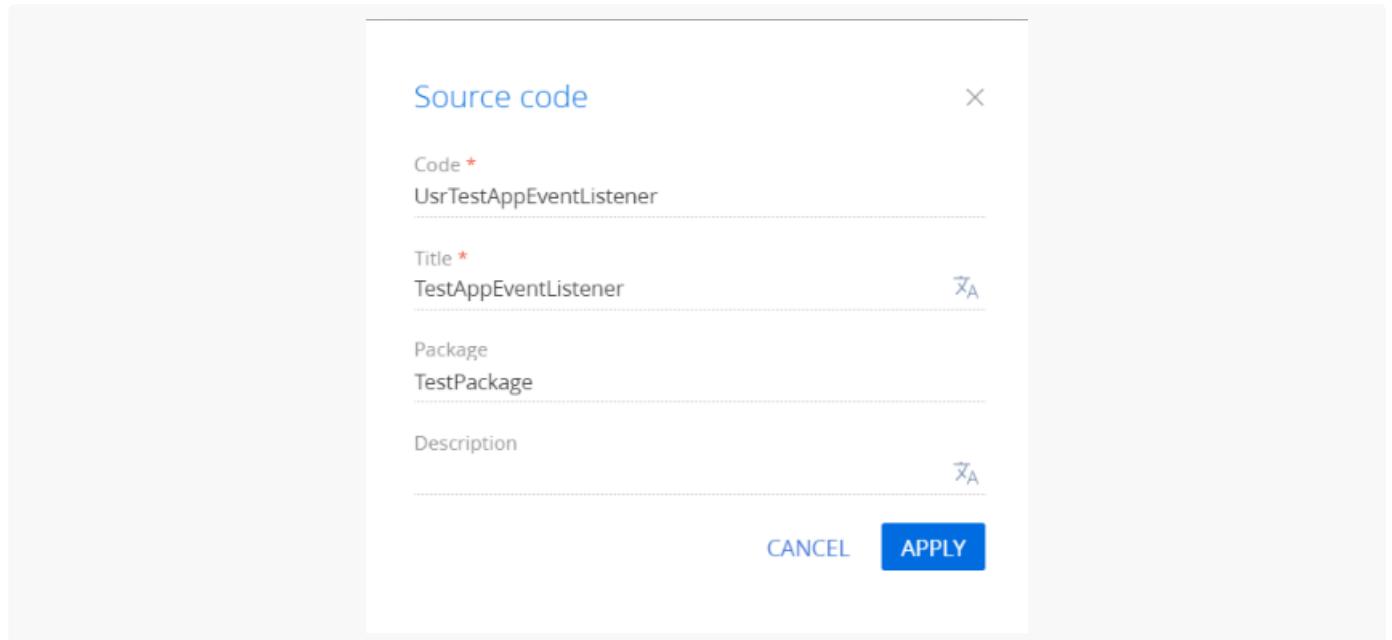
```

### 3. Связать интерфейс и код правила из справочника

Чтобы выполнить связывание реализации интерфейса и буквенного кода правила, указанного в справочнике на шаге 1, создайте класс `TestAppEventListener` (наследник класса `AppEventListenerBase`), в котором выполните связывание интерфейсов.

Чтобы **создать класс**:

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Исходный код ] ([ Add ] —> [ Source code ]).
3. В дизайнере схем заполните **свойства схемы**:
  - [ Код ] ([ Code ]) — "UsrTestAppEventListener".
  - [ Заголовок ] ([ Title ]) — "TestAppEventListener".



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

4. В дизайнере схем добавьте исходный код.

```
TestAppEventListener

namespace Terrasoft.Configuration.Omnichannel.Messaging {
    using Common;
    using Core;
    using Terrasoft.Core.Factories;
    using Web.Common;

    #region Class: TestAppEventListener

    /// <summary>
    /// The class that runs prerequisites for OmnichannelMessaging on the application start.
    /// </summary>
    public class TestAppEventListener: AppEventListenerBase {

        #region Fields: Protected

        protected UserConnection UserConnection {
            get;
            private set;
        }

        #endregion

        #region Methods: Protected

        /// <summary>
        /// Retrieve the user connection from the application event scope.
        /// </summary>
    }
}
```

```

/// </summary>
/// <param name="context">The application event scope.</param>
/// <returns>User connection.</returns>
protected UserConnection GetUserConnection(AppEventArgs context) {
    var appConnection = context.Application["AppConnection"] as AppConnection;
    if (appConnection == null) {
        throw new ArgumentNullException("AppConnection");
    }
    return appConnection.SystemUserConnection;
}

protected void BindInterfaces() {
    ClassFactory.Bind < IOperatorRoutingRule,
    TestOperatorRoutingRule > ("TestRule");
}

#endregion

#region Methods: Public

/// <summary>
/// Handle the application start.
/// </summary>
/// <param name="context">The application event scope.</param>
public override void OnAppStart(AppEventArgs context) {
    base.OnAppStart(context);
    UserConnection = GetUserConnection(context);
    BindInterfaces();
}

#endregion

}

#endregion
}

```

## 4. Перезапустите приложение в IIS

Для применения изменений перезапустите приложение в IIS. После перезапуска приложения и выбора в настройках очереди правила "Test rule", чаты будут распределяться по заданным условиям.

Queue

CLOSE

Name \*

New queue for Supervisor

Routing rule \*

Test rule

To all agents

To an available agent

Chat completion timeout, minutes

## Механизм Feature Toggle



Средний

**Feature toggle** — техника разработки программных продуктов. **Назначение** Feature toggle — возможность подключения дополнительной функциональности в работающем приложении.

В Creatio Feature toggle использует непрерывную интеграцию, которая позволяет сохранить работоспособность приложения и скрыть функциональность на стадии разработки. В исходном коде приложения присутствует блок с дополнительной функциональностью и условный оператор, который определяет состояние подключения функциональности.

Понятие Feature toggle описано на [Википедии](#).

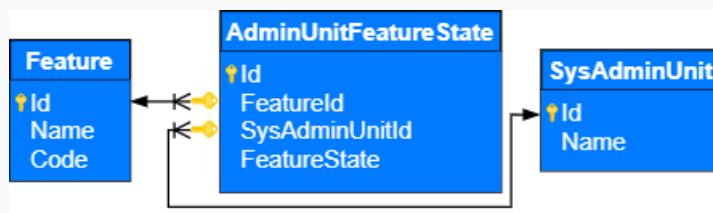
## Хранить сведения о функциональности

Сведения о функциональности хранятся в базе данных приложения.

**Таблицы** базы данных, которые хранят информацию о функциональности:

- [Feature] — перечень функциональности, которая доступна для подключения/отключения. По умолчанию таблица пуста.
- [AdminUnitFeatureState] (колонка [FeatureState]) — информация о состоянии функциональности (подключена/отключена). Таблица [AdminUnitFeatureState] связывает таблицы [Feature] и [SysAdminUnit] базы данных. Таблица [SysAdminUnit] содержит информацию о пользователях и группах пользователей приложения.

Диаграмма взаимосвязи таблиц базы данных, которые хранят информацию о функциональности, представлена на рисунке ниже.



Основные колонки таблицы `[Feature]` базы данных приведены в таблице ниже.

Основные колонки таблицы `[Feature]`

Колонка	Тип	Описание
<code>[Id]</code>	<code>uniqueidentifier</code>	Уникальный идентификатор записи.
<code>[Name]</code>	<code>nvarchar(250)</code>	Имя функциональности.
<code>[Code]</code>	<code>nvarchar( 50 )</code>	Код функциональности.

Основные колонки таблицы `[AdminUnitFeatureState]` базы данных приведены в таблице ниже.

Основные колонки таблицы `[AdminUnitFeatureState]`

Колонка	Тип	Описание
<code>[Id]</code>	<code>uniqueidentifier</code>	Уникальный идентификатор записи.
<code>[FeatureId]</code>	<code>uniqueidentifier</code>	Уникальный идентификатор записи из таблицы <code>[Feature]</code> .
<code>[SysAdminUnitId]</code>	<code>uniqueidentifier</code>	Уникальный идентификатор записи пользователя.
<code>[FeatureState]</code>	<code>int</code>	Состояние функциональности ( <code>1</code> — подключена, <code>0</code> — отключена).

## Подключить дополнительную функциональность

**Способы** подключения дополнительной функциональности:

- Для текущего пользователя приложения.
- Для всех пользователей приложения.

## Подключить функциональность для текущего пользователя приложения

- Перейдите на страницу `[Функциональность]` (`[Features]`), которая используется для точечного

подключения функциональности.

Шаблон адреса страницы функциональности

[Адрес приложения]/0/Nui/ViewModule.aspx#BaseSchemaModuleV2/FeaturesPage

Пример адреса страницы функциональности

<http://mycreatio.com/0/Nui/ViewModule.aspx#BaseSchemaModuleV2/FeaturesPage>

2. Переключите соответствующий переключатель функциональности, которую планируется подключить.
3. Нажмите кнопку [ Сохранить изменения ] ([ Save changes ]).

Всплывающая подсказка отображает статус функциональности для группы пользователей, в которую входит текущий пользователь. Подключение функциональности для текущего пользователя не меняет ее статус для группы пользователей. Т. е. для текущего пользователя создается дополнительное правило приоритет которого выше, чем приоритет правила для группы пользователей, в которую входит текущий пользователь.

Пример страницы [ Функциональность ] ([ Features ]) приведен на рисунке ниже.

Подключить функциональность для всех пользователей приложения

Чтобы **подключить функциональность для всех пользователей приложения**, выполните SQL-запрос к базе данных, который приведен ниже.

## Пример SQL-запроса

MSSQL

```

DECLARE @featureCode varchar(max) = 'NewEmailSettingsWithoutAutoSynchronization', @featureId unique
set @featureId = (select top 1 Id from Feature where Code = @featureCode);
IF @featureId is null
BEGIN
    insert into Feature
        (Name, Code)
    values
        (@featureCode, @featureCode);
    set @featureId = (select top 1 Id from Feature where Code = @featureCode);
END;
delete from AdminUnitFeatureState where FeatureId = @featureId;
insert into AdminUnitFeatureState
    (SysAdminUnitId, FeatureState, FeatureId)
values
    ('A29A3BA5-4B0D-DE11-9A51-005056C00008', 1, @featureId),
    ('720B771C-E7A7-4F31-9CFB-52CD21C3739F', 1, @featureId);

```

PostgreSQL

```

do $$

DECLARE
    featureCode varchar(100) = 'EmailIntegrationV2';
    featureId UUID;
BEGIN
    featureId = (select "Id" from "Feature" where "Code" = featureCode limit 1);
    IF featureId is null THEN
        insert into "Feature"
            ("Name", "Code")
        values
            (featureCode, featureCode);
        featureId = (select "Id" from "Feature" where "Code" = featureCode limit 1);
    end IF;
    delete from "AdminUnitFeatureState" where "FeatureId" = featureId;
    insert into "AdminUnitFeatureState"
        ("SysAdminUnitId", "FeatureState", "FeatureId")
    values
        ('A29A3BA5-4B0D-DE11-9A51-005056C00008', 1, featureId),
        ('720B771C-E7A7-4F31-9CFB-52CD21C3739F', 1, featureId);
end $$;

```

Oracle

```

DECLARE
    existObject NUMBER := 0;
    v_featurecode VARCHAR2(50) := 'LoadAttachmentsInSameProcess';
    v_featureid VARCHAR2(38);
BEGIN
    SELECT COUNT(1) INTO existObject FROM "Feature" WHERE "Code" = v_featurecode;
    if existObject = 0 THEN
        INSERT INTO "Feature" ("Name", "Code") VALUES (:v_featurecode, :v_featurecode);
    END IF;
    SELECT "Id" INTO v_featureid FROM "Feature" WHERE "Code" = v_featurecode AND rownum <= 1;
    DELETE FROM "AdminUnitFeatureState" WHERE "FeatureId" = v_featureid;
    INSERT INTO "AdminUnitFeatureState" ("SysAdminUnitId", "FeatureState", "FeatureId")
        VALUES ('{A29A3BA5-4B0D-DE11-9A51-005056C00008}', 1, v_featureid);
END;
/

```

`featureCode` — переменная, которая содержит необходимое имя функциональности.

С помощью SQL-запроса Creatio позволяет подключить функциональность для группы пользователей и отключить функциональность для текущего пользователя и наоборот.

Чтобы **отключить функциональность для всех пользователей приложения**, выполните SQL-запрос к базе данных, в котором для колонки `[FeatureState]` таблицы `[AdminUnitFeatureState]` установите значение 0.

**Важно.** Creatio предоставляет возможность отключения функциональности, которая была подключена через SQL-запрос к базе данных, только путем выполнения соответствующего SQL-запроса к базе данных. Отсутствует возможность отключения функциональности на странице [Функциональность] ([Features]).

## Реализовать пользовательскую функциональность

### 1. Добавьте **пользовательскую функциональность**.

- Перейдите на страницу [Функциональность] ([Features]), которая используется для точечного подключения функциональности.
- Заполните **свойства добавляемой функциональности**:
  - [Код функциональности] ([Feature code]) — код пользовательской функциональности, которую планируется добавить (обязательное свойство).
  - [Имя функциональности] ([Feature name]) — имя пользовательской функциональности, которую планируется добавить.
  - [Описание функциональности] ([Feature description]) — описание пользовательской

функциональности, которую планируется добавить.

Features

Create feature

Feature code\* NewFeature

Feature name New Feature

Feature description Some feature description

**CREATE FEATURE**

**SAVE CHANGES**

- f. Нажмите кнопку [ Добавить функциональность ] ([ Create feature ]).

В результате пользовательская функциональность добавлена в реестр страницы [ Функциональность ] ([ Features ]).

Features

What can I do for you? > Creatio 8.0.0.5434

Create feature

Feature code\* NewFeature

Feature name New Feature

Feature description Some feature description

**CREATE FEATURE**

**SAVE CHANGES**

New Feature

Some feature description

2. Реализуйте **пользовательскую функциональность в исходном коде**. Пользовательская функциональность определяется в блоке условного оператора, который проверяет состояние подключения функциональности (колонка [FeatureState] таблицы [AdminUnitFeatureState]).

**Варианты** реализации пользовательской функциональности:

- На front-end стороне. Для этого воспользуйтесь инструкцией, которая приведена в пункте [Реализовать пользовательскую функциональность \(front-end\)](#).
- На back-end стороне. Для этого воспользуйтесь инструкцией, которая приведена в пункте [Реализовать пользовательскую функциональность \(back-end\)](#).

## Реализовать пользовательскую функциональность (front-end)

1. Создайте схему модели представления. Для этого воспользуйтесь инструкцией, которая приведена в

статье [Клиентский модуль](#).

2. В дизайнере модуля добавьте исходный код. Для этого используйте шаблон, который приведен ниже.

#### Шаблон реализации пользовательской функциональности

```
/* Метод, в котором определяется дополнительная функциональность. */
someMethod: function() {
    /* Проверка подключения функциональности. */
    if (Terrasoft.Features.getIsEnabled("КодФункциональности")) {
        /* Реализация дополнительной функциональности. */
        ...
    }
    /* Реализация метода. */
    ...
}
```

Для удобства использования в базовой схеме `BaseSchemaViewModel` модели представления определен метод `getIsFeatureEnabled`. Поэтому можно заменить метод `Terrasoft.Features.getIsEnabled` на `this.getIsFeatureEnabled("КодФункциональности")`.

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

Чтобы пользовательская функциональность отобразилась в клиентском коде и загружалась браузером на front-end стороне, обновите страницу.

## Реализовать пользовательскую функциональность (back-end)

1. Создайте схему исходного кода. Для этого воспользуйтесь инструкцией, которая приведена в статье [Исходный код](#).
2. В дизайнере исходного кода добавьте исходный код.

Класс `Terrasoft.Configuration.FeatureUtilities` предоставляет набор расширяющих методов класса `UserConnection`, которые позволяют использовать функциональность `Feature toggle` в схемах исходного кода на back-end стороне приложения. Для этого используйте шаблон, который приведен ниже.

#### Шаблон реализации пользовательской функциональности

```
...
/* Пространство имен, в котором определена возможность переключения дополнительной функциональности */
using Terrasoft.Configuration;
...
/* Метод, в котором будет определяться дополнительная функциональность. */
public void AnyMethod() {
    /* Проверка, подключена ли функциональность. */
    if (UserConnection.GetIsFeatureEnabled("КодФункциональности")) {
```

```

    /* Реализация дополнительной функциональности. */
}
/* Реализация метода. */
...
}

```

3. Установите состояние функциональности, вызвав метод `SetFeatureState`. Для этого используйте шаблон, который приведен ниже.

#### Шаблон вызова метода `SetFeatureState`

```
UserConnection.SetFeatureState("КодФункциональности", FeatureState);
```

## Класс FeatureUtilities



Пространство имен `Terrasoft.Configuration`.

Класс `Terrasoft.Configuration.FeatureUtilities` предоставляет набор расширяющих методов класса `UserConnection`, которые позволяют использовать функциональность `Feature Toggle` в схемах исходного кода на back-end стороне приложения. Также в классе `FeatureUtilities` объявлено перечисление состояний функциональностей (колонка `[FeatureState]` таблицы `[AdminUnitFeatureState]`).

**На заметку.** Полный перечень конструкторов, методов и свойств класса `UserConnection`, его родительских классов, а также реализуемых им интерфейсов, можно найти в [Библиотеке .NET классов](#).

## Методы

`static int GetFeatureState(this UserConnection source, string code)`

Возвращает состояние функциональности.

### Параметры

<code>source</code>	Пользовательское подключение.
<code>code</code>	Код функциональности.

`static int GetFeatureState(this UserConnection source, string code, Guid sysAdminUnitId)`

Возвращает состояние функциональности.

#### Параметры

source	Пользовательское подключение.
code	Код функциональности.
sysAdminUnitId	Уникальный идентификатор записи.

---

```
static bool GetIsFeatureEnabledForAnyUser(this UserConnection source, string code)
```

Признак, который возвращает состояние функциональности для любого пользователя.

#### Параметры

source	Пользовательское подключение.
code	Код функциональности.

---

```
static bool GetIsFeatureEnabledForAllUsers(this UserConnection source, string code)
```

Признак, который возвращает состояние функциональности для всех пользователей.

#### Параметры

source	Пользовательское подключение.
code	Код функциональности.

---

```
static Dictionary<string, int> GetFeatureStates(this UserConnection source)
```

Возвращает состояния всех функциональностей.

#### Параметры

source	Пользовательское подключение.
--------	-------------------------------

---

```
static List<FeatureInfo> GetFeaturesInfo(this UserConnection source)
```

Возвращает информацию относительно всех функциональностей и их состояний.

#### Параметры

**source**

Пользовательское подключение.

```
static void SetFeatureState(this UserConnection source, string code, int state, bool forAllUsers)
```

Устанавливает состояние функциональности.

#### Параметры

**source**

Пользовательское подключение.

**code**

Код функциональности.

**state**

Новое состояние функциональности.

**forAllUsers**

Признак, который устанавливает функциональность для всех пользователей.

```
static void SafeSetFeatureState(this UserConnection source, string code, int state, bool forAllUsers)
```

Установить состояние функциональности или создать функциональность (если она не существует).

#### Параметры

**source**

Пользовательское подключение.

**code**

Код функциональности.

**state**

Новое состояние функциональности.

**forAllUsers**

Признак, который устанавливает функциональность для всех пользователей.

```
static void CreateFeature(this UserConnection source, string code, string name, string description)
```

Создает функциональность.

#### Параметры

source	Пользовательское подключение.
code	Код функциональности.
name	Имя функциональности.
description	Описание функциональности.

---

```
static bool GetIsFeatureEnabled(this UserConnection source, string code)
```

Признак, который проверяет подключение функциональности.

#### Параметры

source	Пользовательское подключение.
code	Код функциональности.

---

```
static bool GetIsFeatureEnabled(this UserConnection source, string code, Guid sysAdminUnitId)
```

Признак, который проверяет подключение функциональности.

#### Параметры

source	Пользовательское подключение.
code	Код функциональности.
sysAdminUnitId	Уникальный идентификатор записи.

## Перечисление FeatureState

Перечисление `FeatureState` определяет состояния функциональностей (колонка `[FeatureState]` таблицы `[AdminUnitFeatureState]`).

Disabled	0	Функциональность отключена.
Enabled	1	Функциональность подключена.
Established	2	Функциональность установлена.

# Страница записи



**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 structure of a CRM record page. At the top right is the **ActionButtonsContainer**, which includes a **CLOSE** button, an **ACTIONS** dropdown, and a **VIEW** button. Below it is the **ActionDashboardContainer**, which displays a message: "You don't have any tasks yet" and "Press F above to add a task". To the left is the **LeftModulesContainer**, which contains fields for Name\*, Type (Customer), Owner (Mary King), Web (www.vertigosys.com), Primary phone (+44 (20) 3427 1374), and a progress bar indicating 95% completion. At the bottom right is the **TabsContainer**, which has tabs for ACCOUNT INFO (selected), CONTACTS AND STRUCTURE, MAINTENANCE, TIMELINE, and CONNECTED TO.

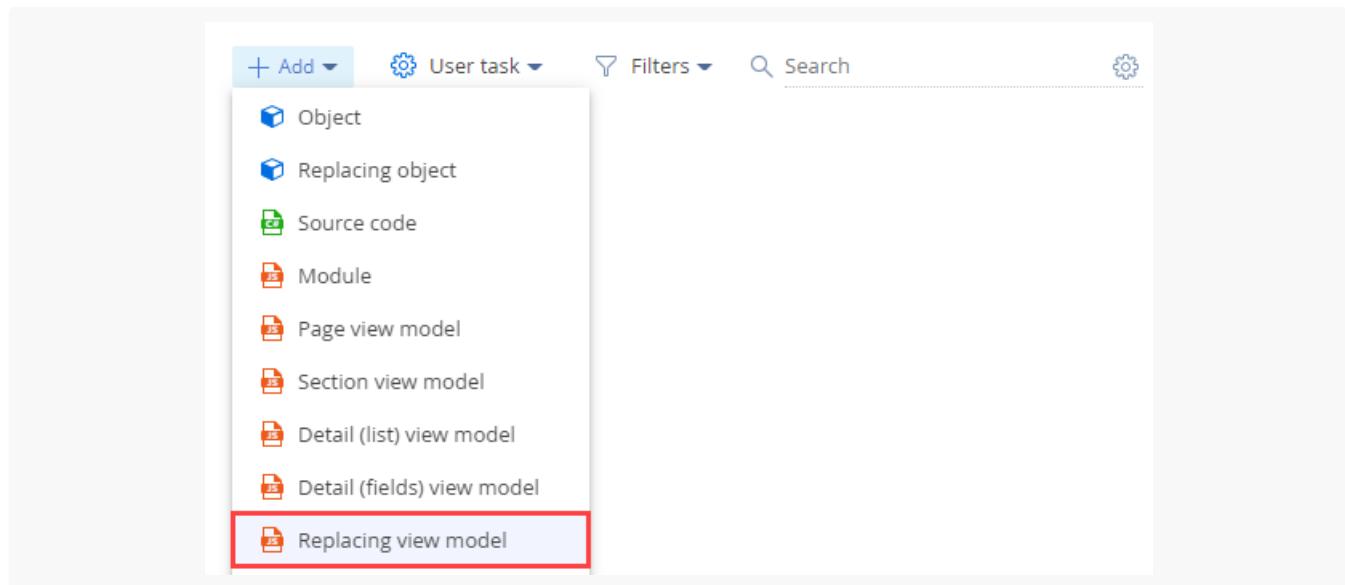
- Контейнер кнопок действий (`ActionButtonsContainer`) — содержит кнопки действий страницы записи.
- Контейнер левой части страницы записи (`LeftModulesContainer`) — содержит основные поля ввода и редактирования данных.
- Контейнер панели действий (`ActionDashboardContainer`) — содержит панель действий 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*]).

Status	Type	Object	Modified on
Under edit page	Client module	11/11/2021, 7:40:37 AM	
ServerSectionV2	Client module	11/11/2021, 8:43:19 AM	

- 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

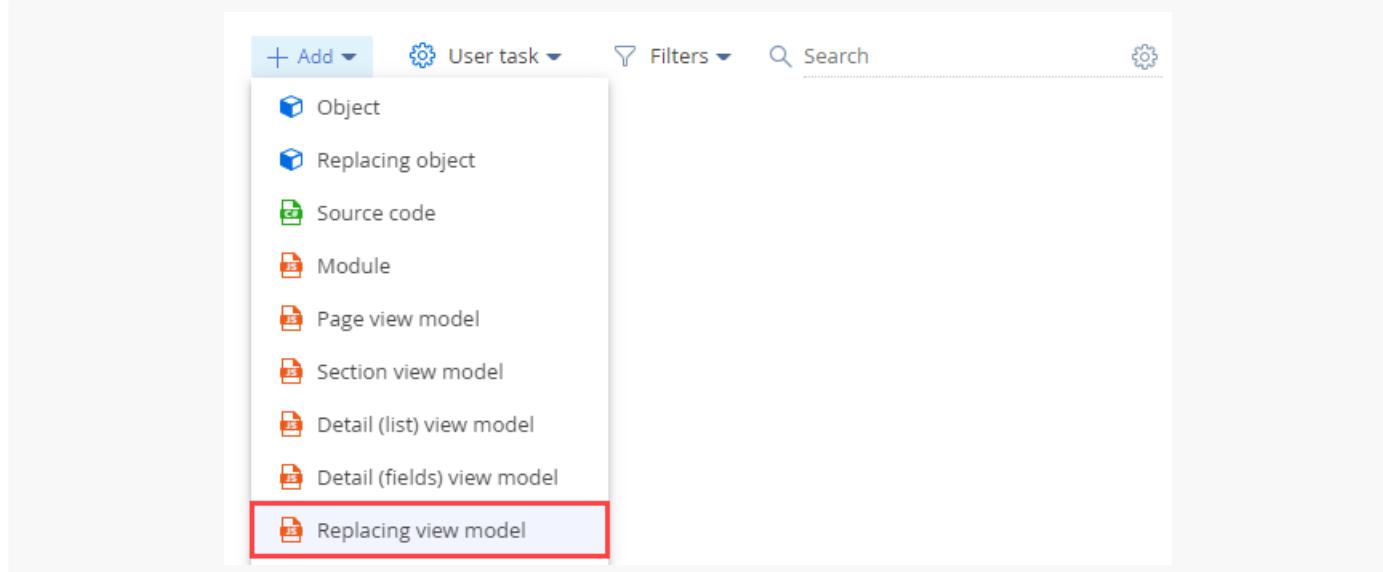


Пример реализован для продуктов линейки Sales Creatio.

**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

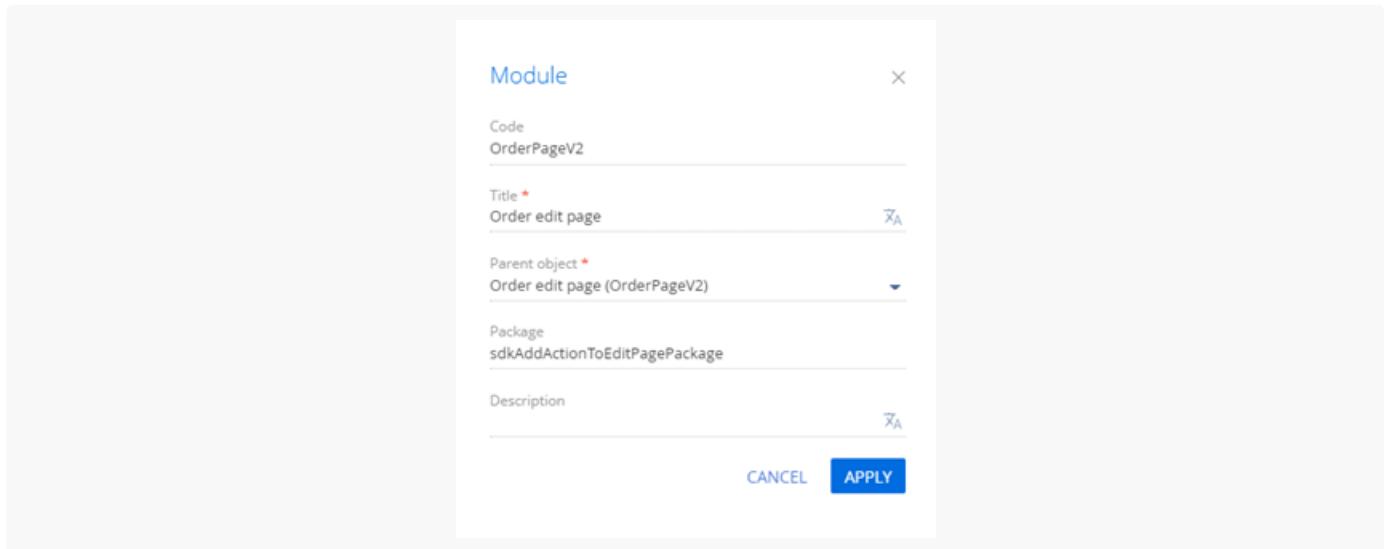
- Перейдите в раздел [ Конфигурация ] ([ *Configuration* ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ *Add* ] —> [ *Replacing view model* ]).



### 3. Заполните **свойства схемы**.

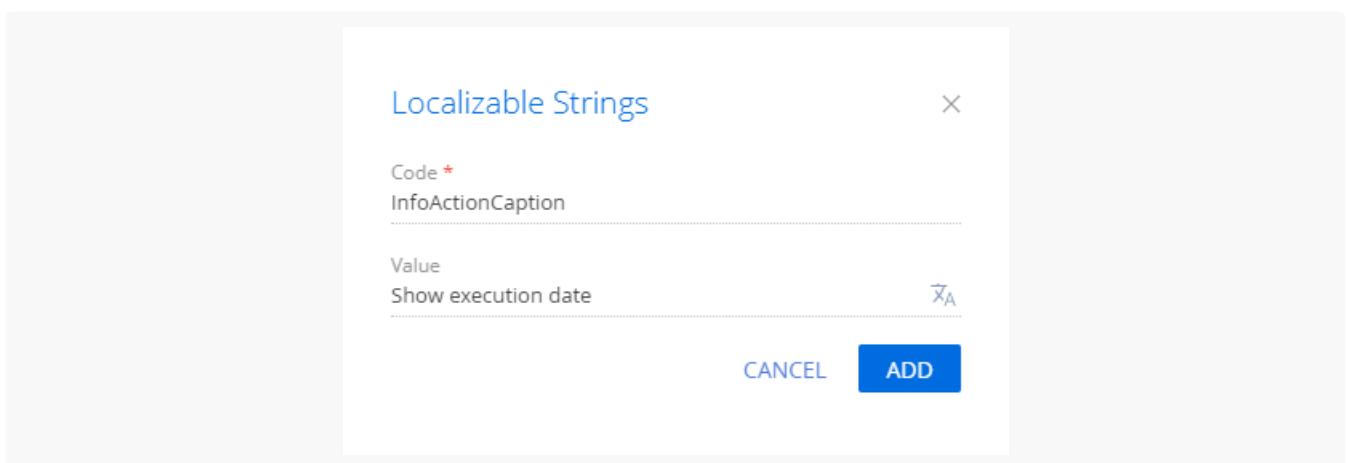
- [ Код ] ([ *Code* ]) — "OrderPageV2".
- [ Заголовок ] ([ *Title* ]) — "Страница редактирования заказа" ("Order edit page").

- [ 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 пропиниций. */
                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 из метода 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. At the top, there's a navigation bar with 'CLOSE', 'ACTIONS ▾', 'PRINT ▾', and 'VIEW ▾'. Below the navigation is a summary section with 'Total, \$' set to 2,725.00 and 'Payment amount, \$' set to 0.00. The main content area has tabs for 'SUMMARY', 'HISTORY', 'APPROVALS', and 'GENERAL INFORMATION'. Under 'GENERAL INFORMATION', it says 'Items: 3 Total: \$ 2,725.00'. Below this is a table of products:

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.

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

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

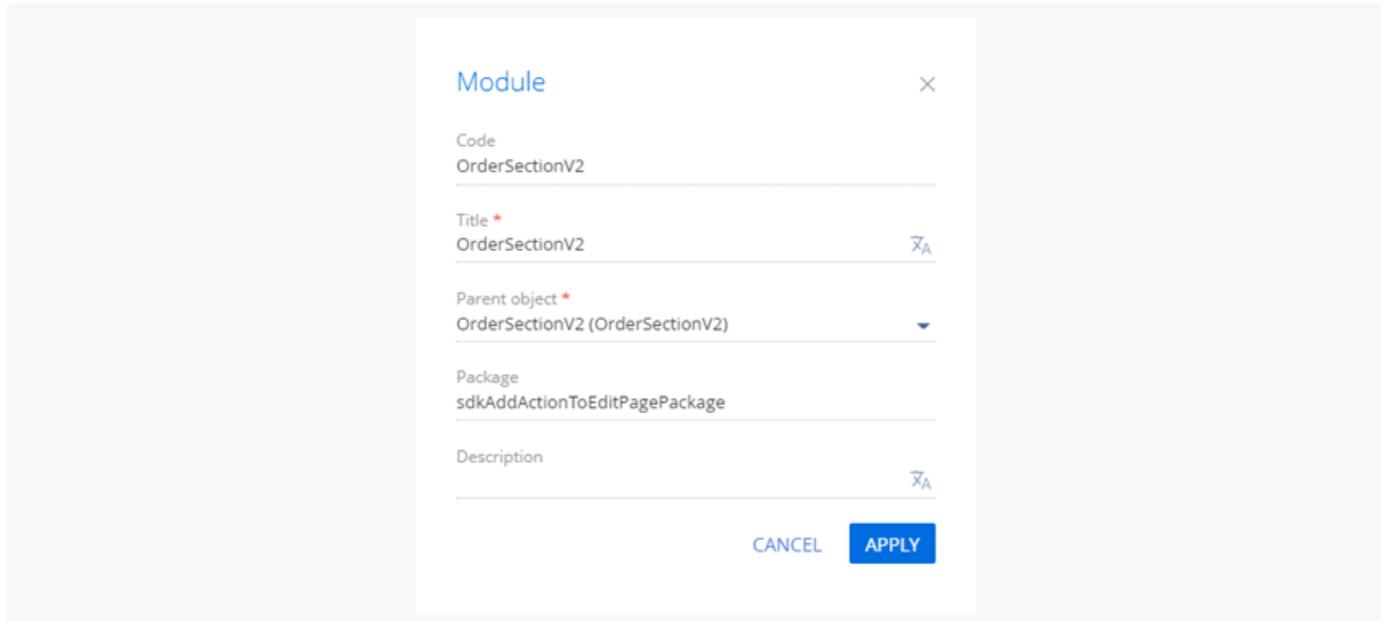
- Локализуемую строку, которая содержит текст пункта меню.
- Метод, который определяет доступность пункта меню.

## 2. Создать схему замещающей модели представления раздела

- 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* ]) — "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**:

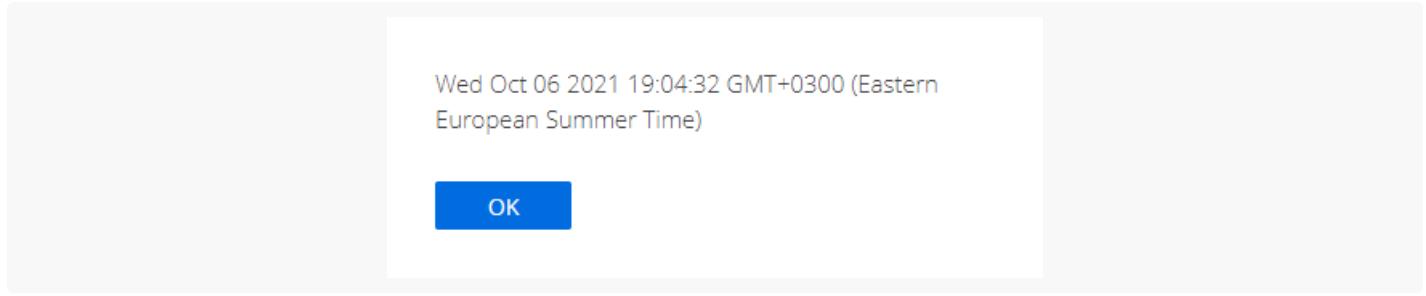
1. Очистите кэш браузера.
2. Обновите страницу раздела [ Заказы ] ([ Orders ]).

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 top right, there's a 'Actions' dropdown menu. One item in this menu, 'Show execution date', is highlighted with a red box. The rest of the menu items are enabled: 'Unfollow the feed', 'Set up access rights', 'Send for approval', and 'New invoice based on this order'. Below the actions, there's a table of products with their details. At the bottom of the card, there are tabs for SUMMARY, HISTORY, APPROVALS, GENERAL INFORMATION, and ATTACHMENTS AND NC >.

## Схема BasePageV2 js



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
<code>UpdatePageHeaderCaption</code>	Адресное	Публикация	Обновляет заголовок страницы.
<code>GridRowChanged</code>	Адресное	Подписка	Получает идентификатор выбранной в реестре записи при ее изменении.
<code>UpdateCardProperty</code>	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.

## Главное меню

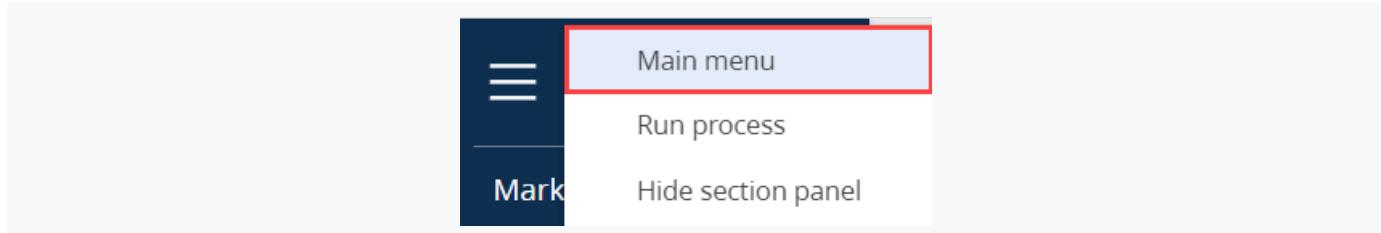


**Главное меню** отображается в рабочей области пользовательского интерфейса приложения после его загрузки. Интерфейс приложения подробно описан в статье [Обзор интерфейса Creatio](#).

## Способы вызова и структура главного меню

**Способы вызова** главного меню:

- Для **CRM-продуктов** Creatio нажмите на кнопку —> [ Главное меню ] ([ Main menu ]) в верхнем меню боковой панели.



- Для **CRM-бандла** продуктов Creatio нажмите на кнопку в верхнем меню боковой панели. Чтобы открыть меню CRM-продукта Creatio, выберите соответствующий пункт меню.



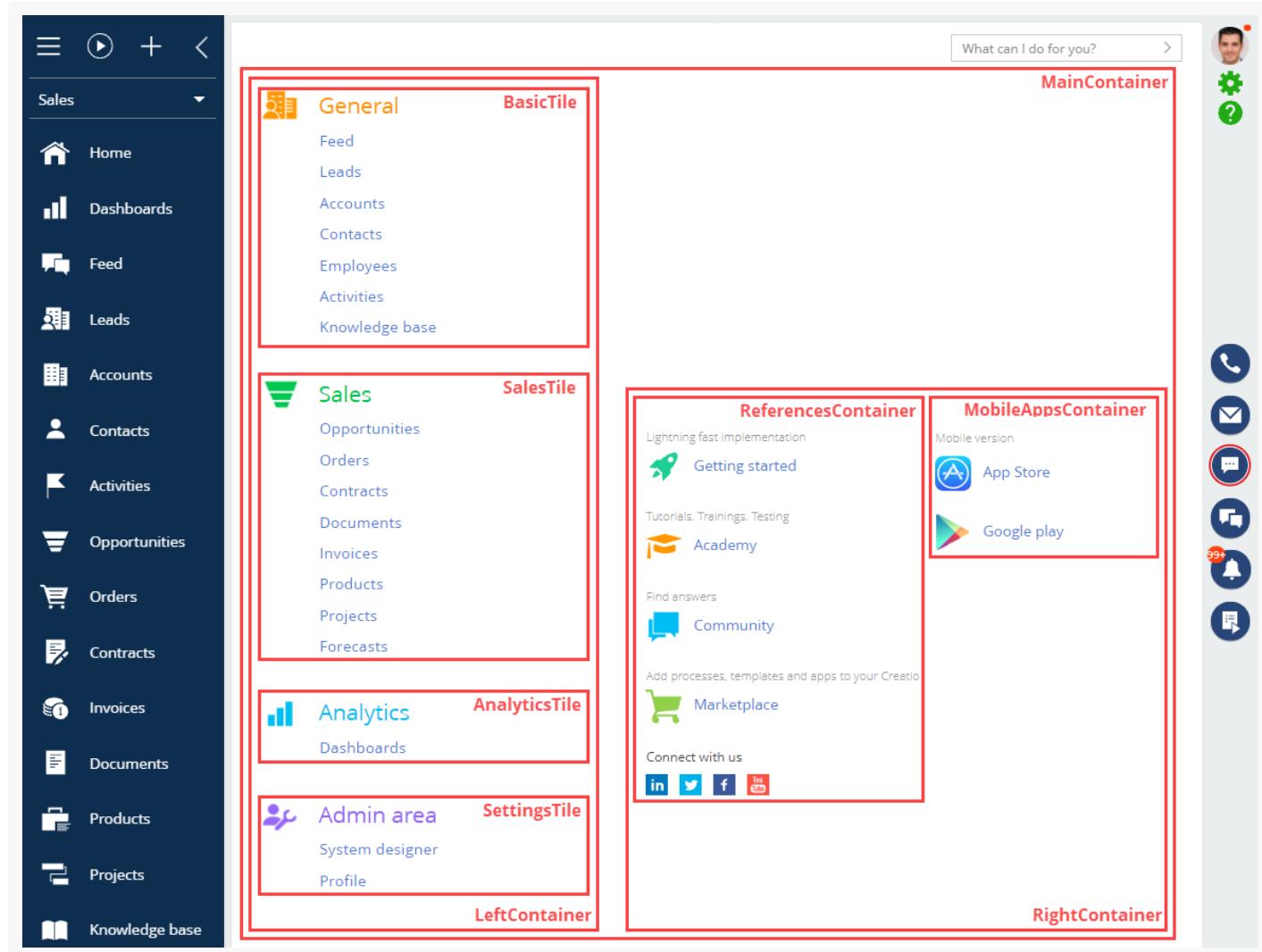
**Составляющие** главного меню:

- Базовая схема объекта `ApplicationMainMenu` пакета `NUI`.
- Схема главного меню продукта, которая наследуется от схемы `SimpleIntro` главного меню для базового продукта пакета `UIv2`. Например, главное меню продукта `SalesEnterprise` реализовано в схеме `EnterpriseIntro` главного меню для продукта `Enterprise` пакета `SalesEnterprise`.

## Контейнеры главного меню

Пункты главного меню интерфейса приложения зависят от продукта. Все элементы интерфейса размещены в соответствующих контейнерах, которые настраиваются в базовой или унаследованной схеме главного меню.

Основные **контейнеры** продукта `SalesEnterprise` представлены на рисунке ниже.



- Основной контейнер меню (`MainContainer`) — содержит все элементы главного меню.
- Контейнер разделов и настроек (`LeftContainer`) — содержит секции команд перехода в разделы и секцию настроек.
- Контейнер ресурсов (`RightContainer`) — содержит секции ссылок на дополнительные ресурсы.

- Контейнер базовой функциональности (`BasicTile`) — содержит команды перехода в разделы, общие для всех продуктов.
- Контейнер продаж (`SalesTile`) — содержит команды перехода в разделы продуктов Sales.
- Контейнер аналитики (`AnalyticsTile`) — содержит команду перехода в раздел [Итоги] ([`Dashboards`]).
- Контейнер настроек (`SettingsTile`) — содержит команды перехода в разделы настроек и в профиль пользователя.
- Контейнер видео (`VideoPanel`) — содержит проигрыватель и название видеоролика.
- Контейнер ссылок (`ReferencesContainer`) — содержит ссылки на обучающие и социальные ресурсы.
- Контейнер ссылок мобильного приложения (`MobileAppsContainer`) — содержит ссылки на скачивание мобильного приложения Creatio из магазинов приложений.

## Интеграция с каналами чатов



Средний

**Назначение** чатов — обработка операторами контакт-центра сообщений из популярных мессенджеров непосредственно в приложении Creatio. **Канал чата** в Creatio — это источник, из которого в приложение добавляются сообщения клиентов. Общий порядок действий по настройке обработки чатов описан в статье [Настроить обработку чатов](#).

**Каналы чатов**, интеграцию с которыми позволяет настроить Creatio:

- Facebook Messenger.
- WhatsApp.
- Telegram.

### Интеграция с каналом Facebook Messenger

**Этапы** предварительной подготовки к настройке интеграции с каналом Facebook Messenger:

1. Проверить наличие доступа к облачным сервисам Creatio.
2. Проверить наличие доступа к сервисам Facebook.

Инструкция по настройке интеграции с Facebook Messenger содержится в статье [Настроить интеграцию с Facebook Messenger](#).

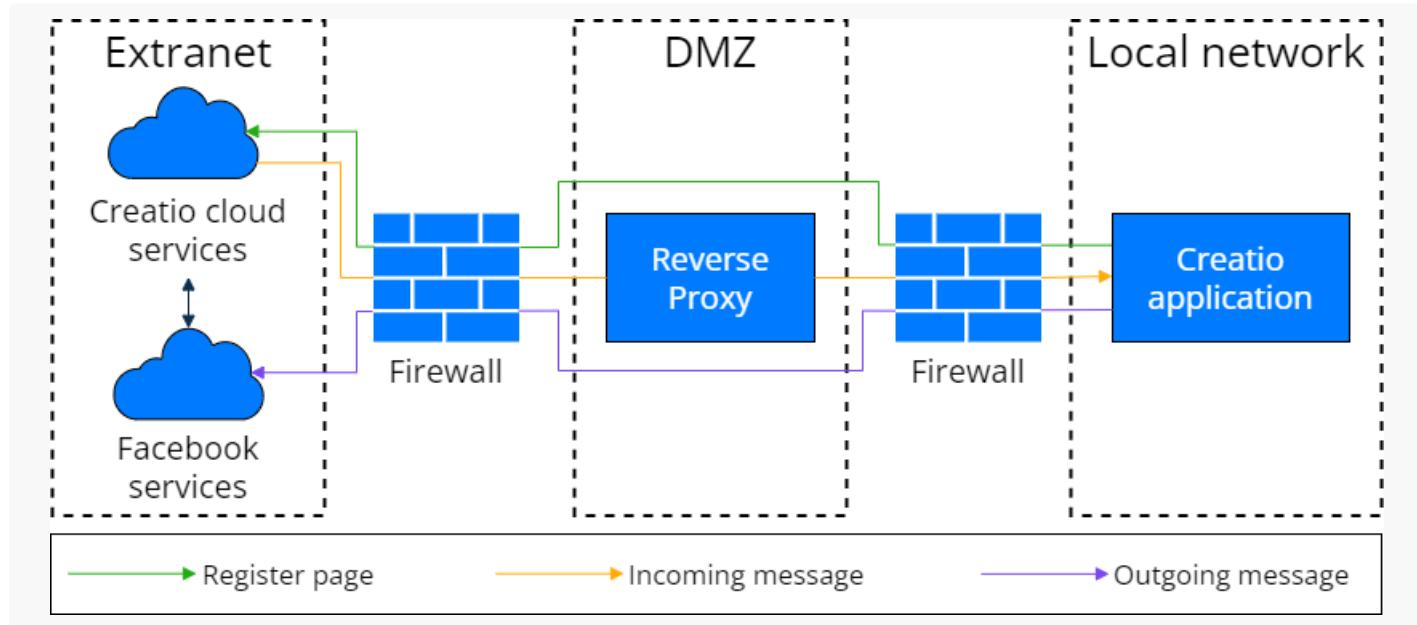
Взаимодействие со страницами Facebook выполняется от имени приложения Creatio Social, которое обращается к облачным сервисам. На уровне облачных сервисов выполняется:

- Подписка на новые входящие сообщения страницы Facebook.
- Создание привязки "подписка-сайт Creatio".

После получения **входящих сообщений** на страницу Facebook облачные сервисы Creatio отправляют сообщение в клиентское приложение Creatio. Если сайт Creatio недоступен, то сообщение помещается в очередь и будет отправлено повторно.

Отправка **исходящих сообщений** выполняется напрямую из приложения Creatio без использования облачных сервисов.

Схема взаимодействия on-site приложения Creatio с каналом Facebook Messenger представлена на рисунке ниже.



## Интеграция с каналом WhatsApp

Возможность интеграции с каналом WhatsApp доступна для приложений Creatio версии 7.18.0 и выше.

**Этапы** предварительной подготовки к настройке интеграции с каналом WhatsApp:

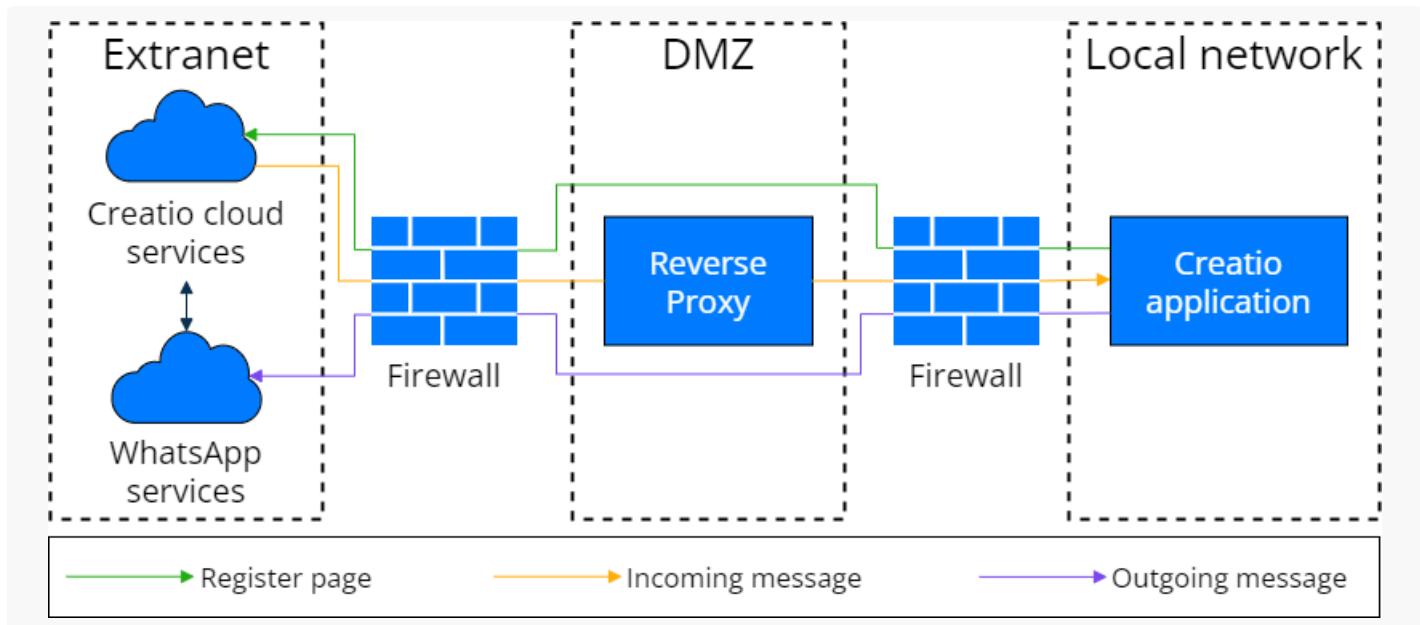
1. Проверить наличие доступа к облачным сервисам Creatio.
2. Создать учетную запись Twilio.
3. Приобрести номер телефона Twilio для получения входящих и отправки исходящих сообщений.

Инструкция по настройке интеграции с WhatsApp содержится в статье [Настройте интеграцию с WhatsApp](#).

После получения **входящих сообщений** на номер телефона Twilio облачные сервисы Creatio отправляют сообщение в клиентское приложение Creatio. Если сайт Creatio недоступен, то сообщение помещается в очередь и будет отправлено повторно.

Отправка **исходящих сообщений** выполняется напрямую из приложения Creatio без использования облачных сервисов.

Схема взаимодействия приложения Creatio, которое развернуто on-site, с каналом WhatsApp представлена на рисунке ниже.



## Интеграция с каналом Telegram

**Этапы** предварительной подготовки к настройке интеграции с каналом Telegram:

1. Создать API для каждого чат-бота Telegram.
2. Указать созданное API в настройках приложения Creatio.

Для интеграции с Telegram промежуточные сервисы не используются. Инструкция по настройке интеграции с Telegram содержится в статье [Настроить интеграцию с Telegram](#).

Для получения **входящих сообщений** приложение Creatio использует [Long pooling технологию](#). Чтобы проверить наличие новых сообщений, приложение периодично отправляет запрос к Telegram API.

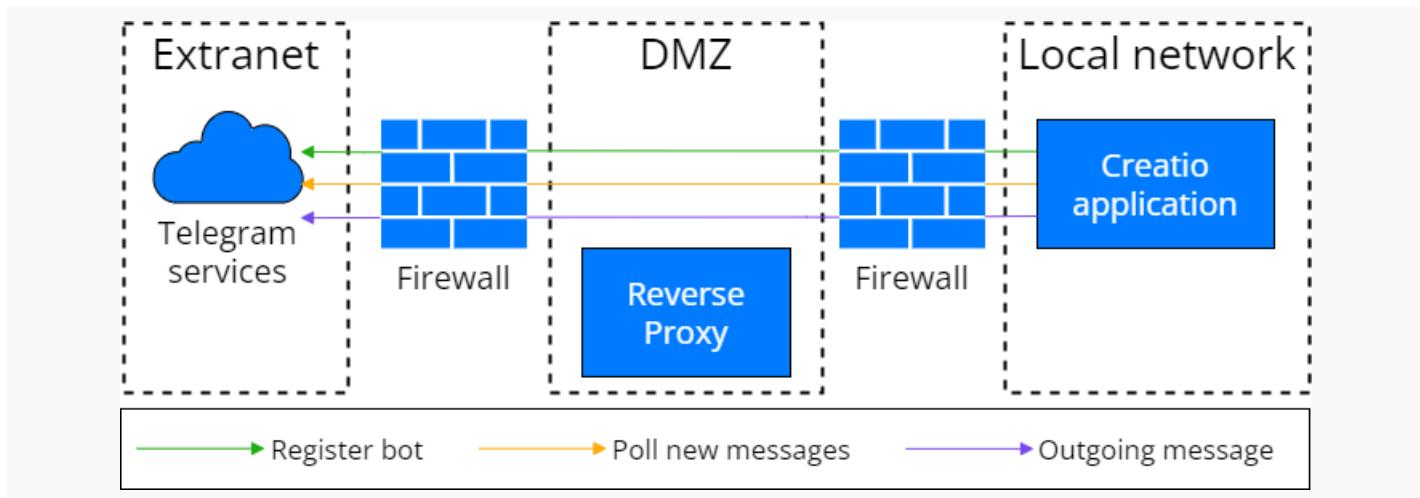
Шаблон строки запроса к Telegram API

```
https://api.telegram.org/bot{token}/getUpdates
```

`token` — токен, который использовался при регистрации канала.

Отправка **исходящих сообщений** выполняется напрямую на адреса сервисов Telegram.

Схема взаимодействия приложения Creatio, которое развернуто on-site, с каналом Telegram представлена на рисунке ниже.



## 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 with the following details:

- Header:** "Additional service / Jordan, AlphaBusiness" and "What can I do for you?".
- Action Dashboard Container:** Shows the status bar with "Distribution" and "HeaderContainer".
- Content Area:**
  - Header:** "ActionDashboardContainer", "HeaderContainer", "ContentContainer".
  - Status Bar:** "NEXT STEPS (0)", "Distribution", "HeaderContainer", "ContentContainer".
  - Tabs:** "LEAD INFO" (highlighted with circle 2), "CUSTOMER NEED DETAILS", "HISTORY", "ATTACHMENTS AND NOTES", "FEED".
  - Lead Record:** Contact name: Jordan, Account name: AlphaBusiness, Web: www.alphabusiness.com, Job title: , Mobile phone: , Email: , Country: United Kingdom.
  - Bottom:** "Similar leads" button.
- Sidebar:** Navigation icons and a sidebar panel showing account information: "Account: Alpha Business", "Web: www.alphabusiness.co.uk", "Industry".
- Right Panel:** Icons for phone, email, messaging, and notifications.

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 within a business process. On the left, there's a diagram of the process flow. A central node is labeled 'Validation item: Approve loan issuance'. An arrow from this node points to a 'Change application data' step. From there, it branches into two parallel paths: 'Add contract' and 'Add contract parameters'. These lead to 'Read contract id' and 'Connect contract with application' respectively. A feedback loop labeled 'Not confirmed' returns to the initial validation item node. On the right, a detailed configuration panel for the validation item is shown:

- Validation item:** Approve loan issuance
- Application:** [#Application#]
- Execute on page:** Preconfigured verification page (highlighted with a red box)
- How to perform validation:** Application approval
- Who performs validation:** 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 details of the 'Validation' activity. At the top, a breadcrumb navigation bar indicates the current step: Product selection > Filling in the ... > Validation > Settlement > Closed success... . Below this, a 'NEXT STEPS (1)' section shows a single step: 'Approve loan issuance'. This step is currently active, indicated by a green border. To the right of the step is a large green 'COMPLETE' button. Above the step, there are icons for phone, email, message, and flag.

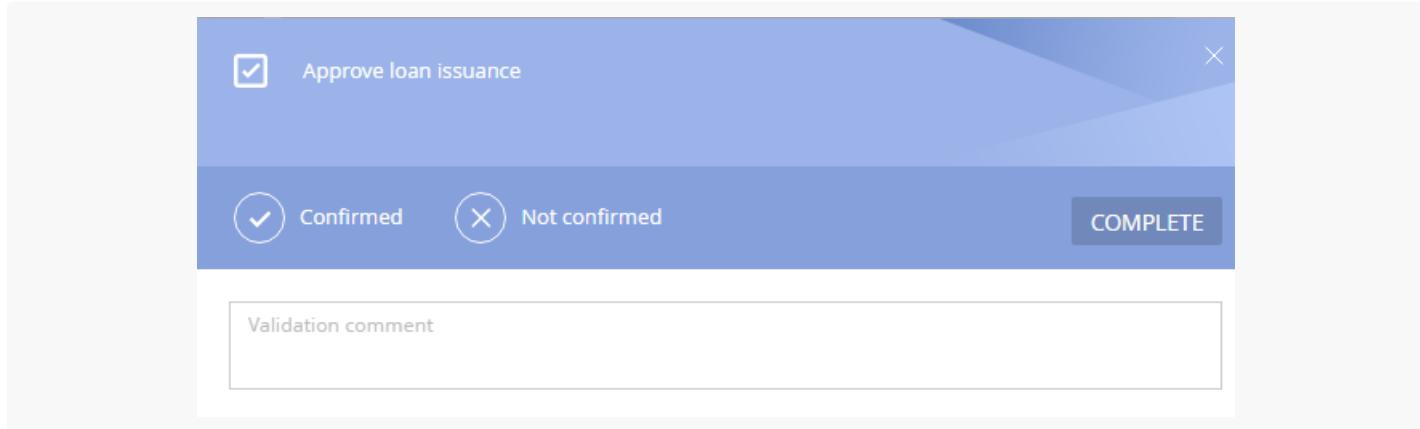
#### Структурные элементы преднастроенной страницы верификации:

- 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

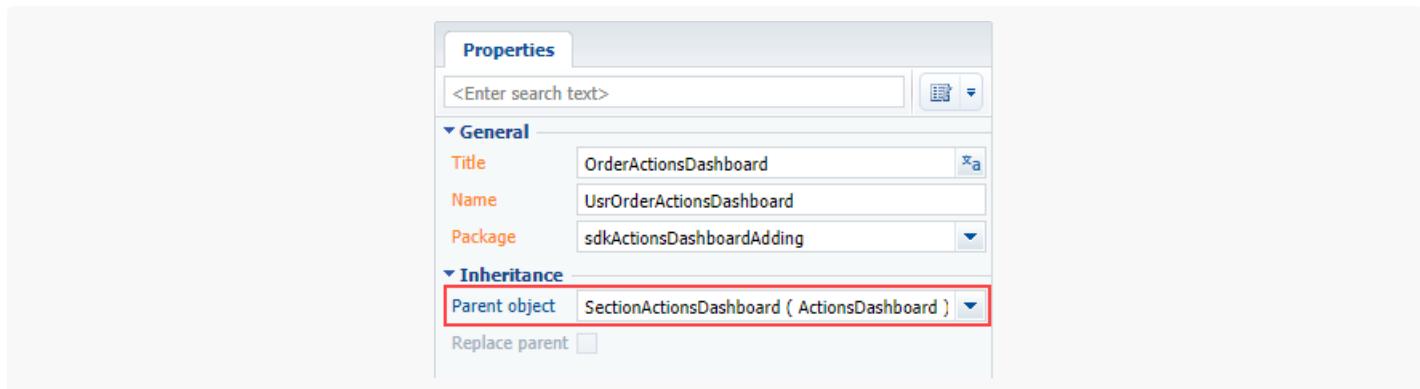
Пакет с реализацией примера можно скачать по [ссылке](#).

### Алгоритм реализации примера

#### 1. Создать схему модели представления OrderActionsDashboard

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

Рис. 1. — Свойства схемы модели представления



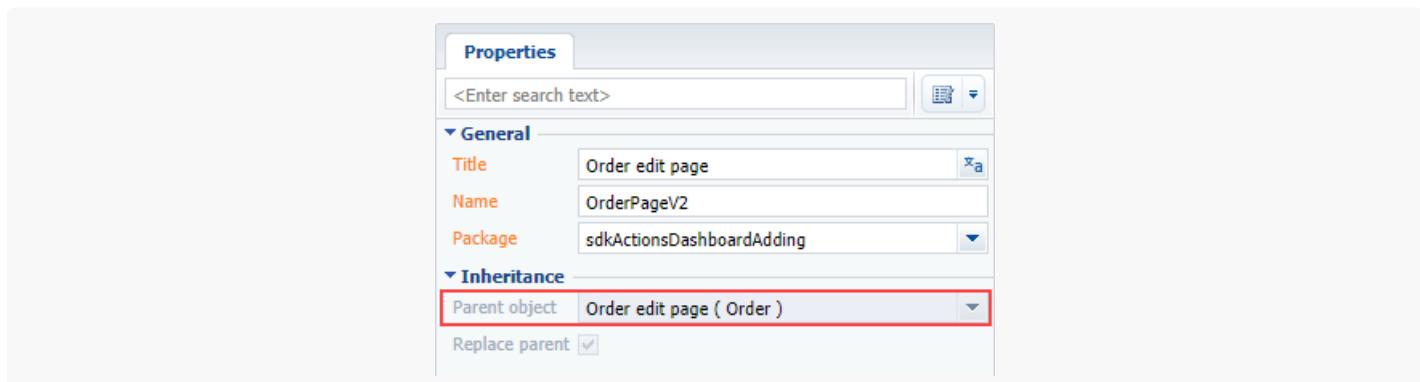
Исходный код схемы модели представления:

```
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

The screenshot shows the Creatio application interface for managing orders. At the top, there's a header with the order number 'ORD-41', a search bar 'What can I do for you?', and a 'Creatio' logo. Below the header is a toolbar with 'CLOSE', 'ACTIONS', and 'VIEW' buttons. A red box highlights the 'NEXT STEPS (2)' section, which contains two items: 'Prepare documents for customer' and 'Call to customer', both dated 5/9/2018 and assigned to 'Supervisor'. The main content area displays customer information (Customer: IT-Plus, Status: 1. Draft) and financial details (Total: \$ 5,389.81, Payment amount: \$ 5,389.81). Below this, a table lists products with columns for Product, Price, Quantity, Unit of measure, Discount, %, and Total. The 'PRODUCTS' tab is currently selected. At the bottom, there are tabs for ORDER DETAILS, DELIVERY, SUMMARY, HISTORY, GENERAL INFORMATION, and APPROV.

## 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

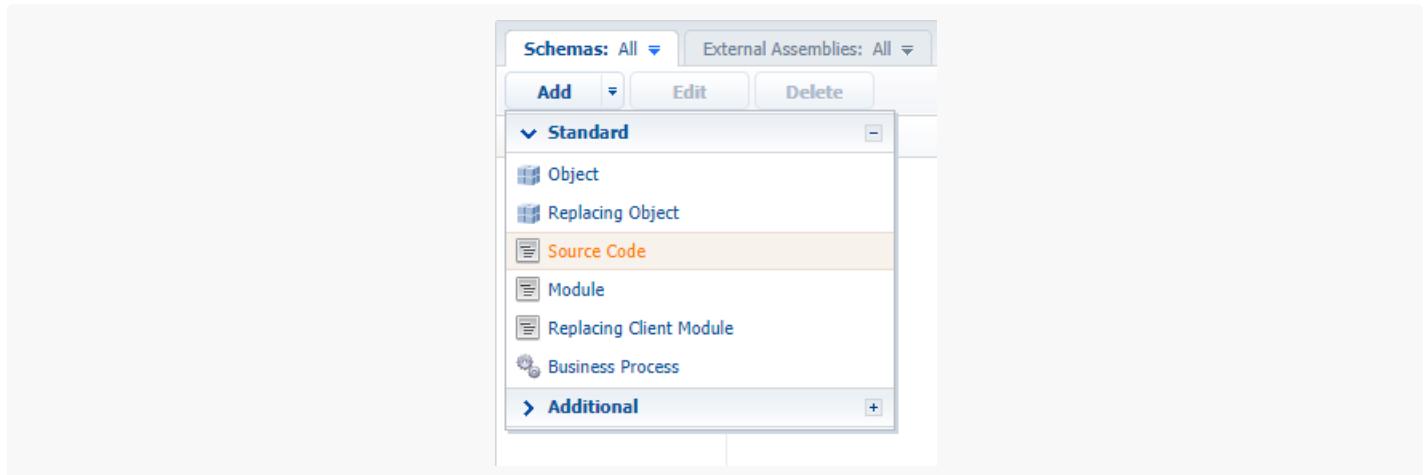
Пакет с реализацией примера можно скачать по [ссылке](#).

### 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).

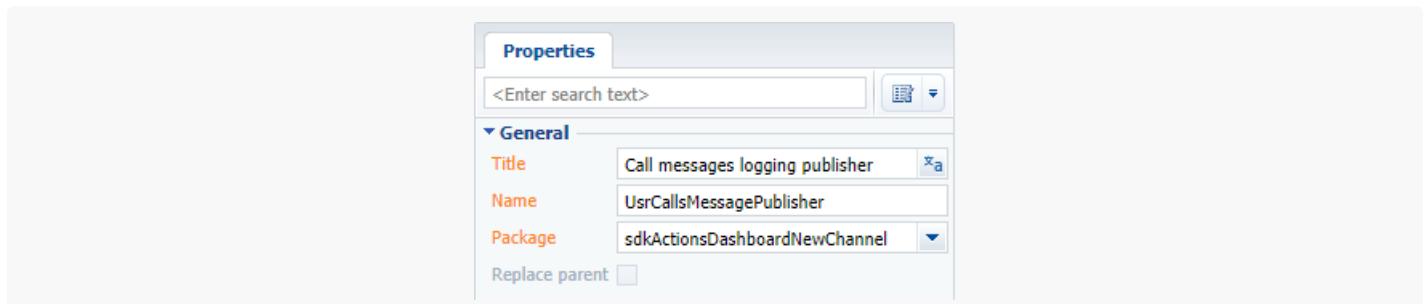
Рис. 1. — Добавление схемы исходного кода



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";
    }
}
}

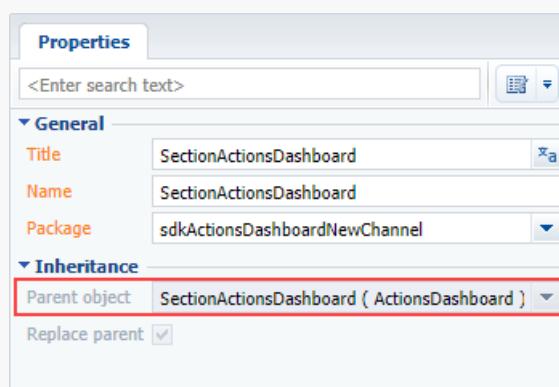
```

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*/
};

}
);

```

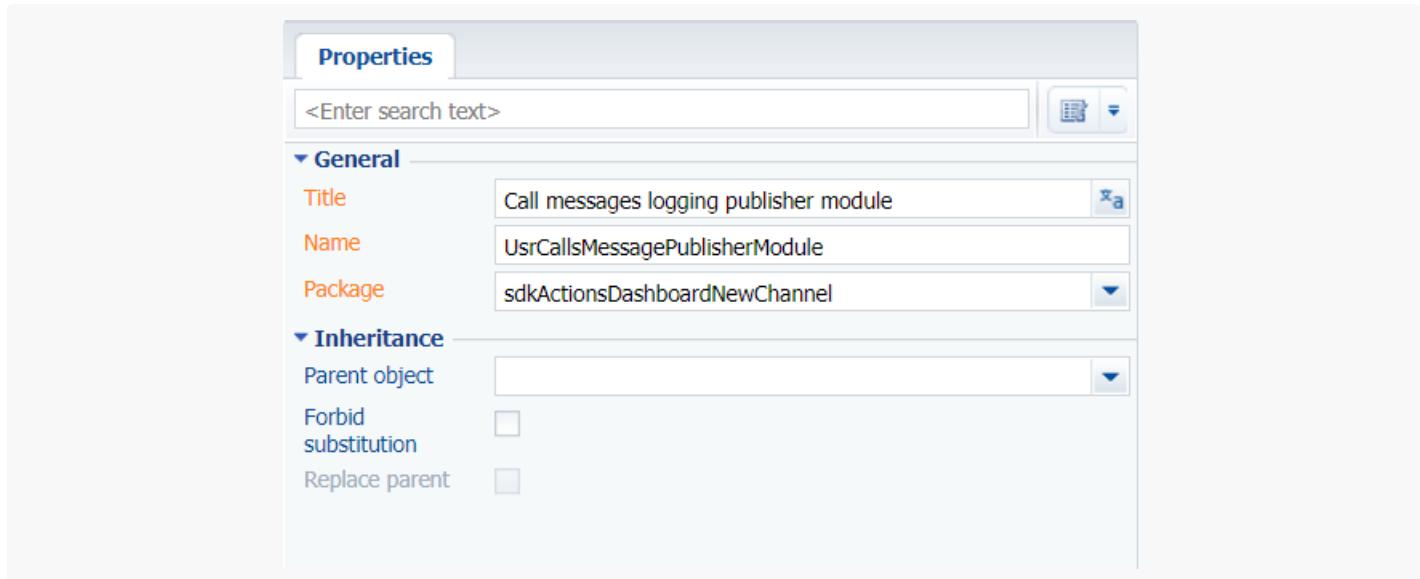
### 3. Создать модуль UsrCallsMessagePublisherModule

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"`.

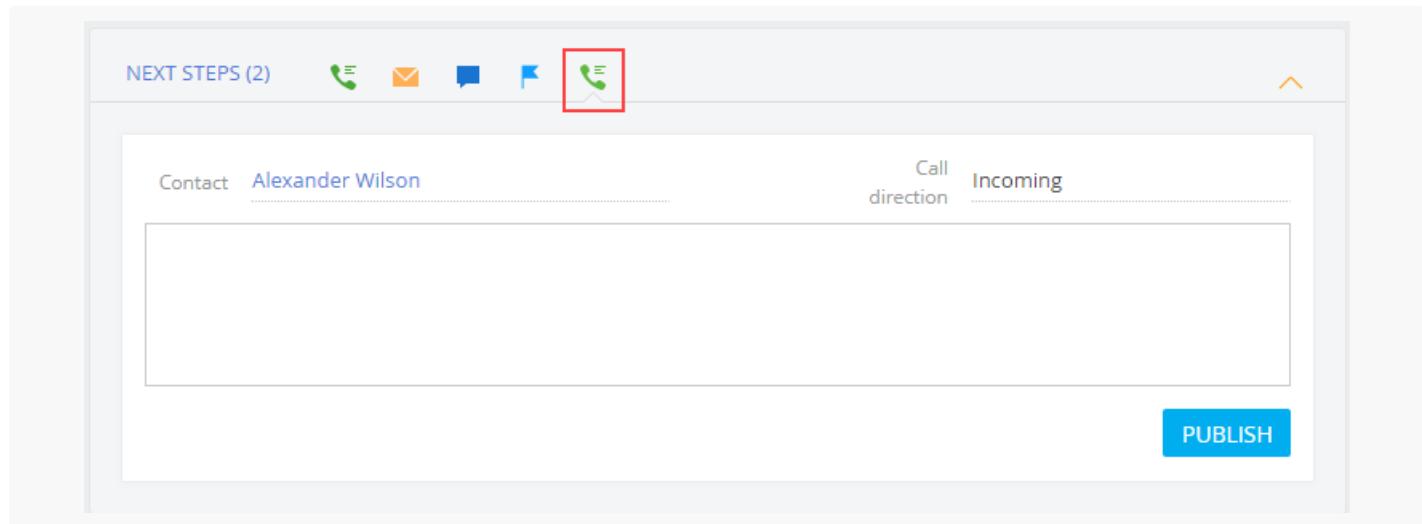
В исходном коде страницы укажите имя схемы объекта, с которым будет работать страница (в данном случае `Activity`), реализуйте логику публикации сообщения и переопределите метод `getServiceConfig`, в котором укажите имя класса из конфигурации.

```
// 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:

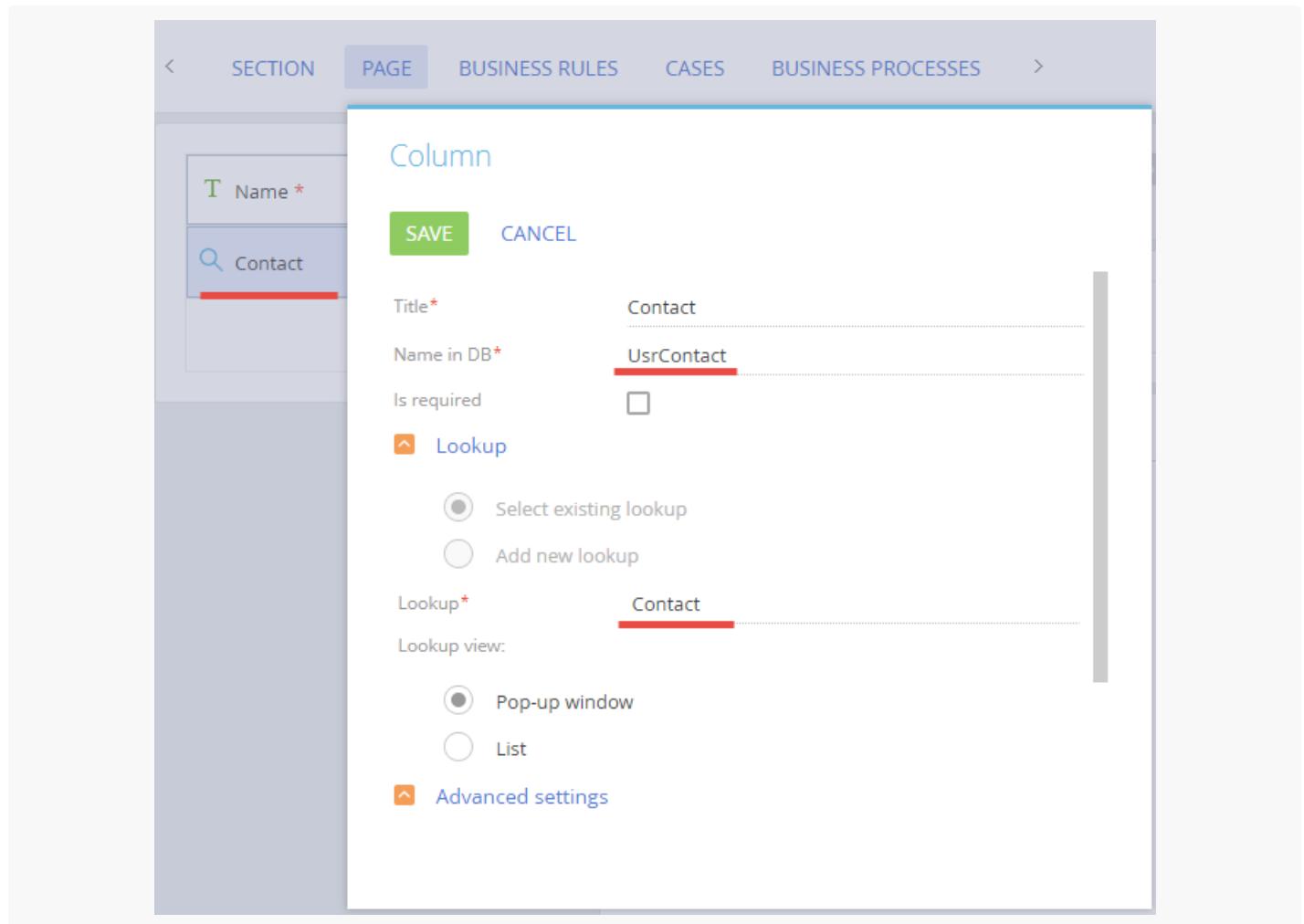
1. Убедитесь, что в справочнике [ *Языки общения* ] ([ *Customer languages* ]) используются английский и испанский языки (рис. 1).

Рис. 1. — Справочник [ Языки общения ]

Customer languages			
<input type="checkbox"/> Filters/folders ▾			
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

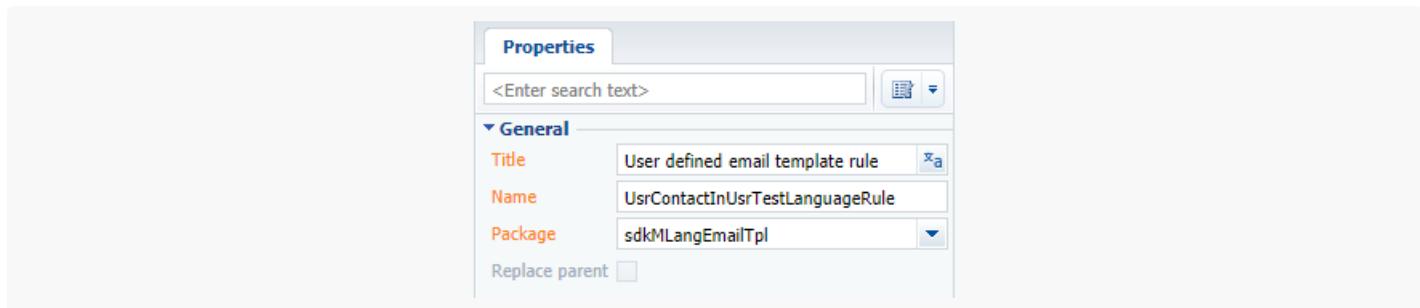
### 1. Добавить правило подбора языка

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]) — "UsrTestLanguagelterator";

- [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 software interface for managing email templates. At the top, there's a header 'Email templates'. Below it is a toolbar with a 'Filters/folders' dropdown. The main area displays a table with two columns: 'Case feedback request notification' and 'Subject'. The 'Subject' column contains the text 'New message on case #[#Number#]'. At the bottom of the table, there's a row labeled 'MLEmailTpptest (US, ES)' which is highlighted with a red underline, indicating it's the selected item.

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 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 content area displays 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

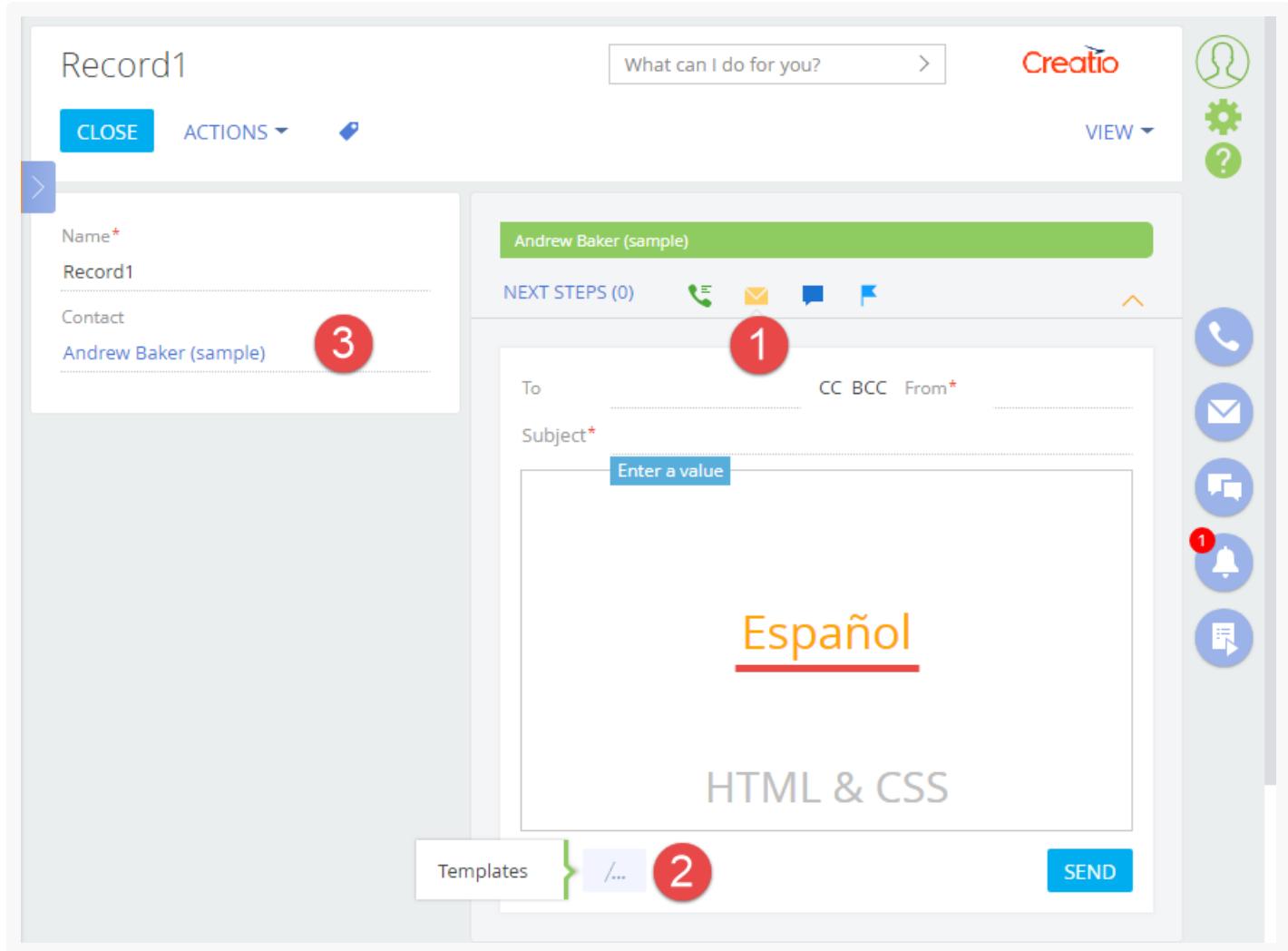


Рис. 7. — Предпочтительный язык контакта

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)

## Создать пользовательскую страницу действия верификации

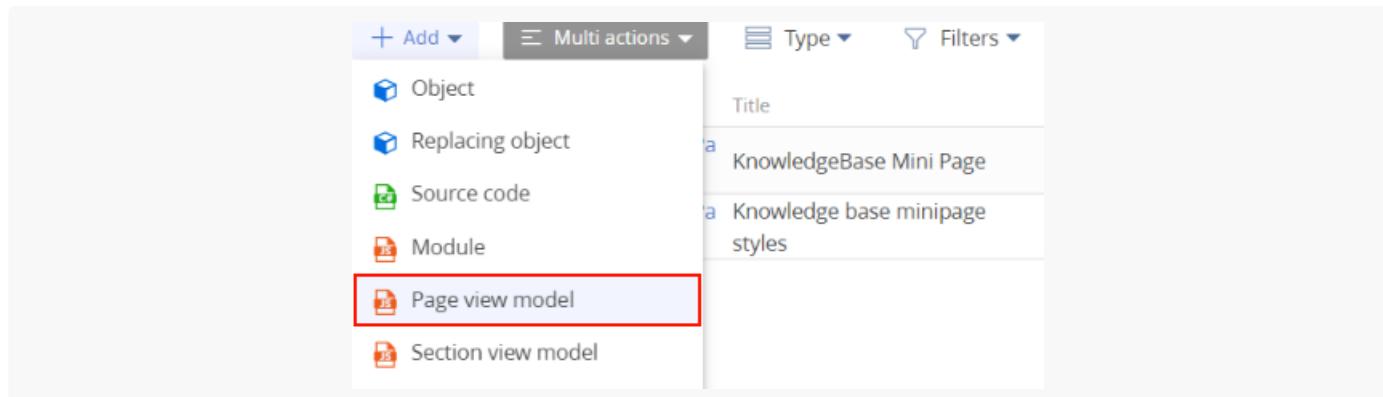
Сложный

**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](#).

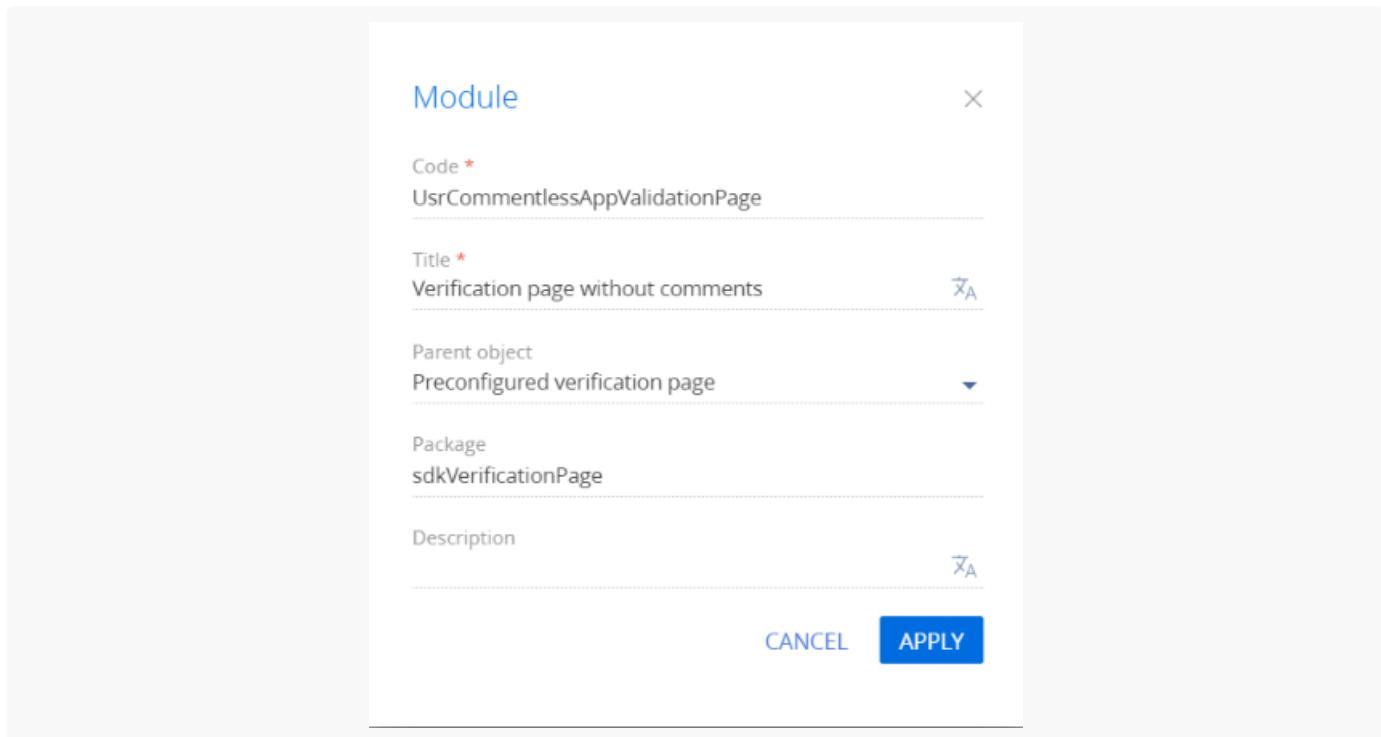
## 1. Создать схему страницы действия верификации

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.

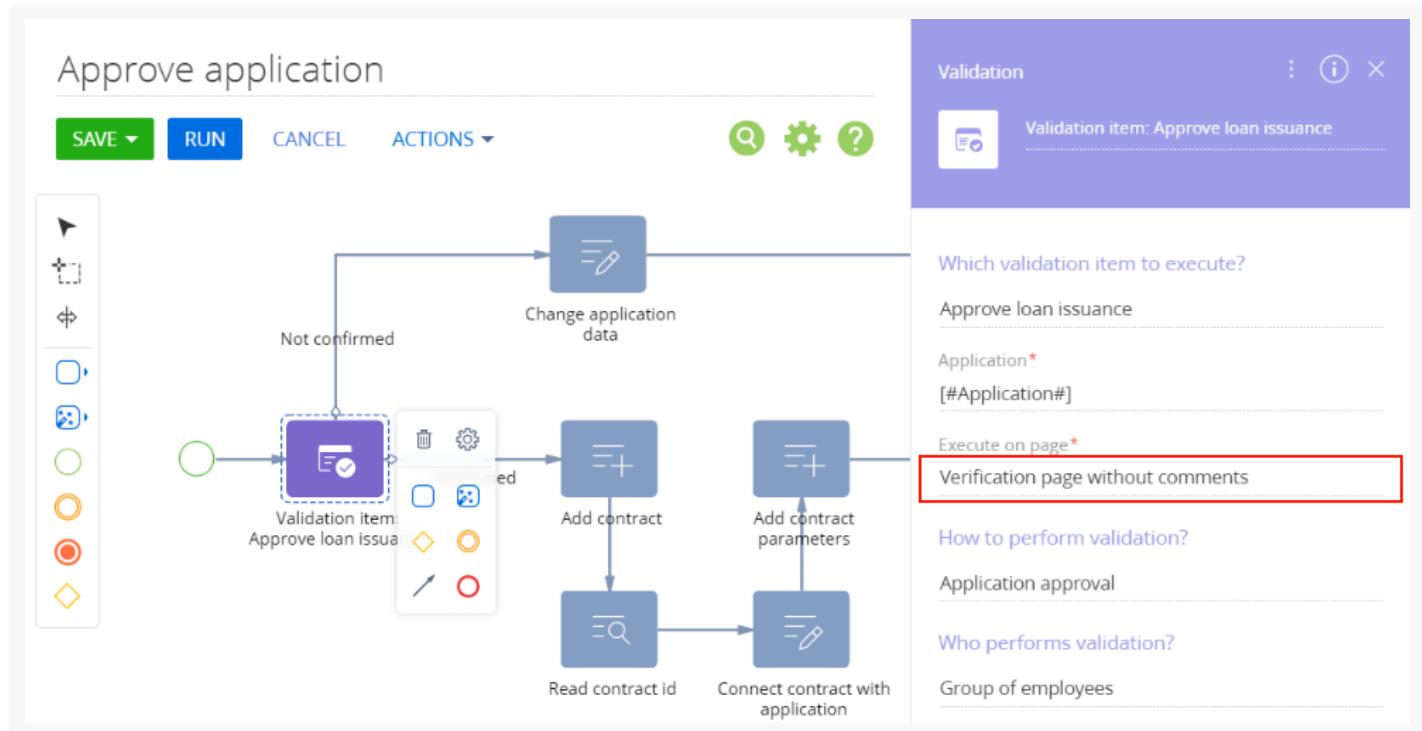
В массиве модификаций diff удалите из родительского элемента поле [ *Комментарий* ].

### **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`.

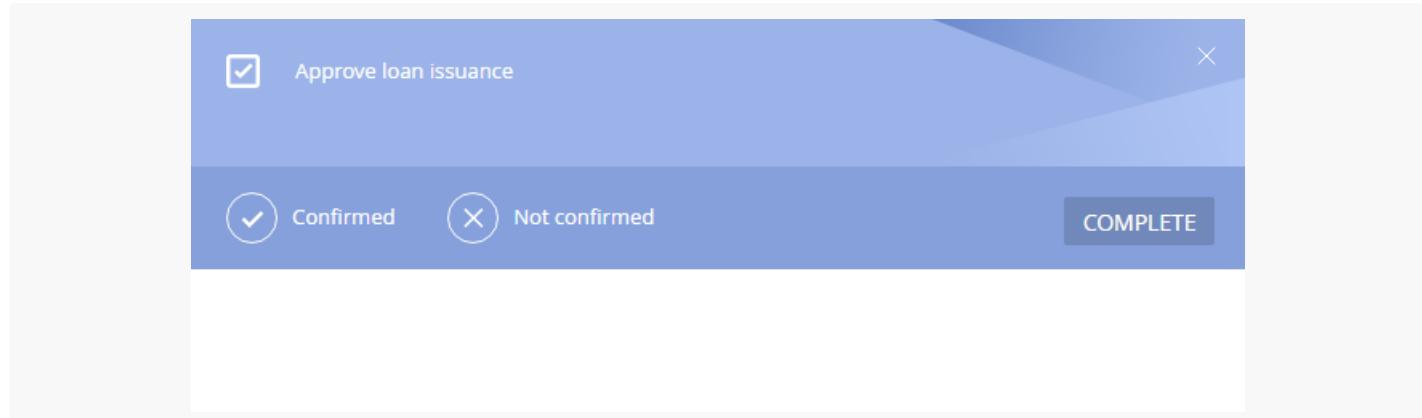


Для применения изменений бизнес-процесс необходимо сохранить.

**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].



# Раздел

**Раздел** — элемент интерфейса, который отражает определенную бизнес-сущность и содержит набор записей. Примерами раздела являются разделы [ Контрагенты ] ([ Accounts ]), [ Контакты ] ([ Contacts ]), [ Активности ] ([ Activities ]) и т. д. Разделы доступны в боковой панели, их можно сгруппировать в рабочие места для удобства работы отдельных ролей. Раздел описан в блоке статей [Раздел](#).

## Контейнеры раздела

Элементы пользовательского интерфейса приложения, которые относятся к разделу, размещены в соответствующих контейнерах. Контейнеры конфигурируются в базовой схеме раздела или схеме замещающей модели представления раздела. Контейнеры зависят от представления раздела.

**Представления** раздела:

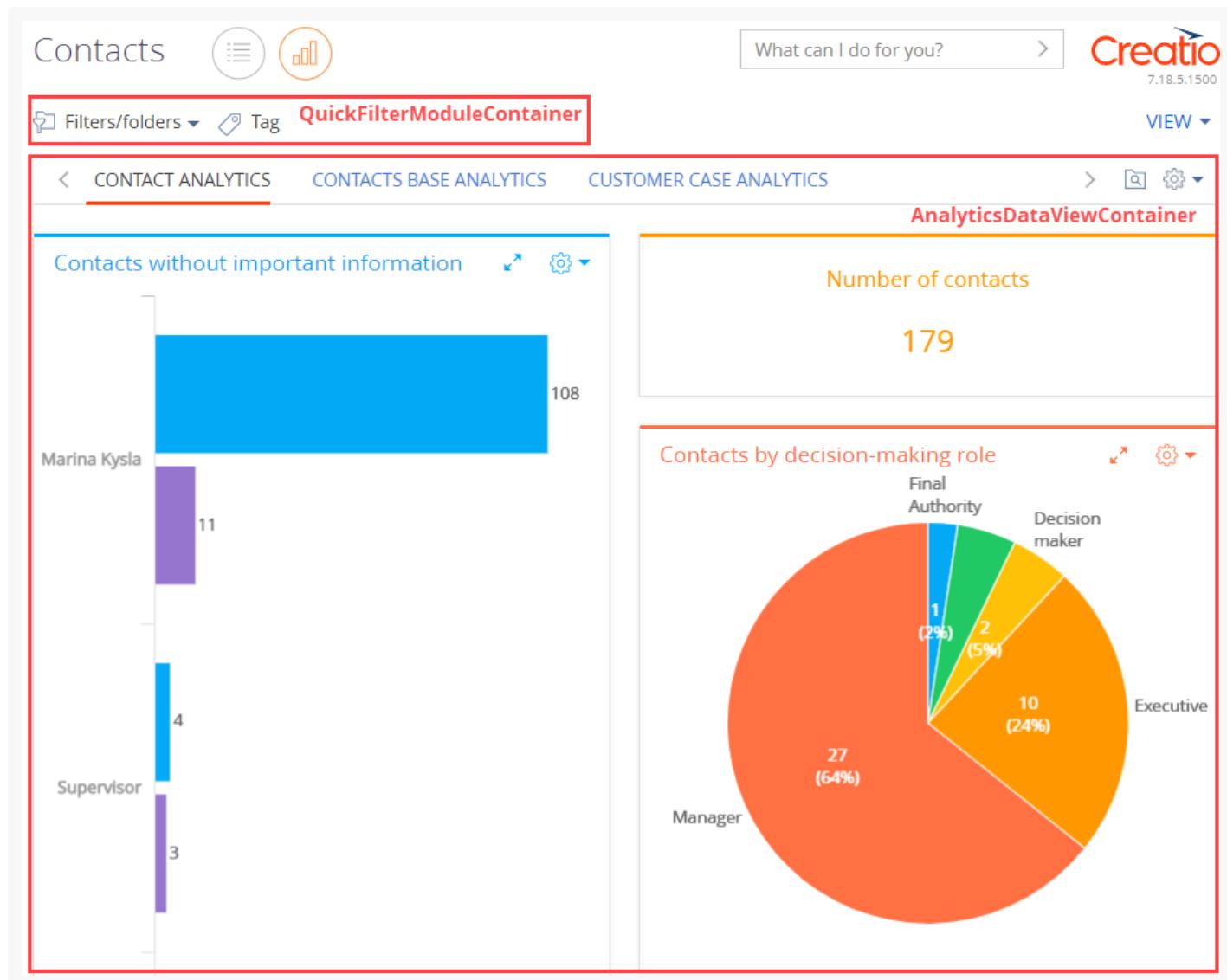
- Реестр раздела.
- Аналитика раздела.

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

Основные **контейнеры реестра** раздела представлены на рисунке ниже.

Alexander Wilson	Job title CEO	Business phone +1 212 542 4238
Account Alpha Business	Email a.wilson@alphabusiness.com	Mobile phone +1 212 854 7512
Alice Phillips	Job title CEO	Business phone +1 212 1440 5222
Account Axiom	Email alice.phillips@axiom.com	Mobile phone +1 212 1204 5477

Основные **контейнеры аналитики** раздела представлены на рисунке ниже.



- Контейнер кнопок действий ( `ActionButtonsContainer` ) — содержит кнопку действия раздела и кнопку с выпадающим списком действий.
- Контейнер фильтров ( `QuickFilterViewContainer` ) — содержит фильтры и теги.
- Контейнер отображения реестра раздела ( `Grid DataViewContainer` ) — содержит записи раздела. В активной записи реестра размещаются кнопки действий редактирования, копирования и удаления текущей записи. Отображается в реестре раздела.
- Контейнер отображения аналитики раздела ( `Analytics DataViewContainer` ) — содержит дашборды раздела. Отображается в аналитике раздела.

## Структура раздела

### Составляющие раздела:

- **Реестр** — компонент, в котором в плиточном или списочном представлении отображается список записей раздела. Отображается в контейнере `Grid DataViewContainer`.
- **Аналитика** — компонент, который используется для визуализации статистических данных с

помощью графиков, единичных показателей или списков. Блоки итогов и пользовательские дашборды отображаются в контейнере `AnalyticsDataViewContainer`.

- **Действия** — функциональный элемент, который представляет набор операций над активной записью реестра раздела. Действия вызываются при помощи кнопок, которые размещены в контейнере `ActionButtonsContainer`, и в активной записи реестра раздела.
- **Фильтр** — инструмент для поиска и сегментации записей реестра по заданным условиям. Отображается в контейнере `QuickFilterViewContainer`.
- **Тег** — метка, которая используется для сегментации записей вручную. Как и фильтр, отображается в контейнере `QuickFilterViewContainer`.

## Реестр

**Реестр** — элемент интерфейса для отображения перечня записей, которые добавлены в раздел или на деталь. Например, перечень контактов в разделе [ Контакты ] ([ *Contacts* ]).

**Виды** реестра раздела:

- **Вертикальный реестр** — способ отображения реестра, при котором можно переключаться между записями, не закрывая страницу.
- **Редактируемый реестр** — реестр записей, которые предоставляет возможность редактирования записей в самом реестре, без перехода к их страницам.

**Представления** реестра раздела:

- **Плиточное представление** — отображает поля записи реестра в несколько строк. Установлено по умолчанию.
- **Списочное представление** — отображает записи в виде простой таблицы, в которой каждой записи соответствует одна строка. При этом последовательность расположения полей в списочном представлении может не совпадать с последовательностью расположения полей в плиточном представлении.

Плиточное представление реестра раздела [ Контакты ] ([ *Contacts* ]) представлено на рисунке ниже.

Реестр			
	Alexander Wilson Account Alpha Business	Job title CEO  Email a.wilson@alphabusiness.com	Business phone +1 212 542 4238  Mobile phone +1 212 854 7512
	Alice Phillips Account Axiom	Job title CEO  Email alice.phillips@axiom.com	Business phone +1 212 1440 5222  Mobile phone +1 212 1204 5477
<a href="#">OPEN</a> <a href="#">COPY</a> <a href="#">DELETE</a>			

Списочное представление реестра раздела [ Контакты ] ([ Contacts ]) представлено на рисунке ниже.

The screenshot shows the 'Contacts' section of the Creatio application. At the top, there are icons for 'New Contact' (green), 'Actions' (blue dropdown), and 'View' (blue dropdown). A search bar says 'What can I do for you?'. The top right corner displays the 'Creatio' logo and the version '7.18.5.1500'. The main area is titled 'Реестр' (Registry) and contains a table with columns: 'Contact name', 'Account', 'Job title', 'Business phone', 'Mobile phone', and 'Email'. Two rows of data are shown:

Contact name	Account	Job title	Business phone	Mobile phone	Email
Nora Wesley	Gtech	Head of department	+1 213829 58 33		TyroneRigg@hotmail.com Запись реестра
Winter Hodge	Console Solutions	Sales manager	+1 212 566 84 89		winterhodge@gmail.com

At the bottom left are buttons for 'OPEN', 'COPY', and 'DELETE'. At the bottom right is the text 'Активная запись реестра' (Active registry record).

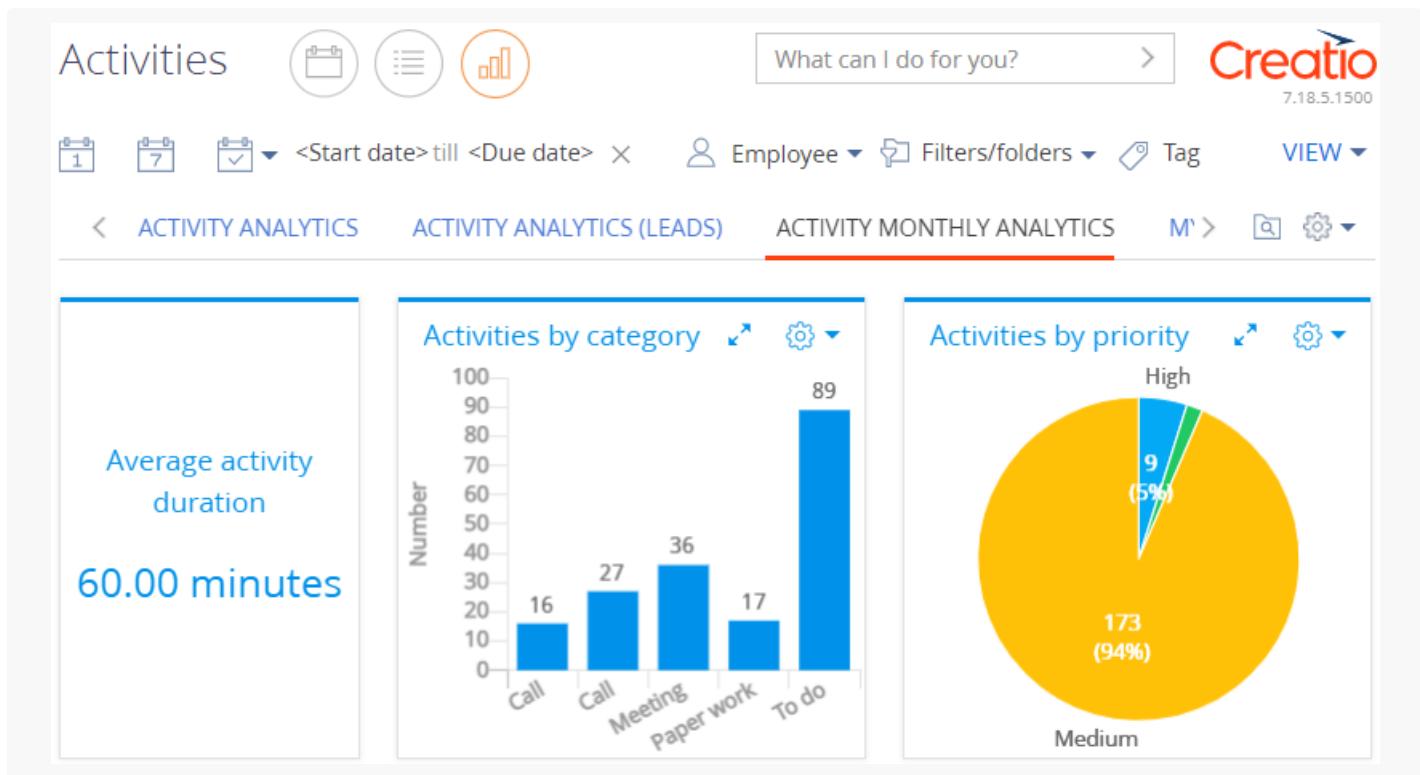
Каждому разделу соответствует конфигурационная схема, которая описывает структуру соответствующей таблицы базы данных с данными записи и содержит определенные инструкции по обработке этих данных. Например, конфигурирование раздела [ Контакты ] ([ Contacts ]) выполняется в схеме `ContactSectionV2`. Все схемы разделов являются наследниками базовой схемы `BaseSectionV2` раздела. Условно каждая строка таблицы соответствует записи реестра раздела. Например, разделу [ Контакты ] ([ Contacts ]) соответствует схема бизнес-объекта `Contact`, которая содержит перечень и свойства колонок таблицы [Contact]. По умолчанию, реестр раздела отображает не все колонки соответствующей таблицы базы данных. Это необходимо для избежания избыточности отображаемых данных. Посмотреть все колонки можно на [странице записи](#) раздела.

Внешний вид реестра, перечень отображаемых полей и сортировку данных позволяет настроить выпадающее меню кнопки [ Вид ] ([ View ]) панели инструментов раздела. Подробнее читайте в статье [Реестр раздела](#).

## Аналитика

**Назначение** аналитики раздела — анализ статистических данных разделов. По умолчанию раздел отображается в представлении реестра. Чтобы **перейти в представление аналитики раздела**, нажмите на кнопку [ Итоги ] ([ Dashboards ]) возле названия раздела. Чтобы **перейти в представление аналитики всех разделов**, откройте раздел [ Итоги ] ([ Dashboards ]).

Пример аналитики раздела [ Активности ] ([ Activities ]) представлены на рисунке ниже.



Аналитическая информация отображается на [дашбордах](#). Дашборды описаны в статье [Дашборды](#) и блоке статей [Аналитика](#). Дашборды размещены на [панели итогов](#). Панель итогов описана в статье [Добавить аналитику в раздел](#).

## Действия

**Действия раздела** — функциональные элементы, которые представляют собой перечень операций с одной или несколькими записями реестра. Действие можно вызвать при помощи кнопок разного вида, которые размещены как в контейнере действий текущего раздела, так и в контейнере активной записи.

Действия раздела [Контакты] ([*Contacts*]) представлены на рисунке ниже.

The screenshot shows the Creatio Contacts module interface. At the top, there's a navigation bar with 'Contacts' (highlighted in blue), search bar, and 'Creatio 7.18.5.1500' logo. Below is a list of contacts with their names, accounts, and email addresses. A context menu is open over the contact 'Andrew Barber' (Email: a.barber@gros.com). The menu is titled 'ACTIONS' and includes options like 'Synchronize contacts', 'Select multiple records', 'Select all', 'Export to Excel', 'Data import', 'Change log setup', 'Add to folder', 'Exclude from folder', 'Show duplicate 'Contacts'', 'Show on map', 'Update the values in the 'Age' column', 'Schedule daily update of the 'Age' column', and 'Mark all email addresses as valid'. The bottom of the menu has buttons for 'OPEN', 'COPY', and 'DELETE'. The entire 'ACTIONS' menu is highlighted with a red box.

Contact Name	Account	Email
Alexander	Alpha Business	alexander@alpha.com
Alice Phillips	Axiom	alice.phillips@axiom.com
Andrew Baker	Accom (sample)	andrew.baker@accom.com
Andrew V.	Apex Solutions	andrew.v@apex.com
Andrew Z.	Infocom	andrew.z@infocom.com

### **Виды действий раздела:**

- Стандартные.
- Дополнительные.

#### **Стандартные** действия раздела [ Контакты ] ([ Contacts ]):

- [ Добавить контакт ] ([ New contact ]) — вызывает всплывающее окно для добавления и сохранения новой записи раздела.
- [ Открыть ] ([ Open ]) — открывает страницу активной записи раздела.
- [ Копировать ] ([ Copy ]) — открывает страницу записи раздела, копирует в нее данные активной записи и при сохранении создает новую запись.
- [ Удалить ] ([ Delete ]) — удаляет активную запись.
- [ Выбрать несколько записей ] ([ Select multiple records ]) — выполняет множественный выбор записей реестра.
- [ Выбрать все ] ([ Select all ]) — выполняет выбор всех записей реестра.
- [ Экспорт в Excel ] ([ Export to Excel ]) — экспортирует все записи реестра текущего раздела в \*.xlsx-файл. Экспорт данных описан в статье [Экспорт в Excel](#).
- [ Импорт данных ] ([ Data import ]) — импортирует в Creatio данные из \*.xlsx-файла. Импорт данных

описан в статье [Импорт из Excel](#).

- [ Настроить журнал изменений ] ([ *Change log setup* ]) — открывает страницу управления логированием и выбора колонок раздела, которые будут логироваться при изменении записи.

**Дополнительные** действия реализуют функциональность, которая зависит от бизнес-логики раздела. Дополнительные действия раздела [ Контакты ] ([ *Contacts* ]):

- [ Запустить синхронизацию ] ([ *Synchronize now* ]) — выполняет синхронизацию Creatio с контактами Google. Синхронизация контактов из Creatio в Google выполняется только для записей, которые отмечены тегом, указанным в настройках синхронизации. Синхронизация контактов описана в статье [Синхронизировать контакты и активности с Google](#).
- [ Добавить аккаунт для синхронизации ] ([ *Add new account for synchronization* ]) — выполняет синхронизацию с Google для Creatio Cloud. Синхронизация описана в статье [Синхронизировать контакты и активности с Google](#).
- [ Поместить в группу ] ([ *Add to folder* ]) — вызывает всплывающее окно для выбора группы, в которую необходимо поместить активную запись.
- [ Исключить из группы ] ([ *Exclude from folder* ]) — исключить активную запись из всех групп, в которую она входит.
- [ Перейти к дублям раздела 'Контакты' ] ([ *Show duplicate 'Contacts'* ]) — открывает дополнительную страницу, которая содержит все предполагаемые дубли контактов. Записи добавляются на эту страницу автоматически после выполнения поиска дублей.
- [ Показать на карте ] ([ *Show on map* ]) — позволяет отобразить на карте местоположение активных контактов. По действию открывается окно с картой, на которой отмечены выбранные в реестре контакты. Если для всех выбранных контактов адрес не заполнен, то действие не выполняется. Если для некоторых контактов адрес не заполнен или заполнен некорректно, то в окне отображается соответствующая информация.
- [ Обновить возраст ] ([ *Update the values in the 'Age' column* ]) — обновляет значения в колонке [ Возраст ] ([ *Age* ]) на странице активного контакта.
- [ Настроить время обновления возраста ] ([ *Schedule daily update of the 'Age' column* ]) — вызывает всплывающее окно для настройки времени ежедневного обновления информации о возрасте контакта.
- [ Снять признак 'Неактуальный' у email-адресов ] ([ *Mark all email addresses as valid* ]) — устанавливает признак [ Является актуальным ] ([ *Valid* ]) email-адресу активной записи контакта.

## Фильтры

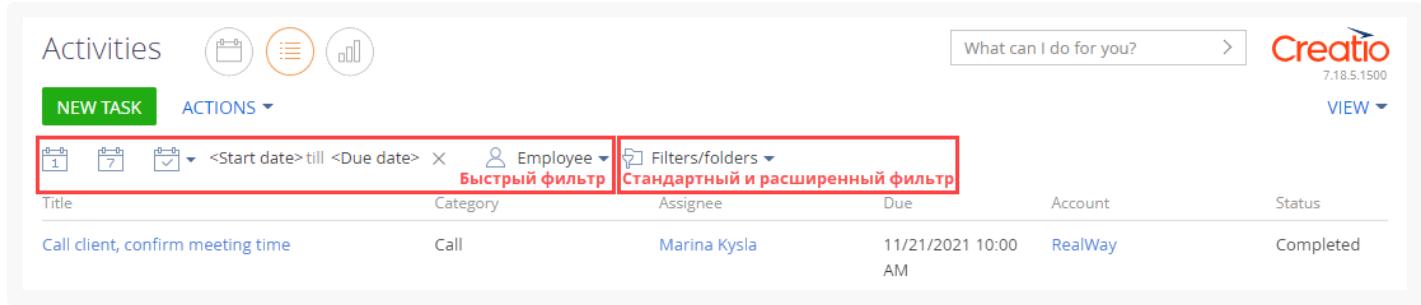
**Фильтр** — инструмент для поиска и сегментации записей системы по заданным условиям. Фильтры могут использоваться как самостоятельный инструмент в разделах Creatio. Также инструменты фильтрации применяются при выполнении других настроек, например, при настройке динамических групп, дашбордов, бизнес-процессов. Элементы управления фильтрами отображаются над реестром разделов приложения. Фильтры описаны в статье [Фильтры](#).

**Виды** фильтров:

- Быстрый фильтр.
- Расширенный фильтр.

- Стандартный фильтр.

Управление быстрым фильтром осуществляется из панели инструментов, а управление стандартным и расширенным фильтрами — из меню [ Фильтры/группы ] ([ *Filters/folders* ]).



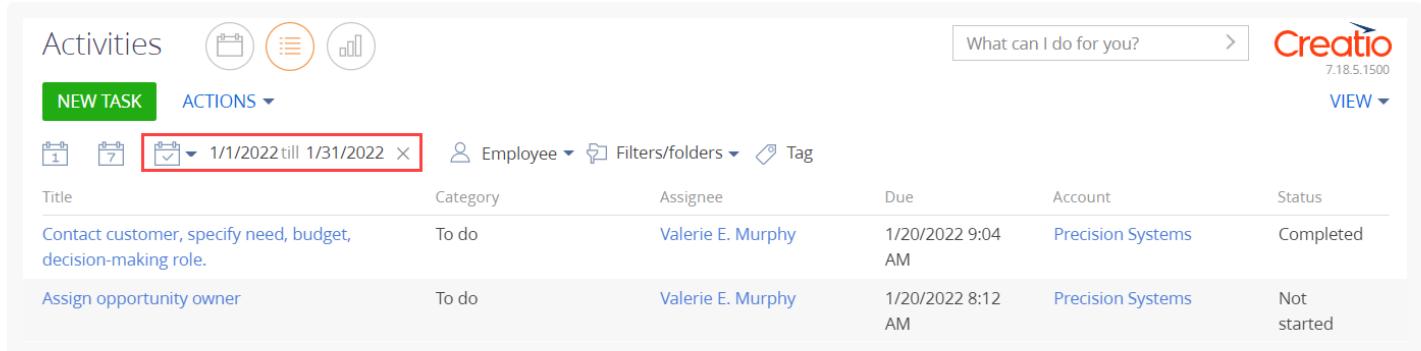
The screenshot shows the 'Activities' section of the Creatio interface. At the top, there are three circular icons: a document with a checkmark, a grid, and a bar chart. Below them are buttons for 'NEW TASK' and 'ACTIONS'. A search bar says 'What can I do for you?'. On the right, the 'Creatio 7.18.5.1500' logo and a 'VIEW' button are visible. The main area displays a table with columns: Title, Category, Assignee, Due, Account, and Status. A row for a task titled 'Call client, confirm meeting time' is shown. Above the table, there are two dropdown menus: 'Employee' and 'Filters/folders'. The 'Filters/folders' menu has two items: 'Быстрый фильтр' (Quick filter) and 'Стандартный и расширенный фильтр' (Standard and extended filter), with the latter also highlighted by a red box.

**Быстрый фильтр** — фильтр, который используется в реестре некоторых разделов Creatio для поиска данных по наиболее часто используемым параметрам (обычно — по периоду и ответственному).

Например, быстрый фильтр присутствует в разделе [ Активности ] ([ *Activities* ]), поскольку чаще всего нужно просматривать активности одного сотрудника за указанный период времени. Быстрый фильтр описан в статье [Фильтры](#).

Чтобы **установить быстрый фильтр**, в выпадающем меню [ Выбрать период ] ([ *Select period* ]) выберите период, по которому планируется отфильтровать записи реестра раздела.

Пример быстрого фильтра, который фильтрует активности по текущему месяцу (в выпадающем меню [ Выбрать период ] ([ *Select period* ]) выбрано значение [ Текущий месяц ] ([ *Current month* ])), представлен на рисунке ниже.



This screenshot shows the same 'Activities' interface as above, but the 'Filters/folders' dropdown menu is now open, revealing several options: 'Быстрый фильтр' (Quick filter), 'Стандартный и расширенный фильтр' (Standard and extended filter), 'Tag', and 'Filters/folders'. The 'Standard and extended filter' option is highlighted with a red box. The rest of the interface and data table are identical to the previous screenshot.

**Стандартный фильтр** — фильтр, который используется для быстрого поиска записей по значениям одной или нескольких колонок текущего раздела. Доступен в реестре большинства разделов.

Стандартный фильтр описан в статье [Фильтры](#).

Чтобы **установить стандартный фильтр**:

- В выпадающем меню [ Фильтры/группы ] ([ *Filters/folders* ]) выберите пункт [ Добавить условие ] ([ *Add filter* ]).
- Выберите поле и значение поля, по которому планируется отфильтровать записи реестра раздела.

Пример стандартного фильтра, который фильтрует активности по значению контрагента (для поля [ Контрагент ] ([ *Account* ]) выбрано значение [ *Estron* ]), представлен на рисунке ниже.

The screenshot shows the 'Activities' section of the Creatio application. At the top, there are three icons: a calendar, a grid, and a bar chart. Below them are buttons for 'NEW TASK' and 'ACTIONS'. A search bar contains the placeholder 'What can I do for you?'. In the top right corner, the 'Creatio' logo is followed by the version '7.18.5.1500' and a 'VIEW' button. The main area displays a table with columns: Title, Category, Assignee, Due, Account, and Status. A single row is shown: 'Visit: Ted Heinrichs, Estron' (Title), 'Meeting' (Category), 'Peter Moore' (Assignee), '12/18/2021 2:30 PM' (Due), 'Estron' (Account), and 'Not started' (Status). Above the table, a filter bar shows 'Employee' and 'Account = Estron' with a red box highlighting the account filter.

**Расширенный фильтр** — фильтр, который состоит из нескольких параметров и сложных условий поиска. Например, при помощи расширенной фильтрации вы можете отобразить в разделе [ Активности ] ([ Activities ]) все встречи по новым клиентам. Также, настроив условия расширенного фильтра, можно сохранить динамическую группу для дальнейшего использования. Расширенный фильтр описан в статье [Фильтры](#).

Чтобы **установить расширенный фильтр**:

- В выпадающем меню [ Фильтры/группы ] ([ Filters/folders ]) выберите пункт [ Перейти в расширенный режим ] ([ Switch to advanced mode ]).
- Выберите объект, колонку объекта и значение колонки, по которой планируется отфильтровать записи реестра раздела.

Пример расширенного фильтра, который фильтрует активности по значению контрагента (для поля [ Название ] ([ Name ]) объекта [ Контрагент ] ([ Account ]) выбрано значение [ Novelty ]) и по значению должности (для поля [ Должность ] ([ Job title ]) объекта [ Контакт ] ([ Contact ]) выбрано значение [ Специалист ] ([ Specialist ])), представлен на рисунке ниже.

The screenshot shows the 'Activities' section with the 'Advanced Mode' filter configuration open. On the left, a sidebar allows setting conditions using 'AND' or 'OR'. Two conditions are selected: 'Account.Name = Novelty' and 'Contact.Job title = Specialist'. On the right, the results table shows four tasks: 'Contact customer, specify need, budget, decision-making role.' assigned to Megan Lewis, 'Contact customer, specify need, budget, decision-making role.' assigned to Megan Lewis, and 'Specify contract terms' assigned to Marina Kysla. The first two tasks are marked as 'Completed' and the third as 'Not started'.

**Группа** — инструмент фильтрации записей путем объединения их по определенным условиям. Группы могут формировать иерархическую структуру. Группы описаны в статье [Группы](#).

**Виды групп:**

- **Динамическая группа** — группа, содержащая только те записи раздела, которые соответствуют заданным условиям фильтрации. Например, динамической может быть группа [ Новые клиенты ] ([ New client ]), поскольку отобрать записи в такую группу можно, используя фильтр по дате создания записи в системе. Наполнение такой группы формируется и обновляется автоматически.
- **Статическая группа** — группа, содержащая только те записи раздела, которые были добавлены в нее пользователем (вручную или при конвертации из динамической группы). Например, статическими может быть группа [ VIP-клиенты ] ([ VIP Customers ]), поскольку решение о включении клиента в

такую группу принимается менеджером или руководителем лично.

Включение записи в группу, как и исключение из нее, возможно только для статических групп. В динамической группе запись отображается автоматически, если она отвечает условиям фильтрации группы. Если запись не соответствует условиям фильтра группы, то она автоматически исключается из группы.

Чтобы **установить группу**:

- В выпадающем меню [ Фильтры/группы ] ([ *Filters/folders* ]) выберите пункт [ Показать группы ] ([ *Show folders* ]).
- Выберите группу, по которой планируется отфильтровать записи реестра раздела.

Пример группы контактов без контрагентов (группа [ Контакты без контрагента ] ([ *Contact not connected to accounts* ])), представлен на рисунке ниже.

Job title	Email	Business phone
Specialist	RalphPulido@gmail.com	+1 31172 416 84 27
Specialist	RaulParson@gmail.com	+1 30776 548 96 64
Specialist	RayCrowden@gmail.com	+1 31839 207 95 94

## Теги

**Тег** — специальная метка, которую можно использовать для сегментации записей вручную. Например, тегировав записи в разделе [ Контакты ] ([ *Contacts* ]), можно выделить VIP-клиентов или определить черный список клиентов. Тегирование записей выполняется вручную. Чтобы **выбрать тег**, нажмите на кнопку [ Tag ] ([ *Tag* ]) и выберите тег, по которому планируется отфильтровать записи реестра раздела. Теги описаны в статье [Теги](#).

**Виды** тегов, которые предоставляет Creatio:

- **Личные** — теги, которые может видеть и использовать только тот сотрудник, который их создал. Любой пользователь системы может создать необходимое количество личных тегов. Ни администраторы системы, ни руководители не смогут увидеть личные теги сотрудников. Личные теги в Creatio отображаются зеленым цветом.
- **Корпоративные** — теги, которые отображаются для всех сотрудников компании. Любой сотрудник может установить или снять корпоративный тег. Создавать новые корпоративные теги могут все сотрудники/роли, которым предоставлено право на операцию [ Управление корпоративными тегами ] ([ *Corporate tags management* ], Код `CanManageCorporateTags`). Корпоративные теги в Creatio отображаются голубым цветом.

- **Публичные** — теги, которые отображаются для всех сотрудников компании, а также для пользователей портала самообслуживания. Любой сотрудник может установить или снять публичный тег. Создавать новые публичные теги могут все сотрудники/роли, которым предоставлено право на операцию [ Управление публичными тегами ] ([ *Public tags management* ], код `CanManagePublicTags`). Публичные теги в Creatio отображаются красным цветом.

Пример VIP-клиентов, которыми являются контакты (выбран корпоративный тег [ *VIP* ] ([ *VIP customer* ])), представлен на рисунке ниже.

The screenshot shows the Creatio Contacts module interface. At the top, there are navigation buttons for 'NEW CONTACT' and 'ACTIONS'. Below the header, there are filters for 'Filters/folders' and a selected tag 'VIP customer'. The main content area displays a contact record for 'Alexander Wilson'. The contact's profile picture is shown, along with their name and title ('Account' at 'Alpha Business'). To the right, detailed information is provided: Job title (CEO), Email (a.wilson@alphabusiness.com), Business phone (+1 212 542 4238), and Mobile phone (+1 212 854 7512).

## Создать раздел

Создать раздел можно с использованием мастера разделов. Чтобы создать раздел, воспользуйтесь инструкцией, которая приведена в статье [Добавить новый раздел](#).

Схемы объектов и клиентские схемы, которые создаются в пользовательском пакете при создании раздела, приведены в таблице ниже.

Схемы, которые создаются мастером разделов

Название	Назначение	Родительский объект
<b>Схемы объектов</b>		
[Имя объекта раздела]	Основной объект раздела.	Базовый объект <code>BaseEntity</code> .
[Имя объекта раздела]Folder	Группа объекта. Служебный объект для корректной группировки записей в разделе. Формирует общую структуру дерева групп раздела.	Базовая группа <code>BaseFolder</code> .
[Имя объекта раздела]InFolder	Объект в группе. Служебный объект для корректной работы группировки записей в разделе. Определяет связи между записями раздела и группами, в которые они входят.	Базовый элемент <code>BaseItemInFolder</code> в группе.
[Имя объекта раздела]File	Объект для детали [Файлы и ссылки] ([Attachments]).	Файл <code>File</code> .
[Имя объекта раздела]Tag	Тег раздела.	Базовый тег <code>BaseTag</code> .
[Имя объекта раздела]InTag	Тег в объекте раздела.	Базовый тег в базовом объекте <code>BaseEntityInTag</code> .
<b>Клиентские схемы</b>		
[Имя объекта раздела]Section	Схема раздела.	Базовая схема <code>BaseSectionV2</code> раздела.
[Имя объекта раздела]Page	Схема страницы записи раздела.	Базовая схема <code>BaseModulePageV2</code> страницы записи раздела.

## Добавить пользовательскую колонку в реестр раздела

Добавление пользовательской колонки в реестр раздела значит, что колонка добавляется в схему бизнес-объекта раздела. При добавлении пользовательской колонки в текущем пакете создается схема

замещающей модели представления схемы, которая унаследует колонки базового объекта. В созданную схему будет добавлена новая пользовательская колонка.

**Инструменты**, которые позволяют добавить пользовательскую колонку в схему бизнес-объекта раздела:

- Мастер разделов.
- Creatio IDE.

Чтобы добавить пользовательскую колонку в реестр раздела с использованием **мастера разделов**, воспользуйтесь инструкцией, которая приведена в статье [Реестр раздела](#).

Чтобы добавить пользовательскую колонку на страницу раздела с использованием **Creatio IDE**:

1. Создайте схему замещающей модели представления страницы раздела, на которой будет размещена пользовательская колонка. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В качестве родительского объекта выберите схему модели представления, которую необходимо заместить.
3. В схему замещающей модели представления страницы раздела добавьте пользовательскую колонку соответствующего типа.
4. Настройте отображение пользовательской колонки в реестре раздела.

## Настроить условия отображения реестра раздела

Creatio позволяет настроить условия отображения реестра раздела. Записи реестра, для которых выполняются заданные условия, отображаются в соответствии с установленными стилями.

Чтобы **настроить условия отображения реестра раздела**:

1. Создайте схему замещающей модели представления страницы раздела. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В качестве родительского объекта выберите схему модели представления, которую необходимо заместить.
3. Настройте **условия отображения реестра раздела**. Для этого в свойстве `methods` реализуйте метод `prepareResponseCollectionItem()`, который переопределяет базовый метод и модифицирует строку данных перед загрузкой в реестр. В методе реализуйте присвоение определенного значения свойству `customStyle`, которое отвечает за отображение строки реестра. Является объектом, который содержит свойства-аналоги CSS-свойств и формирует стиль отображения необходимых записей реестра.

Пример настройки свойства `customStyle` представлен ниже.

### Пример настройки свойства `customStyle`

```
item.customStyle = {
    /* Цвет текста – белый. */
    "color": "white",
    /* Цвет фона – оранжевый. */
}
```

```
"background": "orange"
};
```

## Добавить действие для записей раздела

Приложение предоставляет возможность добавить пользовательские действия как для единичной записи, так и для нескольких записей. Базовое наполнение выпадающего меню [Действия] ([Actions]) страницы раздела реализовано в схеме `BaseSectionV2` пакета `NUI`.

Чтобы **добавить действие для единичной записи раздела**:

1. Создайте схему замещающей модели представления страницы раздела. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В качестве родительского объекта выберите схему модели представления, которую необходимо заместить.
3. Перечень действий раздела является экземпляром класса `Terrasoft.ViewModelCollection`. Каждый элемент перечня действий — модель представления. Настройка действия выполняется в конфигурационном объекте, который позволяет как явно задавать свойства модели представления действий, так и использовать базовый механизм привязки.

Настройте **логику работы пункта меню**. Для этого в свойстве `methods` реализуйте метод `getSectionActions()` — переопределенный метод родительской схемы, который возвращает коллекцию действий раздела. При замещении базовых разделов в методе `getSectionActions()` замещающего модуля необходимо вызвать родительскую реализацию этого метода для инициализации действий родительского раздела. Для этого выполните метод `this.callParent(arguments)`, который возвращает коллекцию действий базового раздела.

Добавление действия в коллекцию выполняется с помощью вызова метода `addItem()`. В качестве параметра методу передается callback-метод `getButtonMenuItem()`. Метод создает экземпляр модели представления действия по конфигурационному объекту, который передан ему в качестве параметра.

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.

Значение первичных колонок выбранных записей хранится в свойстве `SelectedRows` модели представления раздела. Эти значения могут использоваться для получения значений, загруженных в реестр полей выбранных объектов, например, из коллекции данных списочного реестра, которая хранится в свойстве `GridData` модели представления реестра.

Базовая реализация добавления действия представлена ниже.

### BaseSectionV2

```
/**
 * Возвращает коллекцию действий раздела в режиме отображения реестра.
 * @protected
 * @virtual
```

```

* @return {Terrasoft.BaseViewModelCollection} Коллекция действий раздела.
*/
getSectionActions: function() {
    /* Коллекция действий. Экземпляр Terrasoft.BaseViewModelCollection. */
    var actionMenuItems = this.Ext.create("Terrasoft.BaseViewModelCollection");
    /* Добавление действия в коллекцию. В качестве callback-метода передается метод, который инс
actionMenuItems.addItem(
    this.getButtonItem({
        /* Конфигурационный объект настройки действия. */
        ...
    })
);
return actionMenuItems;
}

```

Свойства конфигурационного объекта действия раздела, который передается в качестве параметра в метод `getButtonItem()` представлены в таблице ниже.

Свойства конфигурационного объекта

Свойство	Описание
Type	Тип элемента выпадающего меню [ <i>Действия</i> ] ([ <i>Actions</i> ]). Это свойство позволяет добавить в меню действий горизонтальную линию для отделения блоков меню. Для этого в качестве значения свойства необходимо указать <code>Terrasoft.MenuSeparator</code> . Если значение свойства не указано, то по умолчанию добавляется пункт меню.
Caption	Заголовок пункта меню [ <i>Действия</i> ] ([ <i>Actions</i> ]). Для заголовков рекомендуется использовать <a href="#">локализуемые строки</a> .
Click	Привязка метода-обработчика действия по имени метода.
Enabled	Логическое свойство, которое регулирует доступность пункта меню.
Visible	Логическое свойство, которое регулирует видимость пункта меню.

## Настроить блок быстрых фильтров раздела

Приложение предоставляет возможность реализовать пользовательский блок быстрых фильтров для записей реестра раздела.

Чтобы **настроить блок быстрых фильтров раздела**:

1. Создайте схему замещающей модели представления страницы раздела. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В качестве родительского объекта выберите схему модели представления, которую необходимо заместить.

3. Реализуйте **логику работы фильтрации**. Для этого в свойстве `methods` реализуйте метод `initFixedFiltersConfig()`, который инициализирует фиксированные фильтры. В методе создайте конфигурационный объект с массивом фильтров, присвойте ссылку на объект атрибуту `fixedFiltersConfig` модели представления.

## Удалить раздел

- Проверьте наличие доступа к конфигурации системы и базе данных.
- Снимите блокировку с соответствующего удаляемому разделу файла в SVN-хранилище.
- Удалите раздел, созданный на основании объекта, который планируется удалить. Для этого выполните SQL-запрос.

### SQL-запрос

```
DECLARE @UID UNIQUEIDENTIFIER
DECLARE @ModuleEntityUID UNIQUEIDENTIFIER;
DECLARE @ModuleID UNIQUEIDENTIFIER;
DECLARE @Name NVARCHAR(max) = 'ToDelete';
select @UID = UId from SysSchema where Name Like @Name
select @ModuleEntityUID = Id from SysModuleEntity where
SysEntitySchemaUID = @UID
select @ModuleID = Id from SysModule where SysModuleEntityId = @ModuleEntityUID;
delete from SysModuleInWorkplace where SysModuleId = @ModuleID;
delete from SysModule where Id = @ModuleID;
delete from SysModuleEdit where SysModuleEntityId = @ModuleEntityUID;
delete from SysModuleEntity where Id = @ModuleEntityUID;
delete from SysDetail where EntitySchemaUID = @UID;
delete from SysLookup where SysEntitySchemaUID = @UID;
delete from [Lookup] where SysEntitySchemaUID = @UID;
```

Значение `ToDelete` замените на название схемы пользовательского раздела.

- После очистки базы данных удалите пользовательские схемы в разделе [ Конфигурация ] ([ Configuration ]).

**Порядок** удаления конфигурационных схем:

- `ToDeleteFile`.
- `ToDeleteInFolder`.
- `ToDeleteInTag`.
- `ToDeleteTag`.
- `ToDeleteFolder`.
- `ToDelete`.

# Добавить в раздел действие для единичной записи

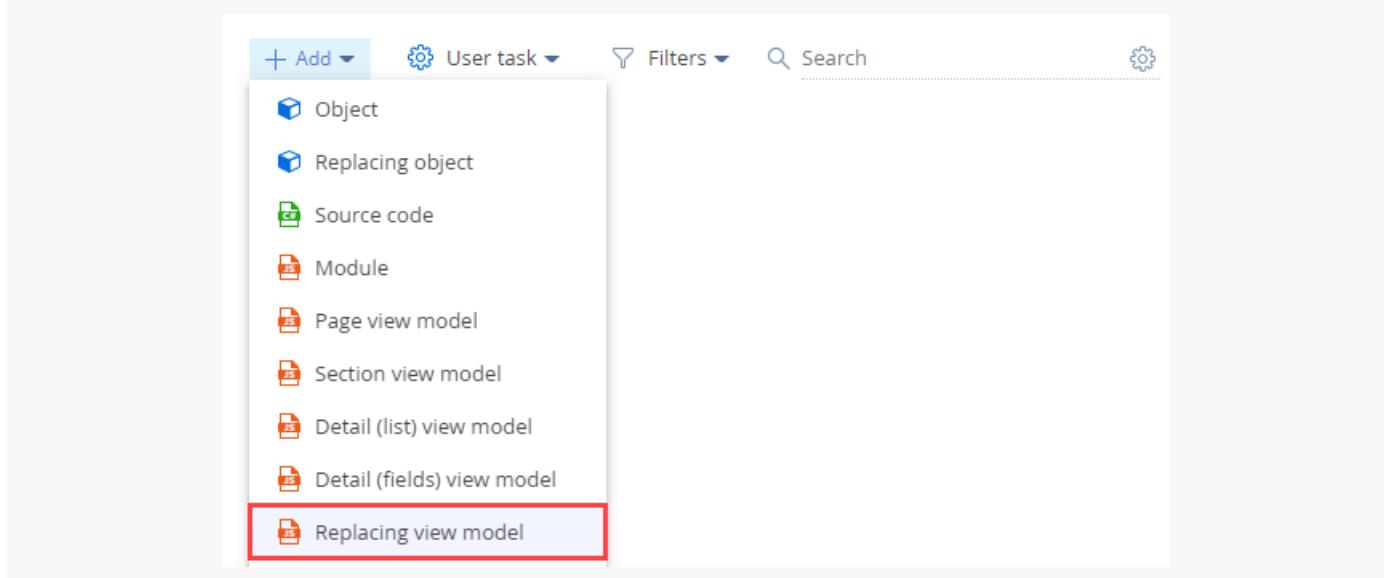
 Средний

Пример реализован для продуктов линейки Sales Creatio.

**Пример.** Для записи реестра раздела [ Заказы ] ([ Orders ]) добавить действие, которое в информационном окне отображает дату создания заказа. Действие активно для заказов, которые находятся на стадии [ Исполнение ] ([ In progress ]).

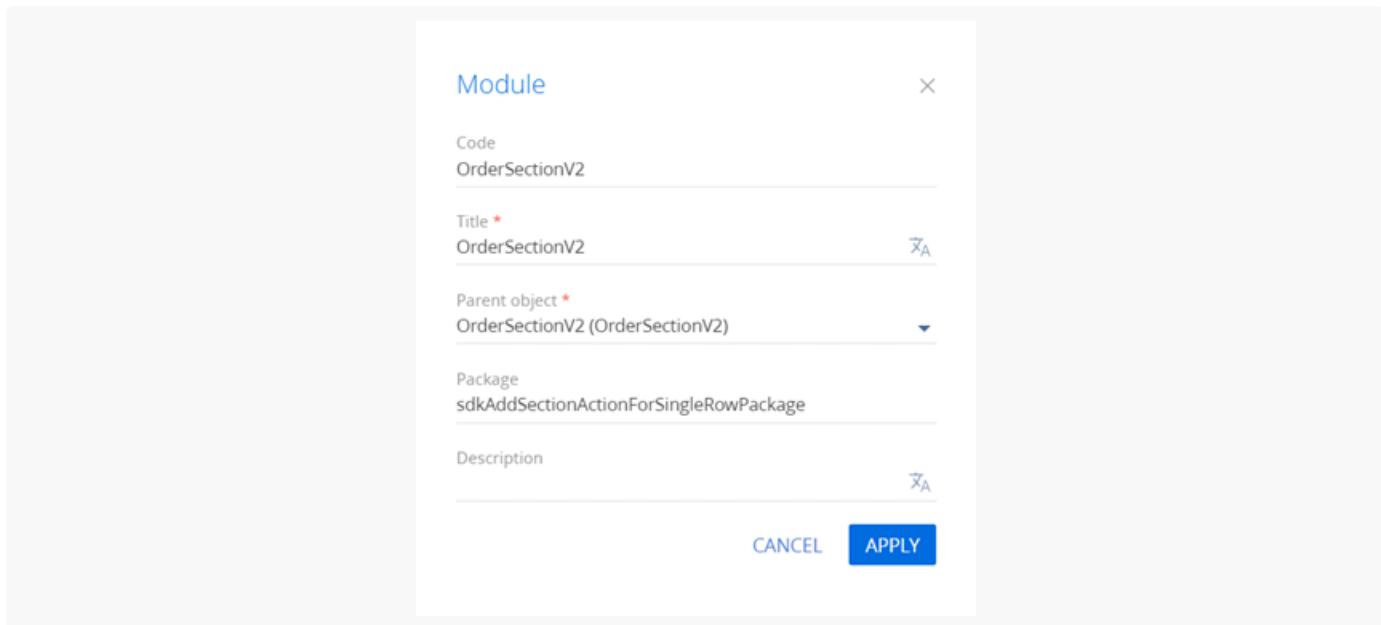
## Создать схему замещающей модели представления раздела

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 ]).



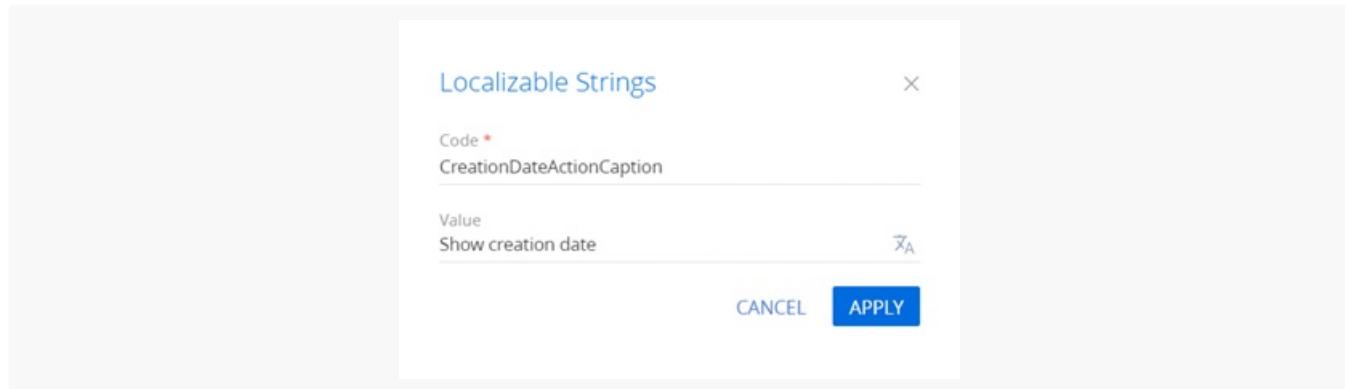
3. Заполните **свойства схемы**.

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



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* ]) — "CreationDateActionCaption".
  - [ *Значение* ] ([ *Value* ]) — "Показать дату создания" ("Show creation 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 выбранный в реестре заказ на стадии [ *Исполнение* ] ([ *In progress* ]).
- `isCustomActionEnabled()` — определяет доступность добавленного пункта меню.
- `showOrderInfo()` — метод-обработчик действия, который отображает в информационном окне дату создания выбранного заказа.
- `getSectionActions()` — переопределенный метод родительской схемы, который возвращает коллекцию действий раздела.

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 раздела представлен ниже.

### OrderSectionV2

```
define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
    return {
        /* Название схемы объекта раздела. */
        entitySchemaName: "Order",
        /* Методы модели представления раздела. */
        methods: {
            /* Проверяет стадию заказа. activeRowId – значение первичной колонки выделенной з
            isRunning: function(activeRowId) {
                /* 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 f
                return selectedOrderStatus.value === OrderConfigurationConstants.Order.OrderS
            },
            /* Определяет доступность пункта меню. */
            isCustomActionEnabled: function() {
                /* Попытка получения идентификатора активной записи. */
                var activeRowId = this.get("ActiveRow");
                /* Если идентификатор определен и статус выбранного заказа [Исполнение], то в
                return activeRowId ? this.isRunning(activeRowId) : false;
            },
            /* Метод-обработчик действия. Отображает в информационном окне дату создания зака
            showOrderInfo: function() {
                var activeRowId = this.get("ActiveRow");
                var gridData = this.get("GridData");
                /* Получает дату создания заказа. Колонка должна быть добавлена в реестр. */
                var dueDate = gridData.get(activeRowId).get("Date");
                /* Отображает информационное окно. */
                this.showInformationDialog(dueDate);
            },
            /* Переопределение базового виртуального метода, который возвращает коллекцию дей
            getSectionActions: function() {
                /* Вызывается родительская реализация метода для получения коллекции пропри
                var actionMenuItems = this.callParent(arguments);
                /* Добавляет линию-разделитель. */
                actionMenuItems.addItem(this.getButtonMenuItem({

```

```

        Type: "Terrasoft.MenuSeparator",
        Caption: ""
    }));
    /* Добавляет пункт меню в список действий раздела. */
    actionMenuItems.addItem(this.getButtonItem({
        /* Привязка заголовка пункта меню к локализуемой строке схемы. */
        "Caption": {bindTo: "Resources.Strings.CreationDateActionCaption"},
        /* Привязка метода-обработчика действия. */
        "Click": {bindTo: "showOrderInfo"},
        /* Привязка свойства доступности пункта меню к значению, которое возвращается
        "Enabled": {bindTo: "isCustomActionEnabled"}
    }));
    /* Возврат дополненной коллекции действий раздела. */
    return actionMenuItems;
}
}
);

```

- 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 действие [ Показать дату создания ] ([ Show creation date ]).

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

**ACTIONS ▾**

- Select multiple records
- Select all
- Export to Excel
- Data import
- Change log setup
- Send for approval
- Show creation date

Account	Contact	Owner
Apex Solutions	Andrew Wayne	Marina Kysla
Apex Solutions	Andrew Wayne	Symon Clarke
Streamline Development	Bruce Clayton	Marina Kysla
Gateway	James Smith	Marina Kysla
Infocom	Andrew Z. Barber	Symon Clarke
Alpha Business	Jordan Anderson	Marina Kysla
Factorial Services	Taylor P. Johnson	Symon Clarke

**OPEN    COPY    DELETE    PRINT**

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

Tue Nov 09 2021 04:00:00 GMT+0200 (Eastern European Standard Time)

OK

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

Account	Contact	Owner
Apex Solutions	Andrew Wayne	Marina Kysla
Apex Solutions	Andrew Wayne	Symon Clarke
Streamline Development	Bruce Clayton	Marina Kysla
Gateway	James Smith	Marina Kysla
Infocom	Andrew Z. Barber	Symon Clarke
Alpha Business	Jordan Anderson	Marina Kysla
Factorial Services	Taylor P. Johnson	Symon Clarke
Alpha Business		Mary King

## Добавить в раздел действие для нескольких записей

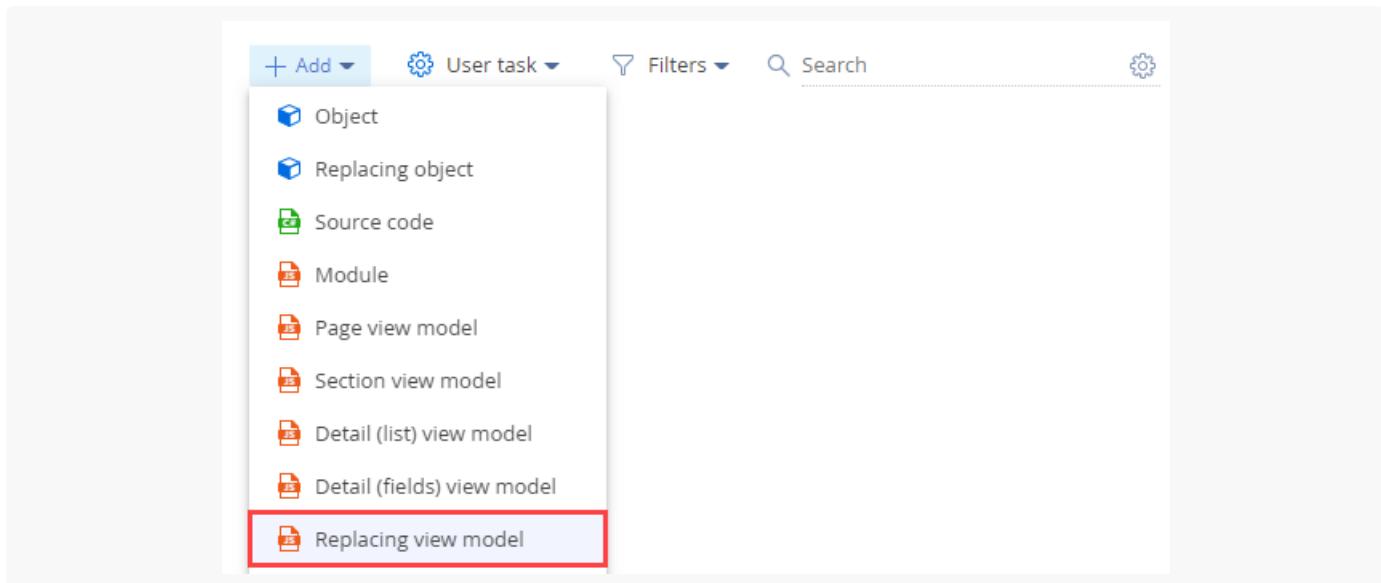


Пример реализован для продуктов линейки Sales Creatio.

**Пример.** Для записей реестра раздела [ Заказы ] ([ Orders ]) добавить действие, которое в информационном окне отображает перечень контрагентов выбранных заказов.

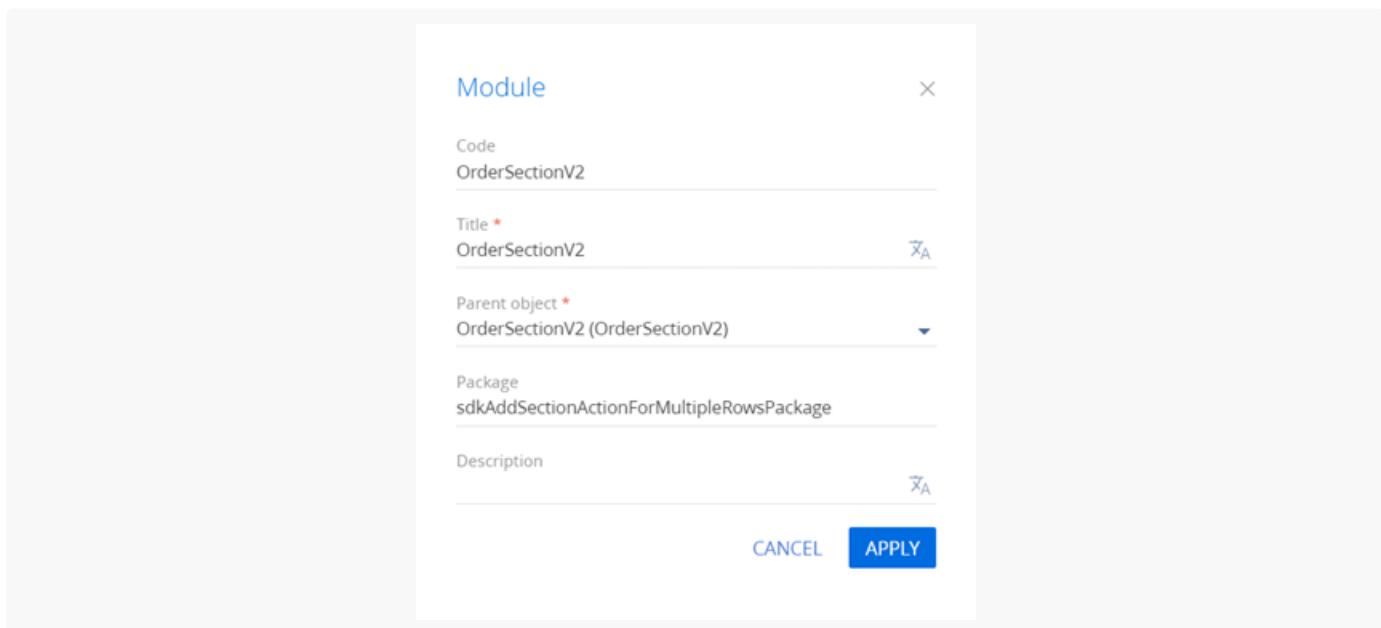
## Создать схему замещающей модели представления раздела

- 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**.**

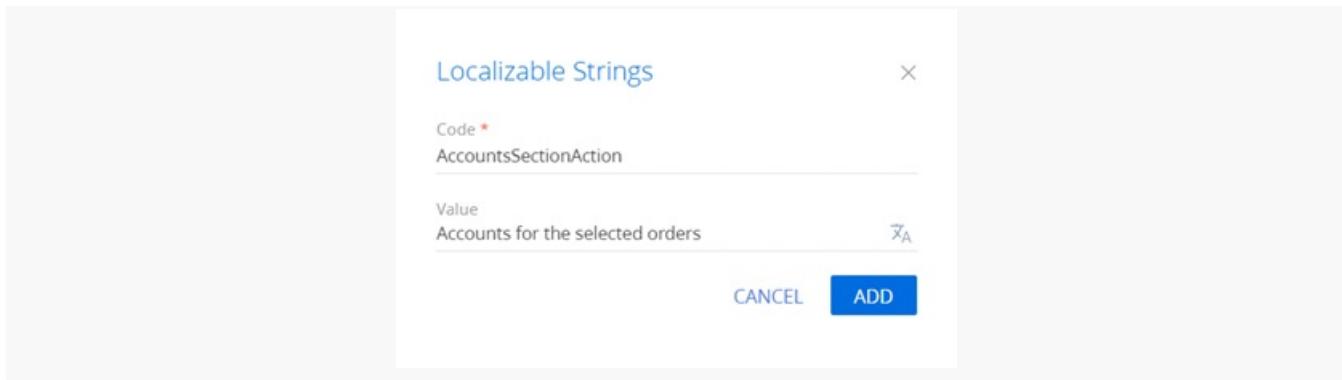
- [ Код ] ([ Code ]) — "OrderSectionV2".
- [ Заголовок ] ([ Title ]) — "Раздел заказа" ("Order section").
- [ Родительский объект ] ([ Parent object ]) — выберите "OrderSectionV2".



**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 ]) — "AccountsSectionAction".
- [ Значение ] ([ Value ]) — "Отобразить контрагентов по выбранным заказам" ("Accounts for the selected orders").



- е. Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).  
 5. Реализуйте **логику работы пункта меню**. Для этого в свойстве `methods` реализуйте **методы**:

- `isCustomActionEnabled()` — определяет доступность добавленного пункта меню.
- `showOrdersInfo()` — метод-обработчик действия, который отображает в информационном окне перечень контрагентов выбранных заказов.
- `getSectionActions()` — переопределенный метод родительской схемы, который возвращает коллекцию действий раздела.

Значение первичных колонок выбранных записей хранится в свойстве `SelectedRows` модели представления раздела. Эти значения могут использоваться для получения значений, загруженных в реестр полей выбранных объектов, например, из коллекции данных списочного реестра, которая хранится в свойстве `GridData` модели представления реестра.

Исходный код схемы замещающей модели представления раздела представлен ниже.

### OrderSectionV2

```
define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
  return {
    /* Название схемы объекта раздела. */
    entitySchemaName: "Order",
    /* Методы модели представления раздела. */
    methods: {
      /* Определяет доступность пункта меню. */
      isCustomActionEnabled: function() {
        /* Попытка получить массив идентификаторов выбранных записей. */
        var selectedRows = this.get("SelectedRows");
        /* Если массив содержит элементы (выбрана хотя бы одна запись в реестре), то
        return selectedRows ? (selectedRows.length > 0) : false;
      },
      /* Метод-обработчик действия. Отображает в информационном окне перечень контраген
      showOrdersInfo: function() {
        /* Получает массив идентификаторов выбранных записей. */
        var selectedRows = this.get("SelectedRows");
        /* Получает коллекцию данных записей реестра. */
        var gridData = this.get("GridData");
      }
    }
  }
});
```

```

/* Переменная для хранения модели объекта выбранного заказа. */
var selectedOrder = null;
/* Переменная для хранения названия контрагента выбранного заказа. */
var selectedOrderAccount = "";
/* Переменная для формирования текста информационного окна. */
var infoText = "";
/* Обработка массива идентификаторов выбранных записей реестра. */
selectedRows.forEach(function(selectedRowId) {
    /* Получает модель объекта выбранного заказа. */
    selectedOrder = gridData.get(selectedRowId);
    /* Получает название контрагента выбранного заказа. Колонка должна быть д
    selectedOrderAccount = selectedOrder.get("Account").displayValue;
    /* Добавляет название контрагента в текст информационного окна. */
    infoText += "\n" + selectedOrderAccount;
});
/* Отображает информационное окно. */
this.showInformationDialog(infoText);
},
/* Переопределение базового виртуального метода, который возвращает коллекцию дей
getSectionActions: function() {
    /* Вызывается родительская реализация метода для получения коллекции проиници
    var actionMenuItems = this.callParent(arguments);
    /* Добавляет линию-разделитель. */
    actionMenuItems.addItem(this.getButtonItem({
        Type: "Terrasoft.MenuSeparator",
        Caption: ""
    }));
    /* Добавляет пункт меню в список действий раздела. */
    actionMenuItems.addItem(this.getButtonItem({
        /* Привязка заголовка пункта меню к локализуемой строке схемы. */
        "Caption": {bindTo: "Resources.Strings.AccountsSectionAction"},
        /* Привязка метода-обработчика действия. */
        "Click": {bindTo: "showOrdersInfo"},
        /* Привязка свойства доступности пункта меню к значению, которое возвраща
        "Enabled": {bindTo: "isCustomActionEnabled"},
        /* Поддержка режима множественного выбора. */
        "IsEnabledForSelectedAll": true
    }));
    /* Возврат дополненной коллекции действий раздела. */
    return actionMenuItems;
}
};
});

```

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

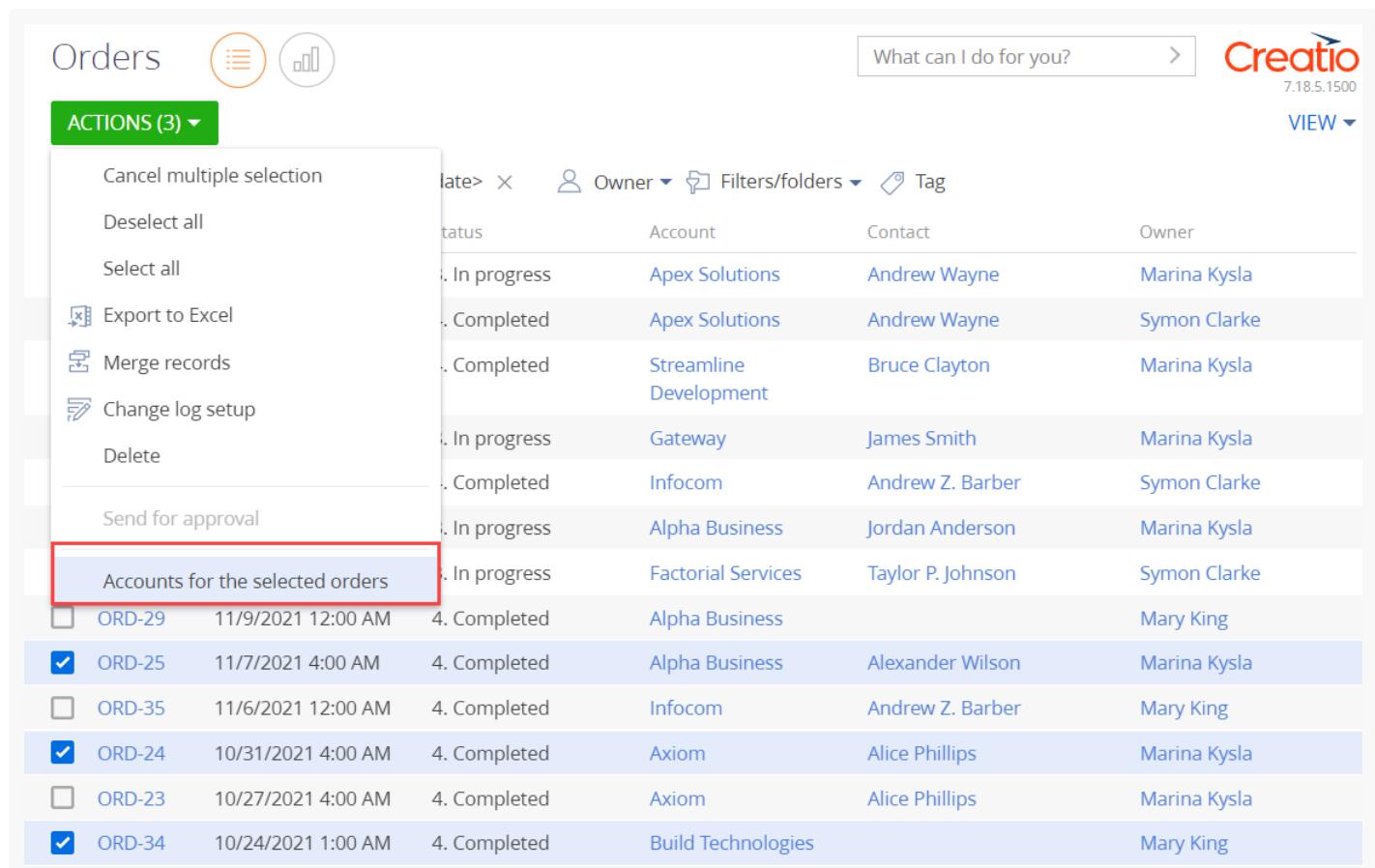
## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

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

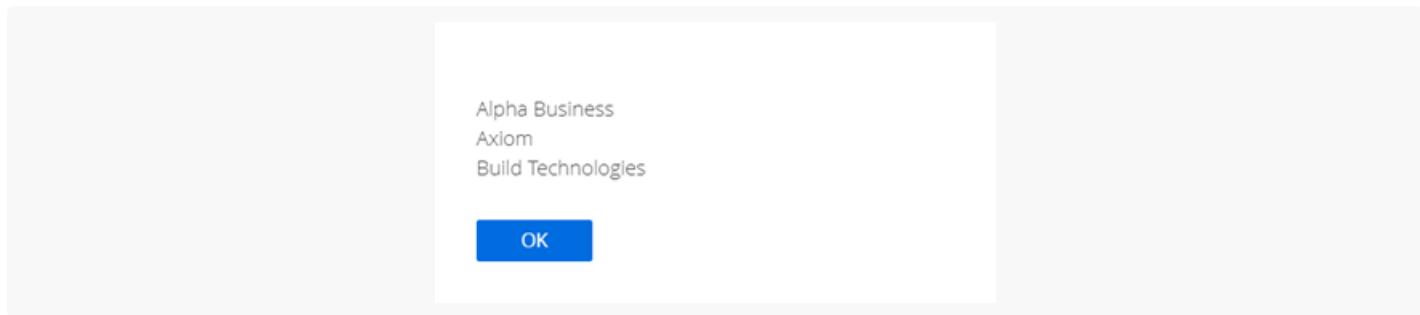
По умолчанию для реестра используется режим выбора одной записи. Для выбора в реестре нескольких записей в меню кнопки [ Действия ] ([ Actions ]) нажмите [ Выбрать несколько записей ] ([ Select multiple records ]). После этого изменяется визуальное представление реестра — появляются элементы для выбора записей.

В результате выполнения примера на страницу заказа добавлено действие [ Отобразить контрагентов по выбранным заказам ] ([ Accounts for the selected orders ]).



Status	Account	Contact	Owner		
In progress	Apex Solutions	Andrew Wayne	Marina Kysla		
Completed	Apex Solutions	Andrew Wayne	Symon Clarke		
Completed	Streamline Development	Bruce Clayton	Marina Kysla		
In progress	Gateway	James Smith	Marina Kysla		
Completed	Infocom	Andrew Z. Barber	Symon Clarke		
In progress	Alpha Business	Jordan Anderson	Marina Kysla		
In progress	Factorial Services	Taylor P. Johnson	Symon Clarke		
Completed	Alpha Business		Mary King		
<input checked="" type="checkbox"/> ORD-25	11/7/2021 4:00 AM	4. Completed	Alpha Business	Alexander Wilson	Marina Kysla
<input type="checkbox"/> ORD-35	11/6/2021 12:00 AM	4. Completed	Infocom	Andrew Z. Barber	Mary King
<input checked="" type="checkbox"/> ORD-24	10/31/2021 4:00 AM	4. Completed	Axiom	Alice Phillips	Marina Kysla
<input type="checkbox"/> ORD-23	10/27/2021 4:00 AM	4. Completed	Axiom	Alice Phillips	Marina Kysla
<input checked="" type="checkbox"/> ORD-34	10/24/2021 1:00 AM	4. Completed	Build Technologies		Mary King

В результате выбора действия [ Отобразить контрагентов по выбранным заказам ] ([ Accounts for the selected orders ]), в информационном окне отображается перечень контрагентов выбранных заказов.



Для отмены режима выбора нескольких записей в меню кнопки [ Действия ] ([ Actions ]) нажмите [ Отменить множественный выбор ] ([ Cancel multiple selection ]).

Если в реестре раздела [ Заказы ] ([ Orders ]) не выбрана ни одна запись, то действие [ Отобразить контрагентов по выбранным заказам ] ([ Accounts for the selected orders ]) неактивно.

Owner	Contact	Owner
Andrew Wayne	Marina Kysla	
Andrew Wayne	Symon Clarke	
Bruce Clayton	Marina Kysla	
James Smith	Marina Kysla	
Andrew Z. Barber	Symon Clarke	

## Настроить обработку действия для нескольких записей раздела

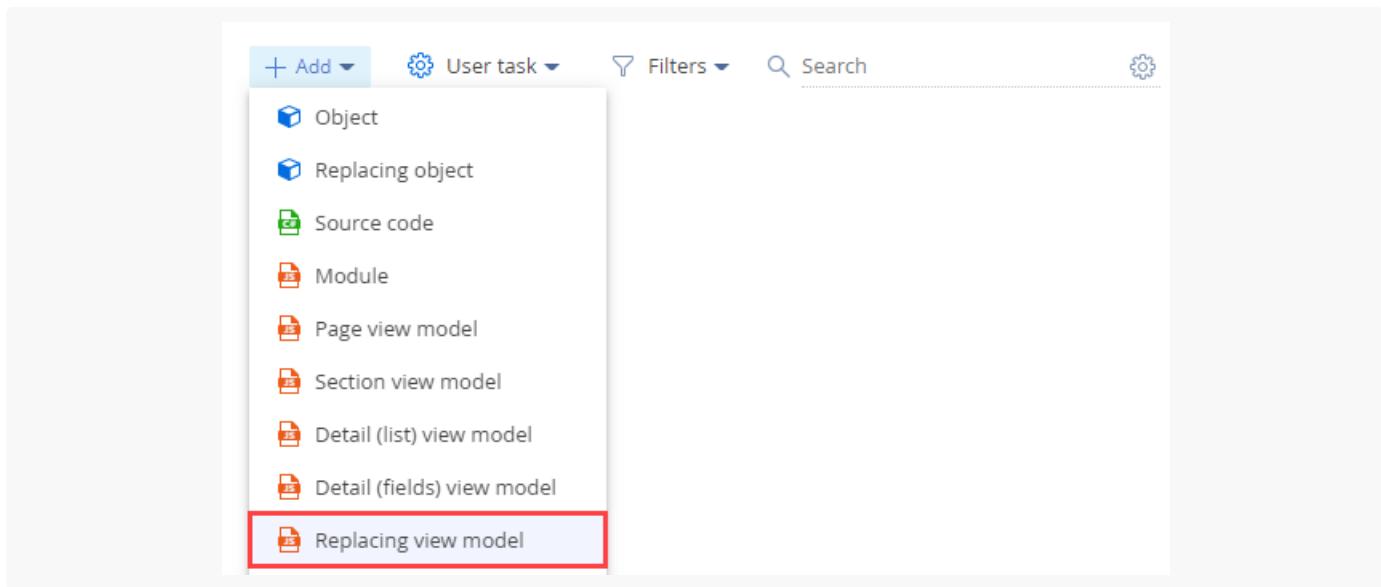


**Пример.** Для реестра раздела [ Активности ] ([ Activities ]) настроить обработку действия. Действие устанавливает состояние [ Завершена ] ([ Completed ]) для выбранных активностей.

## Создать схему замещающей модели представления раздела

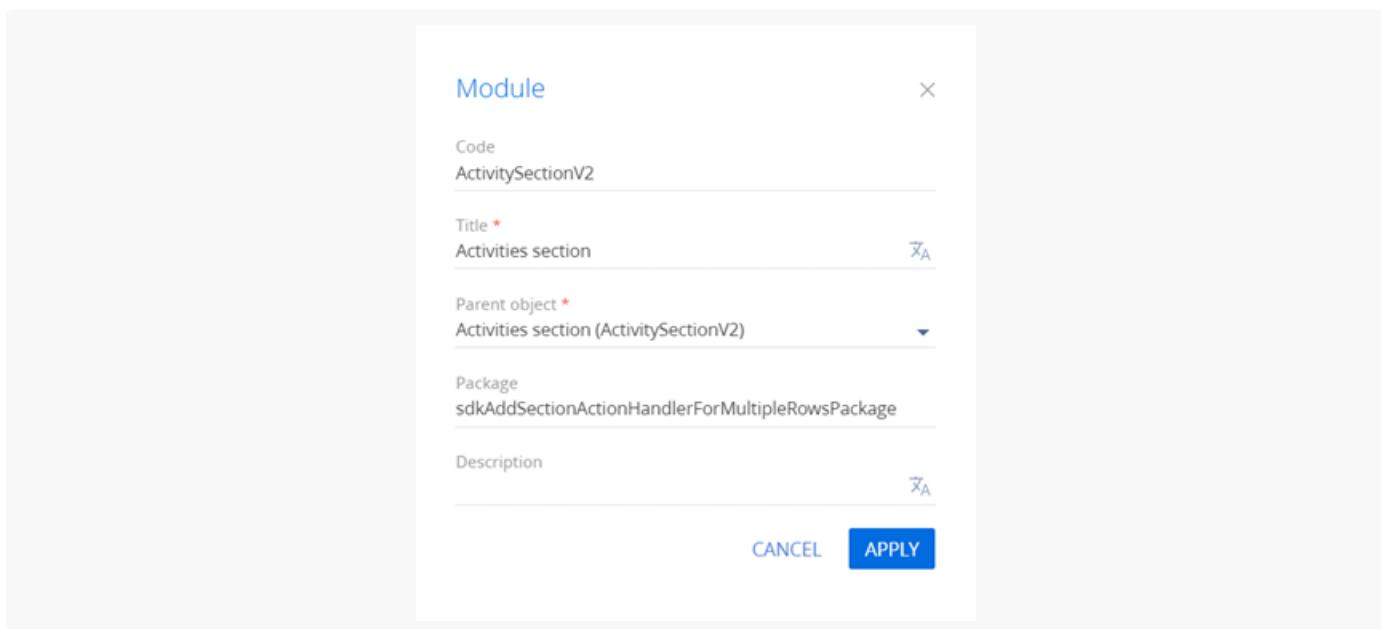
- Перейдите в раздел [ Конфигурация ] ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.

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



3. Заполните **свойства схемы**.

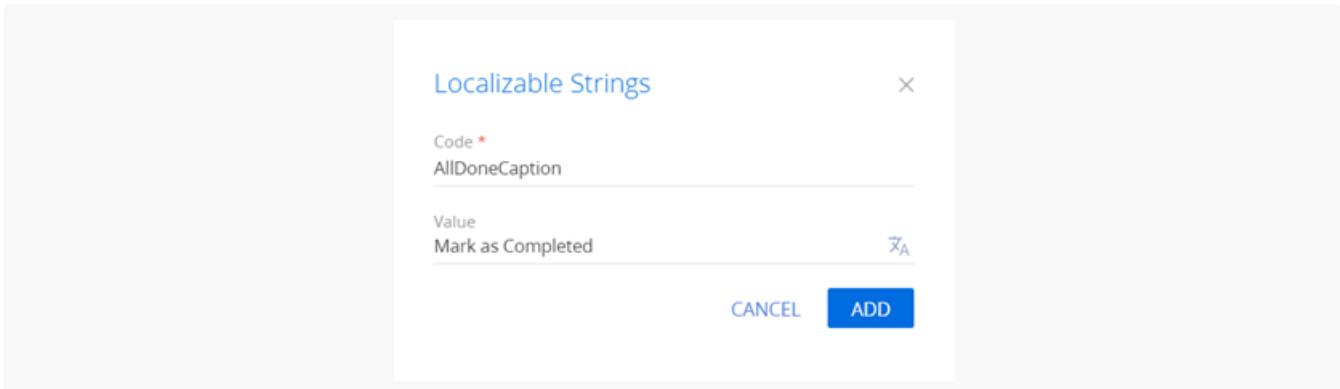
- [ Код ] ([ Code ]) — "ActivitySectionV2".
- [ Заголовок ] ([ Title ]) — "Раздел активности" ("Activities section").
- [ Родительский объект ] ([ Parent object ]) — выберите "ActivitySectionV2".



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 ]) — "AllDoneCaption".

- [ Значение ] ([ Value ]) — "Отметить как "Завершены"" ("Mark as Completed").



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

## 5. Реализуйте логику работы пункта меню.

Для этого в свойстве `methods` реализуйте методы:

- `isCustomActionEnabled()` — определяет доступность добавленного пункта меню.
- `setAllDone()` — метод-обработчик действия, который устанавливает состояние [ Завершена ] ([ *Completed* ]) для выбранных активностей.
- `getSectionActions()` — переопределенный метод родительской схемы, который возвращает коллекцию действий раздела.

Исходный код схемы замещающей модели представления раздела представлен ниже.

### ActivitySectionV2

```
define("ActivitySectionV2", ["ConfigurationConstants"], function(ConfigurationConstants) {
    return {
        /* Название схемы объекта раздела. */
        entitySchemaName: "Activity",
        /* Методы модели представления раздела. */
        methods: {
            /* Определяет доступность пункта меню. */
            isCustomActionEnabled: function() {
                /* Попытка получить массив идентификаторов выбранных записей. */
                var selectedRows = this.get("SelectedRows");
                /* Если массив содержит элементы (выбрана хотя бы одна запись в реестре), то */
                return selectedRows ? (selectedRows.length > 0) : false;
            },
            /* Метод-обработчик действия. Устанавливает для выбранных записей статус [Выполнено]. */
            setAllDone: function() {
                /* Получает массив идентификаторов выбранных записей. */
                var selectedRows = this.get("SelectedRows");
                /* Обработка запускается в случае, если выбрана хотя бы одна запись. */
                if (selectedRows.length > 0) {
                    /* Создает экземпляр класса пакетных запросов. */
                    var batchQuery = this.Ext.create("Terrasoft.BatchQuery");
                    /* Установка состояния для выбранных записей. */
                    batchQuery.set("Status", "Completed");
                    /* Выполнение пакетного запроса. */
                    batchQuery.execute();
                }
            }
        }
    };
});
```

```

/* Обновляет каждую из выбранных записей. */
selectedRows.forEach(function(selectedRowId) {
    /* Создает экземпляр класса UpdateQuery с корневой схемой Activity. */
    var update = this.Ext.create("Terrasoft.UpdateQuery", {
        rootSchemaName: "Activity"
    });
    /* Применяет фильтр для определения записи для обновления. */
    update.enablePrimaryColumnFilter(selectedRowId);
    /* Для колонки [Status] устанавливается значение "Завершена" с помощь
    update.setParameterValue("Status", ConfigurationConstants.Activity.St
    /* Добавляет запрос на обновление записи в пакетный запрос. */
    batchQuery.add(update);
}, this);
/* Выполняет пакетный запрос к серверу. */
batchQuery.execute(function() {
    /* Обновляет реестр. */
    this.reloadGridData();
}, this);
}
},
/* Переопределение базового виртуального метода, который возвращает коллекцию дей
getSectionActions: function() {
    /* Вызывается родительская реализация метода для получения коллекции проиници
    var actionMenuItems = this.callParent(arguments);
    /* Добавляет линию-разделитель. */
    actionMenuItems.addItem(this.getButtonMenuItem({
        Type: "Terrasoft.MenuSeparator",
        Caption: ""
    }));
    /* Добавляет пункт меню в список действий раздела. */
    actionMenuItems.addItem(this.getButtonMenuItem({
        /* Привязка заголовка пункта меню к локализуемой строке схемы. */
        "Caption": { bindTo: "Resources.Strings.AllDoneCaption" },
        /* Привязка метода-обработчика действия. */
        "Click": { bindTo: "setAllDone" },
        /* Привязка свойства доступности пункта меню к значению, которое возвраща
        "Enabled": { bindTo: "isCustomActionEnabled" },
        /* Поддержка режима множественного выбора. */
        "IsEnabledForSelectedAll": true
    }));
    /* Возврат дополненной коллекции действий раздела. */
    return actionMenuItems;
}
}
);
});

```

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

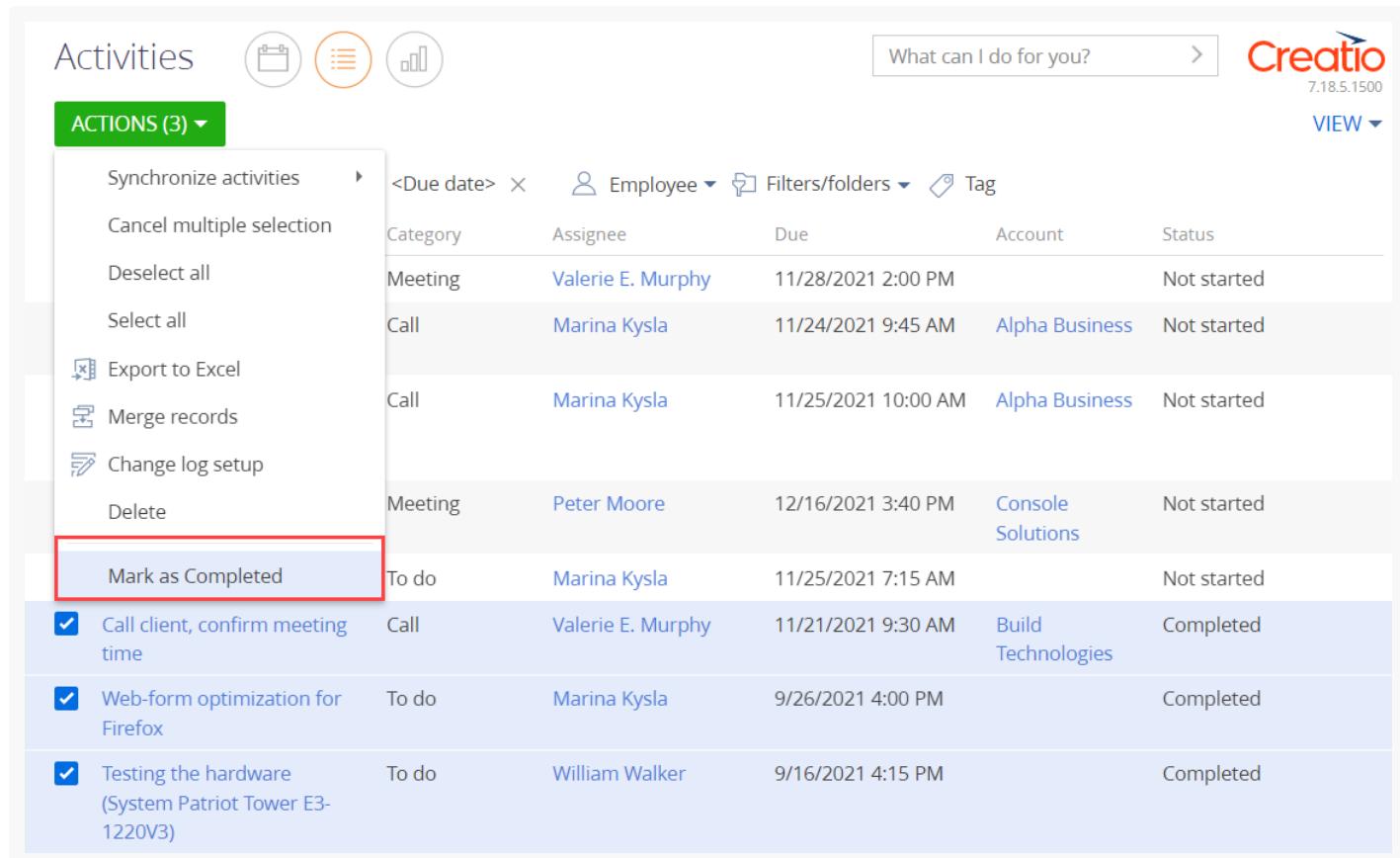
## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

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

По умолчанию для реестра используется режим выбора одной записи. Для выбора в реестре нескольких записей в меню кнопки [ Действия ] ([ Actions ]) нажмите [ Выбрать несколько записей ] ([ Select multiple records ]). После этого изменяется визуальное представление реестра — появляются элементы для выбора записей.

В результате выполнения примера на страницу активности добавлено действие [ Отметить как "Завершены" ] ([ Mark as Completed ]).



The screenshot shows the 'Activities' page in the Creatio application. A context menu is open over a selection of three tasks. The menu is titled 'ACTIONS (3)' and includes options like 'Synchronize activities', 'Cancel multiple selection', 'Deselect all', 'Select all', 'Export to Excel', 'Merge records', 'Change log setup', 'Delete', and 'Mark as Completed'. The 'Mark as Completed' option is highlighted with a red box. The main table lists tasks with columns for Category, Assignee, Due date, Account, and Status. The selected tasks have a blue background.

Category	Assignee	Due	Account	Status
Meeting	Valerie E. Murphy	11/28/2021 2:00 PM		Not started
Call	Marina Kysla	11/24/2021 9:45 AM	Alpha Business	Not started
Call	Marina Kysla	11/25/2021 10:00 AM	Alpha Business	Not started
Meeting	Peter Moore	12/16/2021 3:40 PM	Console Solutions	Not started
To do	Marina Kysla	11/25/2021 7:15 AM		Not started
Call	Valerie E. Murphy	11/21/2021 9:30 AM	Build Technologies	Completed
To do	Marina Kysla	9/26/2021 4:00 PM		Completed
To do	William Walker	9/16/2021 4:15 PM		Completed

В результате выбора действия [ Отметить как "Завершены" ] ([ Mark as Completed ]) для выбранных активностей в колонке [ Статус ] ([ Status ]) установлено значение [ Завершена ] ([ Completed ]).

<input type="checkbox"/> Work with emails, calendar	To do	Marina Kysla	11/25/2021 7:15 AM		Not started
<input type="checkbox"/> Call client, confirm meeting time	Call	Valerie E. Murphy	11/21/2021 9:30 AM	Build Technologies	Completed
<input type="checkbox"/> Web-form optimization for Firefox	To do	Marina Kysla	9/26/2021 4:00 PM		Completed
<input type="checkbox"/> Testing the hardware (System Patriot Tower E3-1220V3)	To do	William Walker	9/16/2021 4:15 PM		Completed
<input type="checkbox"/> Contact customer, specify need, budget, decision-making role.	To do	Mary King	11/21/2021 11:16 AM		Completed

Для отмены режима выбора нескольких записей в меню кнопки [ Действия ] ([ Actions ]) нажмите [ Отменить множественный выбор ] ([ Cancel multiple selection ]).

Если в реестре раздела [ Активности ] ([ Activities ]) не выбрана ни одна запись, то действие [ Отметить как "Завершены" ] ([ Mark as Completed ]) неактивно.

The screenshot shows the 'Activities' module in the Creatio application. At the top, there are navigation icons and a search bar. Below the header, there are buttons for 'NEW TASK' and 'ACTIONS'. A context menu is open, listing options like 'Synchronize activities', 'Select multiple records', 'Select all', 'Export to Excel', 'Data import', and 'Change log setup'. The 'Mark as Completed' option is highlighted with a red box. To the right of the menu, a list of activities is displayed in a table format. The columns are 'Assignee', 'Due', 'Account', and 'Status'. The activities listed are: Valerie E. Murphy (11/28/2021 2:00 PM, Not started), Marina Kysla (11/24/2021 9:45 AM, Alpha Business, Not started), Marina Kysla (11/25/2021 10:00 AM, Alpha Business, Not started), Peter Moore (12/16/2021 3:40 PM, Console Solutions, Not started), and Marina Kysla (11/25/2021 7:15 AM, Not started).

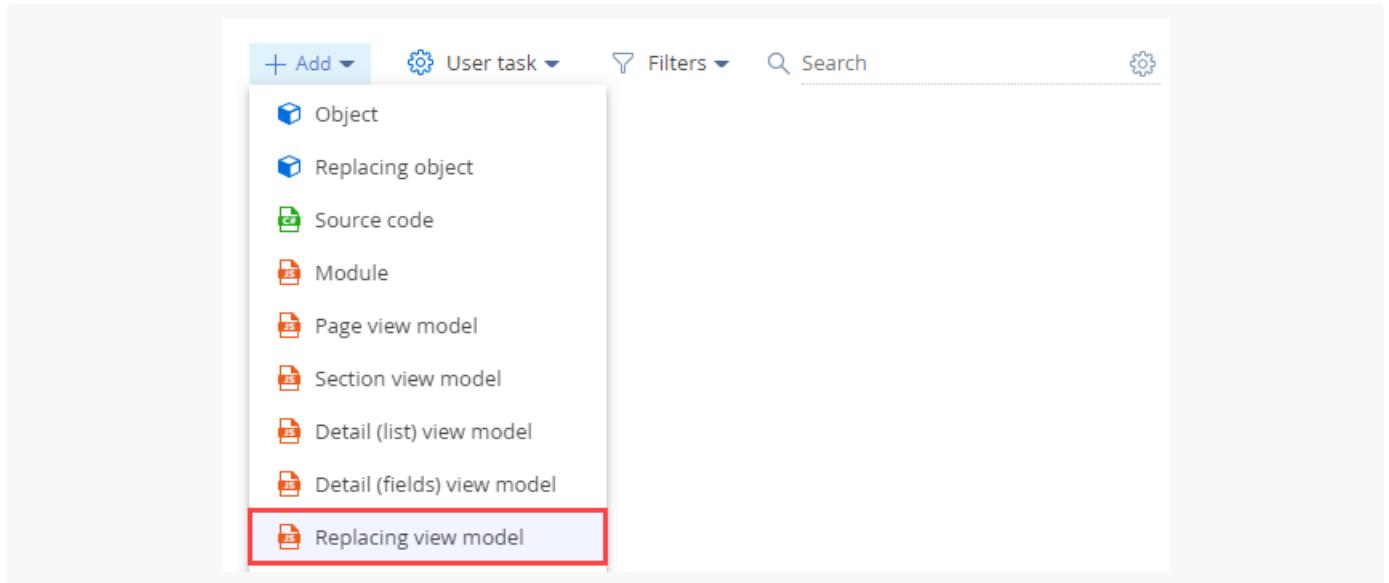
## Настроить обработку действия с использованием справочника для нескольких записей раздела



**Пример.** Для реестра раздела [ Активности ] ([ Activities ]) настроить обработку действия. Действие вызывает окно справочника [ Контакты ] ([ Contacts ]). Выбранный контакт устанавливается в качестве ответственного для выбранных активностей.

## 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 ]) — "ActivitySectionV2".
- [ Заголовок ] ([ Title ]) — "Раздел активности" ("Activities section").
- [ Родительский объект ] ([ Parent object ]) — выберите "ActivitySectionV2".

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

- Code:** ActivitySectionV2
- Title \***: Activities section
- Parent object \***: Activities section (ActivitySectionV2)
- Package:** sdkAddSectionActionHandlerWithLookupPackage
- Description**: (empty)

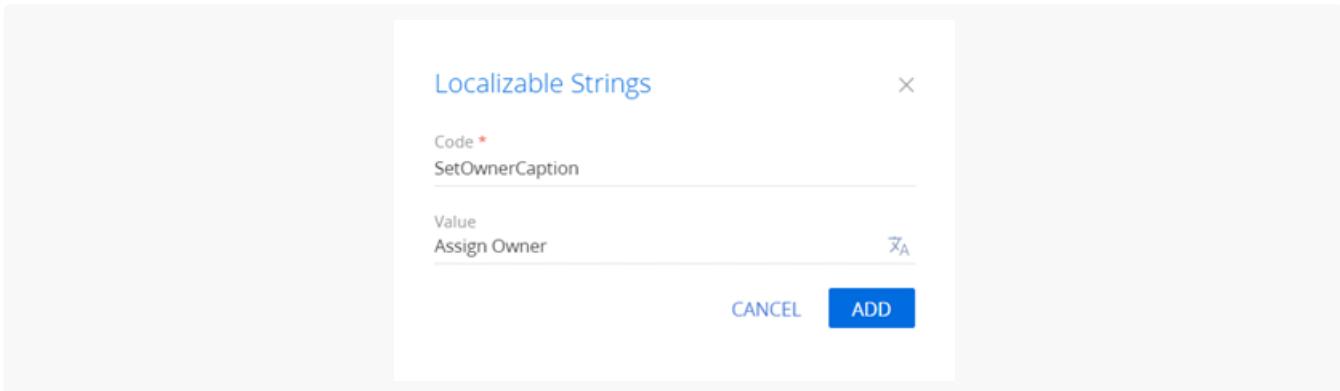
At the bottom right are 'CANCEL' and 'APPLY' buttons.

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

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

b. Заполните **свойства локализуемой строки**.

- [ *Код* ] ([ *Code* ]) — "SetOwnerCaption".
- [ *Значение* ] ([ *Value* ]) — "Назначить ответственного" ("Assign Owner").



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

5. Реализуйте **логику работы пункта меню**. Для этого в свойстве `methods` реализуйте **методы**:

- `isCustomActionEnabled()` — определяет доступность добавленного пункта меню.
- `setOwner()` — метод-обработчик действия, который вызывает открытие справочника [ *Контакты* ] ([ *Contacts* ]).
- `lookupCallback()` — callback-метод, который устанавливает выбранный в справочнике контакт в качестве ответственного для выбранных активностей.
- `getSectionActions()` — переопределенный метод родительской схемы, который возвращает коллекцию действий раздела.

Исходный код схемы замещающей модели представления раздела представлен ниже.

#### ActivitySectionV2

```
define("ActivitySectionV2", ["ConfigurationConstants"], function(ConfigurationConstants) {
    return {
        /* Название схемы объекта раздела. */
        entitySchemaName: "Activity",
        /* Методы модели представления раздела. */
        methods: {
            /* Определяет доступность пункта меню. */
            isCustomActionEnabled: function() {
                /* Попытка получить массив идентификаторов выбранных записей. */
                var selectedRows = this.get("SelectedRows");
                /* Если массив содержит элементы (выбрана хотя бы одна запись в реестре), то
                return selectedRows ? (selectedRows.length > 0) : false;
            }
        }
    }
});
```

```

},
/* Метод-обработчик действия. Вызывает открытие справочника [Контакты]. */
setOwner: function() {
    /* Определяет конфигурацию справочника. */
    var config = {
        /* Схема [Contact]. */
        entitySchemaName: "Contact",
        /* Множественный выбор отключен. */
        multiSelect: false,
        /* Отображаемая колонка – [Name]. */
        columns: ["Name"]
    };
    /* Открывает справочник с определенной конфигурацией и callback-функцией, кот
    this.openLookup(config, this.lookupCallback, this);
},
/* Выполняет установку выбранного в справочнике контакта в качестве ответственног
lookupCallback: function(args) {
    /* Идентификатор выбранной из справочника записи. */
    var activeRowId;
    /* Получает выбранные в справочнике записи. */
    var lookupSelectedRows = args.selectedRows.getItems();
    if (lookupSelectedRows && lookupSelectedRows.length > 0) {
        /* Получает Id первой выбранной в справочнике записи. */
        activeRowId = lookupSelectedRows[0].Id;
    }
    /* Получает массив идентификаторов выбранных записей. */
    var selectedRows = this.get("SelectedRows");
    /* Обработка запускается в случае, если выбрана хотя бы одна запись и в справ
    if ((selectedRows.length > 0) && activeRowId) {
        /* Создает экземпляр класса пакетных запросов. */
        var batchQuery = this.Ext.create("Terrasoft.BatchQuery");
        /* Обновляет каждую из выбранных записей. */
        selectedRows.forEach(function(selectedRowId) {
            /* Создает экземпляр класса UpdateQuery с корневой схемой Activity. */
            var update = this.Ext.create("Terrasoft.UpdateQuery", {
                rootSchemaName: "Activity"
            });
            /* Применяет фильтр для определения записи для обновления. */
            update.enablePrimaryColumnFilter(selectedRowId);
            /* Для колонки [Owner] устанавливается значение, равное идентификатор
            update.setParameterValue("Owner", activeRowId, this..Terrasoft.DataVa
            /* Добавляет запрос на обновление записи в пакетный запрос. */
            batchQuery.add(update);
        }, this);
        /* Выполняет пакетный запрос к серверу. */
        batchQuery.execute(function() {
            /* Обновляет реестр. */
            this.reloadGridData();
        }, this);
    }
}

```

```

        },
        /* Переопределение базового виртуального метода, который возвращает коллекцию действий */
getSectionActions: function() {
    /* Вызывается родительская реализация метода для получения коллекции проприетарных действий */
    var actionMenuItems = this.callParent(arguments);
    /* Добавляет линию-разделитель. */
    actionMenuItems.addItem(this.getButtonItem({
        Type: "Terrasoft.MenuSeparator",
        Caption: ""
    }));
    /* Добавляет пункт меню в список действий раздела. */
    actionMenuItems.addItem(this.getButtonItem({
        /* Привязка заголовка пункта меню к локализуемой строке схемы. */
        "Caption": { bindTo: "Resources.Strings.SetOwnerCaption" },
        /* Привязка метода-обработчика действия. */
        "Click": { bindTo: "setOwner" },
        /* Привязка свойства доступности пункта меню к значению, которое возвращается из метода */
        "Enabled": { bindTo: "isCustomActionEnabled" },
        /* Поддержка режима множественного выбора. */
        "IsEnabledForSelectedAll": true
    }));
    /* Возврат дополненной коллекции действий раздела. */
    return actionMenuItems;
}
};

};

});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

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

По умолчанию для реестра используется режим выбора одной записи. Для выбора в реестре нескольких записей в меню кнопки [ Действия ] ([ Actions ]) нажмите [ Выбрать несколько записей ] ([ Select multiple records ]). После этого изменяется визуальное представление реестра — появляются элементы для выбора записей.

В результате выполнения примера на страницу заказа добавлено действие [ Назначить ответственного ] ([ Assign Owner ]).

The screenshot shows the 'Activities' screen in the Creatio application. At the top, there are three icons: a calendar, a list, and a chart. To the right is a search bar with the placeholder 'What can I do for you?' and a 'VIEW' dropdown. The main area has a header 'Activities' and a 'ACTIONS (3)' dropdown menu. The menu items are: 'Synchronize activities', 'Cancel multiple selection', 'Deselect all', 'Select all', 'Export to Excel', 'Merge records', 'Change log setup', 'Delete', and 'Assign Owner'. The 'Assign Owner' item is highlighted with a red box. Below the menu is a table with columns: Category, Assignee, Due, Account, and Status. The table contains several rows of activity data.

Category	Assignee	Due	Account	Status	
Meeting	Valerie E. Murphy	11/28/2021 2:00 PM		Not started	
Call	Marina Kysla	11/24/2021 9:45 AM	Alpha Business	Not started	
Call	Marina Kysla	11/25/2021 10:00 AM	Alpha Business	Not started	
Meeting	Peter Moore	12/16/2021 3:40 PM	Console Solutions	Not started	
To do	Marina Kysla	11/25/2021 7:15 AM		Not started	
<input checked="" type="checkbox"/> Call client, confirm meeting time	Call	Valerie E. Murphy	11/21/2021 9:30 AM	Build Technologies	Completed
<input checked="" type="checkbox"/> Web-form optimization for Firefox	To do	Marina Kysla	9/26/2021 4:00 PM		Completed
<input checked="" type="checkbox"/> Testing the hardware (System Patriot Tower E3-1220V3)	To do	William Walker	9/16/2021 4:15 PM		Completed

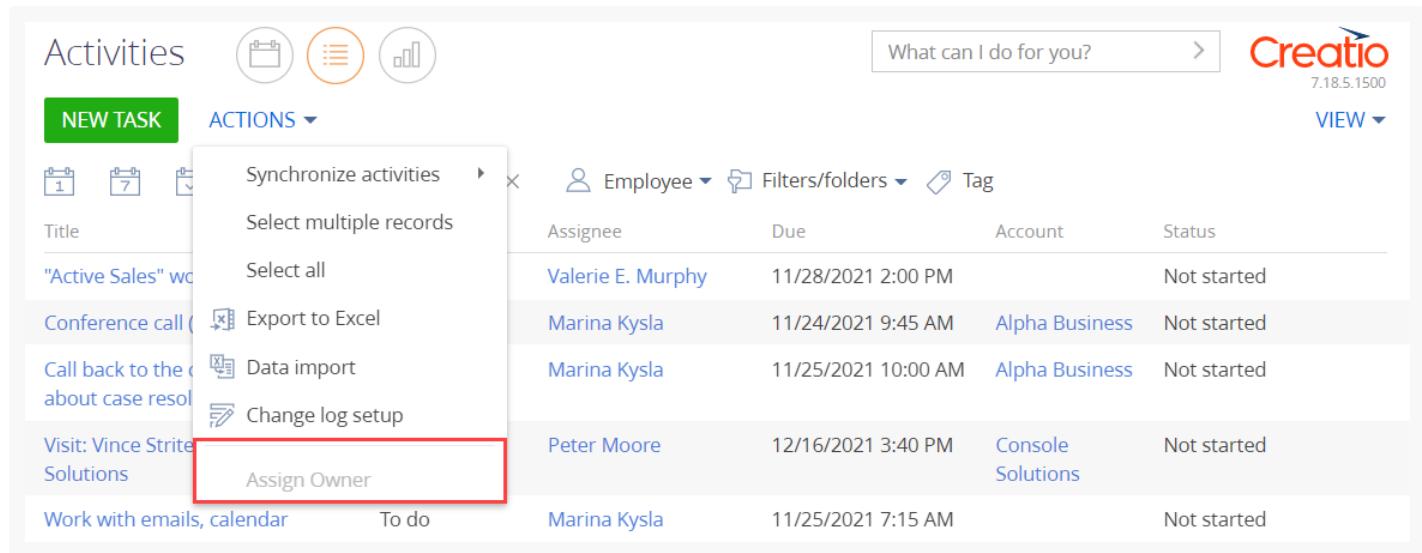
В результате выбора действия [ Назначить ответственного ] ([ *Assign Owner* ]) отображается окно справочника [ Контакты ] ([ *Contacts* ]). Выбранный контакт устанавливается в колонке [ Ответственный ] ([ *Assignee* ]) в качестве ответственного для выбранных активностей.

The screenshot shows a list of contacts in the 'Contacts' window. A red box highlights the contact 'Tran Manzo'. The table has columns: Name, Phone, Email, and Note. The contact 'Tran Manzo' appears in three rows of the table.

Name	Phone	Email	Note
Tran Manzo			
Tran Manzo			
Tran Manzo			
Mary King			

Для отмены режима выбора нескольких записей в меню кнопки [ Действия ] ([ *Actions* ]) нажмите [ Отменить множественный выбор ] ([ *Cancel multiple selection* ]).

Если в реестре раздела [ Активности ] ([ *Activities* ]) не выбрана ни одна запись, то действие [ Назначить ответственного ] ([ *Assign Owner* ]) неактивно.



The screenshot shows the 'Activities' section of the Creatio application. At the top, there are navigation icons for 'Activities', 'Calendar', 'List', and 'Report'. A search bar says 'What can I do for you?'. On the right, the 'Creatio' logo is visible with the version '7.18.5.1500'. Below the header, there's a 'NEW TASK' button and a 'VIEW' dropdown. A context menu is open over a task titled 'Visit: Vince Strike Solutions', with the 'Assign Owner' option highlighted. The main area displays a table of tasks:

Assignee	Due	Account	Status
Valerie E. Murphy	11/28/2021 2:00 PM		Not started
Marina Kysla	11/24/2021 9:45 AM	Alpha Business	Not started
Marina Kysla	11/25/2021 10:00 AM	Alpha Business	Not started
Peter Moore	12/16/2021 3:40 PM	Console Solutions	Not started
Marina Kysla	11/25/2021 7:15 AM		Not started

## Настроить условия отображения реестра раздела

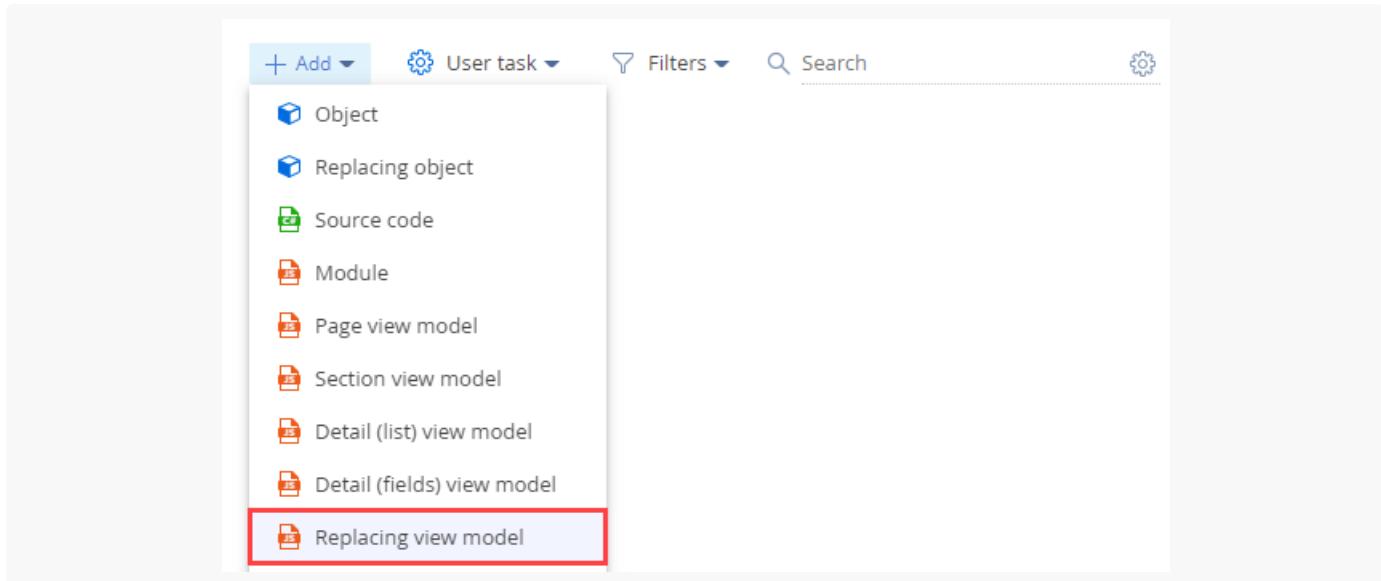
 Средний

Пример реализован для продуктов линейки Sales Creatio.

**Пример.** Для реестра раздела [ Заказы ] ([ *Orders* ]) выделять цветом заказы, которые находятся на стадии [ Исполнение ] ([ *In progress* ]).

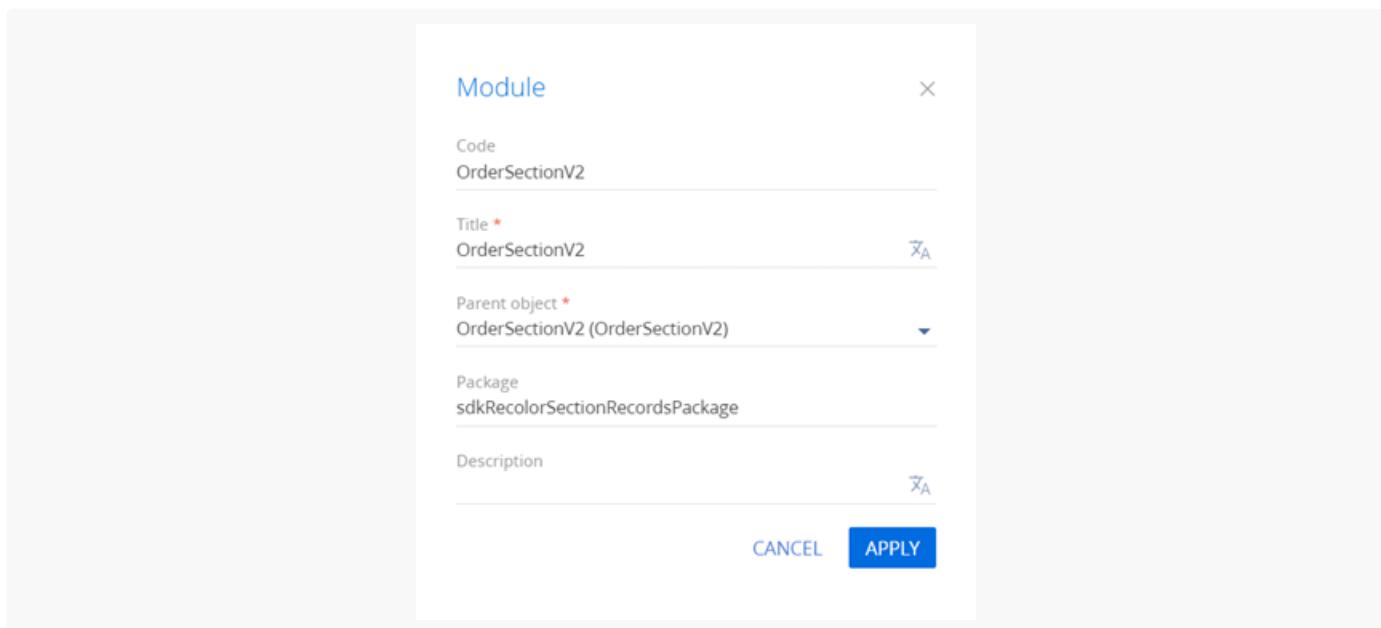
## Создать схему замещающей модели представления раздела

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* ]).



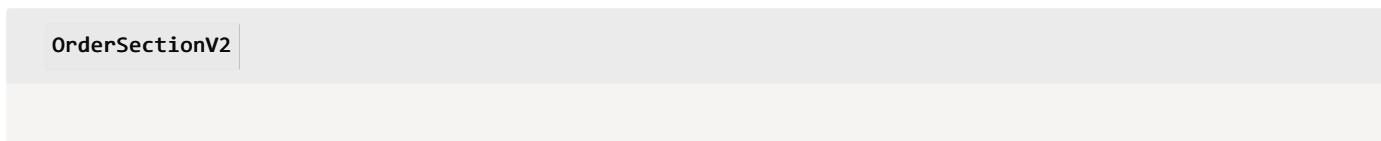
### 3. Заполните **свойства схемы**.

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



### 4. Настройте **условия отображения реестра раздела**. Для этого в свойстве `methods` реализуйте метод `prepareResponseCollectionItem()`, который переопределяет базовый метод, модифицирует строку данных перед загрузкой в реестр, а также добавляет настраиваемые стили к определенным строкам реестра.

Исходный код схемы замещающей модели представления раздела представлен ниже.



```

define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
return {
    /* Название схемы объекта раздела. */
    entitySchemaName: "Order",
    /* Методы модели представления раздела. */
    methods: {
        /* Переопределение базового метода, который модифицирует строку данных перед загр
        prepareResponseCollectionItem: function(item) {
            /* Вызывает базовый метод. */
            this.callParent(arguments);
            item.customStyle = null;
            /* Определяет статус заказа. */
            var running = item.get("Status");
            /* Если состояние заказа "Исполнение", меняется стиль записи. */
            if (running.value === OrderConfigurationConstants.Order.OrderStatus.Running)
                item.customStyle = {
                    /* Цвет текста – белый. */
                    "color": "white",
                    /* Цвет фона – зеленый. */
                    "background": "#8ecb60"
                };
            }
        }
    );
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

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

В результате выполнения примера заказы, которые находятся на стадии [ Исполнение ] ([ In progress ]), выделяются цветом.

Number	Date	Status	Account	Contact	Owner
ORD-30	11/18/2021 12:00 AM	3. In progress	Apex Solutions	Andrew Wayne	Marina Kysla
ORD-31	11/17/2021 12:00 AM	4. Completed	Apex Solutions	Andrew Wayne	Symon Clarke
ORD-26	11/14/2021 4:00 AM	4. Completed	Streamline Development	Bruce Clayton	Marina Kysla
ORD-28	11/13/2021 1:00 AM	3. In progress	Gateway	James Smith	Marina Kysla
ORD-33	11/13/2021 12:00 AM	4. Completed	Infocom	Andrew Z. Barber	Symon Clarke
ORD-32	11/11/2021 12:00 AM	3. In progress	Alpha Business	Jordan Anderson	Marina Kysla
ORD-7	11/9/2021 4:00 AM	3. In progress	Factorial Services	Taylor P. Johnson	Symon Clarke

## Настроить блок быстрых фильтров раздела

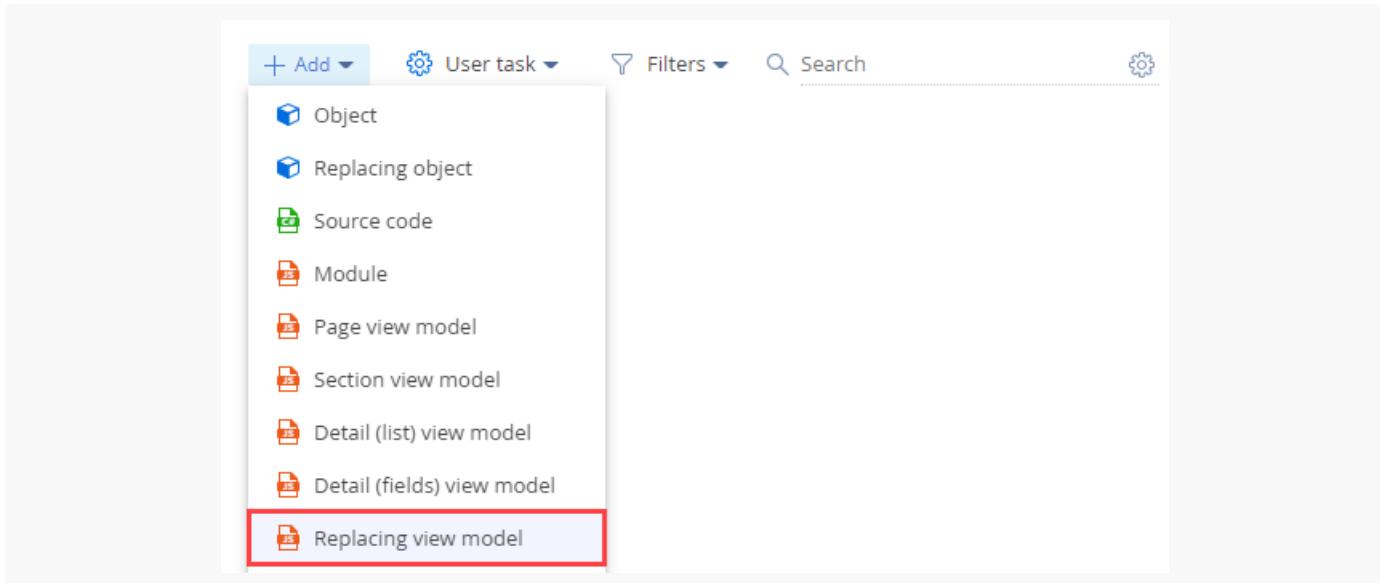


Средний

**Пример.** В раздел [ Договоры ] ([ Contracts ]) добавить блок быстрых фильтров. Договоры фильтруются по дате начала договора и по ответственному.

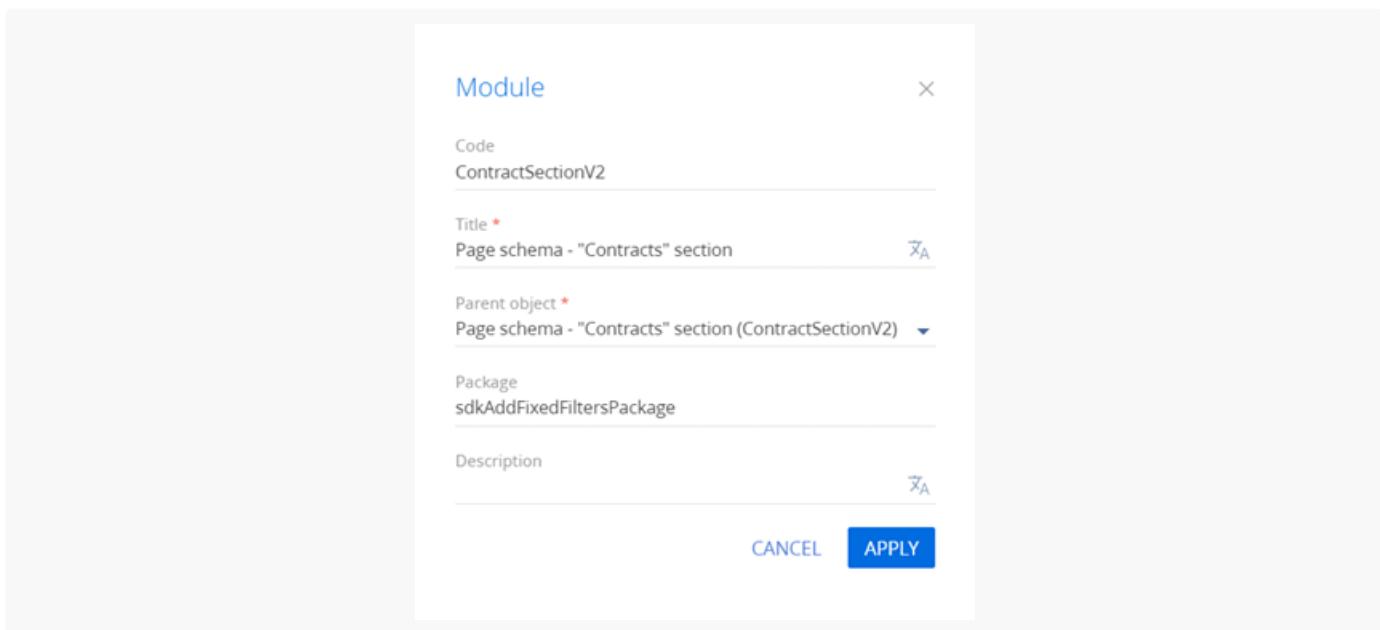
## Создать схему замещающей модели представления раздела

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**.

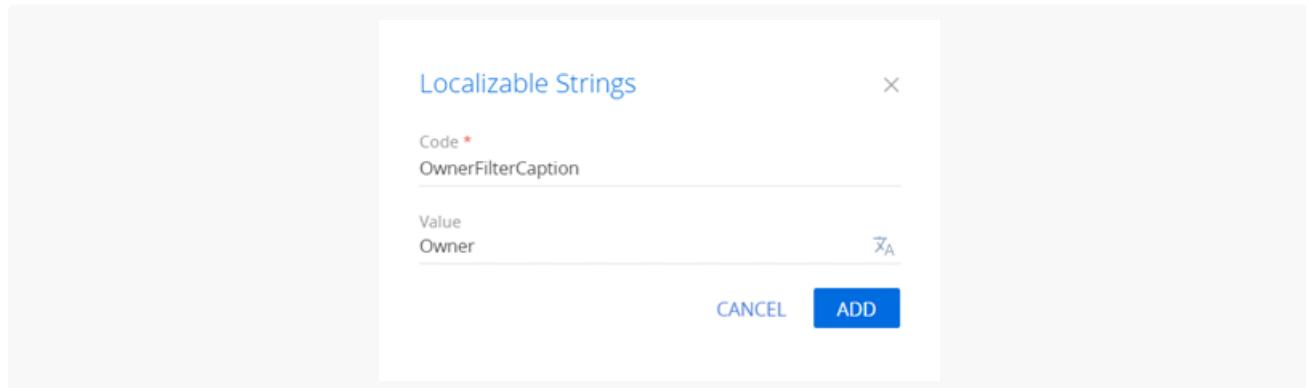
- [ Код ] ([ Code ]) — "ContractSectionV2".
- [ Заголовок ] ([ Title ]) — "Схема страницы раздела \"Договоры\" ("Page schema - "Contracts" section").
- [ Родительский объект ] ([ Parent object ]) — выберите "ContractSectionV2".



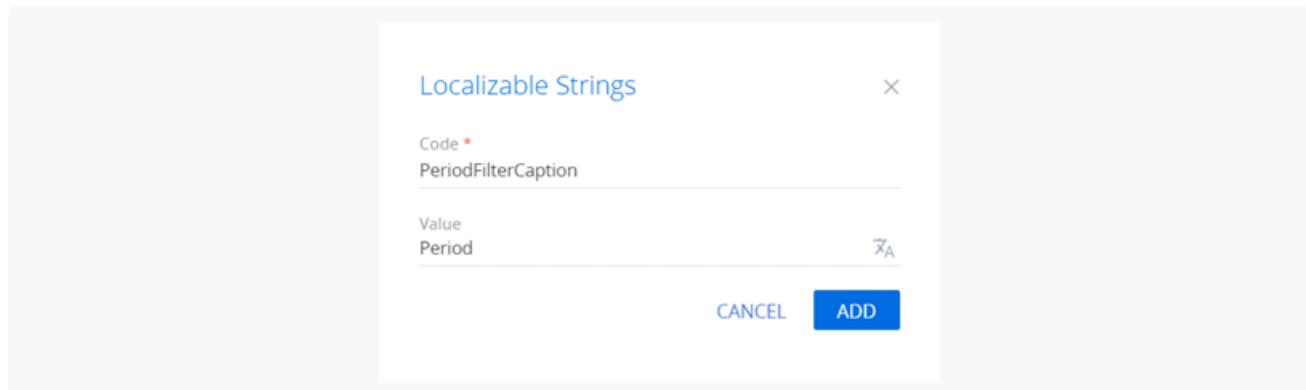
### 4. Добавьте **локализуемые строки** с названиями фильтров.

- Добавьте локализуемую строку, которая содержит **название фильтра по ответственному сотруднику**.
  - В контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) нажмите кнопку +.
  - Заполните **свойства локализуемой строки**.
    - [ Код ] ([ Code ]) — "OwnerFilterCaption".

- [ Значение ] ([ Value ]) — "Ответственный" ("Owner").



- е. Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).
- б. Добавьте локализуемую строку, которая содержит **название фильтра по периоду**.
  - а. В контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) нажмите кнопку +.
  - б. Заполните **свойства локализуемой строки**.
    - [ Код ] ([ Code ]) — "PeriodFilterCaption".
    - [ Значение ] ([ Value ]) — "Период" ("Period").



- е. Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).
5. Реализуйте **логику работы фильтрации**. Для этого в свойстве `methods` реализуйте метод `initFixedFiltersConfig()`. В методе создайте конфигурационный объект с массивом фильтров `PeriodFilter` и `OwnerFilter`, присвойте ссылку на объект атрибуту `fixedFiltersConfig` модели представления.

Исходный код схемы замещающей модели представления раздела представлен ниже.

#### ContractSectionV2

```
define("ContractSectionV2", ["BaseFiltersGenerateModule"], function(BaseFiltersGenerateModule
  return {
    /* Название схемы объекта раздела. */
    entitySchemaName: "Contract",
```

```

/* Методы модели представления раздела. */
methods: {
    /* Инициализирует фиксированные фильтры. */
    initFixedFiltersConfig: function() {
        /* Создает конфигурационный объект. */
        var fixedFilterConfig = {
            /* В качестве схемы объекта для фиксированных фильтров указывается схема
            entitySchema: this.entitySchema,
            /* Массив фильтров. */
            filters: [
                /* Фильтр периода. */
                {
                    /* Название фильтра. */
                    name: "PeriodFilter",
                    /* Заголовок фильтра. */
                    caption: this.get("Resources.Strings.PeriodFilterCaption"),
                    /* Тип данных – дата. */
                    dataType: this.Terrasoft.DataValueType.DATE,
                    /* Дата начала периода фильтрации. */
                    startDate: {
                        /* Фильтруются данные из колонки [Date]. */
                        columnName: "StartDate",
                        /* Значение по умолчанию – начало текущей недели. */
                        defValue: this.Terrasoft.startOfWeek(new Date())
                    },
                    /* Дата завершения периода фильтрации – завершение текущей недели
                    dueDate: {
                        columnName: "StartDate",
                        defValue: this.Terrasoft.endOfWeek(new Date())
                    }
                },
                /* Фильтр ответственного. */
                {
                    /* Название фильтра. */
                    name: "Owner",
                    /* Заголовок фильтра. */
                    caption: this.get("Resources.Strings.OwnerFilterCaption"),
                    /* Фильтрация данных из колонки [Owner]. */
                    columnName: "Owner",
                    /* Значение по умолчанию – контакт текущего пользователя, который
                    defValue: this.Terrasoft.SysValue.CURRENT_USER_CONTACT,
                    /* Тип данных – справочник. */
                    dataType: this.Terrasoft.DataValueType.LOOKUP,
                    /* Фильтр. */
                    filter: BaseFiltersGenerateModule.OwnerFilter
                }
            ]
        };
        /* Атрибуту [FixedFilterConfig] присваивается ссылка на созданный конфигураци

```

```

        this.set("FixedFilterConfig", fixedFilterConfig);
    }
}
);
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ Договоры ] ([ Contracts ]).

В результате выполнения примера в разделе [ Договоры ] ([ Contracts ]) отображается блок фиксированных фильтров, который позволяет фильтровать договоры как по дате их начала, так и по ответственному сотруднику.

Number	Start date	Type	Account	Owner	Status
201 (sample)	4/11/2021	Contract	Accom (sample)	Supervisor	Draft
322	10/29/2014	Contract	Axiom	Marina Kysla	Signed
324	10/28/2016	Contract	Alpha Business	Marina Kysla	Signed

## Схема BaseSectionV2 js

Основы

**BaseSectionV2** — базовая схема раздела. Предоставляет базовую логику раздела. Реализована в пакете **NUI**. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Все схемы разделов должны наследовать схему **BaseSectionV2**.

## Сообщения

Сообщения базового раздела

Название	Режим	Направление	Описание
Rerender Module	Адресное	Публикация	Повторно отобразить сообщение модуля дашбордов.
ReloadData OnRestore	Широковещательное	Подписка	Указывает на необходимость перезагрузки данных при следующей перезагрузке.
Selected Package Result	Адресное	Подписка	Выбранный результат.

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

---

`ChartEditSchemaName` TEXT

Схема редактирования `SchemaName`.

---

`IsEmptyChart` BOOLEAN

Признак пустой схемы.

---

`AnalyticsChartActiveRow` GUID

Активная строка схемы аналитики.

---

`AnalyticsGridData` COLLECTION

Коллекция представлений аналитики реестра.

---

`IsAnalyticsPrintButtonVisible` BOOLEAN

Видимость кнопки печати аналитической формы.

---

`AnalyticsData` COLLECTION

Коллекция данных аналитики.

---

---

```
IsAnalyticsActionButtonsContainerVisible BOOLEAN
```

Признак видимости кнопок аналитических действий.

---

```
AnalyticsDataViewName TEXT
```

Посмотреть название раздела аналитики.

---

```
IsBindDataActionVisible BOOLEAN
```

Признак привязки данных.

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

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

## Методы

---

```
checkCanManageAnalytics()
```

Проверяет, имеет ли пользователь право выполнять CRUD-операции с расписанием. Права доступа устанавливаются системной операцией [ *Настройка аналитики* ] ([ *Analytics setup* ], код `CanManageAnalytics` ).

---

```
onCanManageAnalytics(result)
```

Устанавливает атрибут `CanManageAnalytics` в зависимости от значения запрошенных системной настройки [ *Отображать Демо ссылки* ] ([ *Display demo links* ], код `ShowDemoLinks` ) и системной операции [ *Настройка аналитики* ] ([ *Analytics setup* ], код `CanManageAnalytics` ).

# Деактивация записей объектов



Сложный

Creatio предоставляет возможность деактивировать записи объектов системы для исключения их из бизнес-логики. Это может понадобиться, например, если данные устарели и больше не используются.

## Использование в дизайнере объекта

Функциональность включается специальным свойством [ *Разрешить деактивацию записей* ] ([ *Allow record deactivation* ]) в дизайнере объектов и становится доступной после публикации объекта.

## Behavior

- Represents Structure of Database View
- Is object available as section on SSP
- Virtual
- Allow records deactivation
- Update change log

Функциональность деактивации записей объектов доступна для всех объектов, но автоматическая фильтрация записей работает только в выпадающих списках, на странице выбора из справочника и в быстрых фильтрах. На страницах с содержимым справочников, в расширенных фильтрах и разделах автоматический фильтр не применяется.

## Использование в программном коде

За включение или отключение фильтрации по неактивным записям отвечает параметр `UseRecordDeactivation` класса `EntitySchemaQuery`. По умолчанию этот параметр имеет значение `false`. Если ему присвоить значение `true`, то в запрос на выборку данных из объекта, где включена деактивация записей, будет добавлен фильтр, исключающий неактивные записи.

### Пример использования в клиентском коде

```
var esq = Ext.create("Terrasoft.EntitySchemaQuery", {
    rootSchemaName: "MyCustomLookup",
    useRecordDeactivation: true
});
```

### Пример использования в серверном коде

```
var esq = new EntitySchemaQuery(userConnection.EntitySchemaManager, "ContactType") {
    UseRecordDeactivation = true
};
esq.PrimaryQueryColumn.IsAlwaysSelect = true;
```

Результирующий SQL-запрос в данном случае будет иметь следующий вид:

### SQL-запрос

```
SELECT
[ContactType].[Id] [Id]
```

```
FROM [dbo].[ContactType] [ContactType] WITH(NOLOCK)
WHERE
    [ContactType].[RecordInactive] = 0
```

# 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 [Настроить поля страницы](#).

## Добавить простое поле с использованием Creatio IDE

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

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

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

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

Ниже рассмотрен один из способов.

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

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

## Добавить поле с изображением

1. Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в

статье [Разработка конфигурационных элементов](#).

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

## Добавить вычисляемое поле

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

В Creatio работа вычисляемых полей основана через подписку на события изменения атрибутов схемы модели представления страницы. При изменении значений колонок схемы объекта должно 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 объявите свойство `dependencies`, которое содержит массив конфигурационных объектов. Свойства свойства `dependencies`:
    - `columns` — массив имен колонок, от значений которых зависит значение текущей колонки.
    - `methodName` — имя метода-обработчика, который вызывается при изменении значения хотя бы одной из перечисленных колонок.
  - d. В свойстве `methods` реализуйте:
    - Метод-обработчик события изменения колонки, от которой зависит вычисляемая колонка.
    - `onEntityInitialized()` — переопределенный базовый виртуальный метод. Срабатывает после выполнения инициализации схемы объекта страницы записи. В метод `onEntityInitialized()` добавьте вызов метода-обработчика, который обеспечит расчет вычисляемого поля в момент открытия страницы записи, а не только после изменений в колонках-зависимостях.
  - g. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля.

## Добавить мультивалютное поле

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

Мультивалютное поле **позволяет**:

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

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

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

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

- Колонку для хранения суммы в базовой валюте.

В схеме объекта может быть определена только одна колонка — для хранения общей суммы в выбранной валюте. Остальные колонки могут являться виртуальными, если бизнес-задача не предполагает хранения в базе данных их значений. Они могут быть определены как атрибуты в схеме модели представления.

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

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

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

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

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

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

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

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

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

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

- `primaryAmount` — наименование колонки, которая содержит сумму в базовой валюте.

- `currency` — наименование колонки, которая ссылается на справочник валют.
- `rate` — наименование колонки, которая содержит курс валюты.
- `generator` — генератор элемента управления. Для мультивалютного поля укажите `MultiCurrencyEditViewGenerator.generate`.

## Реализовать валидацию поля

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

Чтобы **реализовать валидацию поля**:

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

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

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

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

## Установить для поля значение по умолчанию

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

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

При создании нового объекта некоторые поля страницы необходимо заполнить соответствующими значениями. В этом случае для колонок объекта, которые соответствуют этим полям, в 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 <b>типы</b> колонок, для которых можно установить константу в качестве значения по умолчанию: <ul style="list-style-type: none"> <li>• [ Стока ] ([ String ]). </li> <li>• [ Число ] ([ Number ]). </li> <li>• [ Справочник ] ([ Lookup ]). </li> <li>• [ Логическое ] ([ Boolean ]). </li> </ul>
[ 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 ])	<b>Системные переменные</b> — глобальные переменные, которые хранят информацию о настройках системы. Значение системной переменной устанавливается на уровне ядра приложения, а не на уровне пользователя.

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

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

2. В схеме замещающей модели представления настройте для поля **значение по умолчанию**.

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

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

## Настроить обязательность для поля

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В объявлении класса модели представления в качестве зависимостей добавьте модули `BusinessRuleModule` и `ConfigurationConstants`.
3. В схеме замещающей модели представления настройте **обязательность для поля**.

Для этого в свойстве `rules`:

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

## Настроить фильтрацию значений справочного поля

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

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

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

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

- d. В массив модификаций `diff` добавьте конфигурационный объект с настройками расположения поля.

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

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

1. Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В объявлении класса модели представления в качестве зависимостей добавьте модуль `BusinessRuleModule`.
3. В схеме замещающей модели представления настройте **фильтрацию значений справочного поля**.

Для этого в свойстве `attributes`:

- В свойстве `dataValueType` укажите значение `LOOKUP`, которое устанавливает тип данных колонки. Типы данных колонки представлены перечислением `Terrasoft.core.enums.DataValueType`.
- В свойстве `lookupListConfig` укажите конфигурационный объект поля-справочника.

#### **Опциональные свойства** свойства `lookupListConfig`:

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

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

Параметры метода `add()`

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

## Настроить условия блокировки поля

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

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

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

Чтобы **настроить условия блокировки поля**:

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

- a. В свойстве `attributes` включите блокировку с помощью атрибута `IsModelItemsEnabled`.

**Способы** включения блокировки поля:

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

### Включение блокировки полей страницы записи (способ 1)

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

### Включение блокировки полей страницы записи (способ 2)

```
"IsModelItemsEnabled": {
    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.

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

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

### Пример настройки массива модификаций `diff`

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

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

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

## Настроить исключения блокировки поля

Creatio предоставляет возможность исключить блокировку для полей и деталей.

Чтобы **настроить исключения блокировки поля**:

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

Пример исключения блокировки поля и детали (способ 1)

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

Пример исключения поля и детали (способ 2)

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

```

        }
        return this.callParent(arguments);
    }

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

```

- г. 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

- Создайте схему замещающего объекта. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
- 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](#).
- Создайте схему замещающей модели представления страницы записи, на которой размещено поле. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
- В схеме замещающей модели представления настройте **условия отображения поля**.
  - 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.
  - 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.

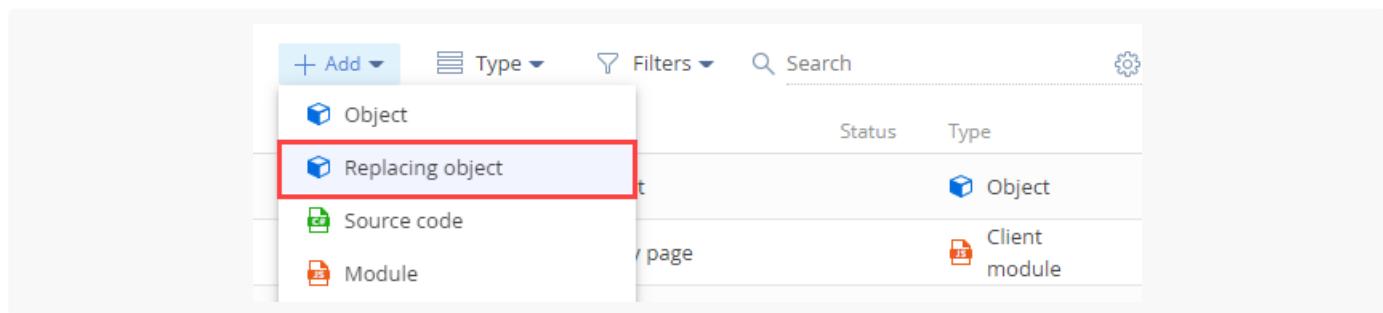
# Добавить поле на страницу записи с использованием новой колонки

 Средний

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

## 1. Создать схему замещающего объекта

- 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](#)]).



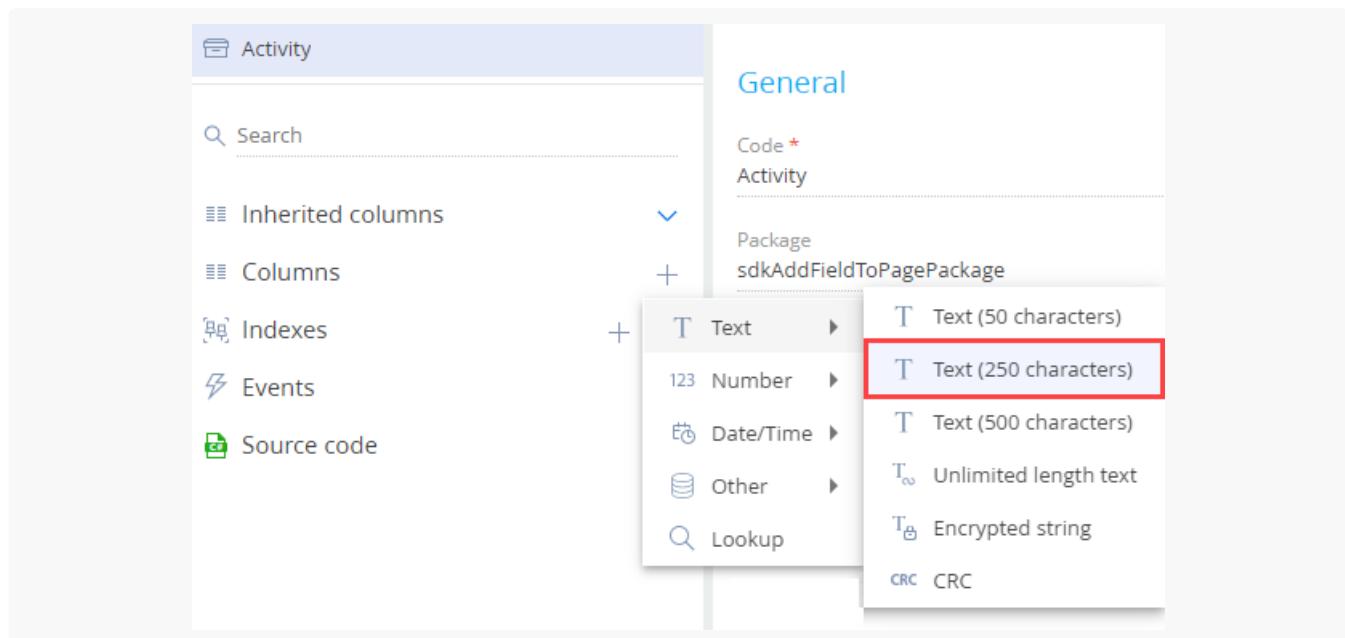
### 3. Заполните **свойства схемы**.

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

General	
Code *	Activity
Package	sdkAddFieldToPagePackage
Title *	
Activity	
Description	
<b>Inheritance</b>	
Parent object *	Activity
<input checked="" type="checkbox"/> Replace parent	

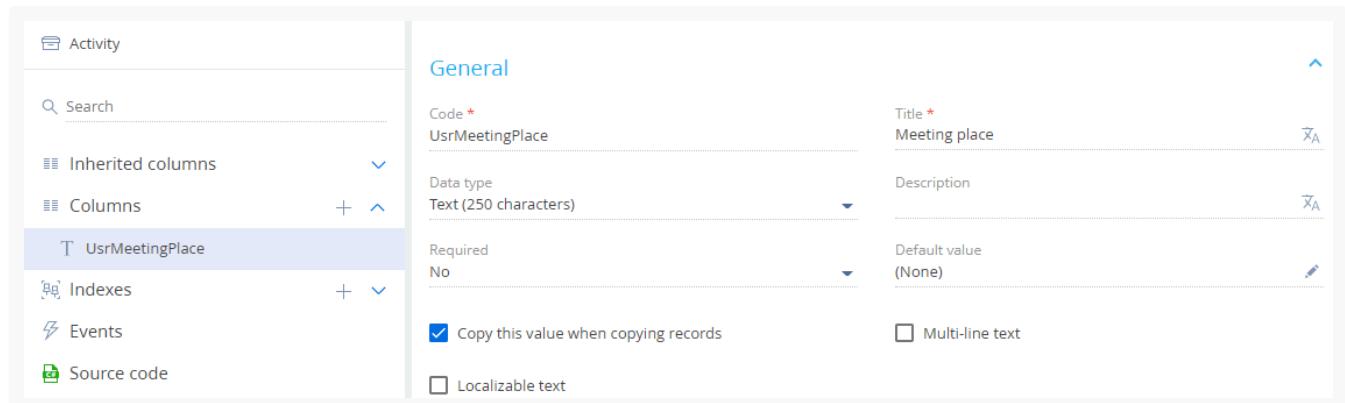
### 4. В схему добавьте **колонку**.

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



c. Заполните **свойства добавляемой колонки**.

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

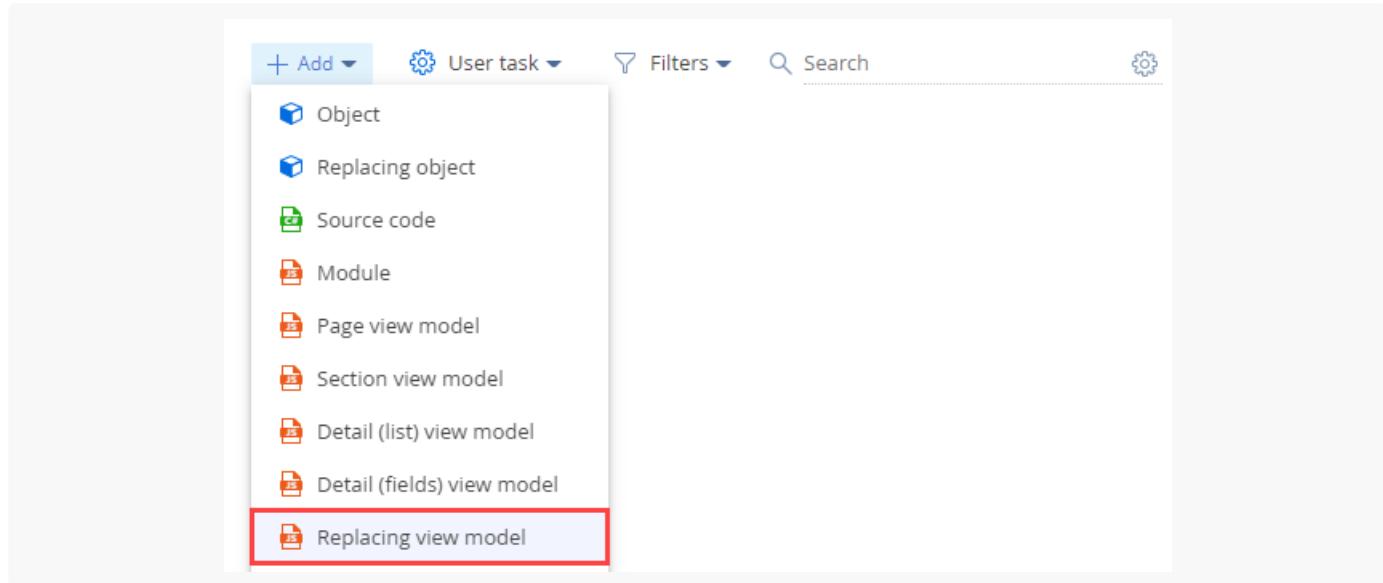


- На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]), а затем [ Опубликовать ] ([ Publish ]).

## 2. Создать схему замещающей модели представления страницы активности

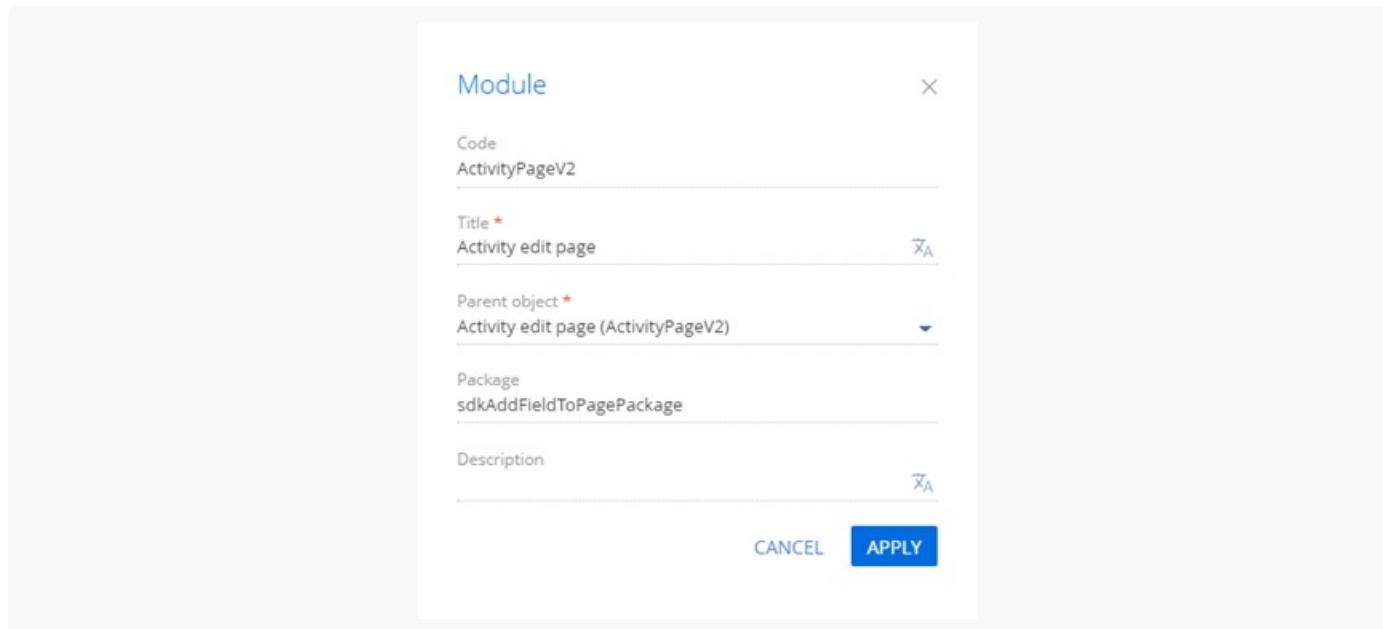
- [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель

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



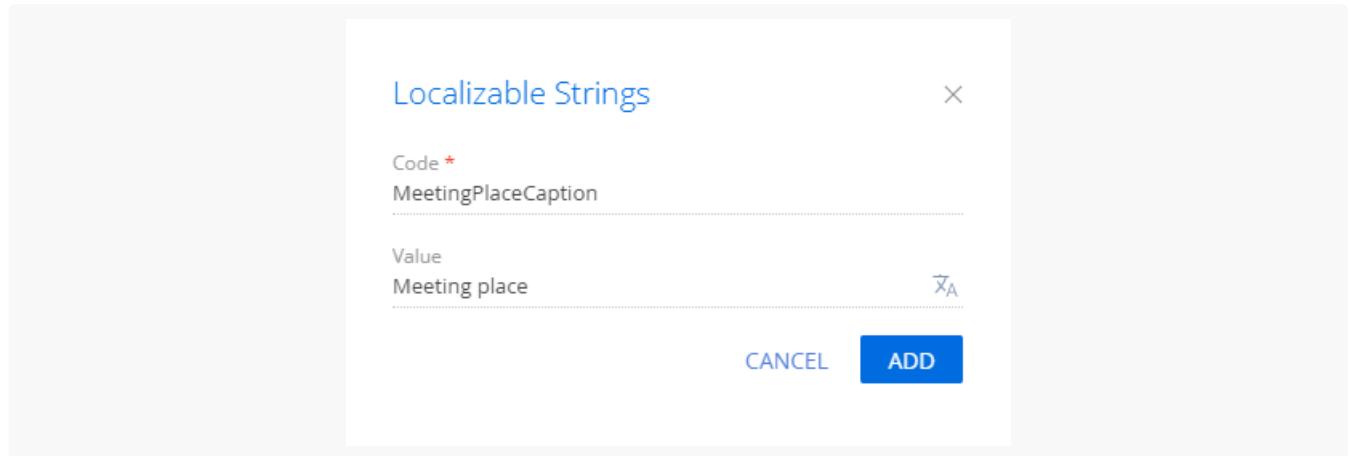
### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "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*/
);
});

```

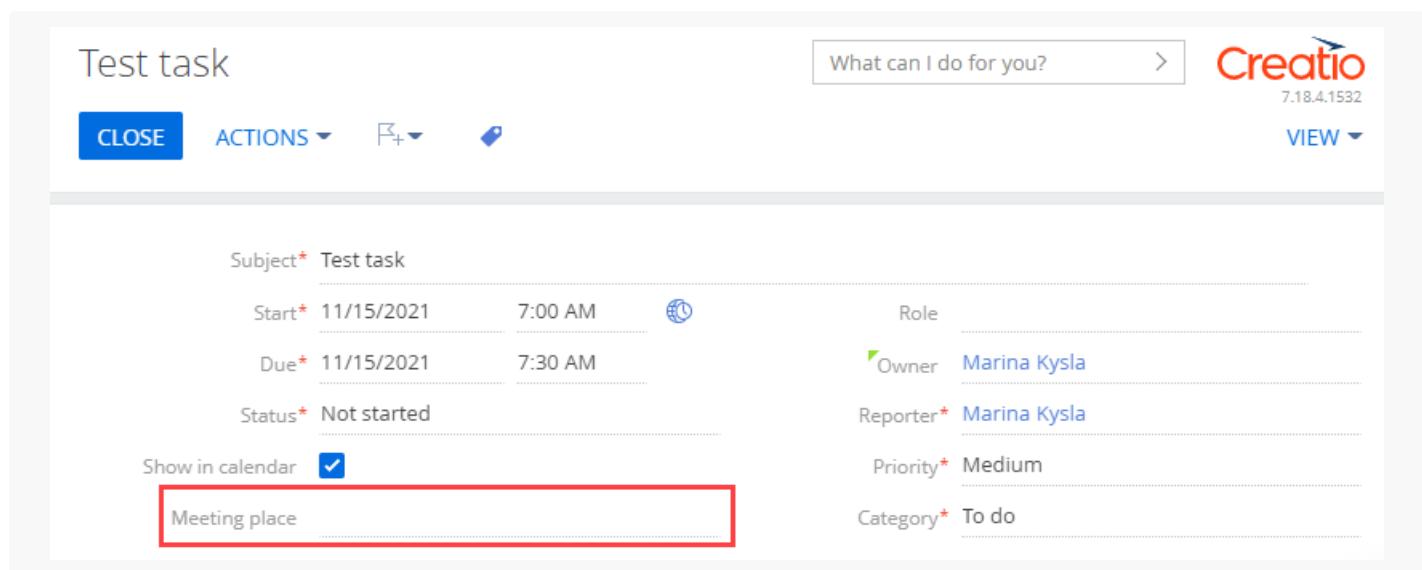
- На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

## 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 ]).



The screenshot shows a task record in the Creatio application. The task details are as follows:

- Subject:** Test task
- Start:** 11/15/2021 at 7:00 AM
- Due:** 11/15/2021 at 7:30 AM
- Status:** Not started
- Show in calendar:** checked
- Meeting place:** (This field is highlighted with a red border.)
- Role:** Owner: Marina Kysla, Reporter: Marina Kysla
- Priority:** Medium
- Category:** To do

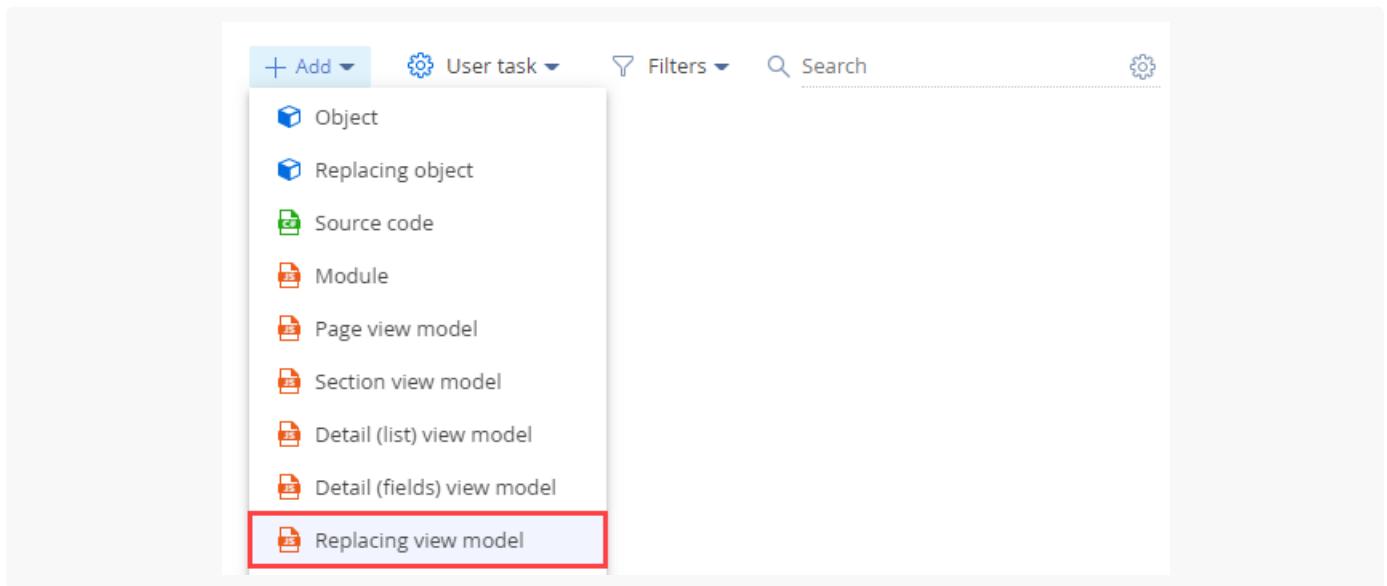
## Добавить поле на страницу записи с использованием существующей колонки

 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 fields filled:

- 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

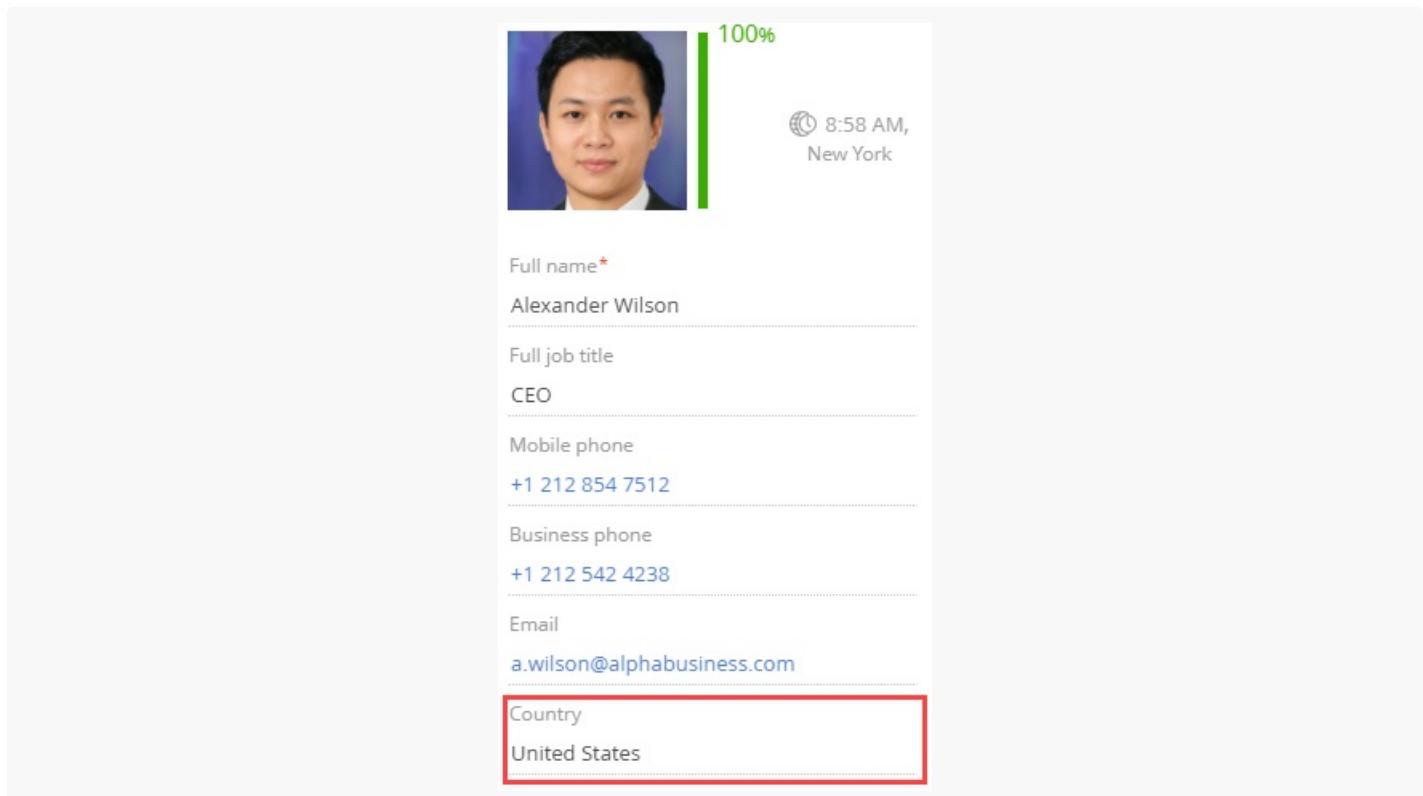
```
define("ContactPageV2", [], function() {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение поля на странице записи. */
        diff: [
            /* Метаданные для добавления на страницу поля [Страна]. */
            {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



100%

8:58 AM,  
New York

Full name\*

Alexander Wilson

Full job title

CEO

Mobile phone

+1 212 854 7512

Business phone

+1 212 542 4238

Email

a.wilson@alphabusiness.com

Country

United States

## Добавить поле с изображением на страницу записи



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

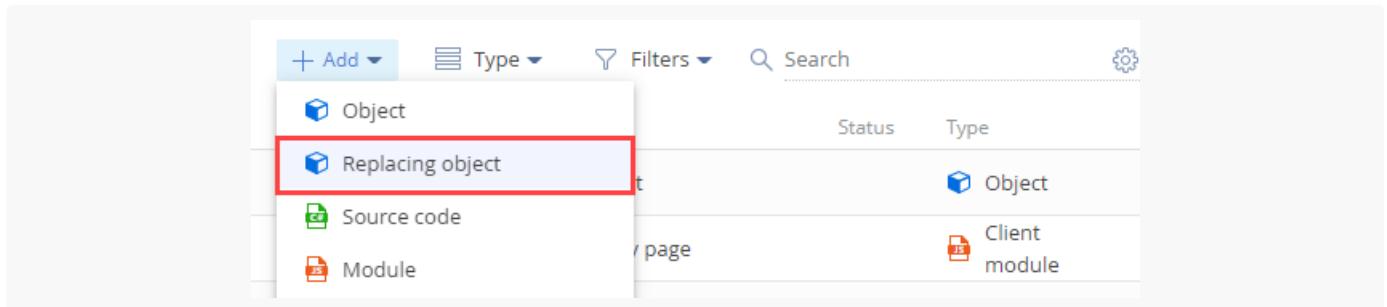


### 1. Создать схему замещающего объекта

- 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 ]).



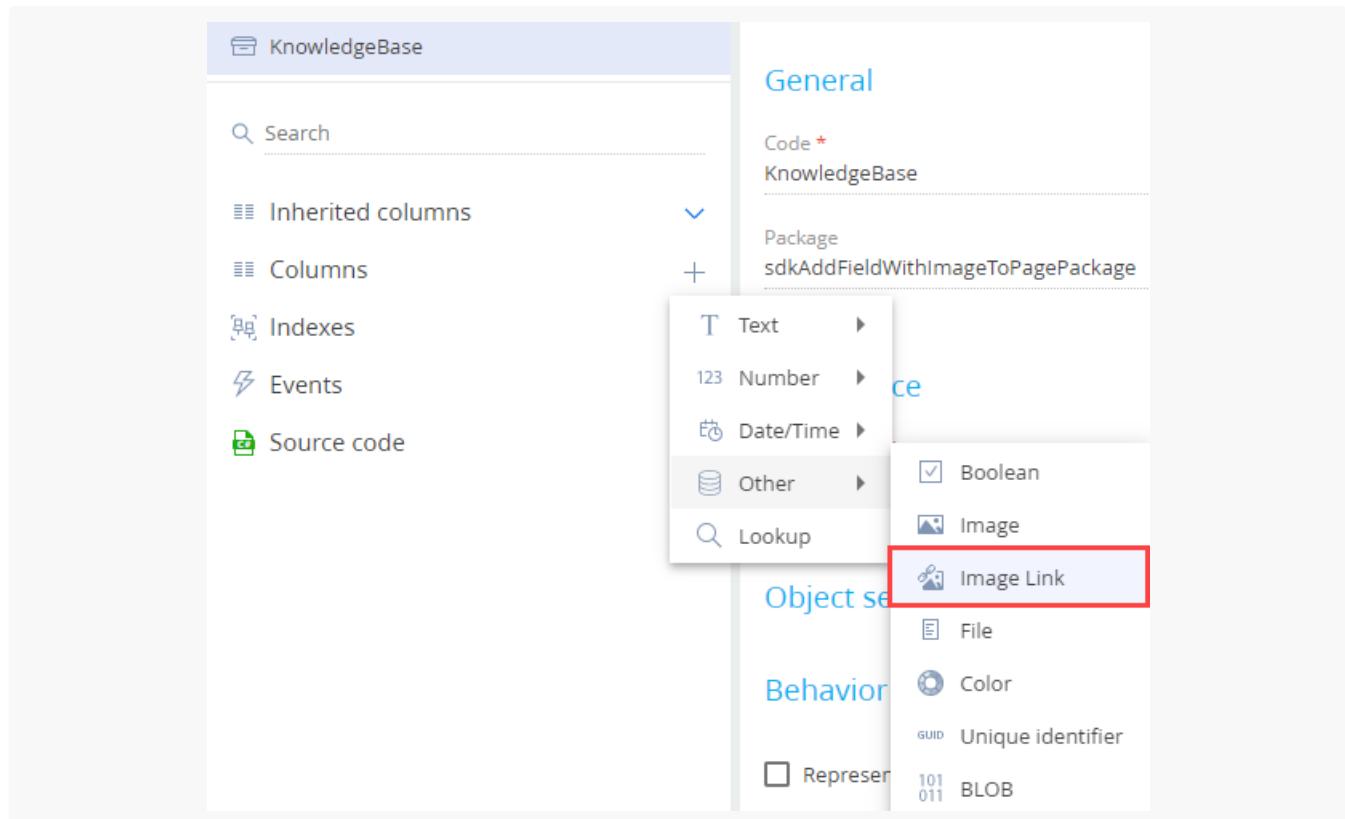
- Заполните **свойства схемы**.

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

The screenshot shows the 'General' tab of a schema configuration. It includes fields for 'Code' (KnowledgeBase), 'Title' (Knowledge base article), 'Package' (sdkAddFieldWithImageToPagePackage), and 'Description'. In the 'Inheritance' section, 'Parent object' is set to 'KnowledgeBase' and the 'Replace parent' checkbox is checked.

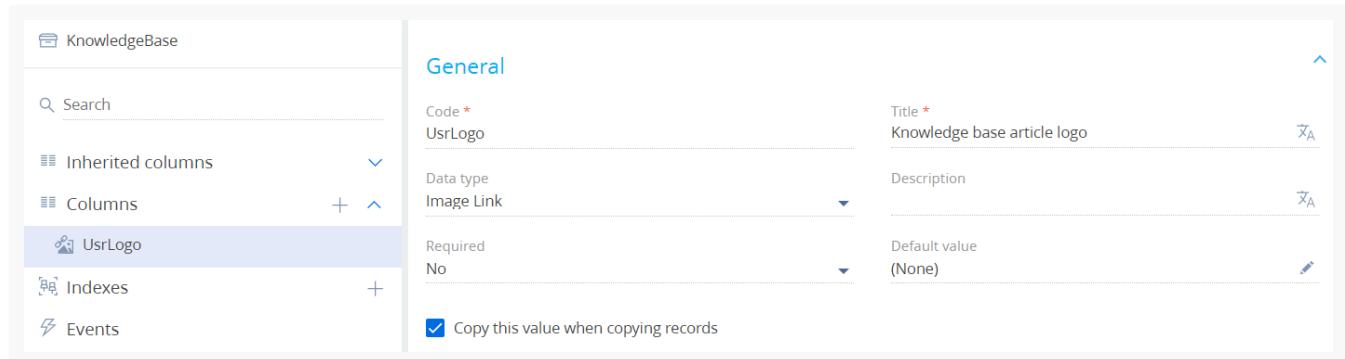
- В схему добавьте **колонку**.

- 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**.

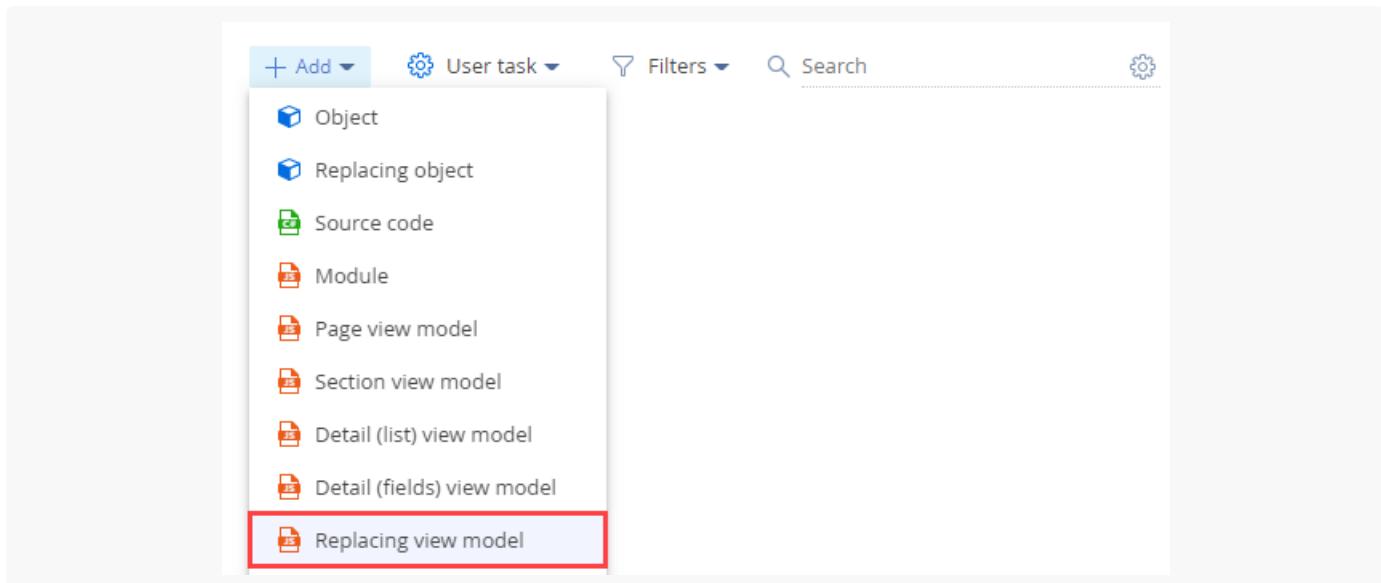
- [ Код ] ([ Code ]) — "UsrLogo".
- [ Заголовок ] ([ Title ]) — "Логотип статьи базы знаний" ("Knowledge base article logo").



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

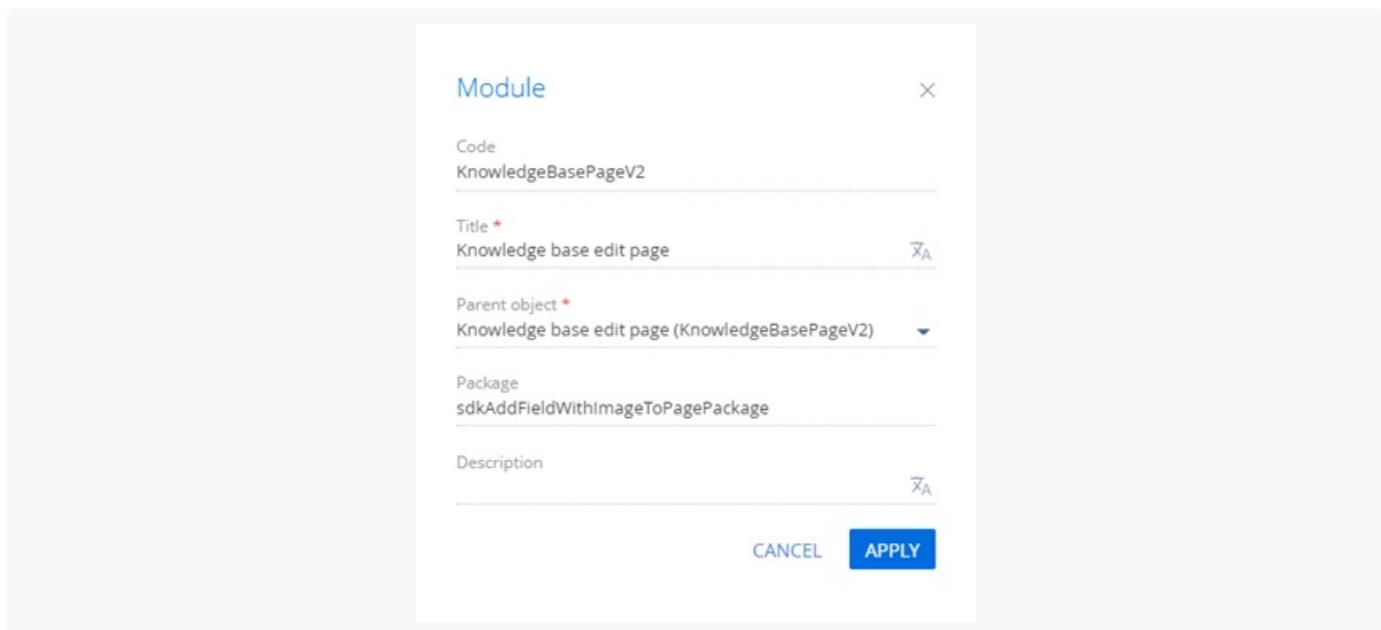
## 2. Создать схему замещающей модели представления страницы статьи базы знаний

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 ]).



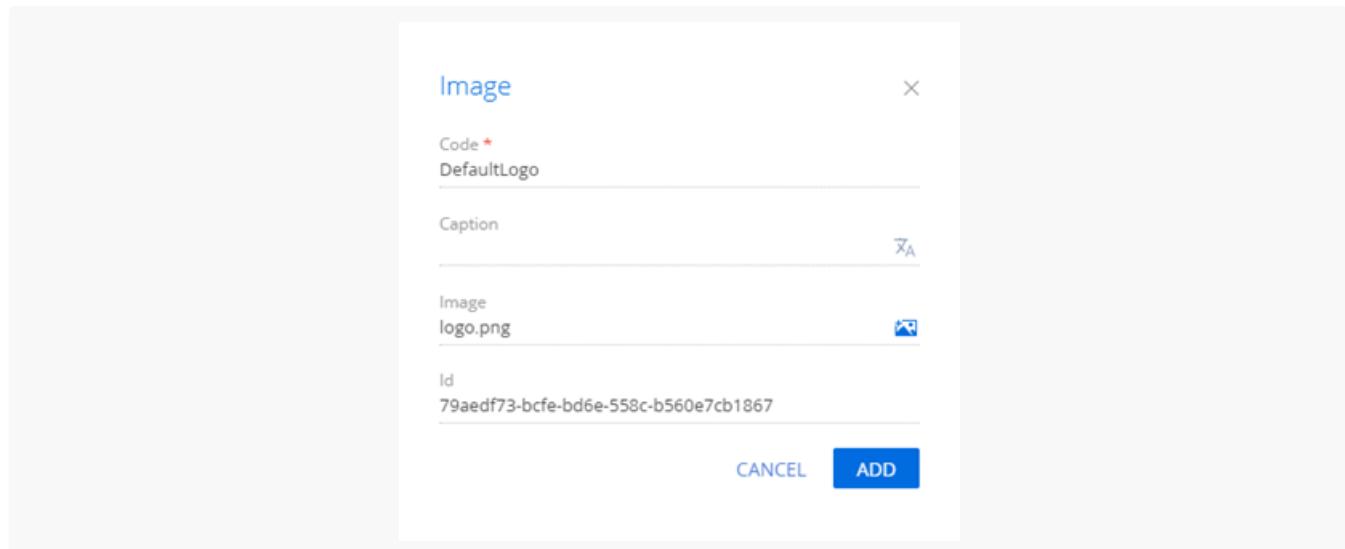
### 3. Заполните **свойства схемы**.

- [ 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.Defau
    },
    /* 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 о
        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 изображе
    {
        /* 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 list of objects in a software application. The top navigation bar includes 'Add', 'Type', 'Filters', 'Search', and a gear icon. Below the bar is a table with columns 'Status' and 'Type'. The table contains four rows: 'Object' (Status: Draft, Type: Object), 'Replacing object' (Status: Draft, Type: Object), 'Source code' (Status: Draft, Type: Client module), and 'Module' (Status: Draft, Type: Client module). A red box highlights the 'Replacing object' row.

### 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 main sections: 'General' and 'Inheritance'.

**General** tab:

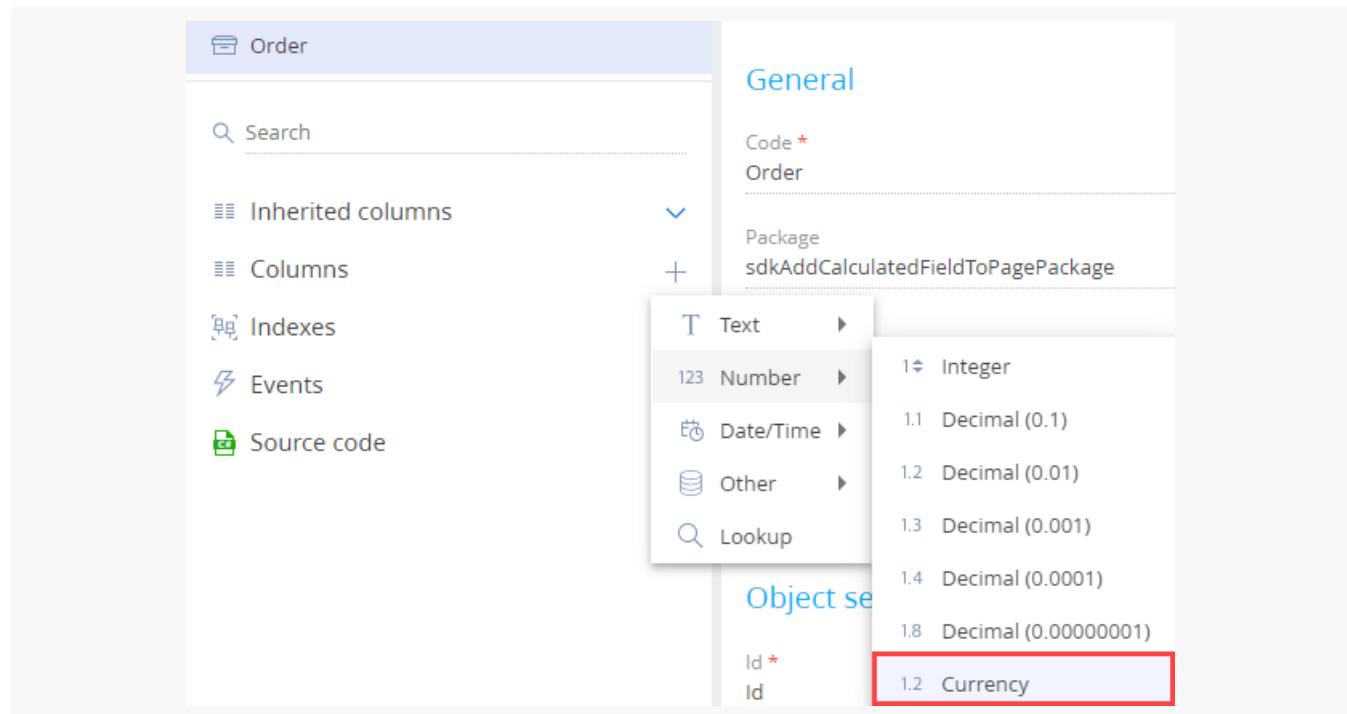
- Code \***: Order
- Title \***: Order
- Package**: sdkAddCalculatedFieldToPagePackage
- Description**

**Inheritance** tab:

- Parent object \***: Order
- Replace parent

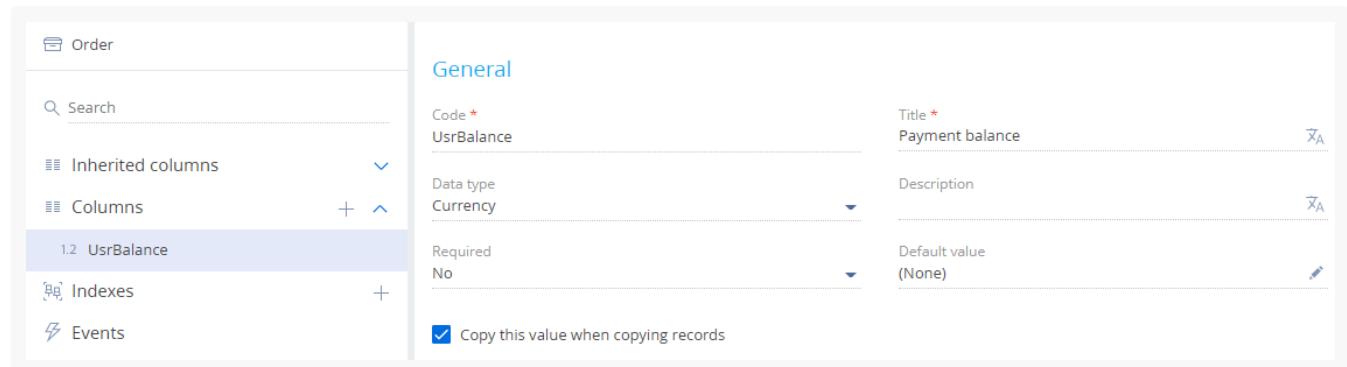
### 4. В схему добавьте **колонку**.

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



### с. Заполните свойства добавляемой колонки.

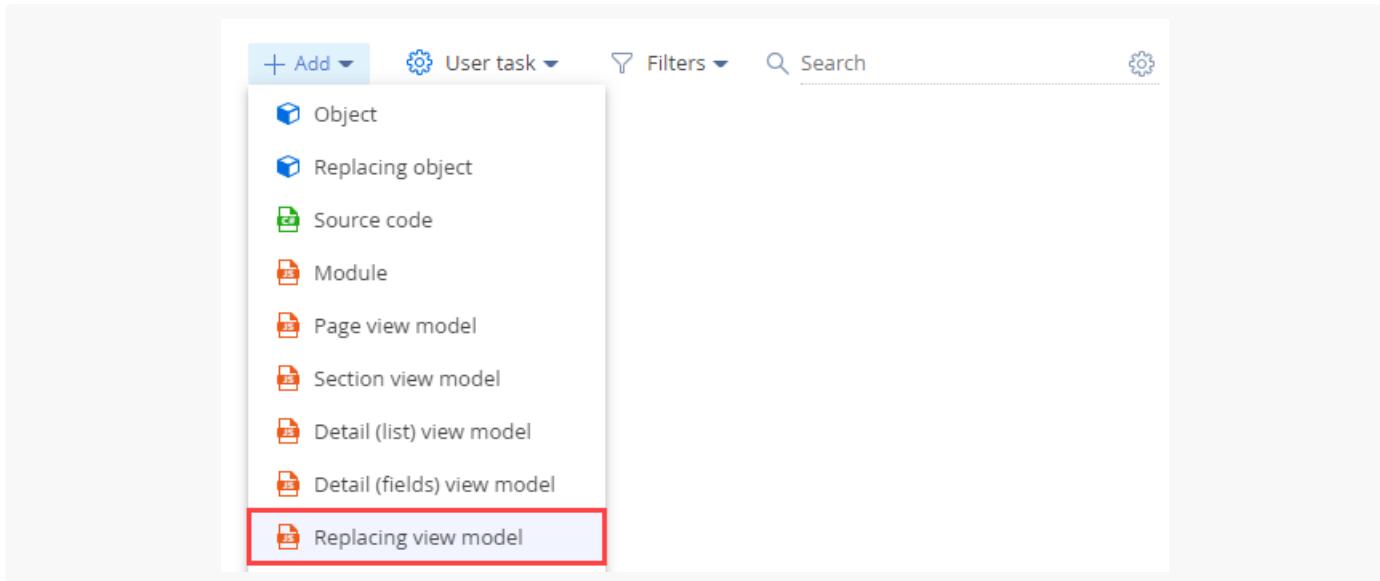
- [ Код ] ([ Code ]) — "UsrBalance".
- [ Заголовок ] ([ Title ]) — "Остаток для оплаты" ("Payment balance").



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

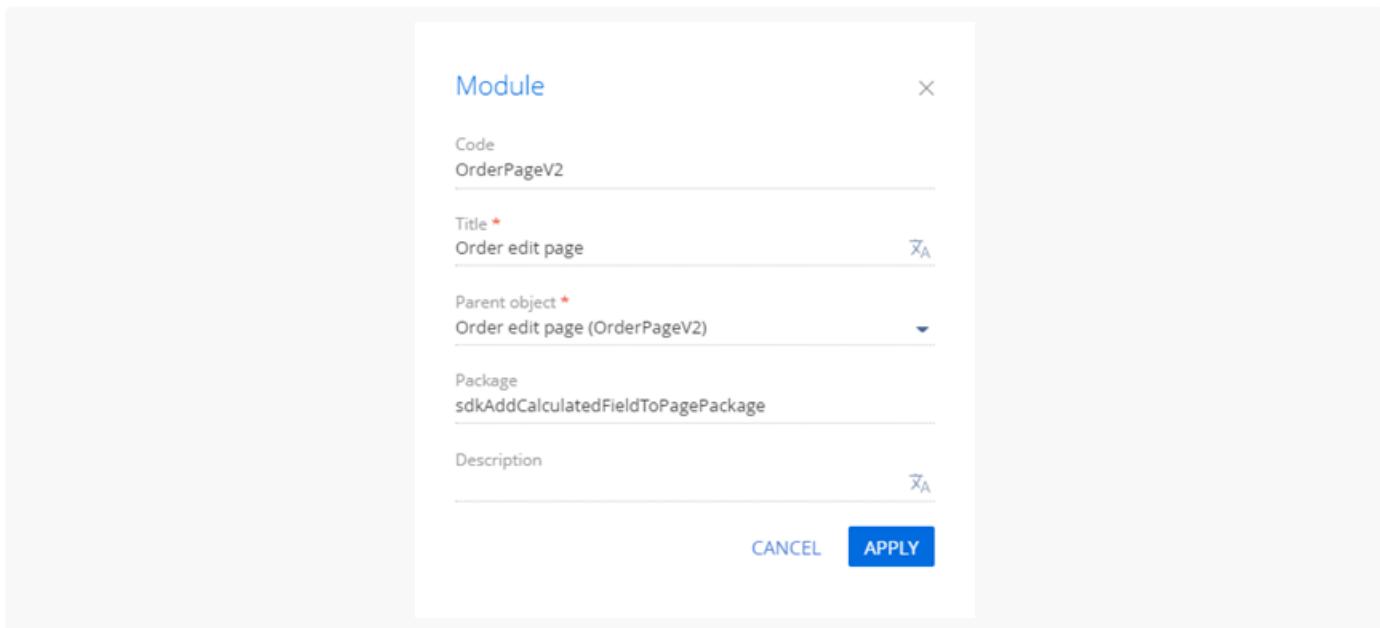
## 2. Создать схему замещающей модели представления страницы заказа

- [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

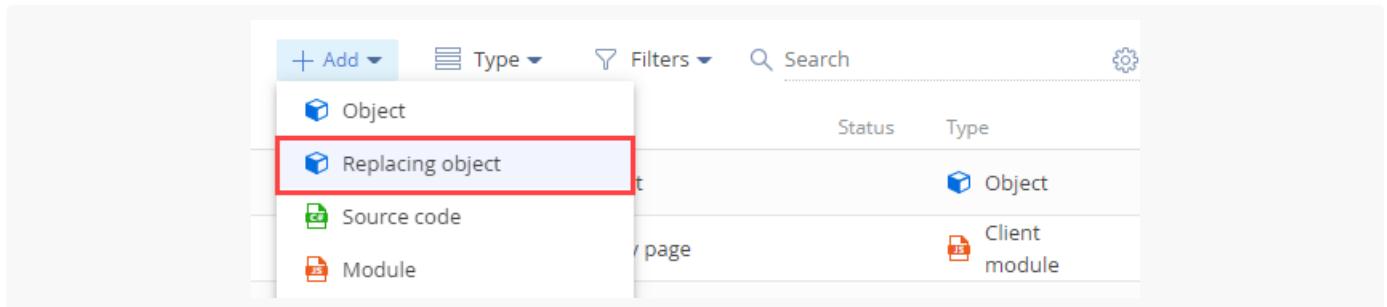
**Пример.** Добавить мультивалютное поле [ Общая сумма ] ([ Amount ]) на страницу проекта.

### 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 ]).



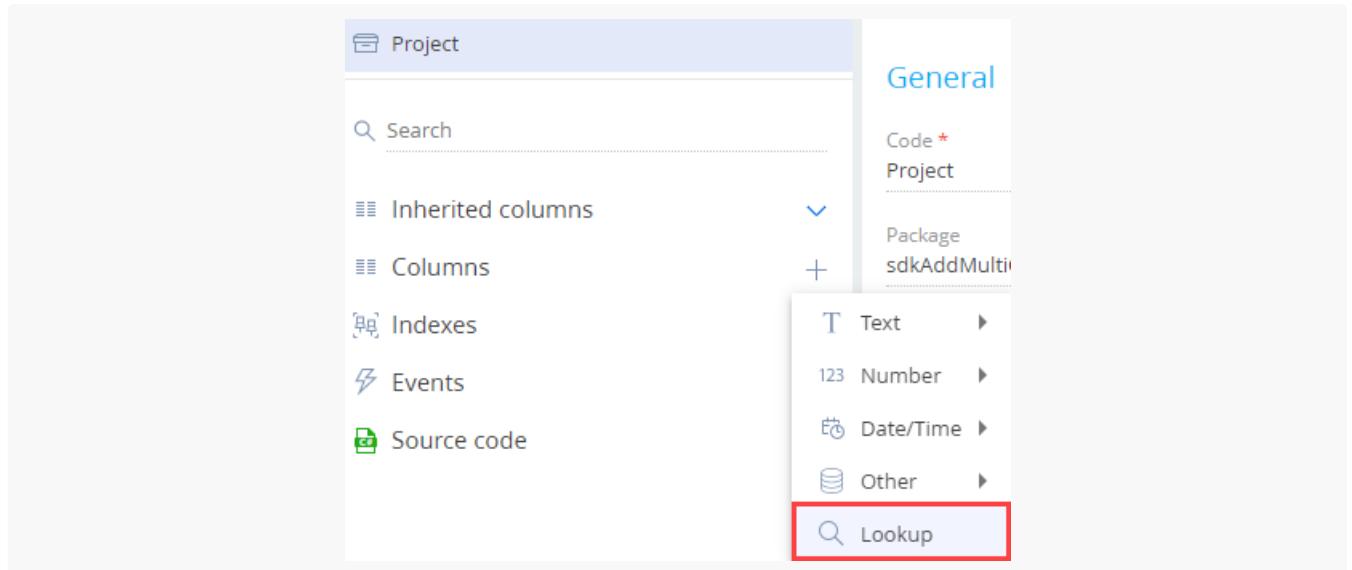
- Заполните **свойства схемы**.

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

The screenshot shows the 'General' tab of a schema configuration. It includes fields for 'Code \*' (set to 'Project'), 'Title \*' (set to 'Project'), 'Package' (set to 'sdkAddMultiCurrencyFieldToPagePackage'), and 'Description'. Below this is the 'Inheritance' tab, which shows 'Parent object \*' set to 'Project' and a checked checkbox for 'Replace parent'.

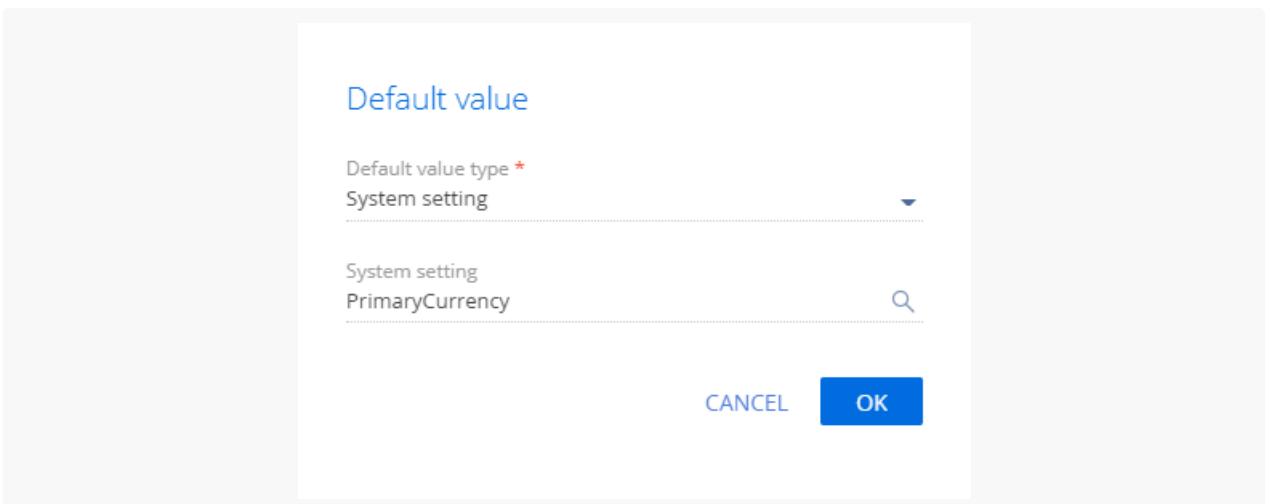
- В схему добавьте **колонку**.

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

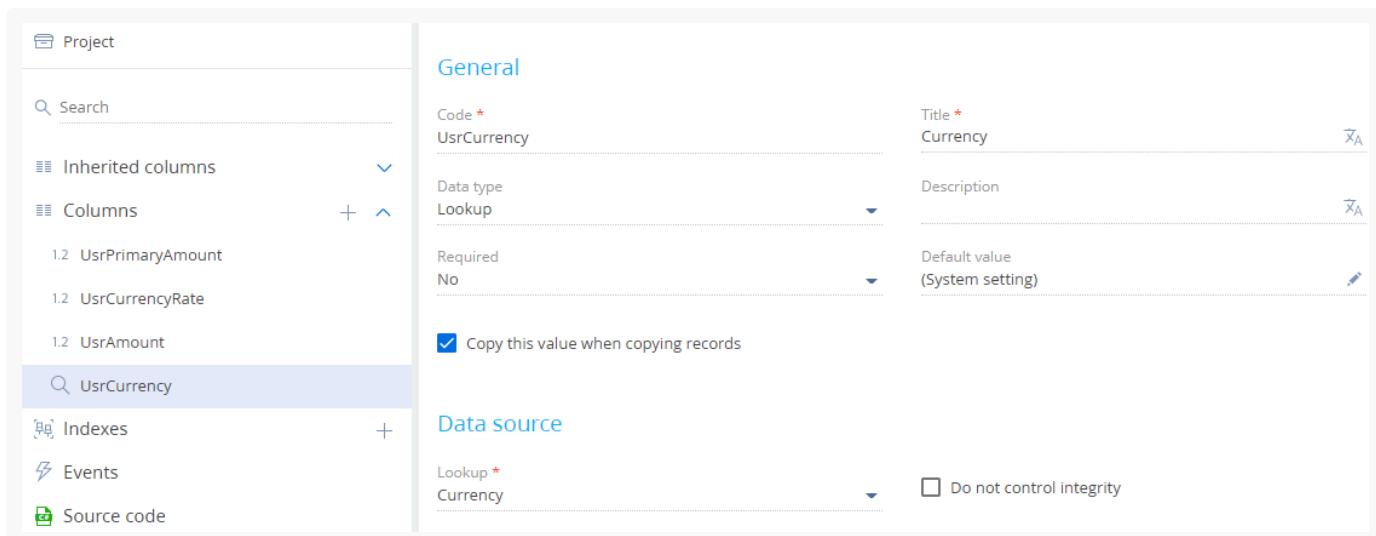


с. Заполните **свойства добавляемой колонки**.

- [ 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.

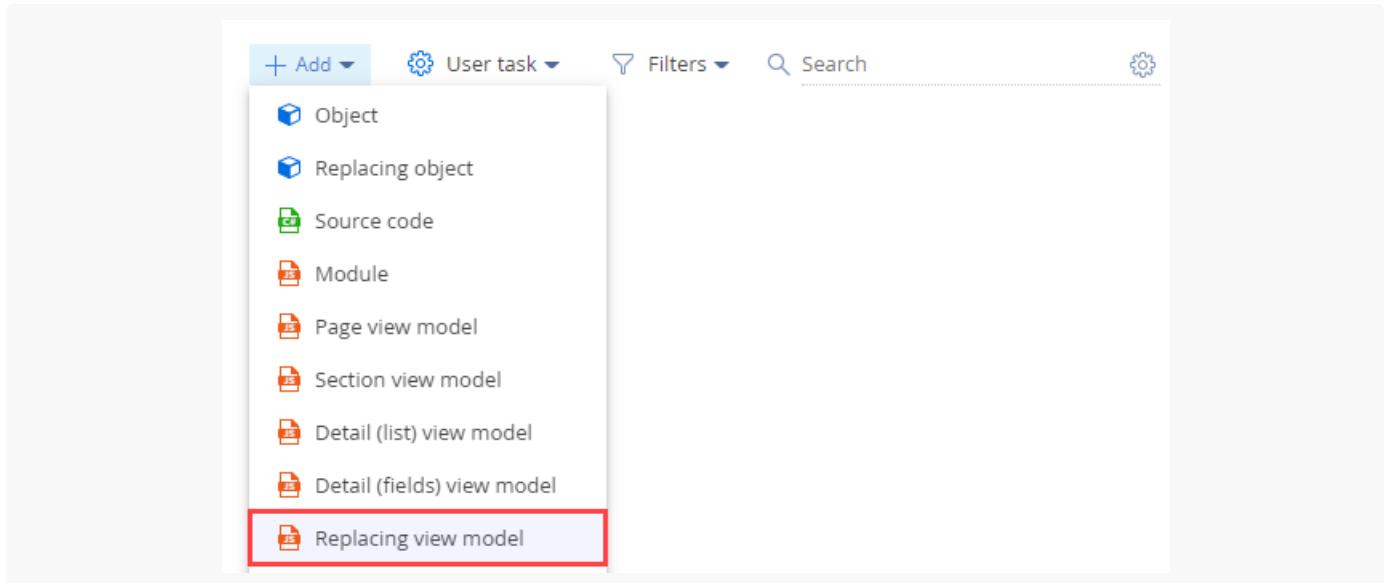
Свойства добавляемых колонок

[ 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 ]).

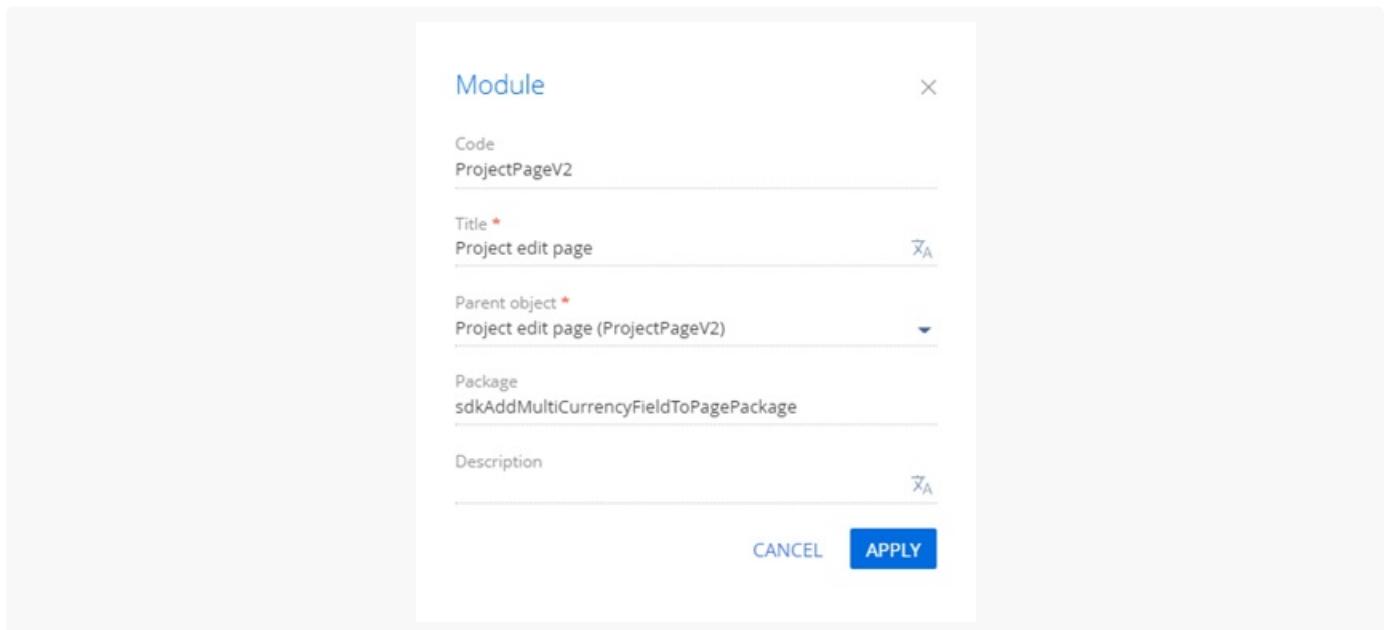
## 2. Создать схему замещающей модели представления страницы проекта

- 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 INFO STRUCTURE FINANCIAL INDICATORS HISTORY ATTACHMENTS AND NOTES >

Contact Andrew Wayne

Type\* Maintenance

Значение поля автоматически пересчитывается после выбора валюты в выпадающем списке.

Software Documentation Development

What can I do for you? > Creatio 7.18.4.1532

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

Name\* Software Documentation Development

Status\* In progress

Owner\* Sarah M. Richards

Amount, \$ ▾ 16.00

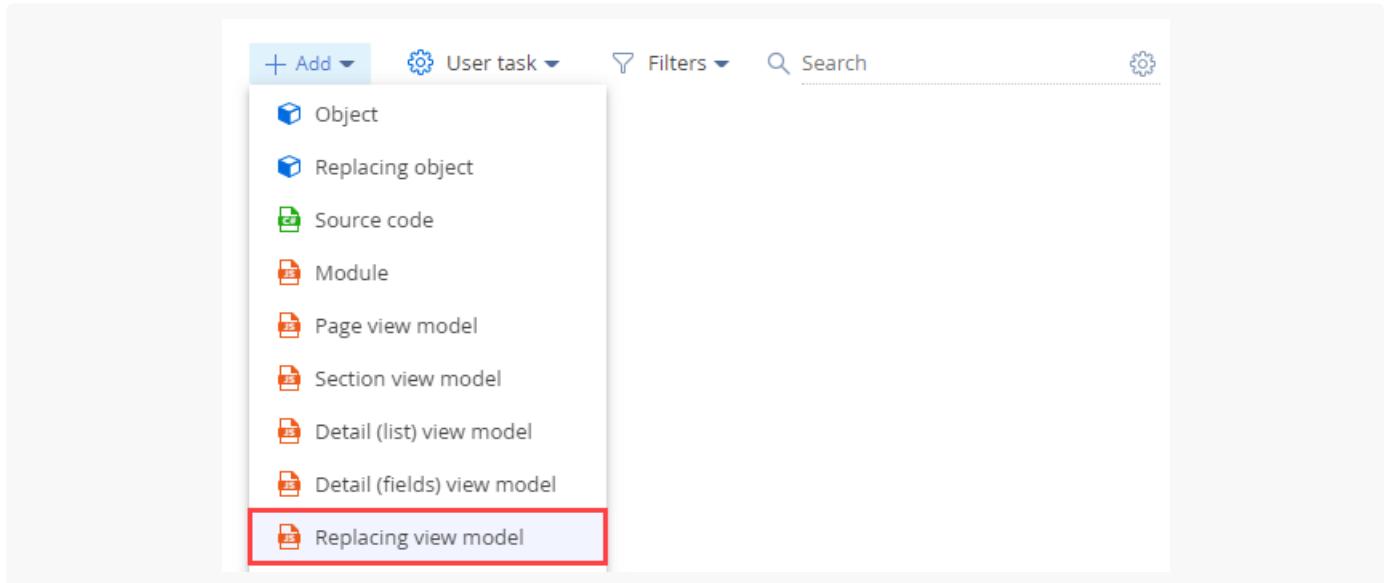
## Реализовать валидацию поля типа [Дата/Время] на странице записи



**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".

**Module**

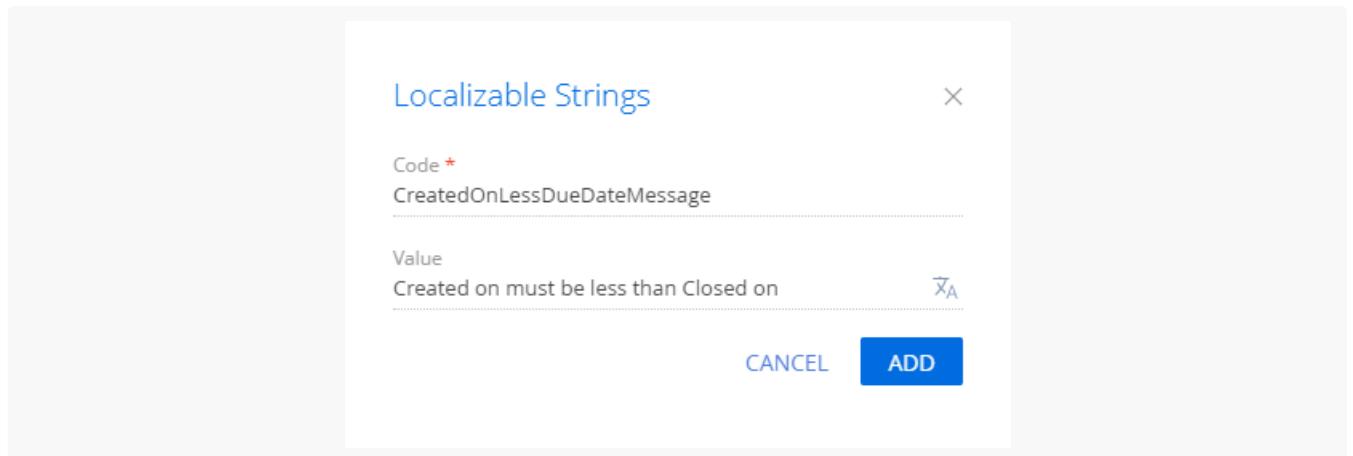
Code	OpportunityPageV2
Title *	Opportunity edit page
Parent object *	Opportunity edit page (OpportunityPageV2)
Package	sdkDateTimeFieldValidationPackage
Description	

CANCEL    **APPLY**

### 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 ]).

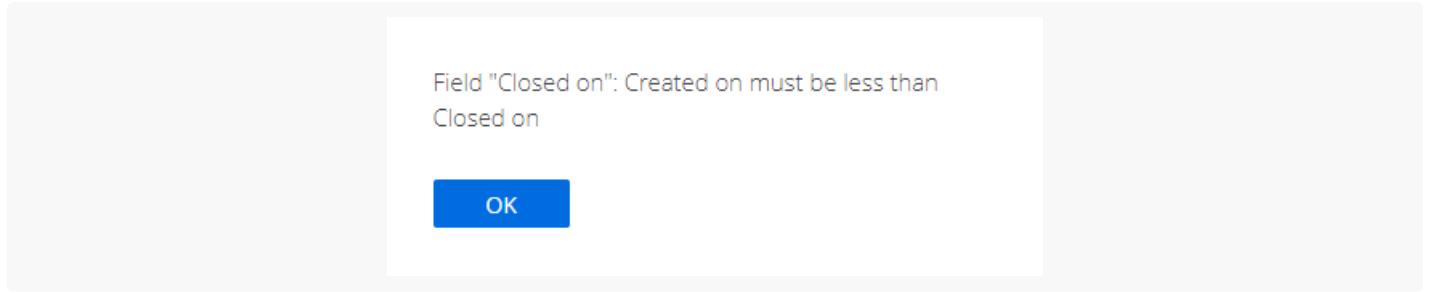
## Результат выполнения примера

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.

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



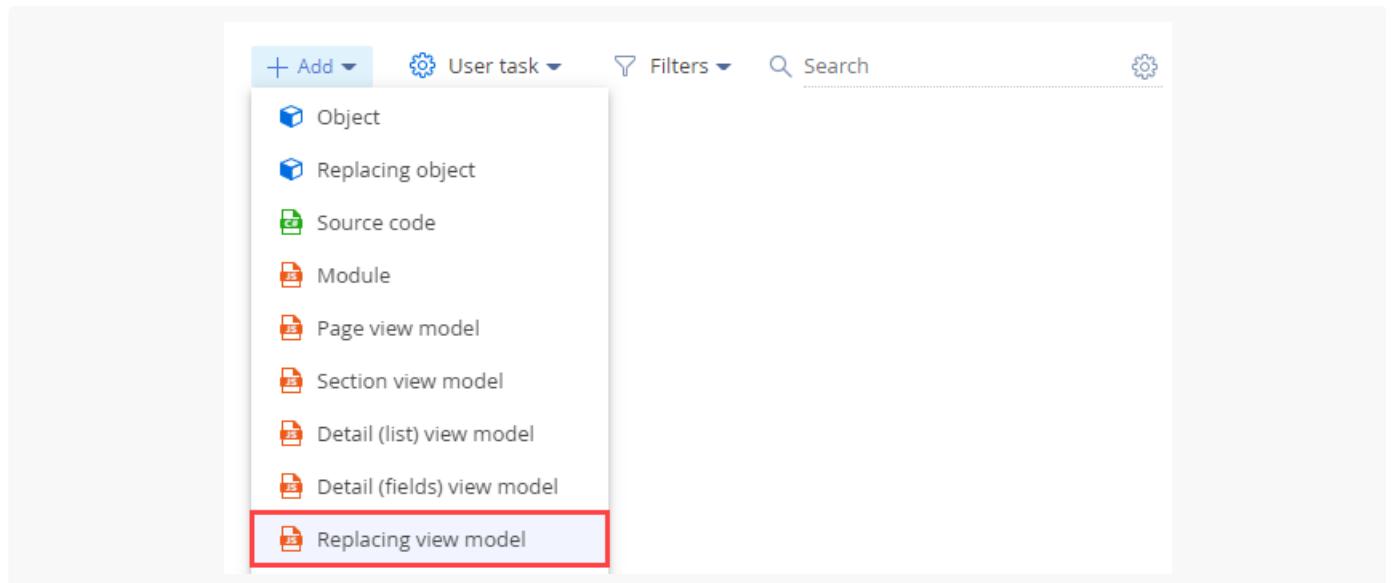
## Реализовать валидацию поля типа [Строка] на странице записи

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

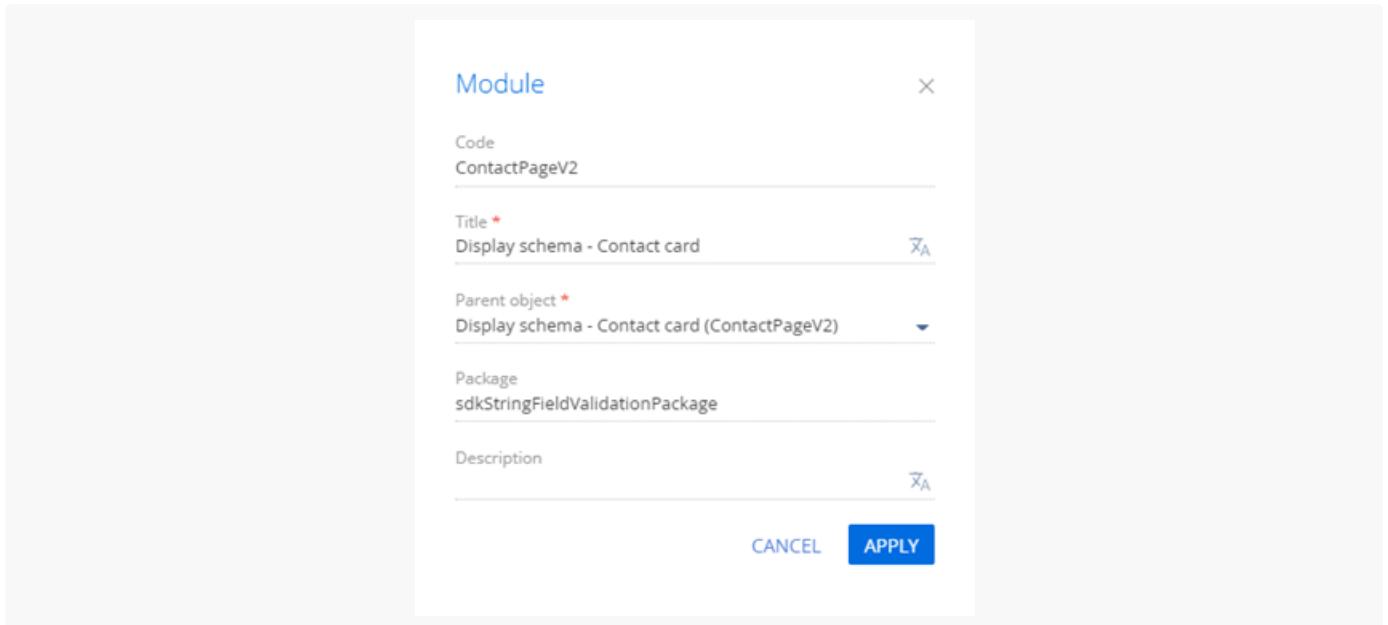
1. [Перейдите в раздел \[ Конфигурация \]](#) ([ *Configuration* ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Замещающая модель представления ] ([ *Add* ] —> [ *Replacing view model* ]).



3. Заполните **свойства схемы**.

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

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

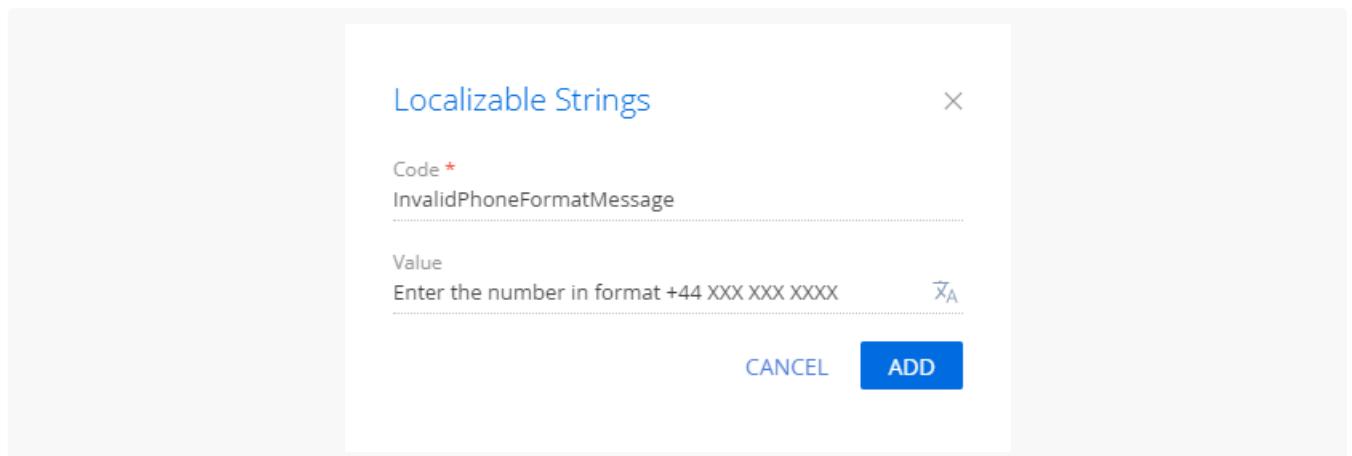


#### 4. Добавьте локализуемую строку.

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 возвращается пустой объект. */
                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

SAVE CANCEL ACTIONS ▾

**NEXT STEPS (8)**

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

Full name\*  
Alexander Wilson

Full job title  
CEO

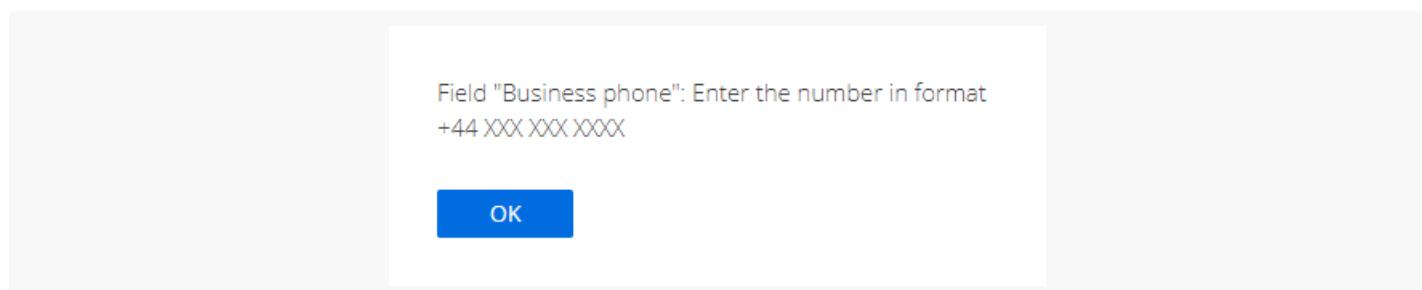
Mobile phone  
+1 212 854 7512

Business phone  
+1 212 542 4238

Enter the number in format +44 XXX XXX XXXX  
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

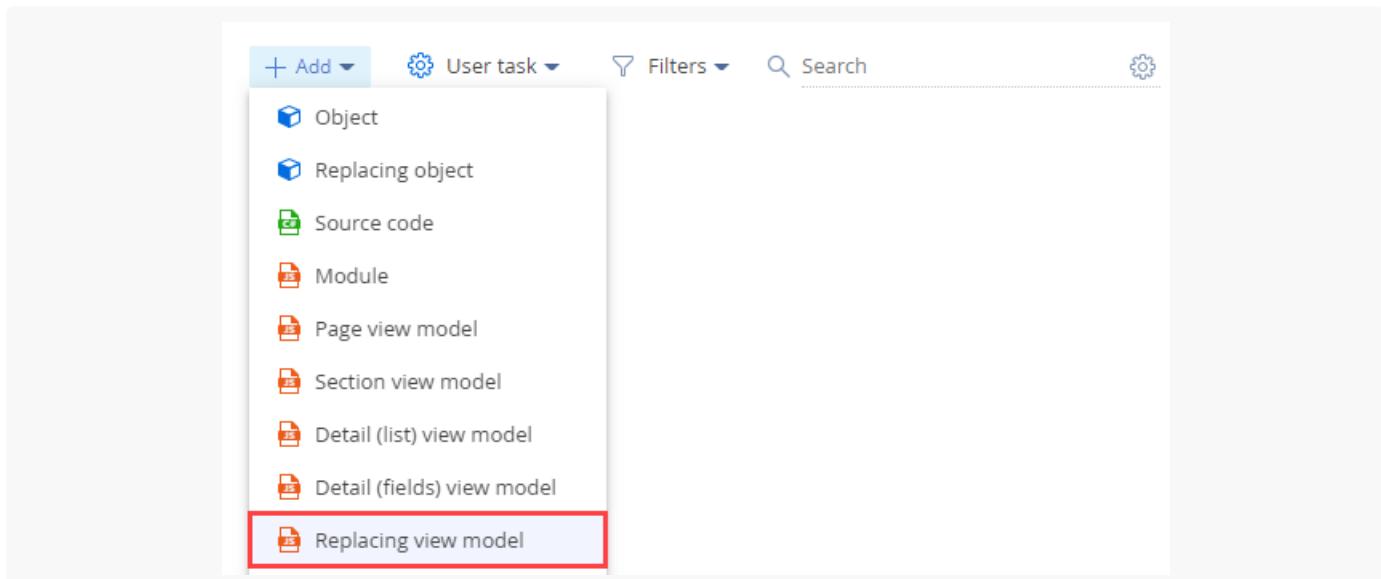
# Поля на странице записи

 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 ]).

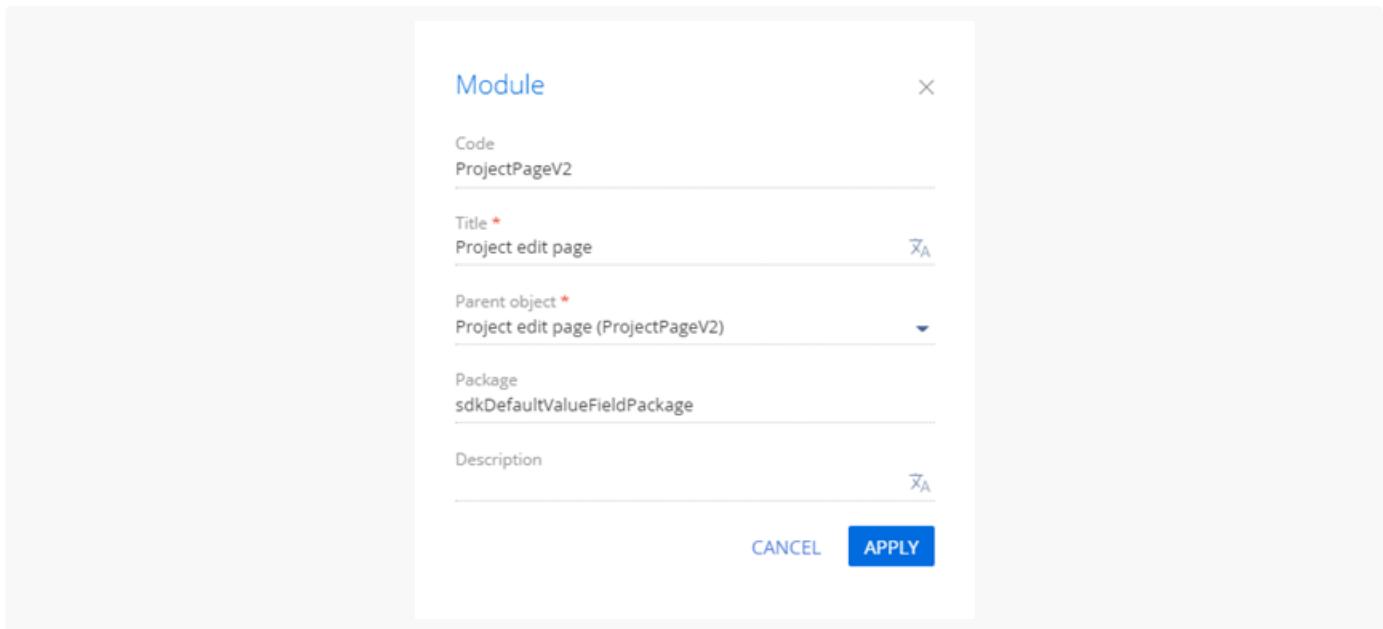
## Создать схему замещающей модели представления страницы проекта

- 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. 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 <b>Completion %</b> <b>Calculate automatically</b> <input type="checkbox"/> <b>Start</b> <b>11/19/2021</b> <b>End</b>		Contact  <b>Type*</b>	<b>Duration</b> 0 min <b>Deadline</b> <b>11/29/2021</b>			

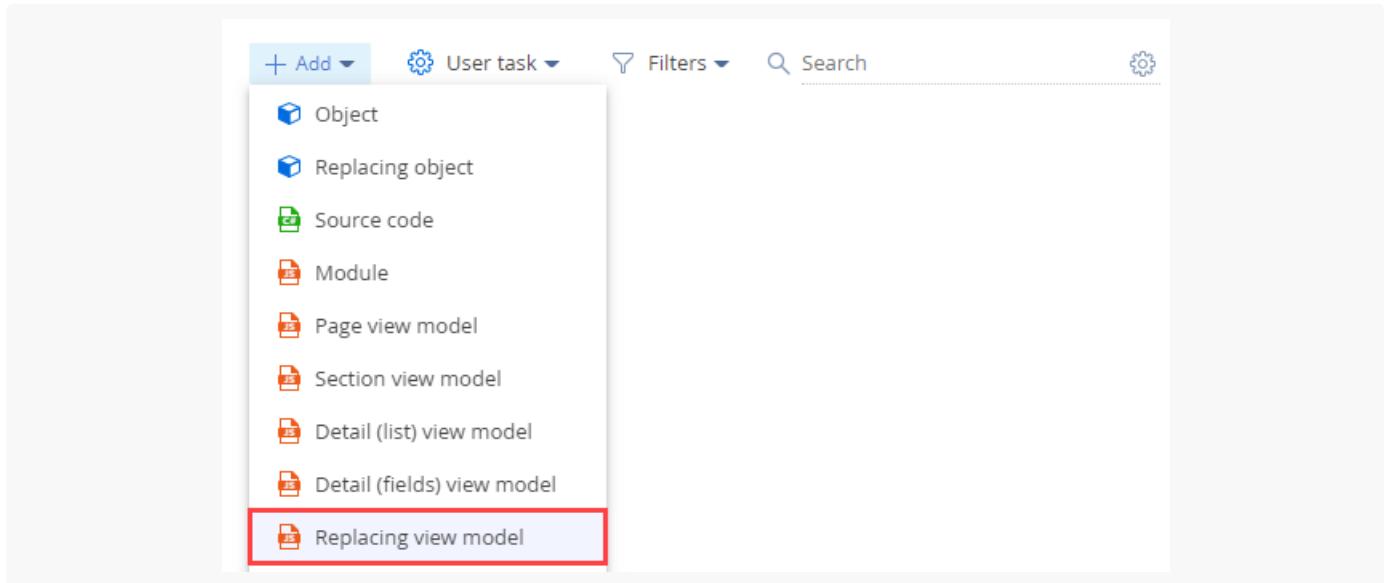
## Настроить обязательность для поля на странице записи

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 dialog has the following fields:

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

At the bottom are **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

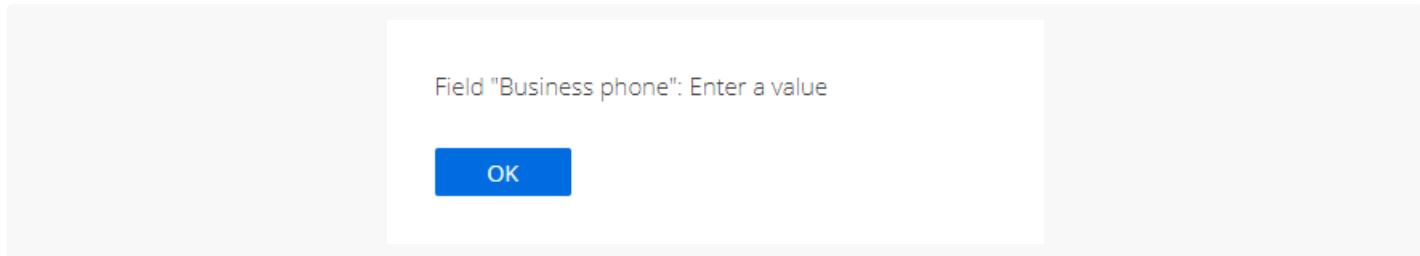
Чтобы **посмотреть результат выполнения примера**:

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").

CONTACT INFO		CONNECTED TO		Maintenance	Timeline	Engagement	Website Events
Type	Customer						
Title	Mr.						
Recipient's name	Dr. Wayne						
Age	49						
Email	a.wayne@apex.co.uk						
Owner	Marina Kysla						
Gender	Male						
Preferred language	English (United States)						

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").

## Настроить фильтрацию значений справочного поля на странице записи

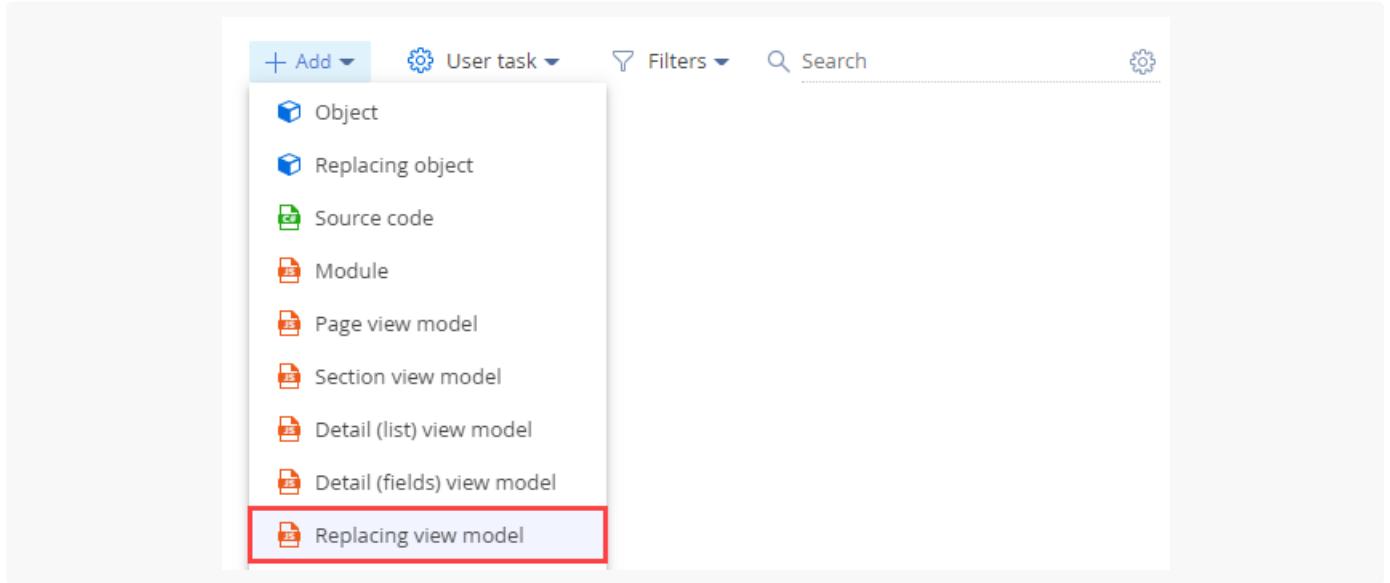


**Пример.** Настроить фильтрацию контактов, которые доступны для выбора при заполнении справочного поля [ Ответственный ] ([ Owner ]) страницы контрагента. В окне выбора контактов отображать:

- 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 is titled 'Module'. It contains 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;
                        }
                    ]
                }
            }
        };
    });
});
```

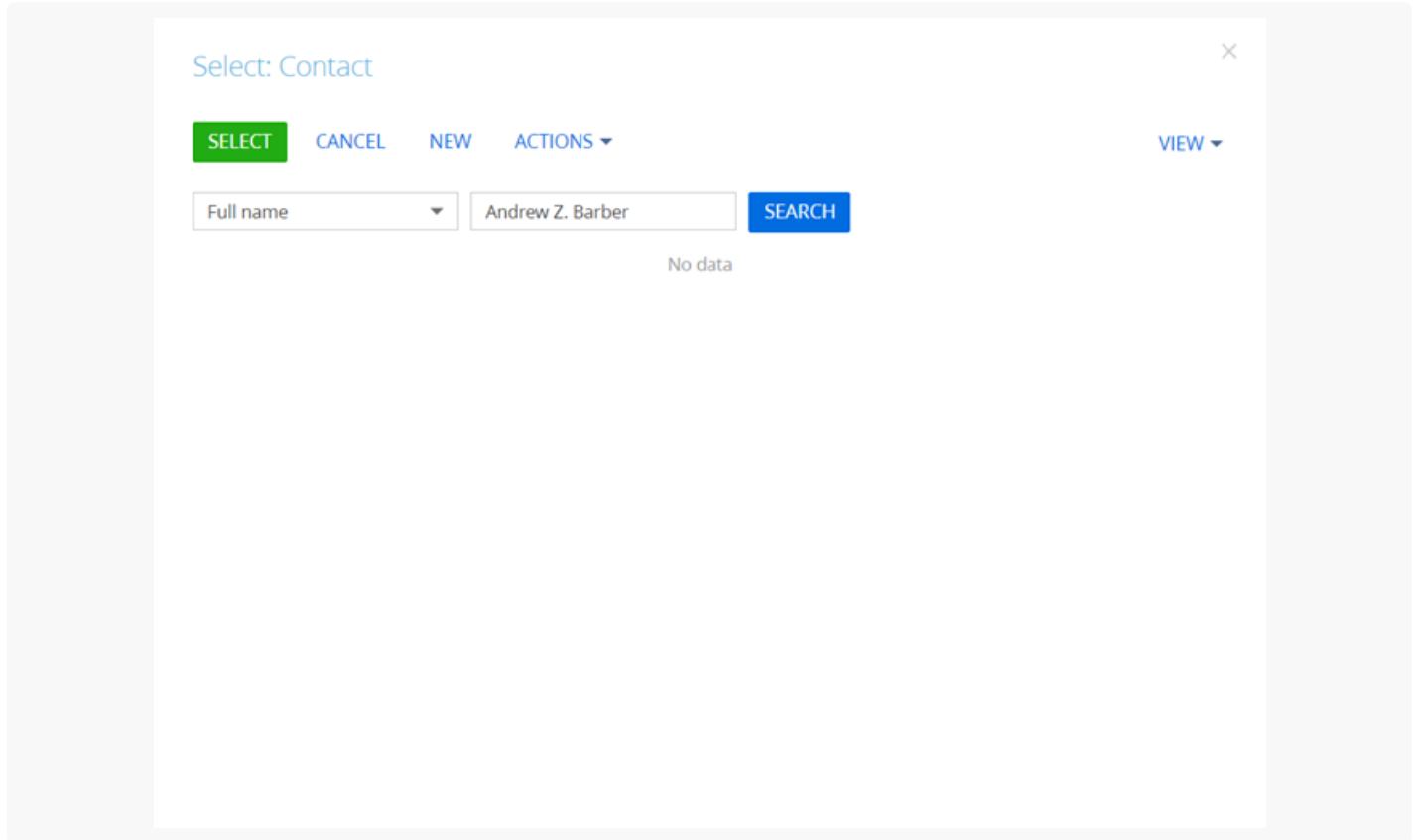
5. На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

## Результат выполнения примера

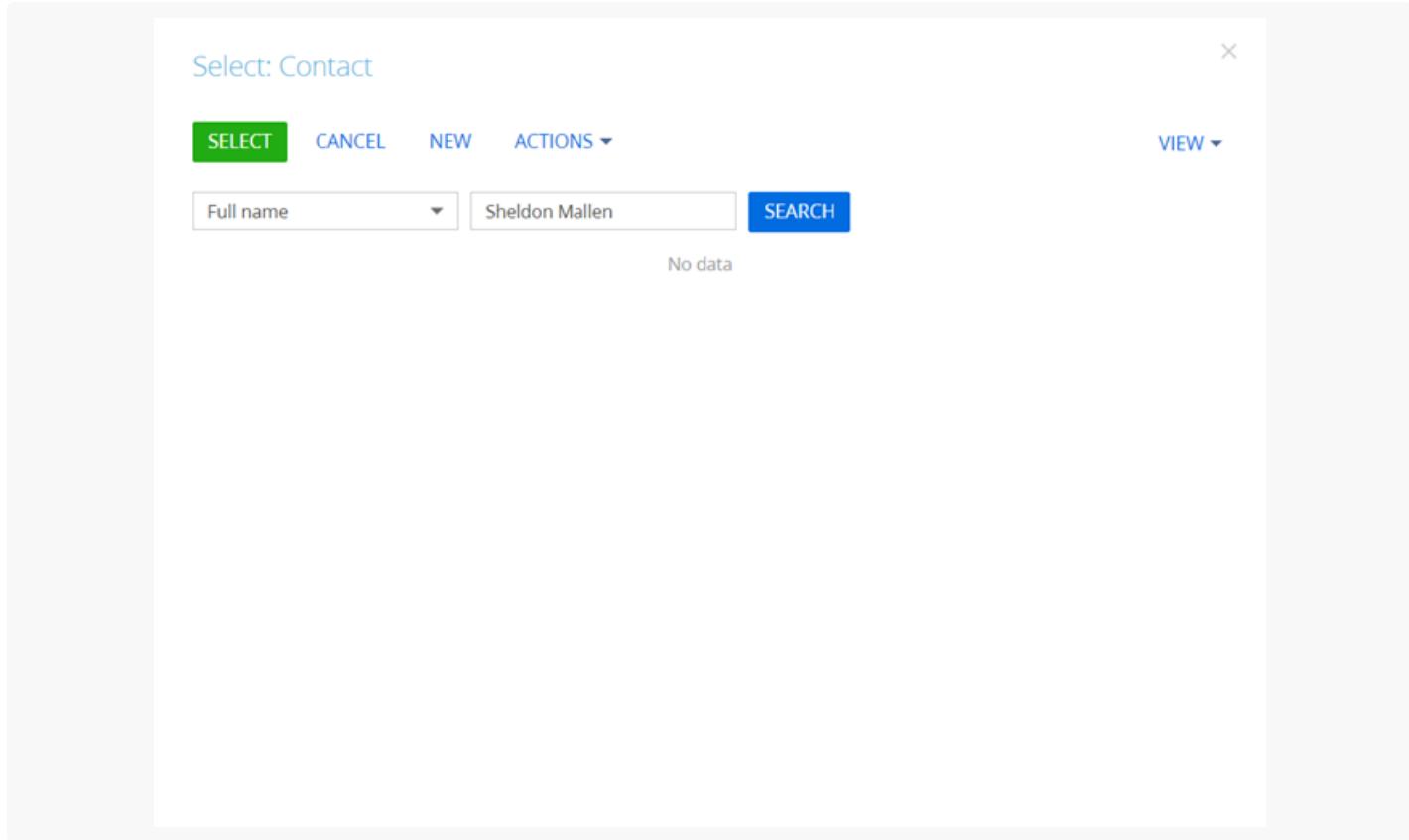
Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ Контрагенты ] ([ Accounts ]).

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.



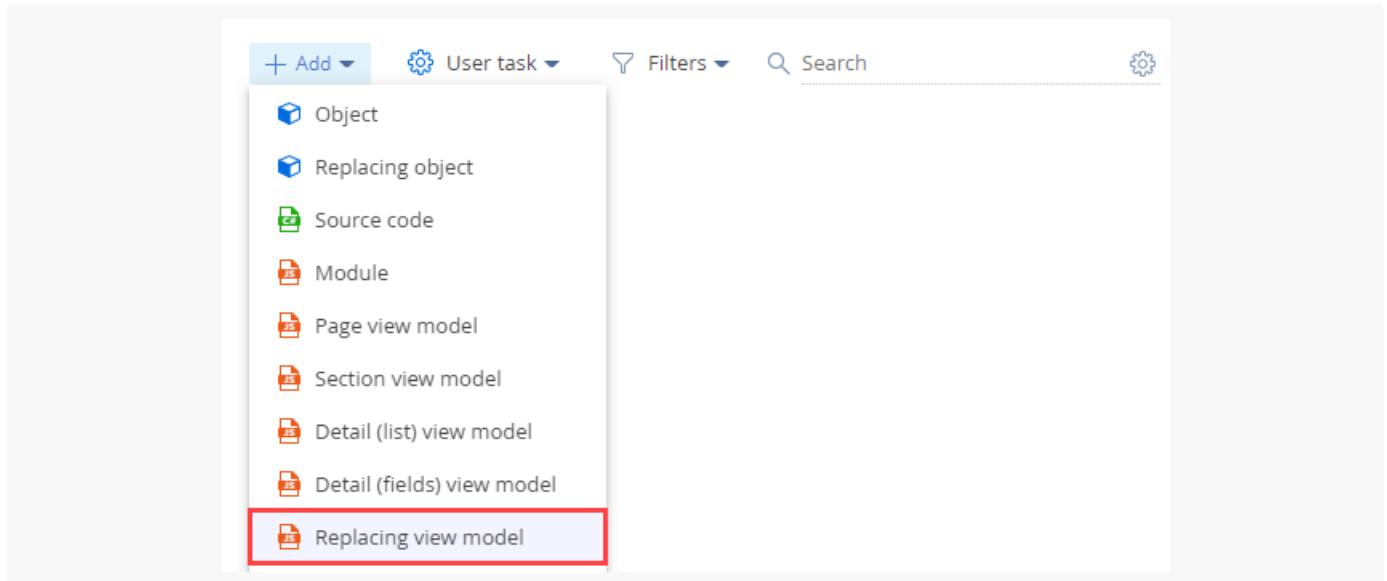
## Настроить фильтрацию значений связанных справочных полей на странице записи

 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* ]).

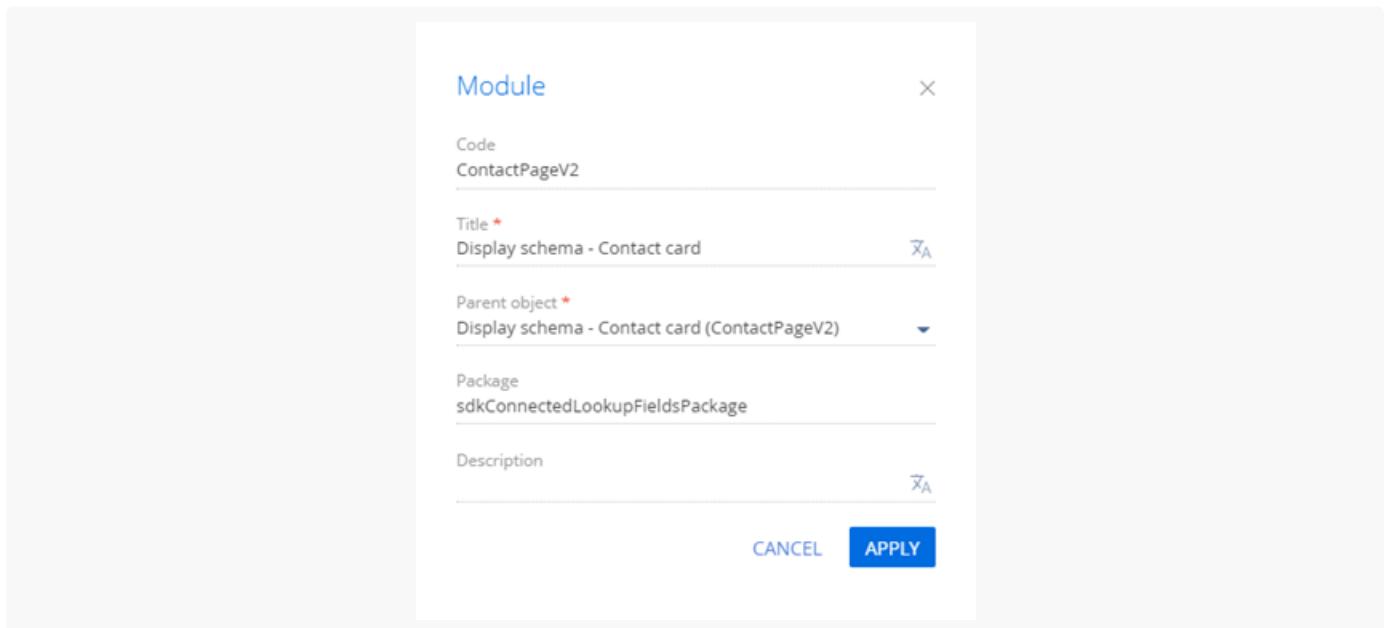
## Создать схему замещающей модели представления страницы контакта

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

Uni

United Arab Emirates

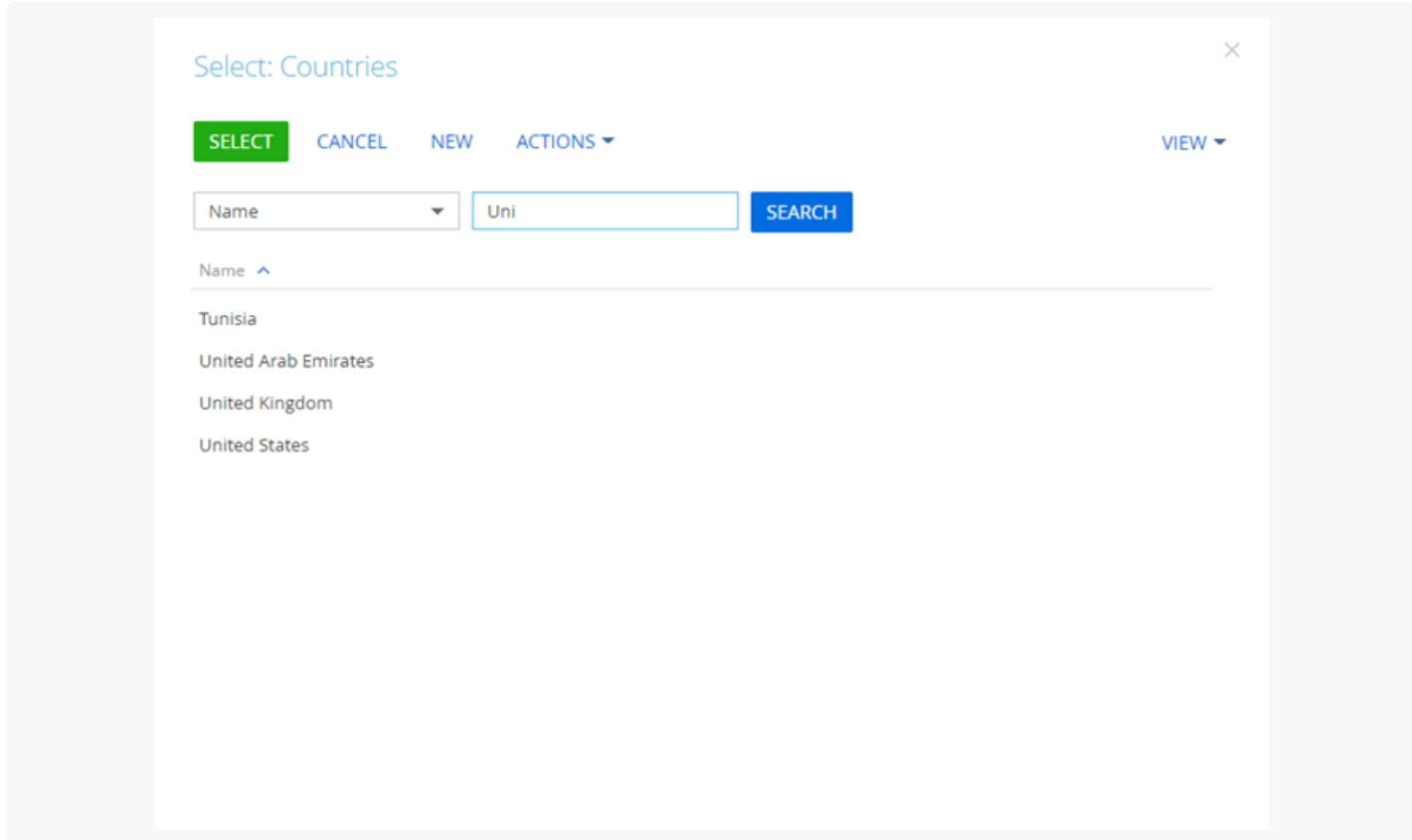
United Kingdom

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

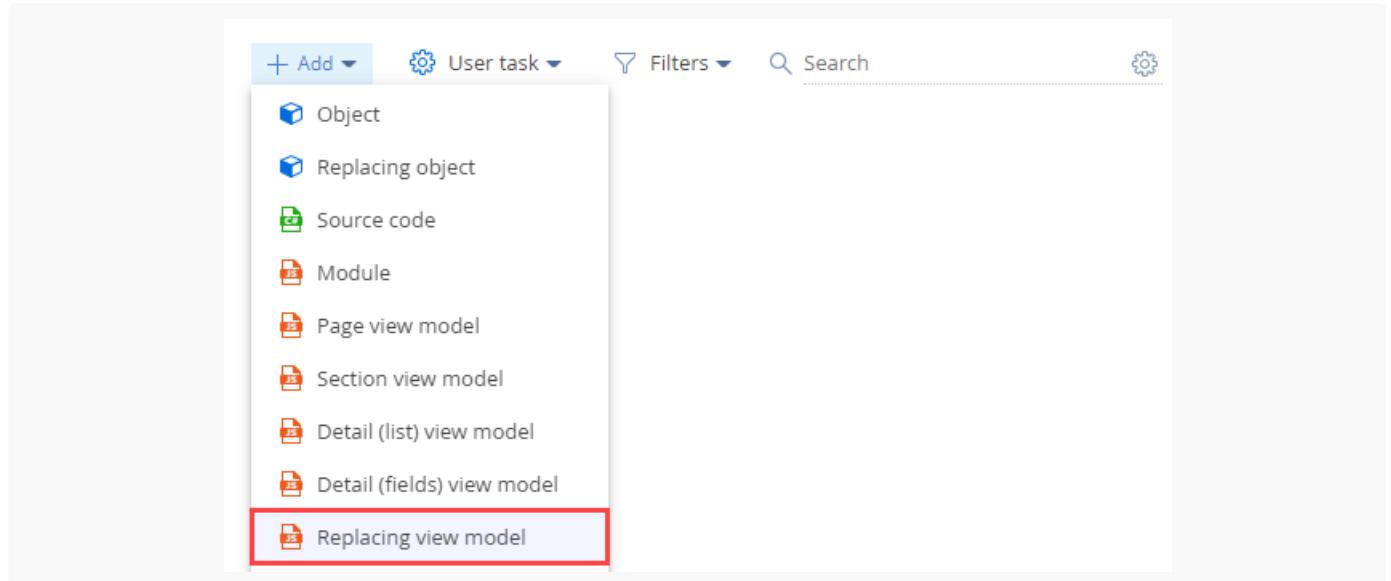
Средний

**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".

The dialog box is titled 'Module'. It contains the following fields:

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

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

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

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

- а. 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 ]).

The screenshot shows a contact record for Alexander Wilson. At the top right, there is a green progress bar labeled '100%' and a timestamp '5:00 AM, New York'. Below the photo, the contact's full name is listed as 'Alexander Wilson'. Under 'Full job title', it says 'CEO'. The 'Mobile phone' field is highlighted with a red border, while the 'Business phone' field below it is not. A small lock icon is visible next to the 'Business phone' field.

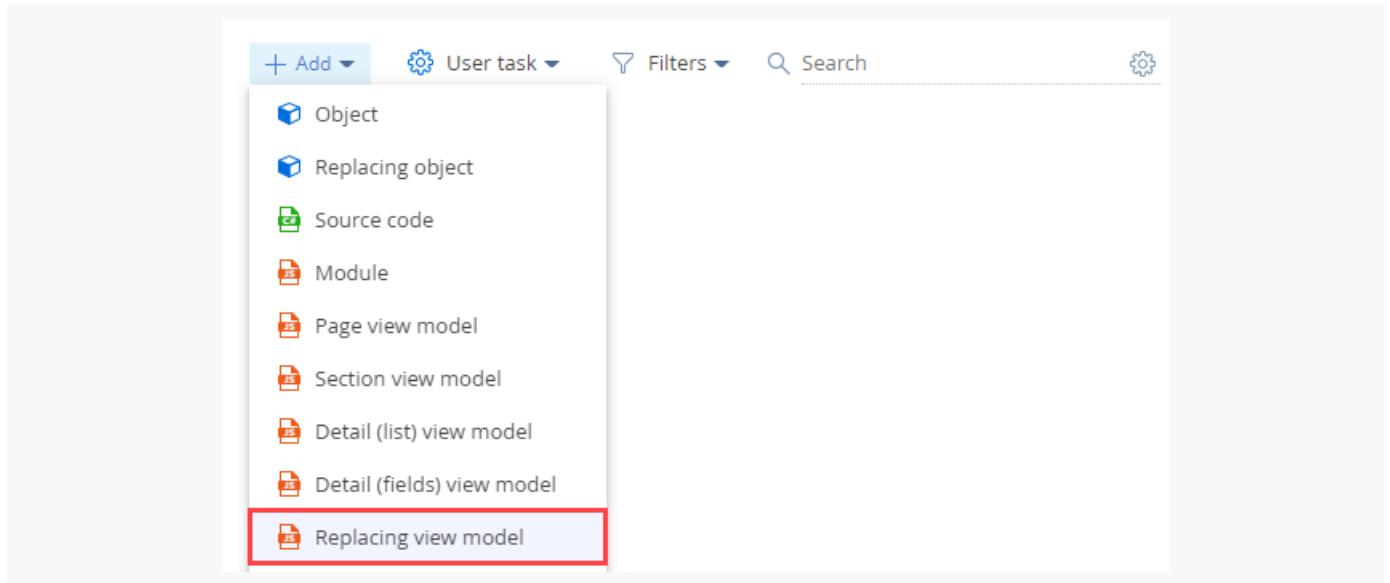
## Настроить исключения блокировки полей на странице записи

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 ]) ([ Category ]) 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:

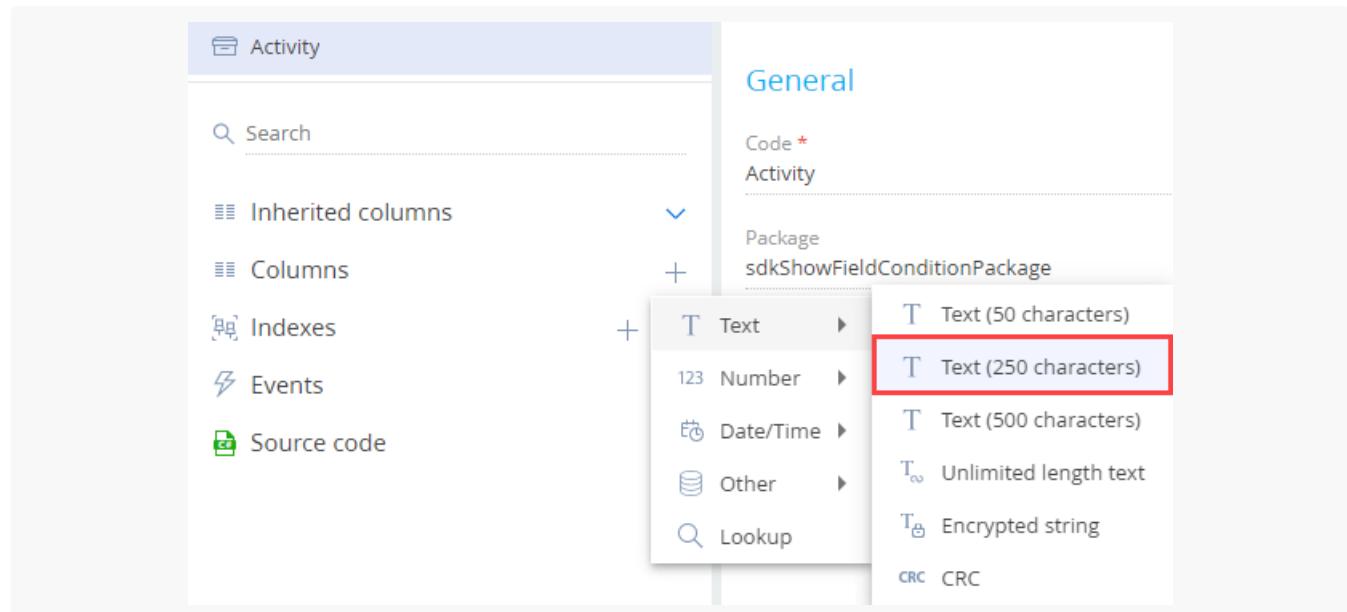
- Code \***: Activity
- Title \***: Activity
- Package**: sdkShowFieldConditionPackage
- Description**: (empty)

The 'Inheritance' tab shows:

- Parent object \***: Activity
- Replace parent

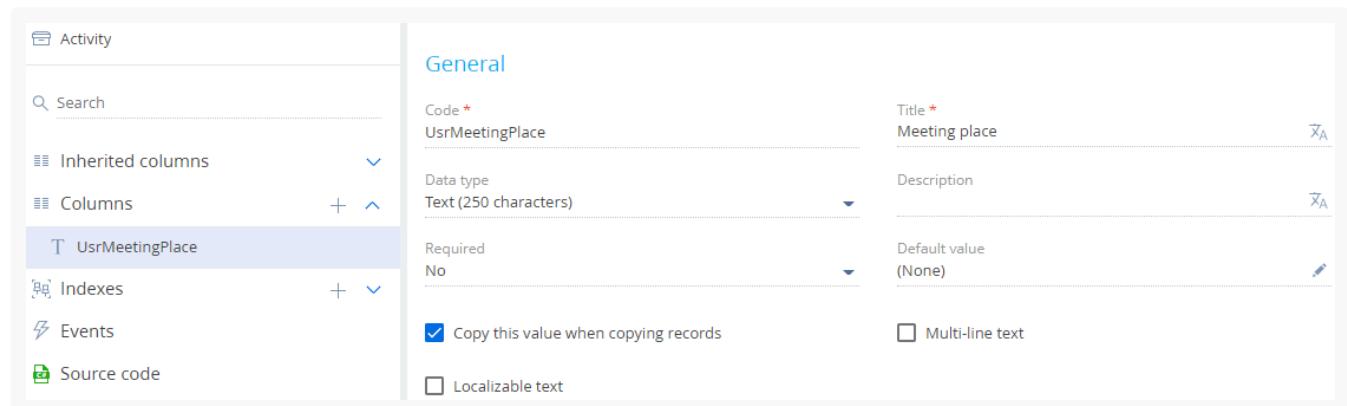
### 4. В схему добавьте **колонку**.

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



с. Заполните **свойства добавляемой колонки**.

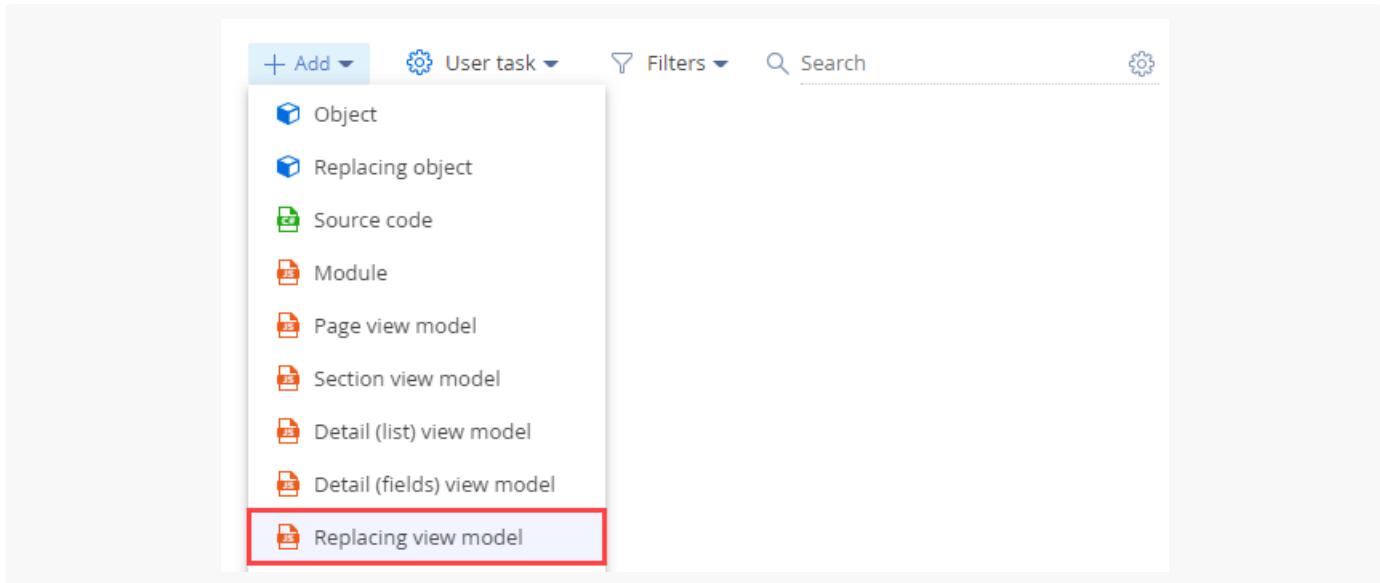
- [ Код ] ([ Code ]) — "UsrMeetingPlace".
- [ Заголовок ] ([ Title ]) — "Место встречи" ("Meeting place").



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

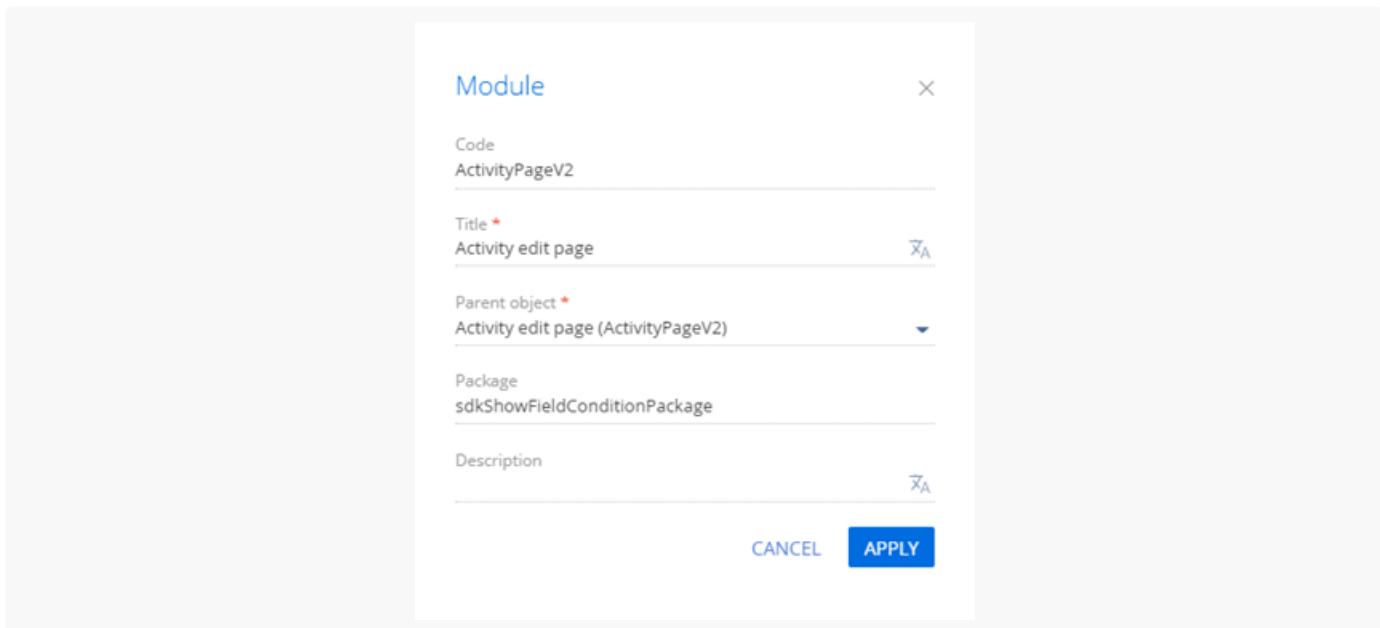
## 2. Создать схему замещающей модели представления страницы активности

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 ]).



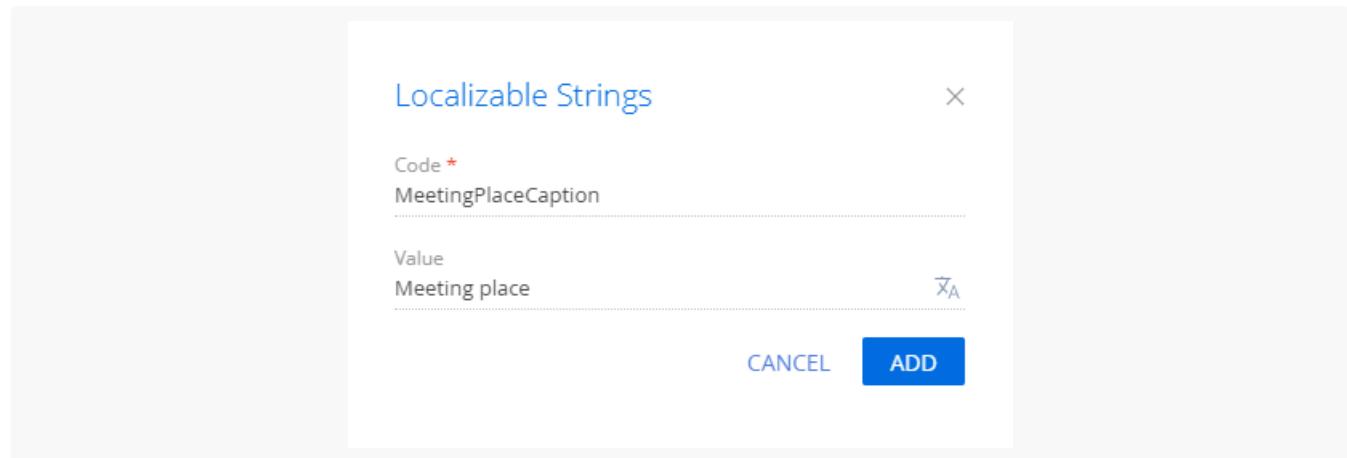
### 3. Заполните **свойства схемы**.

- [ 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? >

**CREATIO**  
7.18.4.1532

**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.

- 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}".

The screenshot shows the 'Product code mask' configuration screen in the Creatio application. At the top left is the title 'Product code mask'. Below it are two buttons: 'SAVE' (blue) and 'CANCEL' (light blue). On the right side of the header is the 'Creatio' logo with the version '7.18.4.1532'. The main area contains several configuration fields:

- Name\***: Product code mask
- Type\***: Unlimited length text
- Default value**: ART\_{0:00000}
- Description**: (empty)
- Code\***: ProductCodeMask
- Cached**:
- Save value for current user**:

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

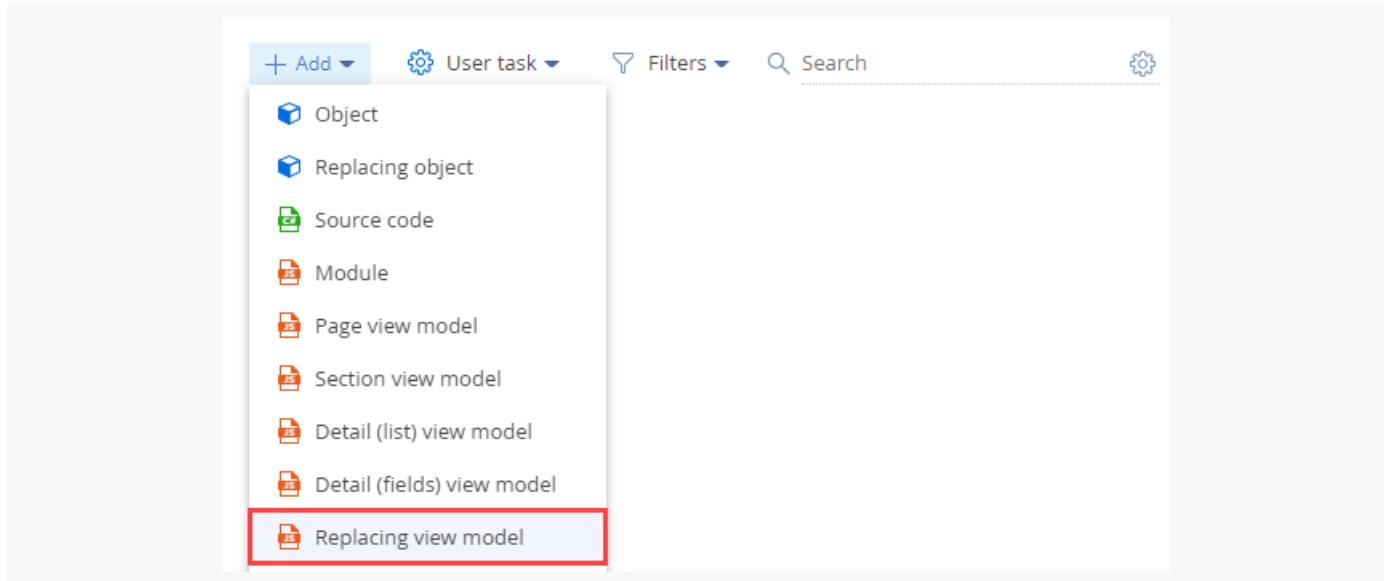
- Перейдите в дизайнер системы по кнопке .
- В блоке [ Настройка системы ] ([ System setup ]) перейдите по ссылке [ Системные настройки ] ([ System settings ]).
- На панели инструментов раздела нажмите на кнопку [ Добавить настройку ] ([ Add setting ]).
- Заполните **свойства системной настройки**.
  - [ 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 in the Creatio application. At the top left is the title 'Product last number'. Below it are two buttons: 'SAVE' (blue) and 'CANCEL' (light blue). On the right side of the header is the 'Creatio' logo with the version '7.18.4.1532'. The main area contains several configuration fields:

- Name\***: Product last number
- Type\***: Integer
- Default value**: 0
- Description**: (empty)
- Code\***: ProductLastNumber
- Cached**:
- Save value for current user**:

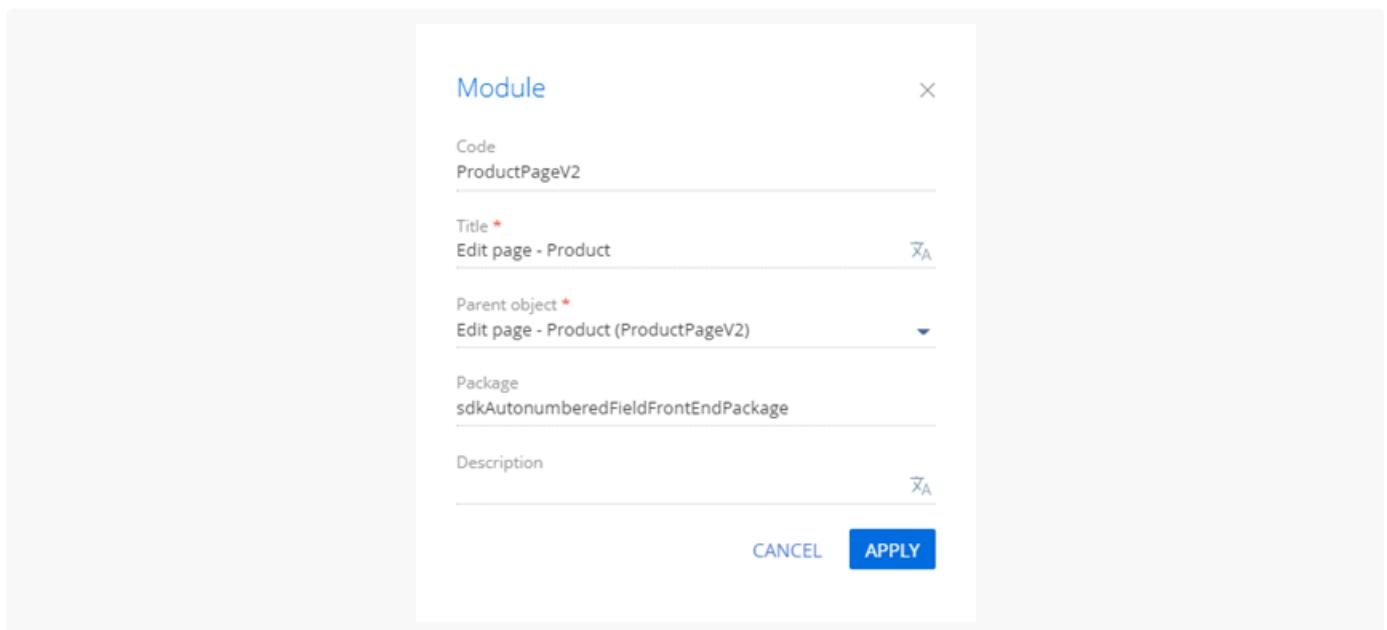
## 2. Создать схему замещающей модели представления страницы продукта

- 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);
                    });
                }
            }
        }
    };
});

```

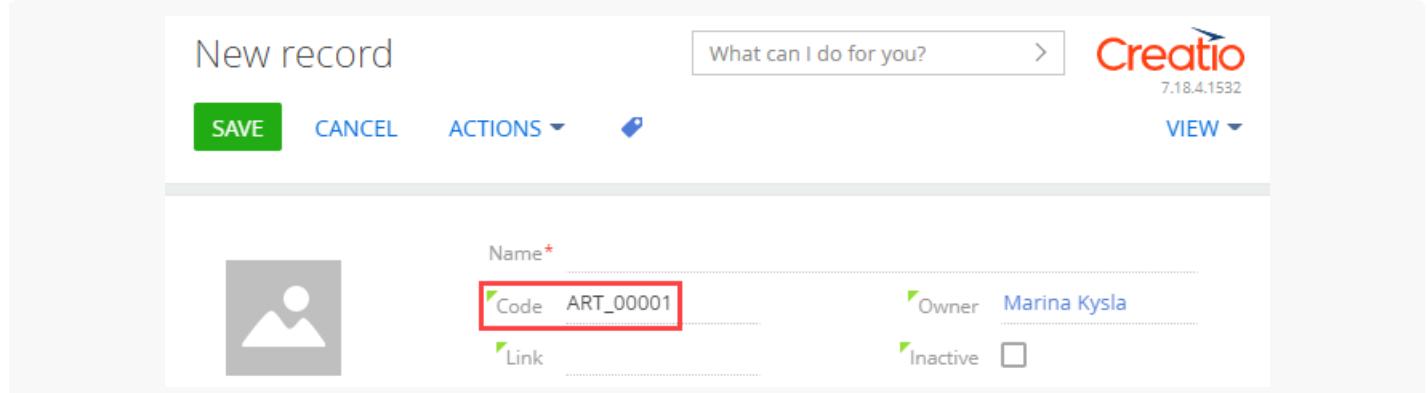
- На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

## 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 box. The 'Code' field has a green checkmark icon and a small 'Link' button below it.

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

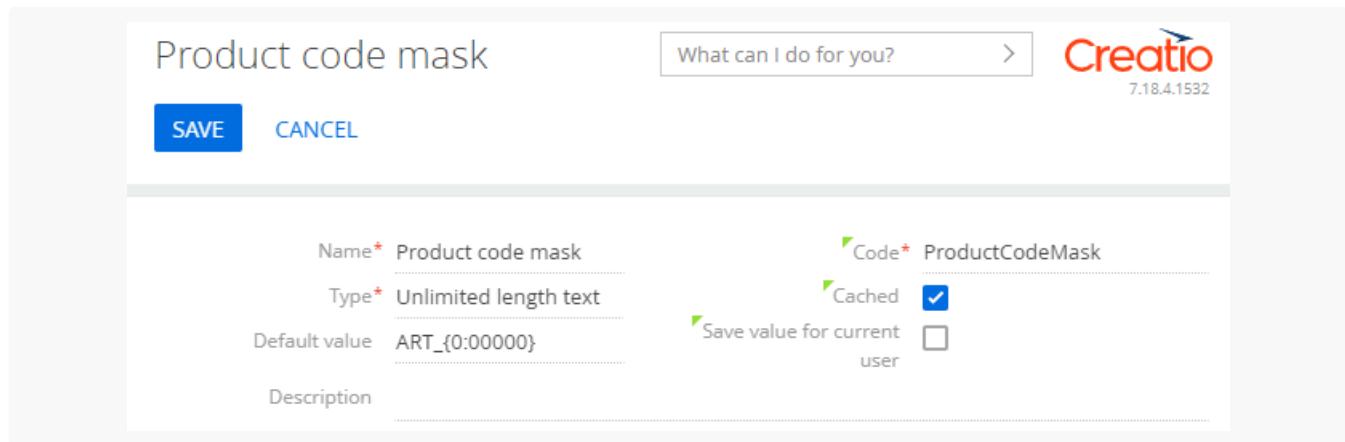
 Сложный

**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 a configuration interface for a field named "Product last number". The field is defined as follows:

- Name:** Product last number
- Type:** Integer
- Default value:** 0
- Code:** ProductLastNumber
- Cached:**
- Save value for current user:**

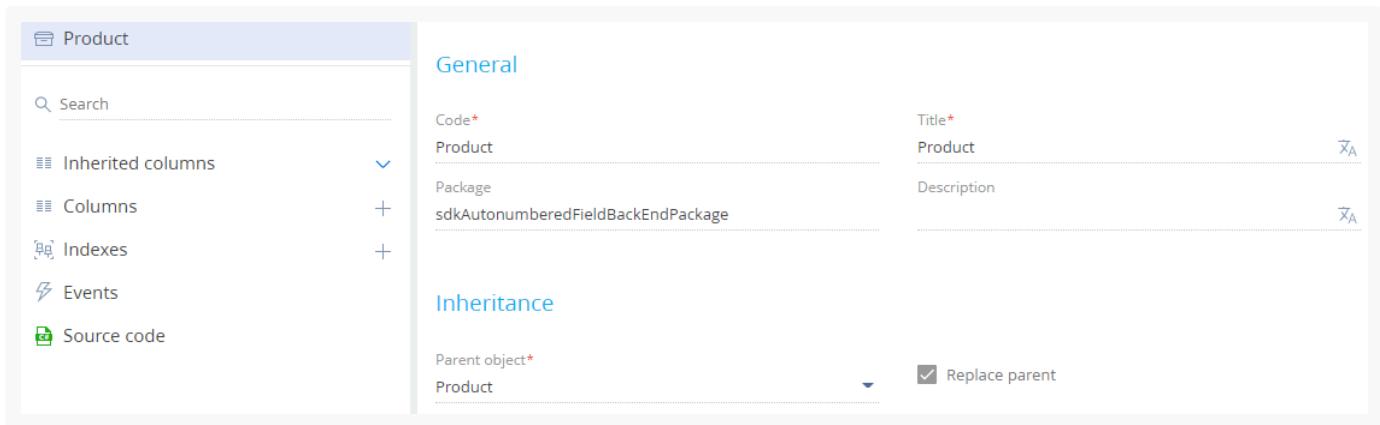
## 2. Создать схему замещающего объекта

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 the 'Add' dropdown menu with the 'Replacing object' option highlighted by a red box. Other options visible include 'Object', 'Source code', and 'Module'.

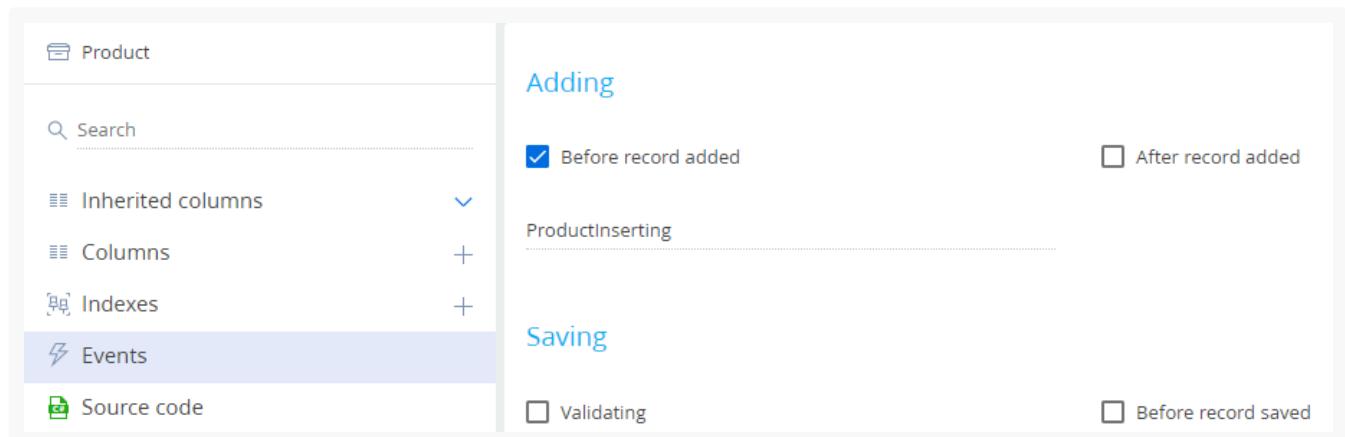
### 3. Заполните **свойства схемы**.

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



**4. В схему добавьте **событие**.**

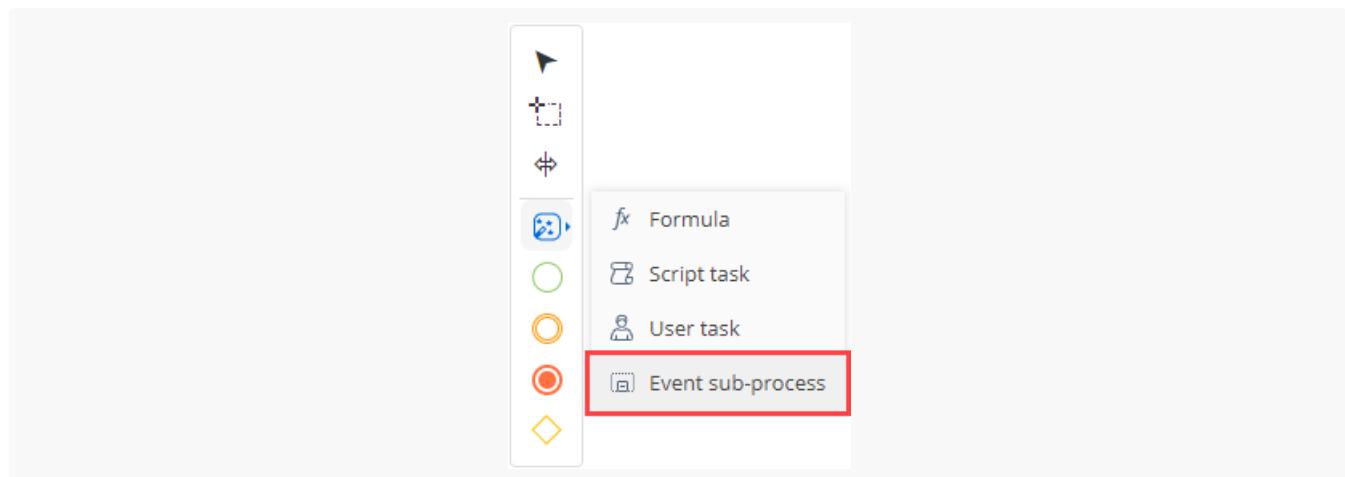
- Перейдите в узел [ События ] ([ Events ]) структуры объекта.
- В блоке [ Добавление ] ([ Adding ]) установите признак [ Перед добавлением записи ] ([ Before record added ]). Событию присвоено имя `ProductInserting`.



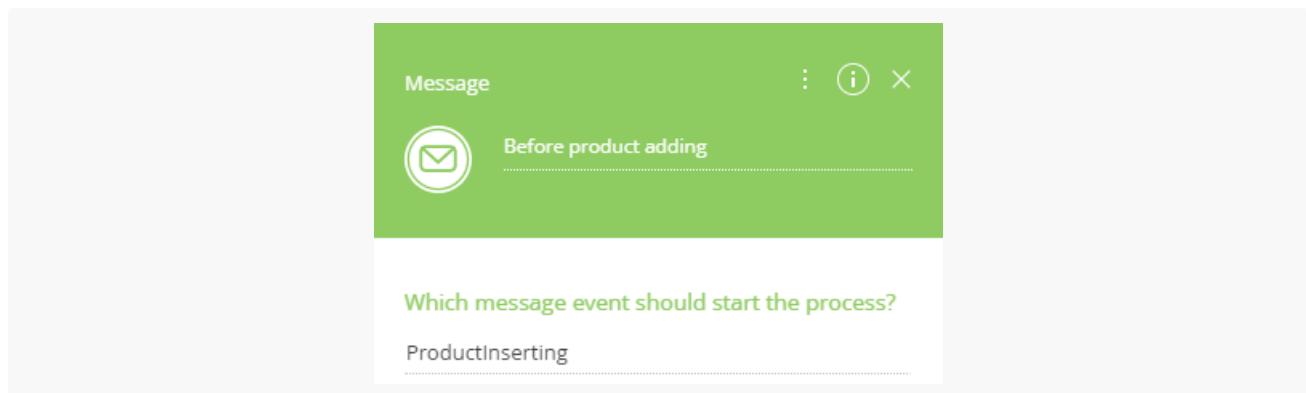
- На панели инструментов дизайнера объектов нажмите [ Сохранить ] ([ Save ]).

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

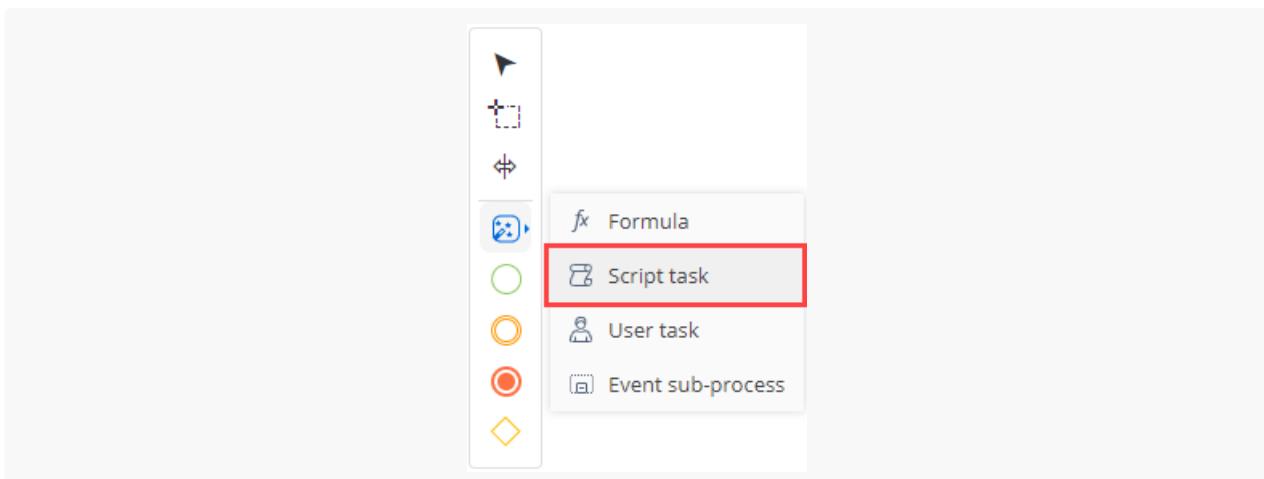
- На панели инструментов дизайнера объектов нажмите [ Открыть процесс ] ([ Open process ]).
- В области элементов дизайнера нажмите [ Действия системы ] ([ System actions ]) и разместите элемент [ Событийный подпроцесс ] ([ Event sub-process ]) в рабочей области дизайнера процессов.



- 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.



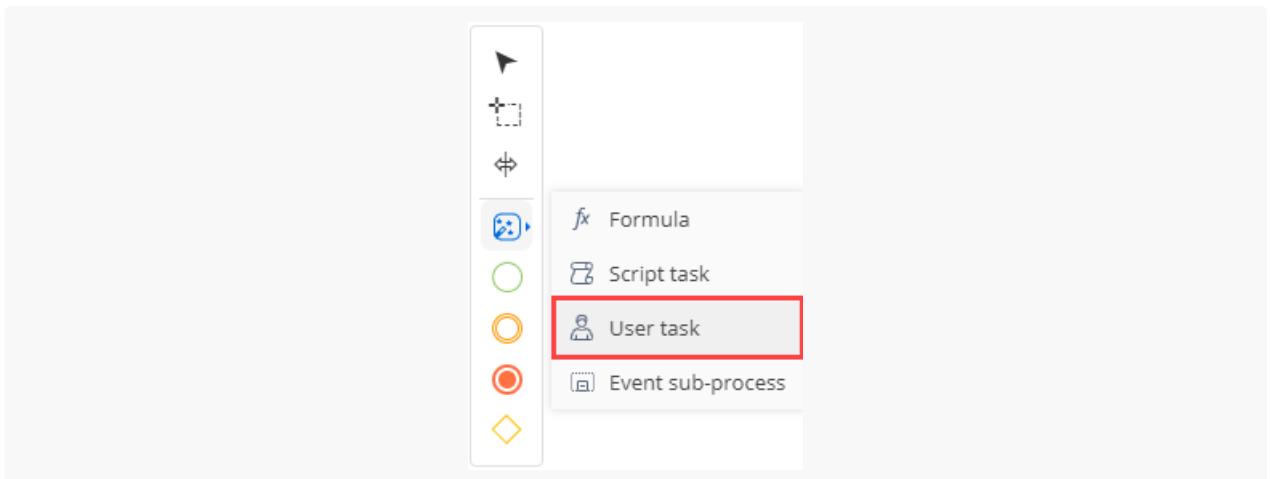
- b. Действию системы [ Задание-сценарий ] ([ Script task ]) добавьте имя "Определить схему объекта для генерации номера" ("Get entity schema to generate number").
- c. Добавьте код действия системы [ Задание-сценарий ] ([ Script task ]).

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

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

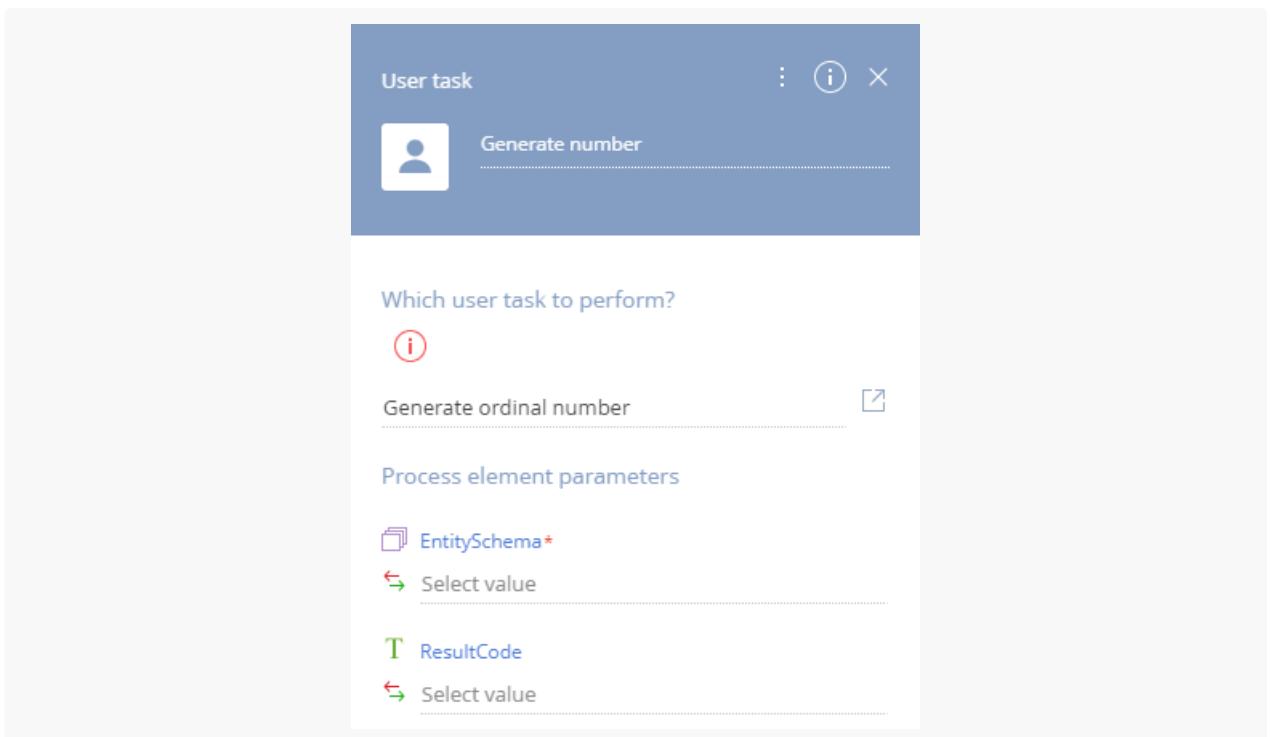
UserTask1 — код действия системы [ Выполнить действие процесса ] ([ User task ]). Выполнить генерацию номера ( Generate number ), настройка которого описана на [следующем шаге](#). Изменить код можно в расширенном режиме настройки действия системы [ Выполнить действие процесса ] ([ User task ]).

- 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.



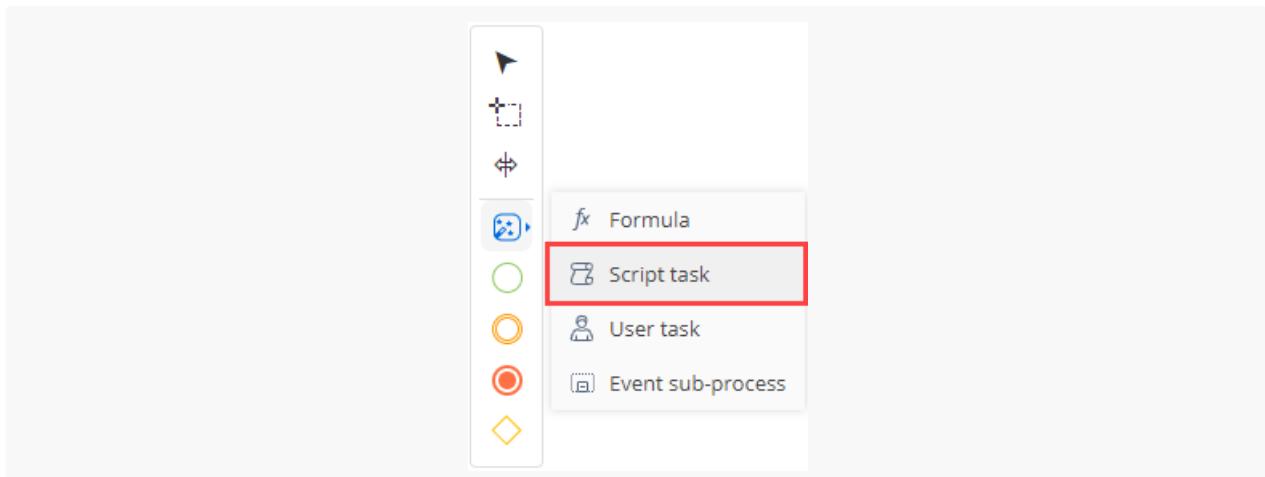
b. Заполните **свойства действия системы**.

- [ 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.

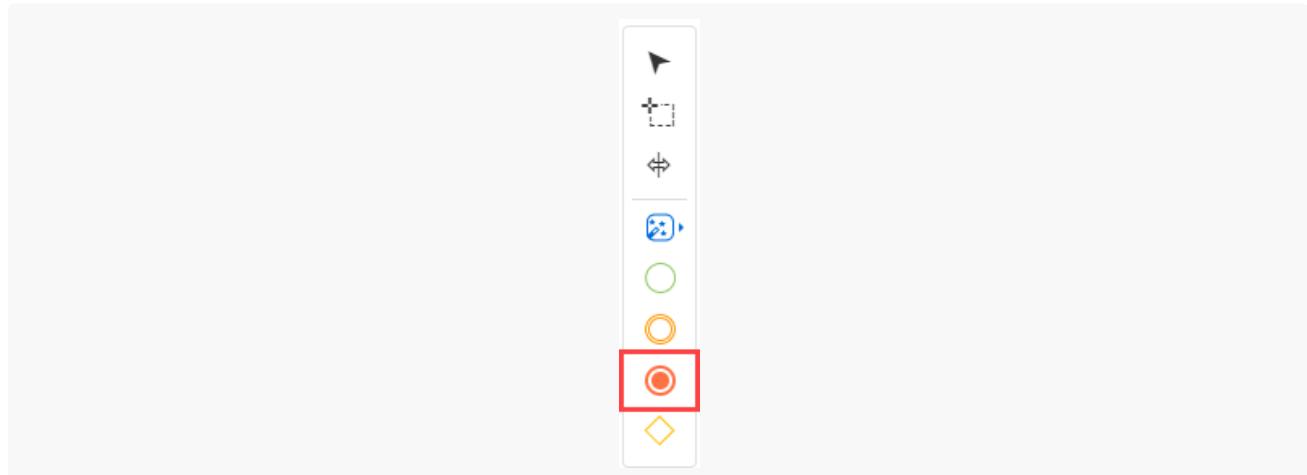


- b. Действию системы [ Задание-сценарий ] ([ Script task ]) добавьте имя "Записать полученный номер в колонку объекта" ("Save number to entity column").
- c. Добавьте код действия системы [ Задание-сценарий ] ([ Script task ]).

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

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

b. Заполните **свойства условного потока**.

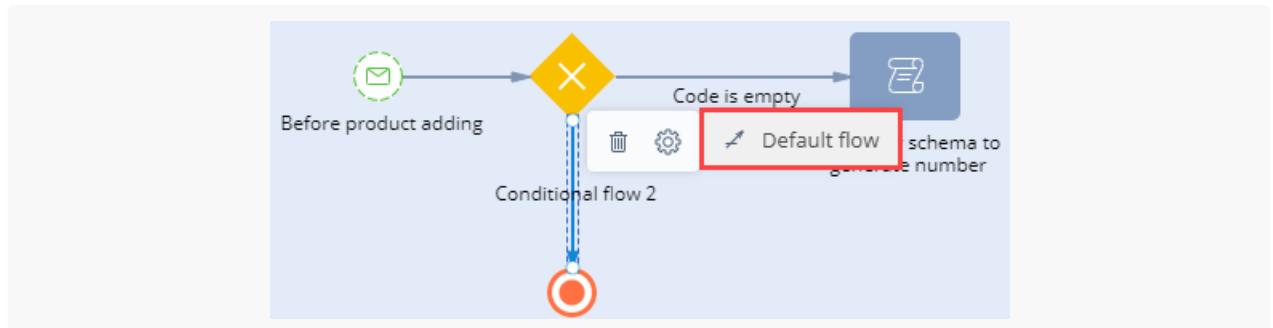
- [ Заголовок ] ([ Title ]) — "Код не заполнен" ("Code is empty").
- [ Условие перехода ] ([ Condition to move down the flow ]).
  - На панели настройки элементов в свойстве [ Условие перехода ] ([ Condition to move down the flow ]) нажмите кнопку .
  - Задайте формулу.

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

- Сохраните изменения.

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

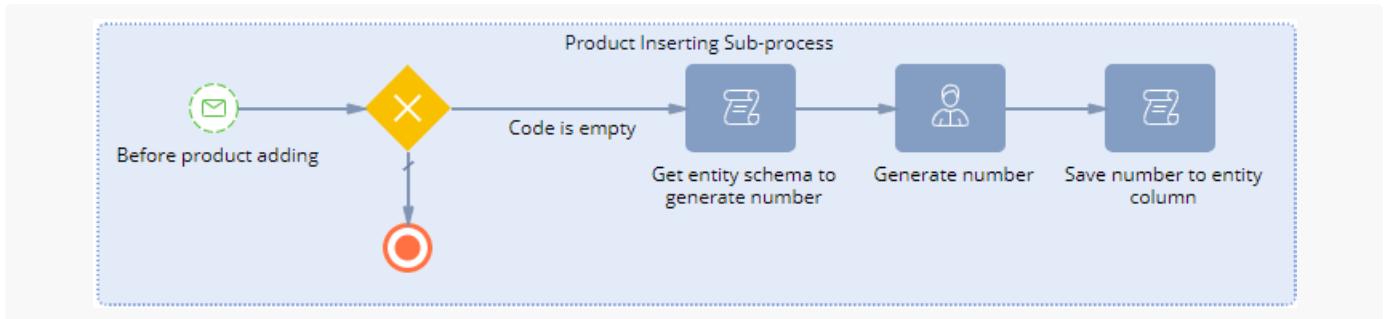
- В меню логического оператора [ Исключающее "ИЛИ" ] ([ Exclusive gateway (OR) ]) нажмите на кнопку и соедините логический оператор [ Исключающее "ИЛИ" ] ([ Exclusive gateway (OR) ]) с событием [ Останов ] ([ Terminate ]).
- Трансформируйте поток управления в поток по умолчанию. Для этого в меню потока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



c. Настройте **потоки управления**.

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

Событийный подпроцесс представлен на рисунке ниже.



- 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"/>	

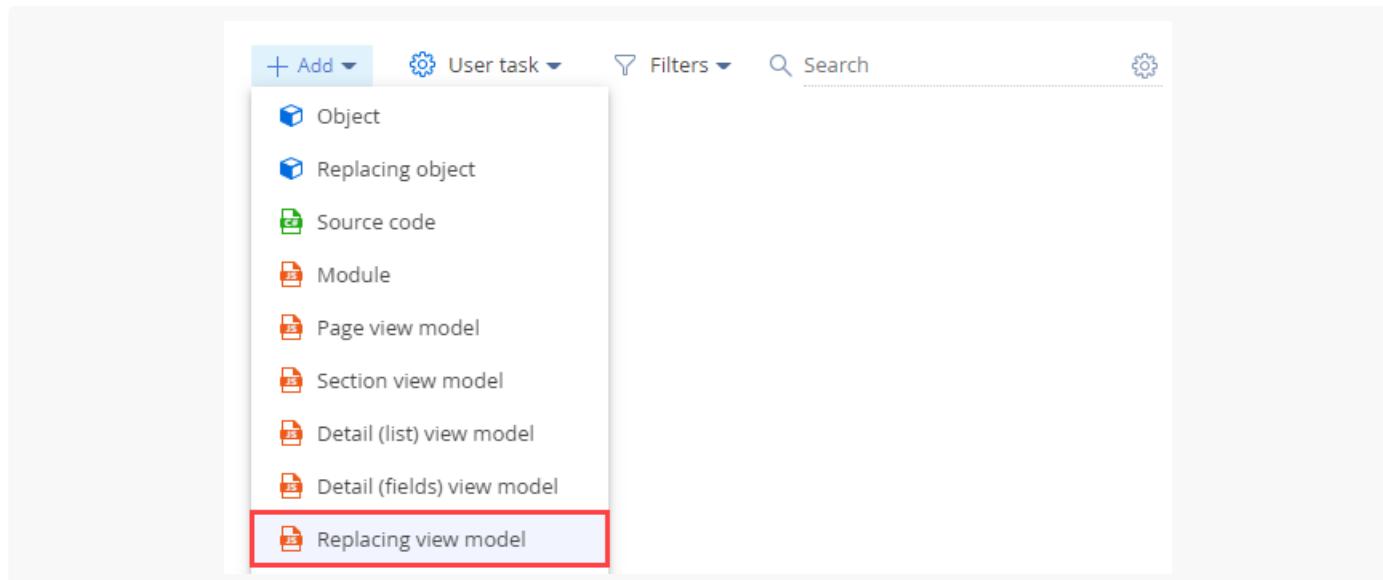
## Добавить информационную кнопку к полю на странице записи

Средний

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

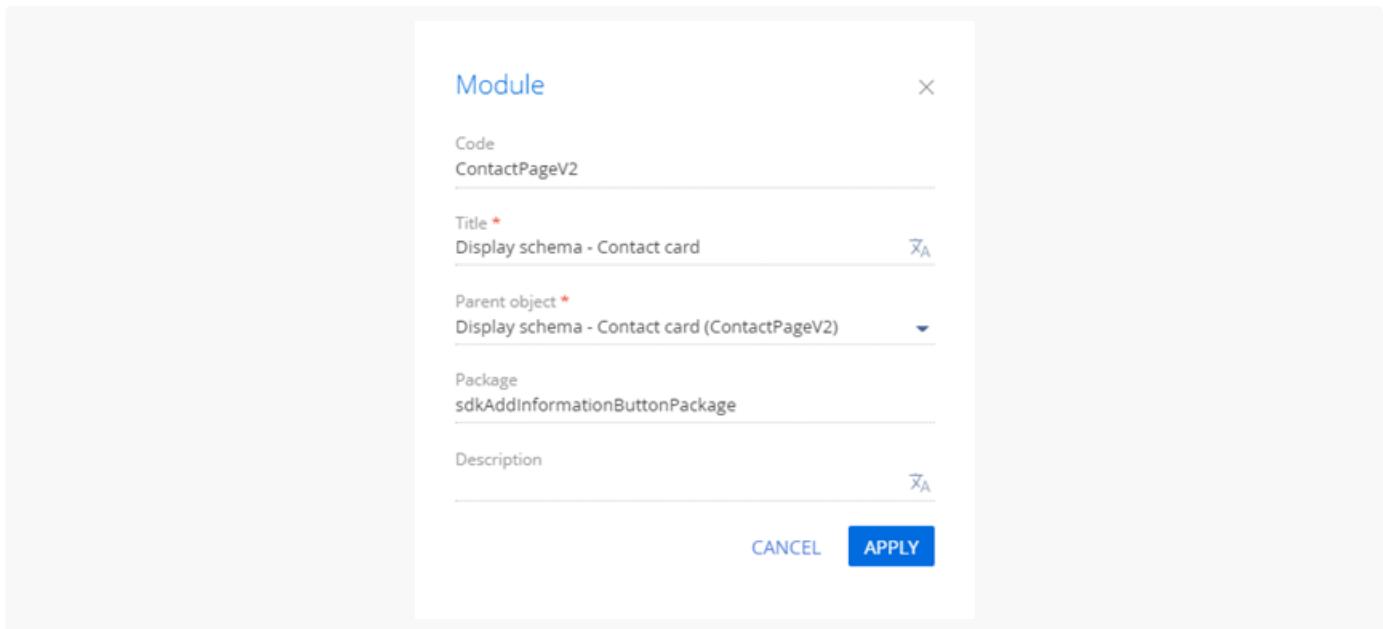
## Создать схему замещающей модели представления страницы контакта

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* ]).



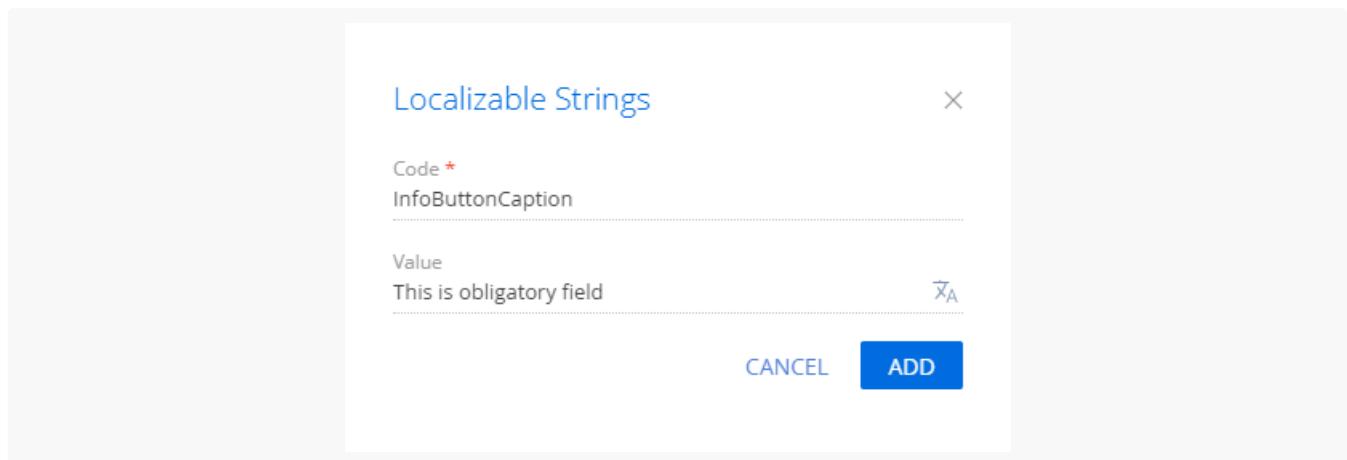
### 3. Заполните **свойства схемы**.

- [ *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

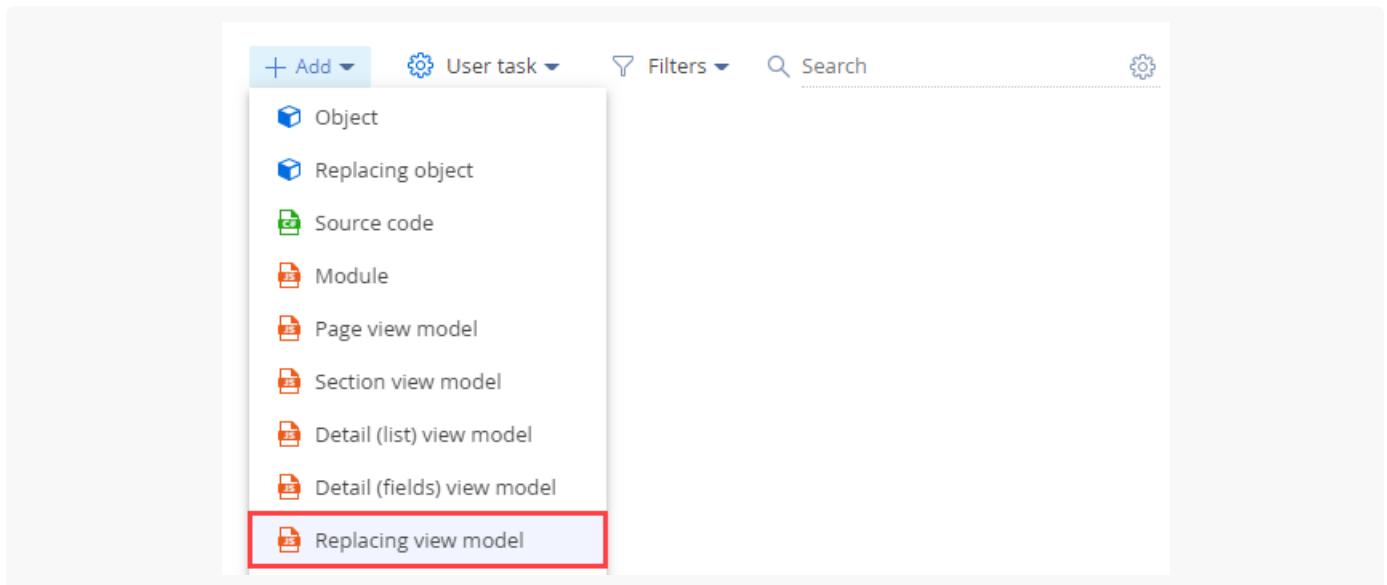
Средний

**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 dialog box is titled 'Module'. It contains the following fields:

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

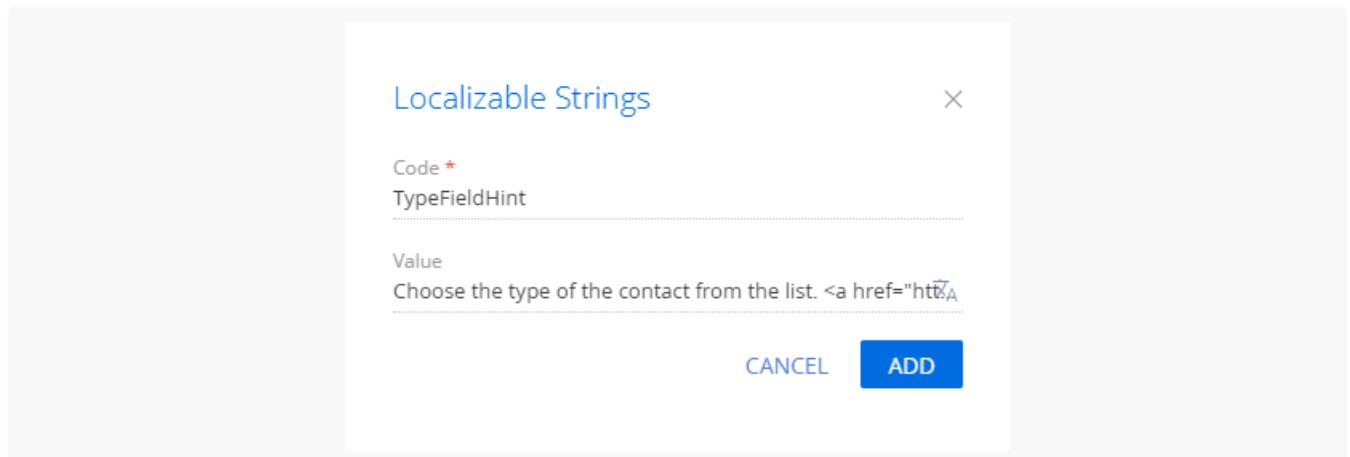
At the bottom are 'CANCEL' and 'APPLY' buttons.

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. <a href="https://academy.terrasoft.ua/docs/user/bazis\_platformy/interfejs/stranitsy\_zapisey/stranicy\_zapisej" target="\_blank">Узнать больше</a>" ("Choose the type of the contact from the list. <a href="https://academy.creatio.com/docs/user/platform\_basics/user\_interface/record\_pages\_shortcut/record\_pages" target="\_blank">Read more</a>").



- e. Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).  
 5. Настройте **всплывающую подсказку к полю** [ Тип ] ([ Type ]) страницы контакта. Для этого в массив модификаций `diff` добавьте конфигурационный объект поля на странице.

Исходный код схемы замещающей модели представления страницы контакта представлен ниже.

```
ContactPageV2

define("ContactPageV2", [], function () {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение всплывающей подсказки. */
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для добавления к полю всплывающей подсказки. */
            {
                /* Выполняется операция изменения существующего элемента. */
                "operation": "merge",
                /* Мета-имя изменяемого поля. */
                "name": "Type",
                /* Мета-имя родительского контейнера, в котором изменяется поле. */
                "parentName": "ContactGeneralInfoBlock",
                /* Поле изменяется изменяется в коллекции элементов родительского элемента. */
                "propertyName": "items",
                /* Свойства, передаваемые в конструктор элемента. */
                "values": {
                    /* Всплывающая подсказка для поля Type. */
                    "TypeFieldHint": {
                        /* Код строки. */
                        "Code": "TypeFieldHint",
                        /* Значение строки. */
                        "Value": "Choose the type of the contact from the list. <a href='https://academy.creatio.com/docs/user/platform_basics/user_interface/record_pages_shortcut/record_pages' target='_blank'>Read more</a>"
                    }
                }
            }
        ]
    }
})
```

```

/* 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. The Timeline section shows a task: 'Contact customer, specify need, budget, decision-making role.' assigned to Valerie E. Murphy on 10/3/2021. A modal window titled 'Choose the type of the contact from the list' is open, with 'Customer' selected. Other options listed are 'Type' and 'Customer'. The contact's details also show 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.

**Sales**

**Accounts**

**Contacts**

**Vertigo Systems**

Primary contact  
Peter Moore

Web  
www.vertigosys.com

Address  
83 Ashton Street

Primary phone  
+44 (20) 3427 1374

City  
London

Type  
Customer

Country  
United Kingdom

**SeparateModeActionButtons LeftContainer**

**SeparateModeActionButtons RightContainer**

**SeparateModeActionButtonsContainer**

**VIEW**

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

**Sales**

**Home**

**Dashboards**

**Feed**

**Leads**

**Accounts**

**Contacts**

**VERTIGO SYSTEMS**

95%

Name\*  
Vertigo Systems

Type  
Customer

Owner  
Mary King

Enrich data

NEXT STEPS (0)

You don't have any tasks yet

Press **F** above to add a task

**CLOSE** **ACTIONS** **VIEW**

**CombinedModeActionButtons CardLeftContainer**

**CombinedModeActionButtons CardRightContainer**

**CombinedModeActionButtonsCardContainer**

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

**Sales**

**Home**

**Dashboards**

**Feed**

**Leads**

**Accounts**

**Contacts**

**0%**

Name\*  
Marina Kysla

Type

Owner  
Marina Kysla

Enrich data

NEXT STEPS (0)

You don't have any tasks yet

Press **F** above to add a task

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

**ActionButtonsContainer**

**LeftContainer**

**RightContainer**

## 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 [ Стока (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`.

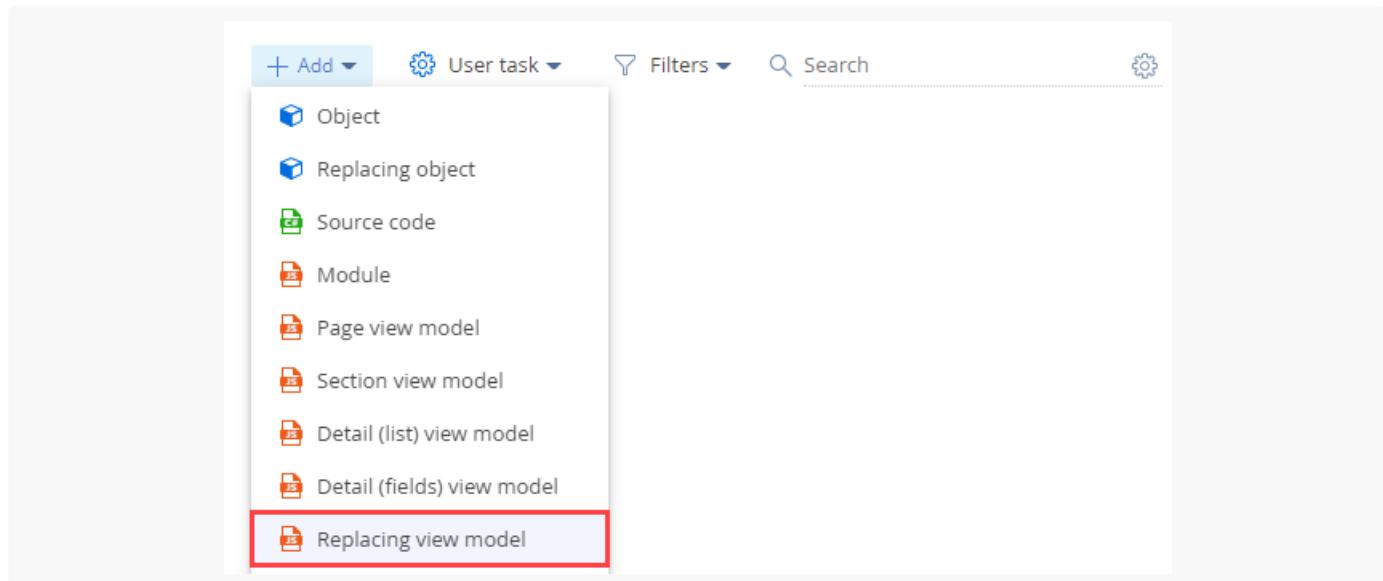
## Добавить кнопку на панель инструментов раздела



**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.

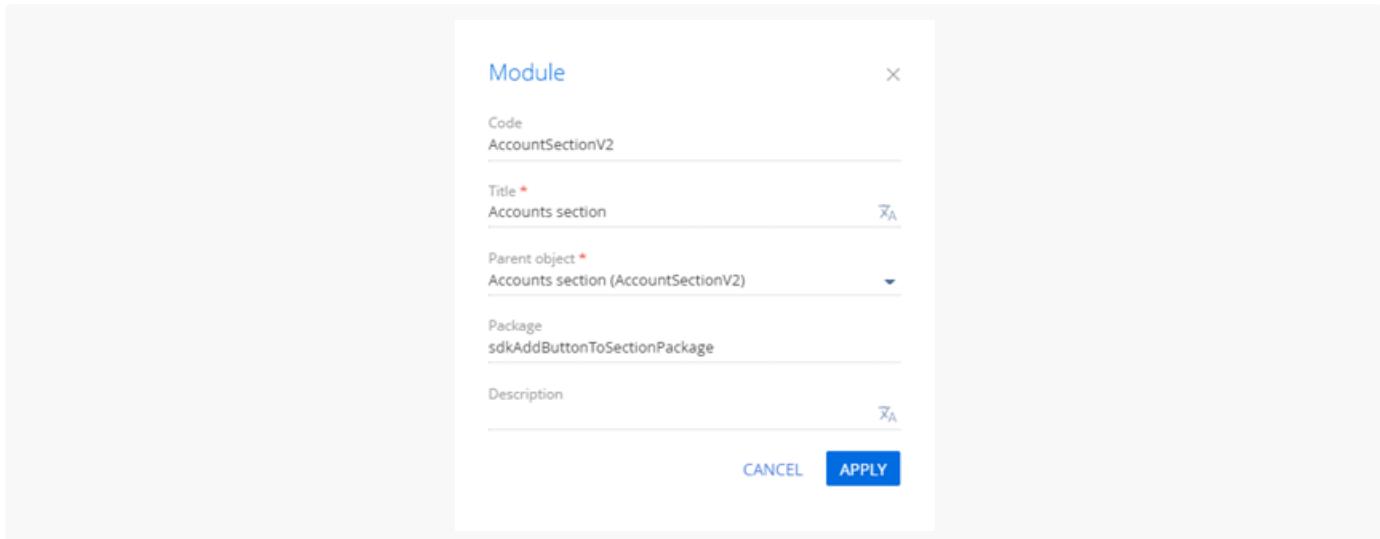
## Создать схему замещающей модели представления раздела

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 ]).



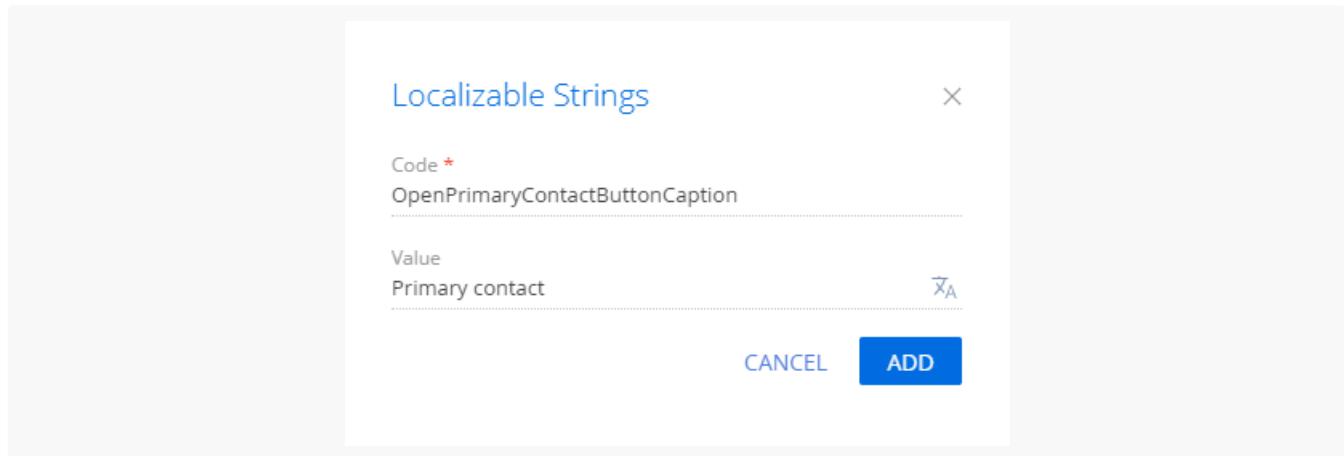
### 3. Заполните **свойства схемы**.

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



#### 4. Добавьте локализуемую строку.

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

The screenshot shows the 'Accounts' screen in the Creatio application. On the left is a dark sidebar with navigation links: Sales, Dashboards, Feed, Leads, and Accounts (which is selected and highlighted with a red border). The main area displays a list of accounts. The first account listed is 'Vertigo Systems', with its details: Web (www.vertigosys.com), Primary phone (+44 (20) 3427 1374), Type Customer, and Country United Kingdom. Below it is another account, 'Sunrise Investments', with similar details. At the top of the main area, there are buttons for 'NEW ACCOUNT', 'ACTIONS', and 'PRIMARY CONTACT'. The 'PRIMARY CONTACT' button is highlighted with a red box.

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

This screenshot shows the same 'Accounts' screen as the previous one, but with a vertical toolbar on the right side. The toolbar contains several icons: a user profile, gear, question mark, telephone, envelope, speech bubble (highlighted with a red circle), message, bell, and a square. The account list is identical to the previous screenshot, showing 'Vertigo Systems' and 'Sunrise Investments'.

## Добавить кнопку на панель инструментов страницы записи в совмещенном режиме

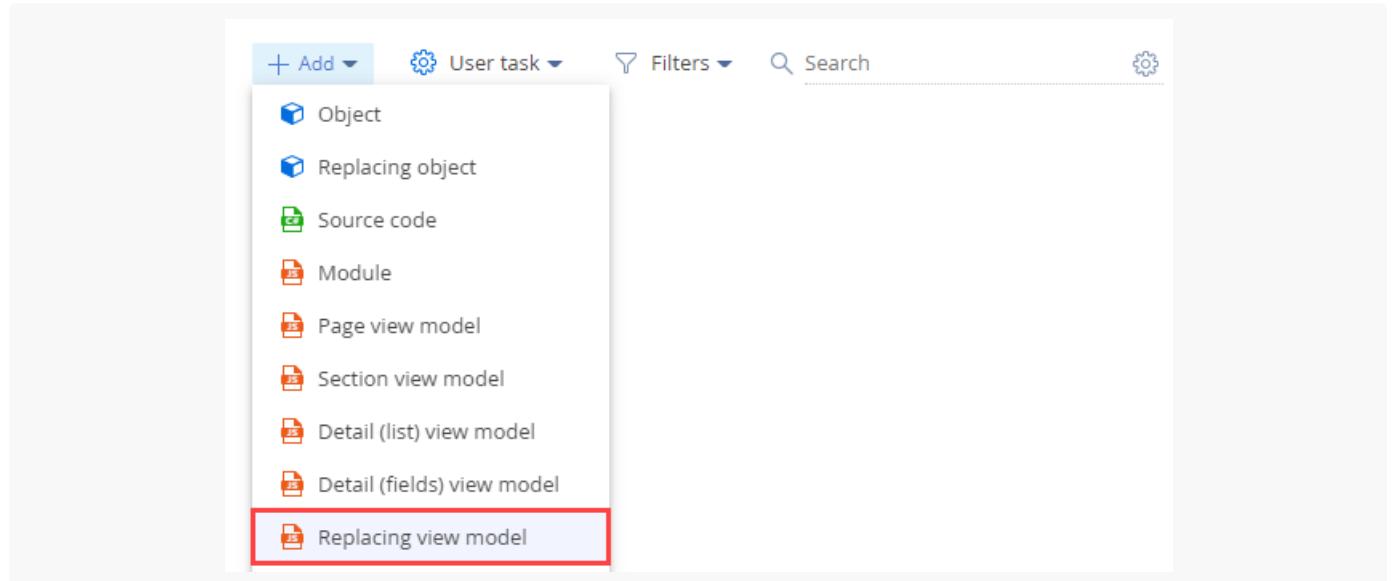
Сложный

**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.

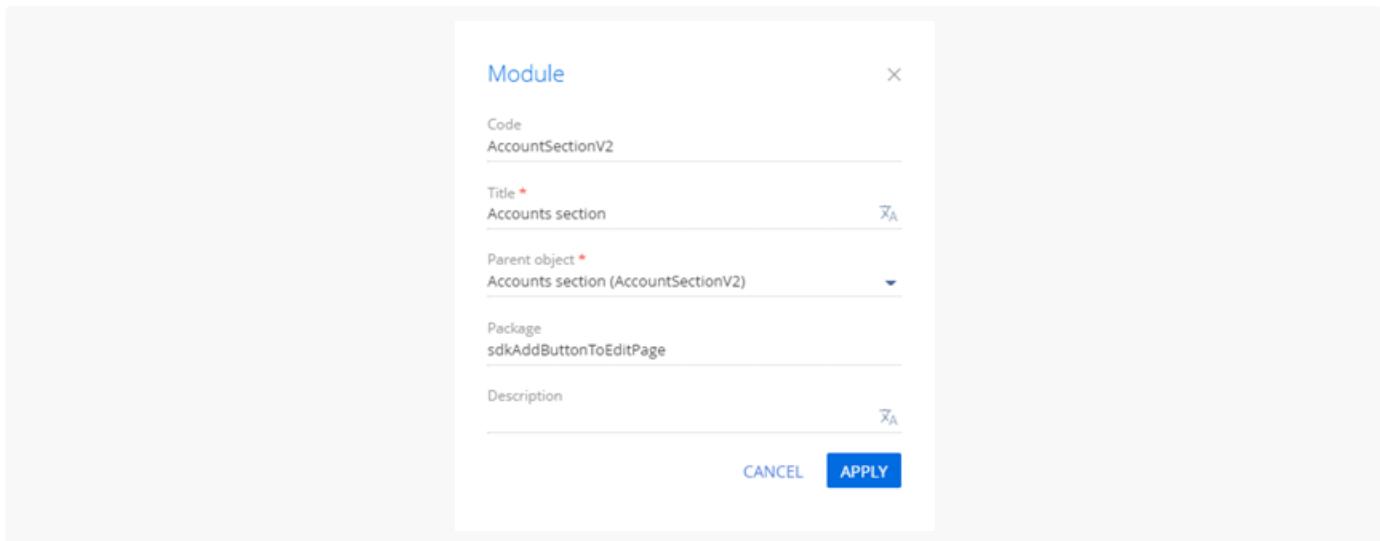
## Создать схему замещающей модели представления раздела

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 ]).



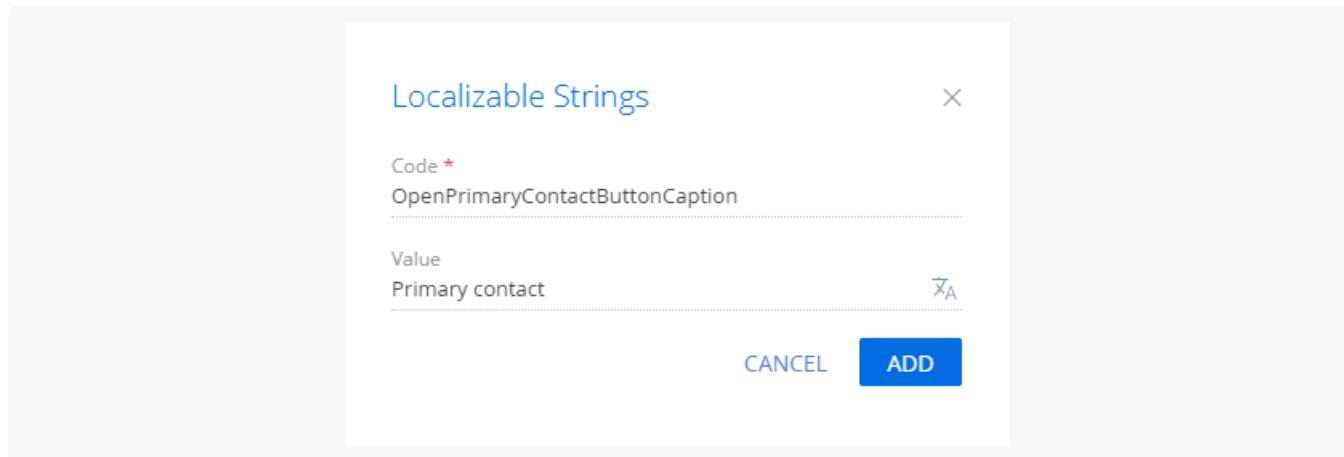
### 3. Заполните **свойства схемы**.

- [ 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.

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

## Добавить кнопку на панель

# инструментов страницы добавления записи

 Средний

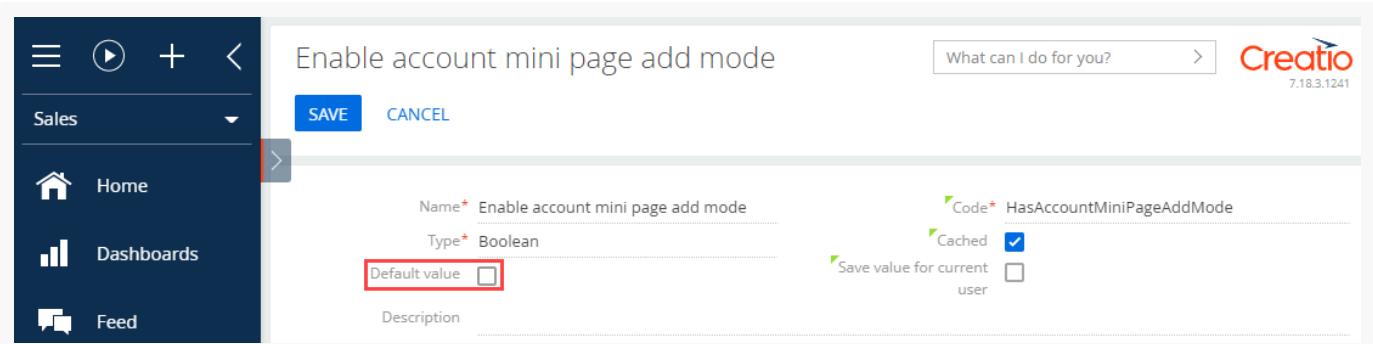
**Пример.** Добавить кнопку на панель инструментов страницы добавления контрагента. Кнопка активна после добавления основного контакта контрагента. При нажатии на кнопку открывается страница основного контакта этого контрагента.

## 1. Изменить способ добавления контрагента

По умолчанию для добавления контрагента используется мини-карточка.

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

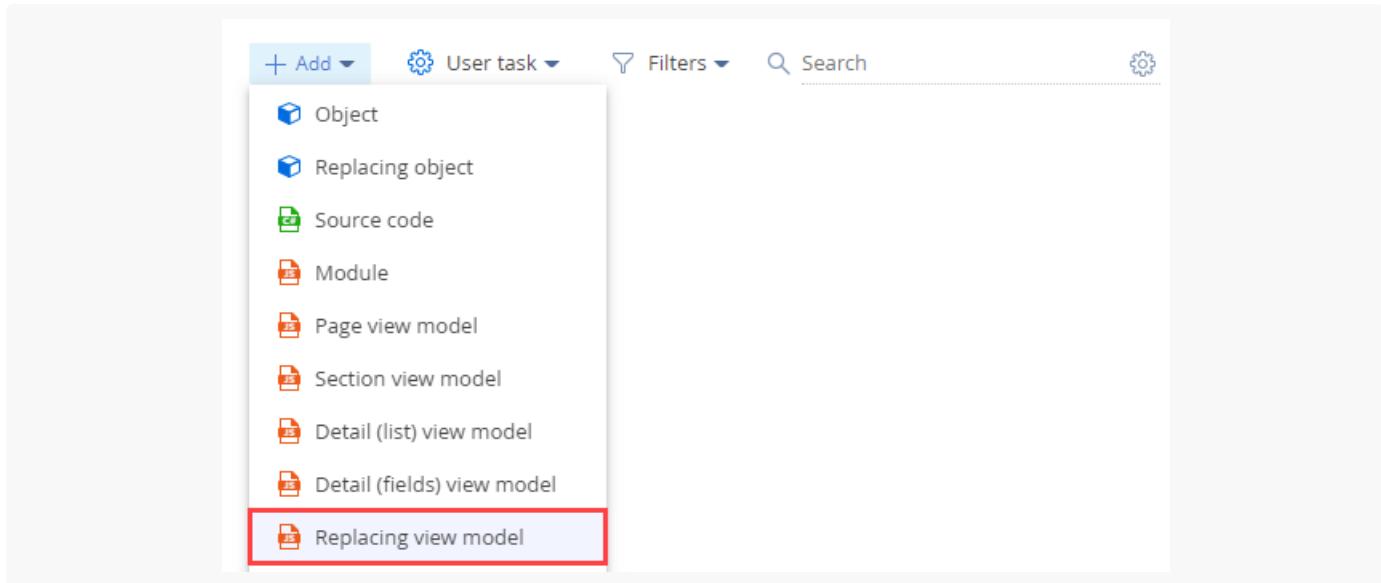
- Перейдите в дизайнер системы по кнопке .
- В блоке [ Настойка системы ] ([ System setup ]) перейдите по ссылке [ Системные настройки ] ([ System settings ]).
- Выберите настройку [ Использовать миникарточку добавления контрагента ] ([ Enable account mini page add mode ], код `HasAccountMiniPageAddMode`).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



- Выполните повторный вход в приложение.

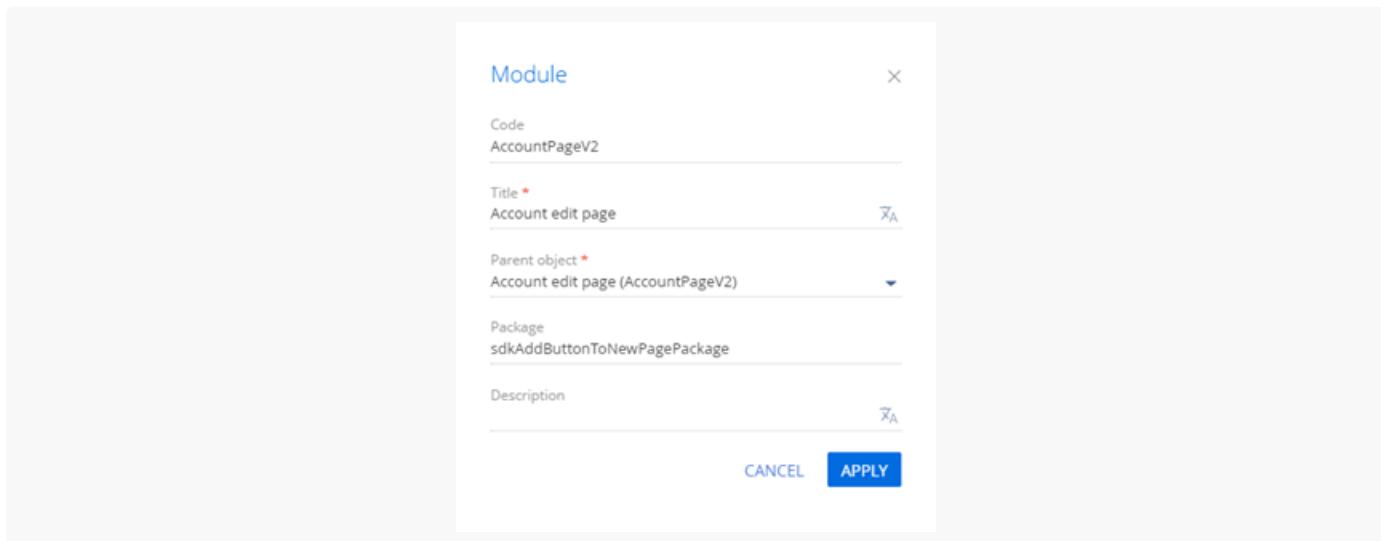
## 2. Создать схему замещающей модели представления страницы контрагента

- 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 ]).



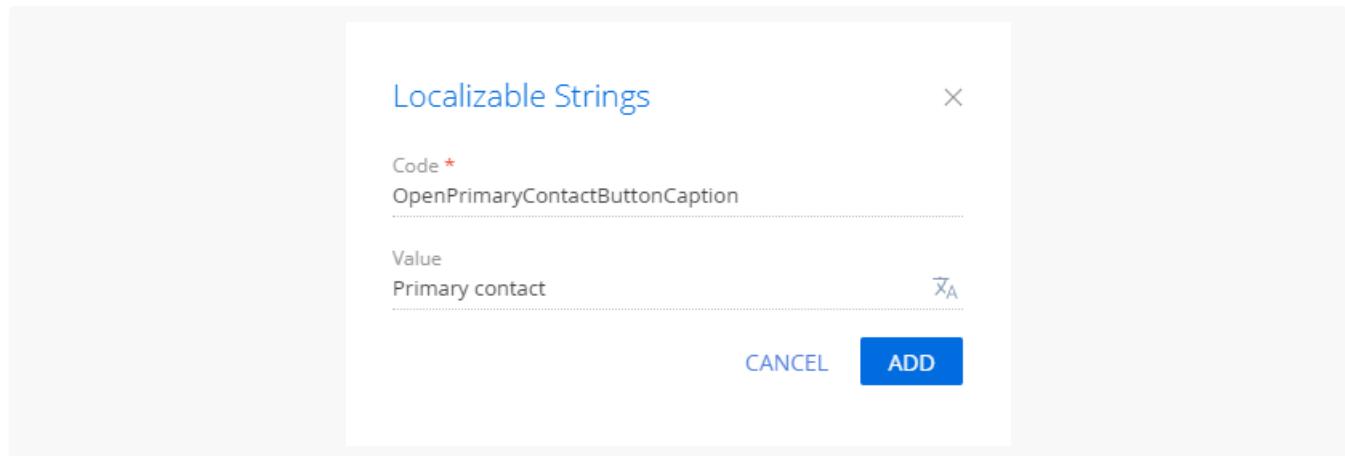
### 3. Заполните **свойства схемы**.

- [ 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

Industry

Communication options +

Addresses

Banking details

Noteworthy events + :

No data

Configuration items + :

No data

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.

New record

What can I do for you? >

**PRIMARY CONTACT**

SAVE CANCEL ACTIONS ▾

VIEW ▾

Primary phone

Category

Industry

Annual revenue

Communication options +

Addresses

Banking details

Noteworthy events + :

No data

Configuration items + :

No data

Primary contact  
James Smith

Full job title  
Senior Account Manager

Mobile phone  
+44 (787) 121 4006

Business phone  
+44 (15) 1432 4926

Email  
smith@gateway-invest.co.uk

## 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 search results page with the following columns: Status and Type. The items 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	

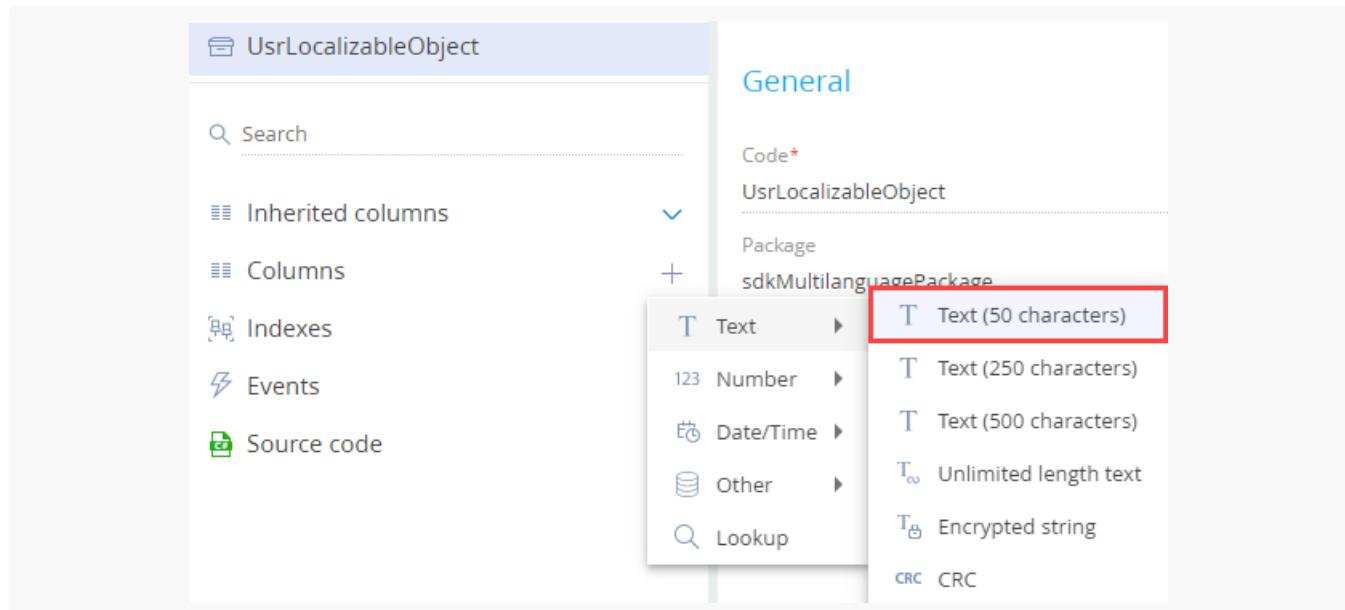
  

**Inheritance**

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

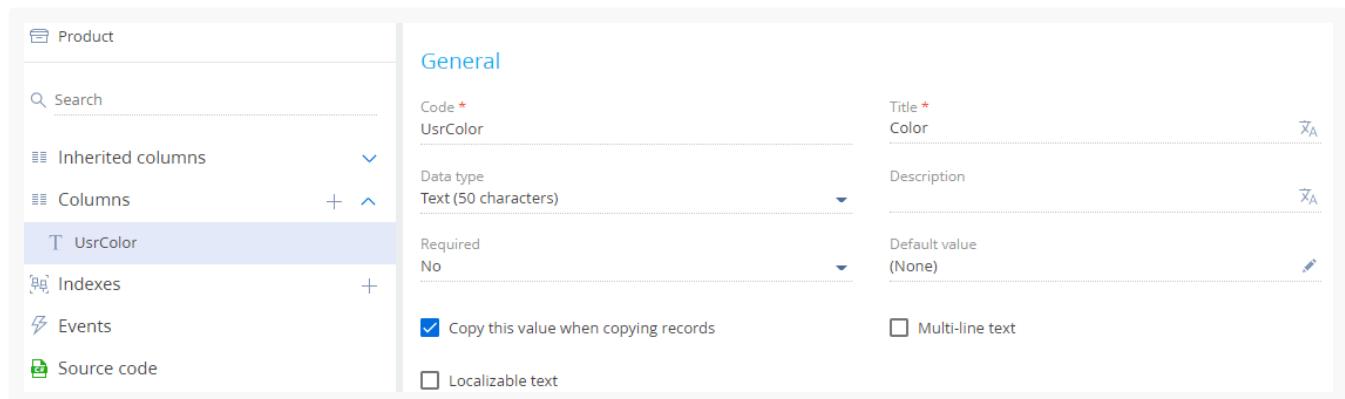
### 4. В схему добавьте **колонку**.

- 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**.

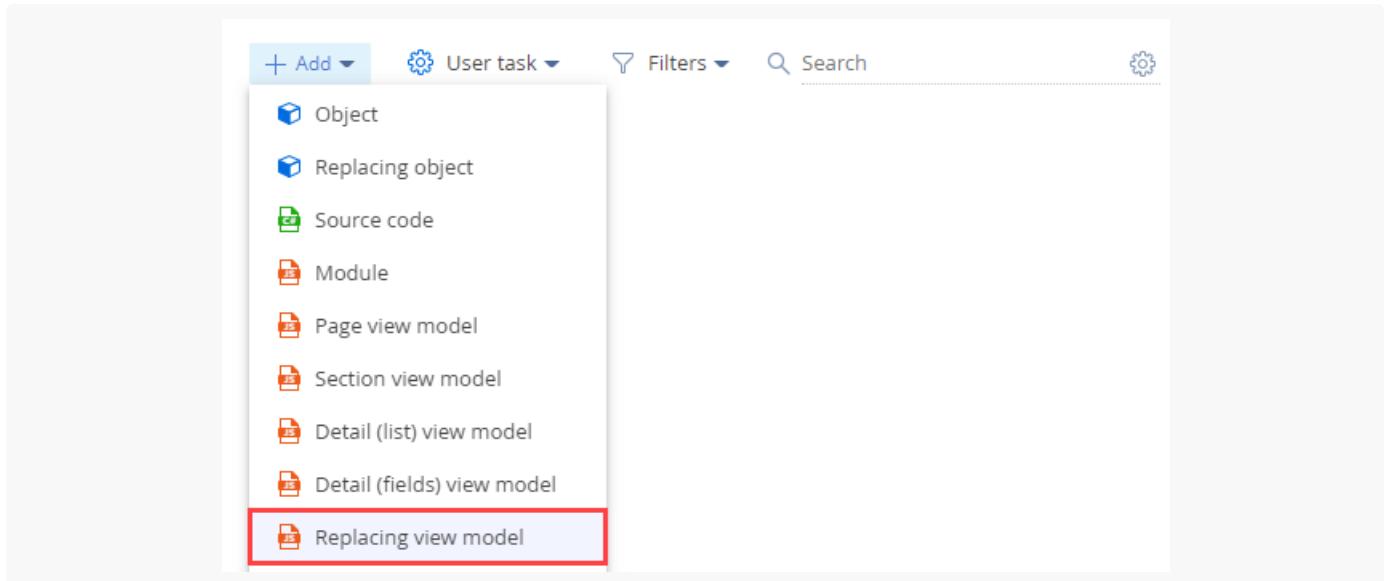
- [ Код ] ([ Code ]) — "UsrColor".
- [ Заголовок ] ([ Title ]) — "Цвет" ("Color").



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

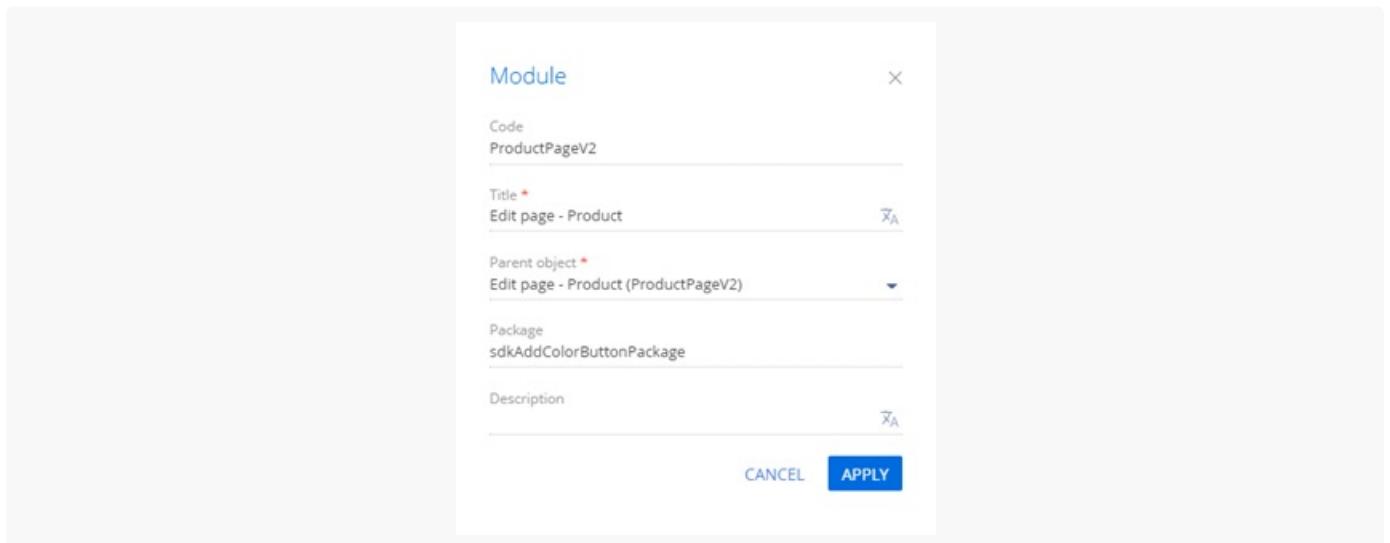
## 2. Создать схему замещающей модели представления страницы контрагента

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.

The screenshot shows a software application window titled "Installing software". On the left is a sidebar with icons for Opportunities, Orders, Contracts, Invoices, Documents, and Products. The main area displays a record for "Installing software" with fields for Name, Code, Link, Owner, and Inactive status. A color palette is overlaid on the screen, with a red box highlighting it. Below the palette, there are tabs for GENERAL INFORMATION, FEATURES, and ATTACHMENTS.

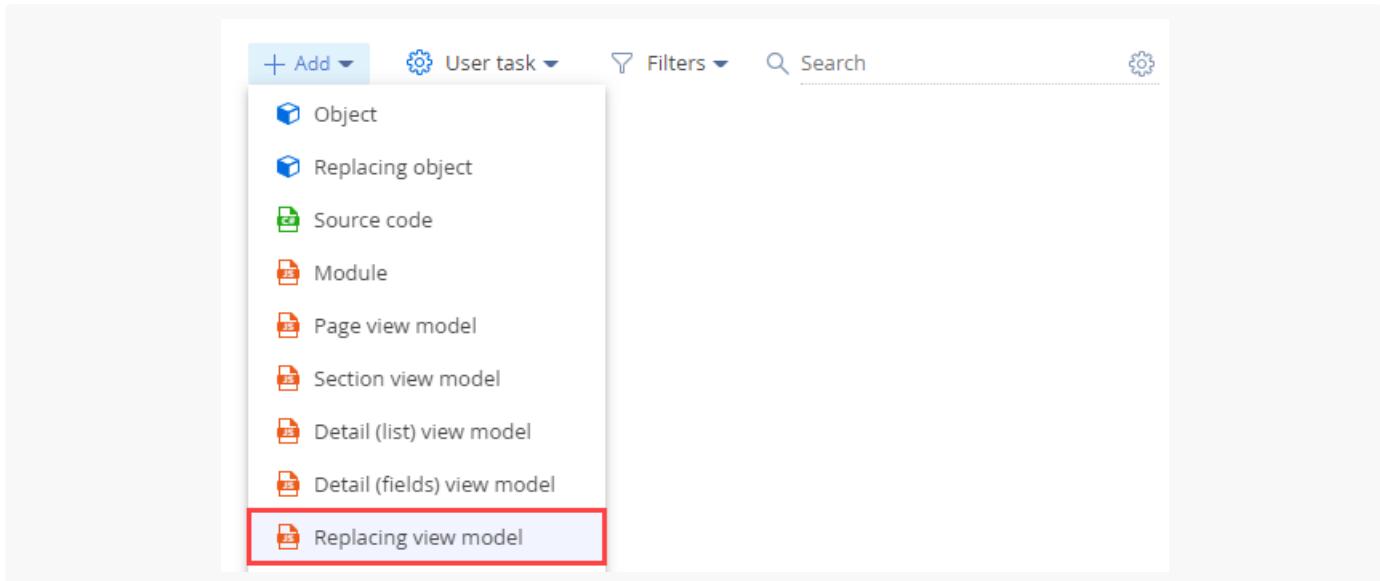
## Добавить к кнопке всплывающую подсказку

Сложный

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

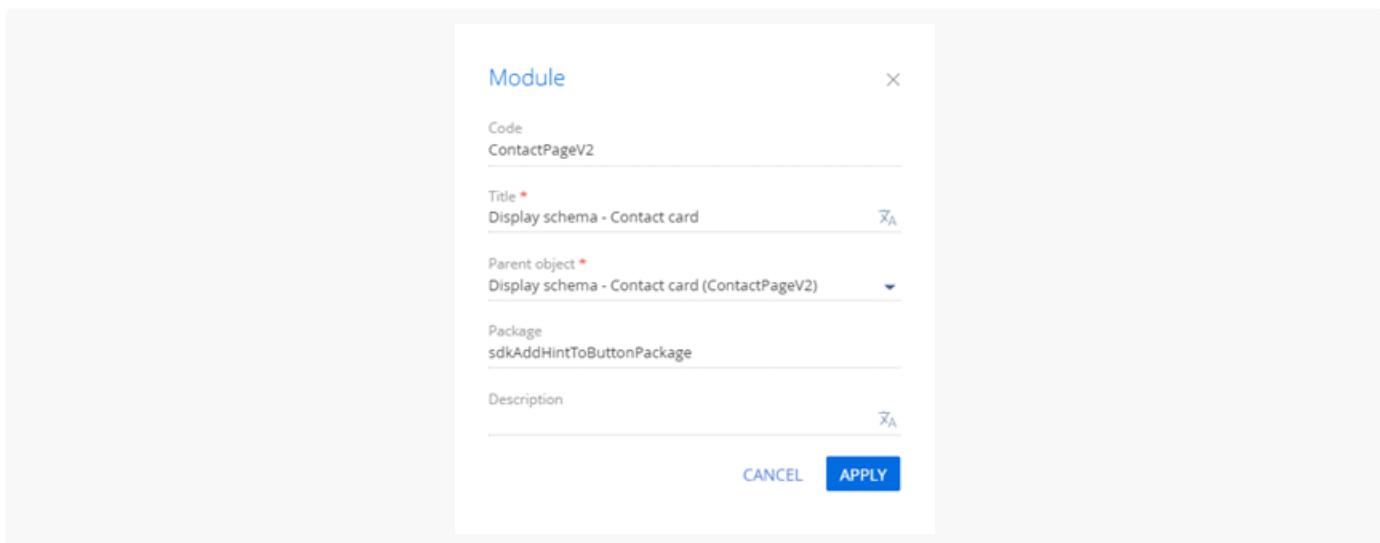
## Создать схему замещающей модели представления страницы контакта

- 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 ]).



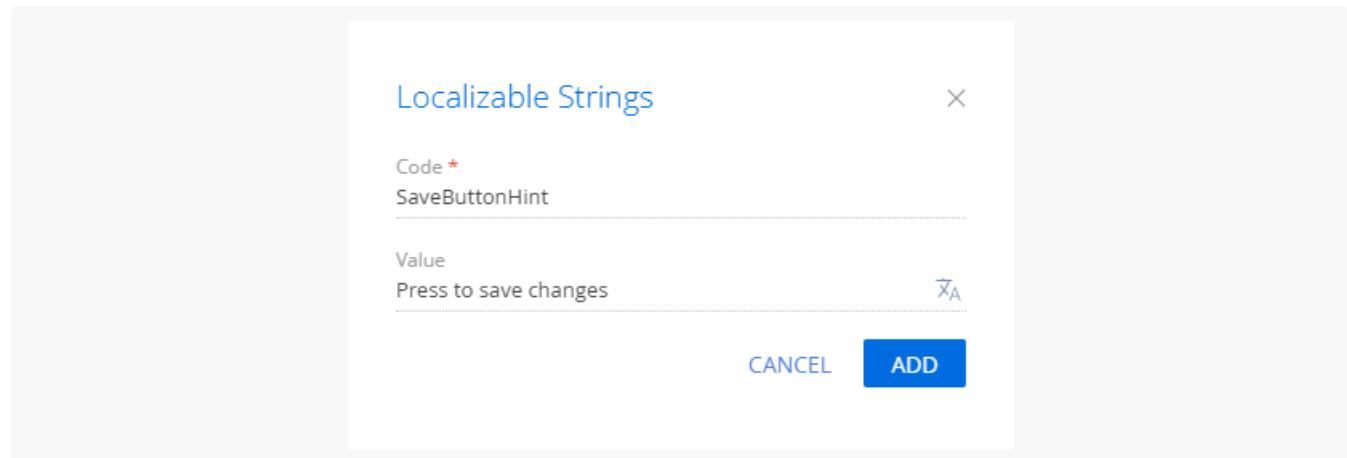
### 3. Заполните **свойства схемы**.

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



### 4. 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)

```

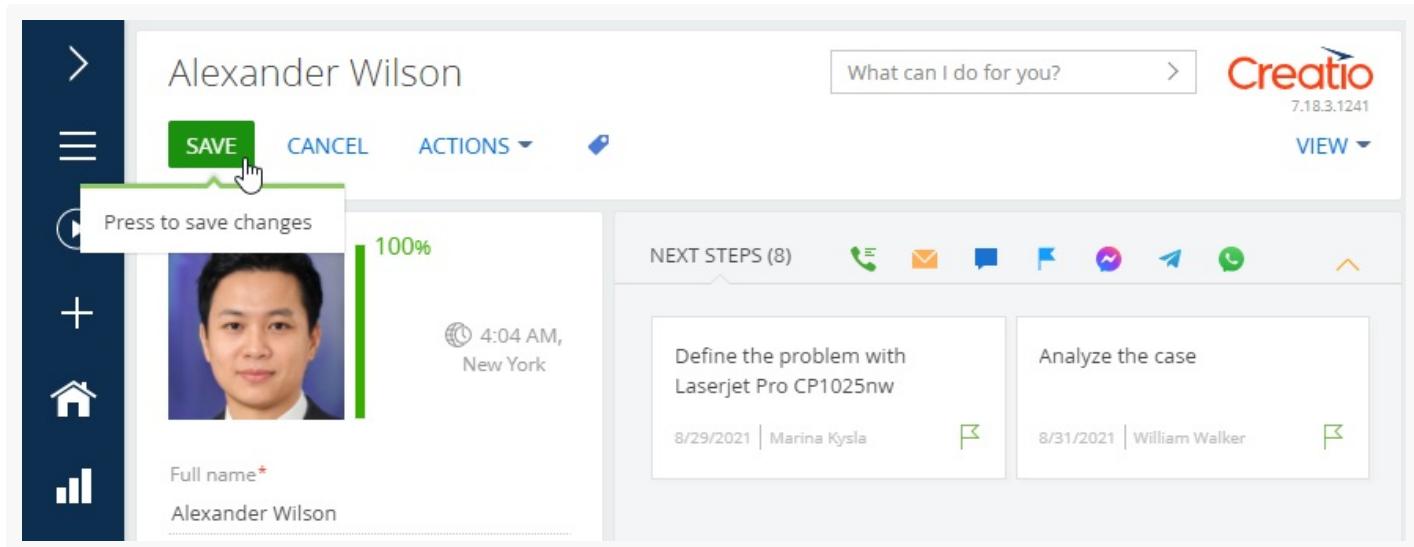
define("ContactPageV2", [], function () {
    return {
        /* Название схемы объекта страницы записи. */
        entitySchemaName: "Contact",
        /* Отображение всплывающей подсказки. */
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для добавления к кнопке всплывающей подсказки. */
            {
                /* 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.



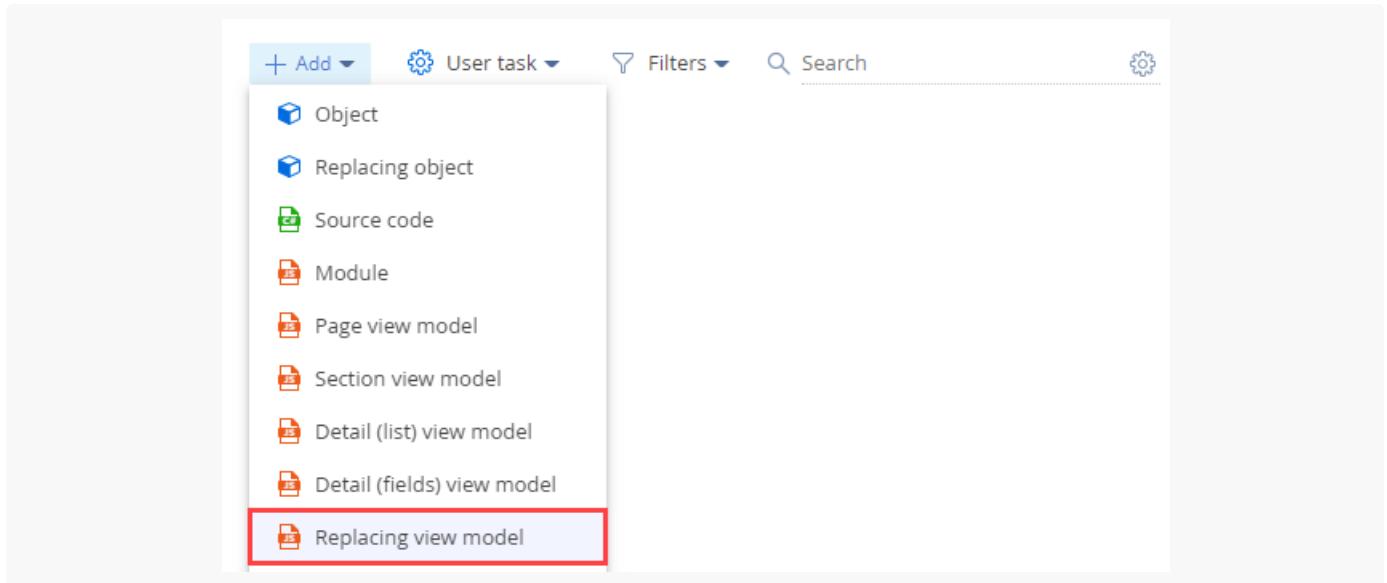
## Добавить кнопку в строку записи раздела

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.

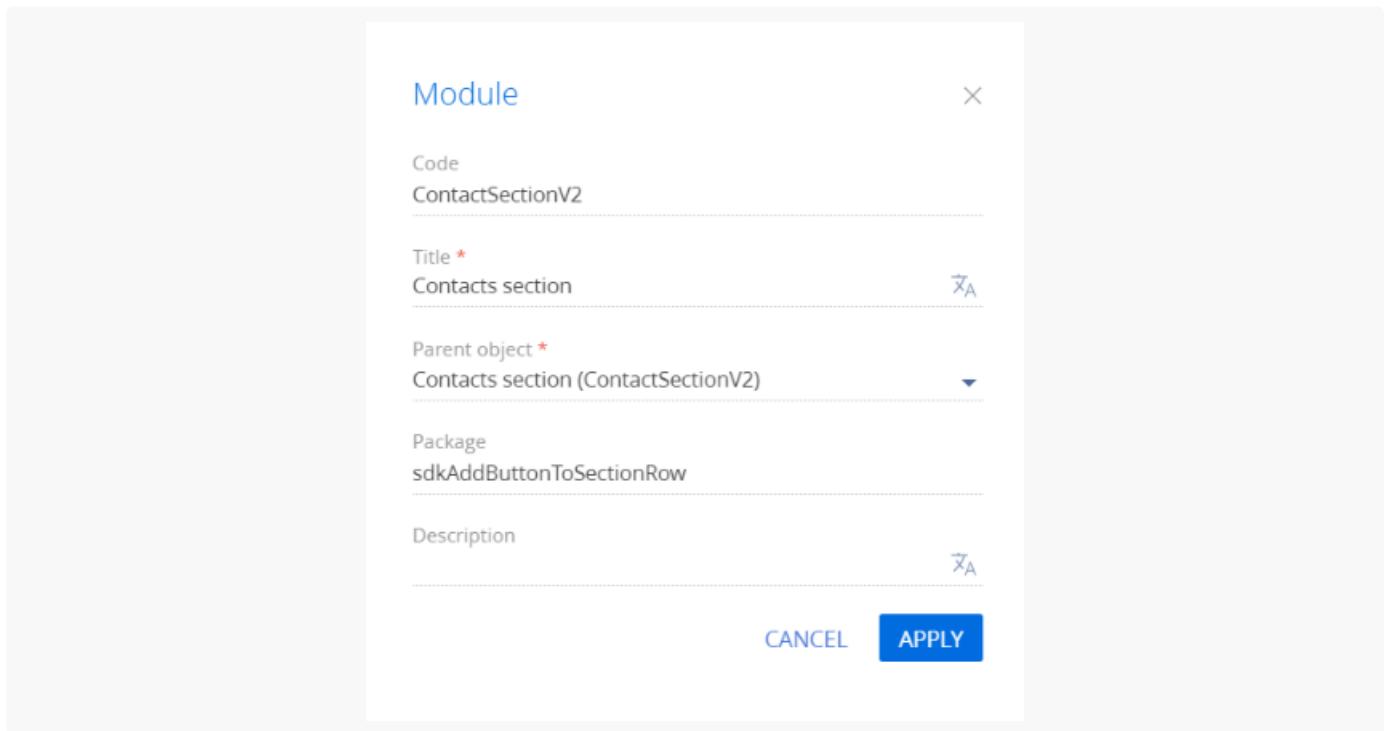
## Создать схему замещающей модели представления раздела

- 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 ]).



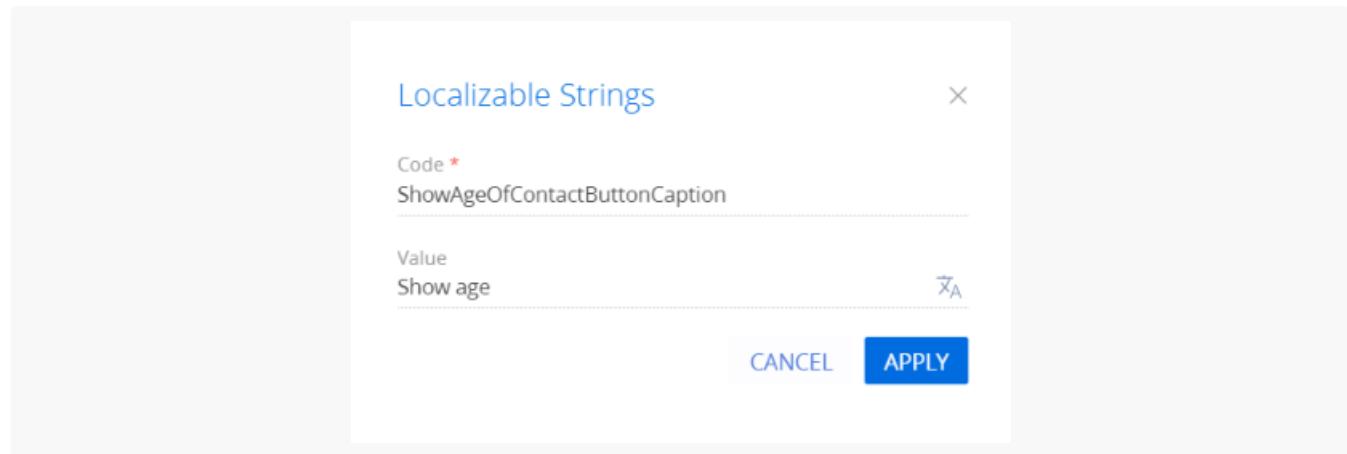
### 3. Заполните **свойства схемы**.

- [ 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*/
};

});

```

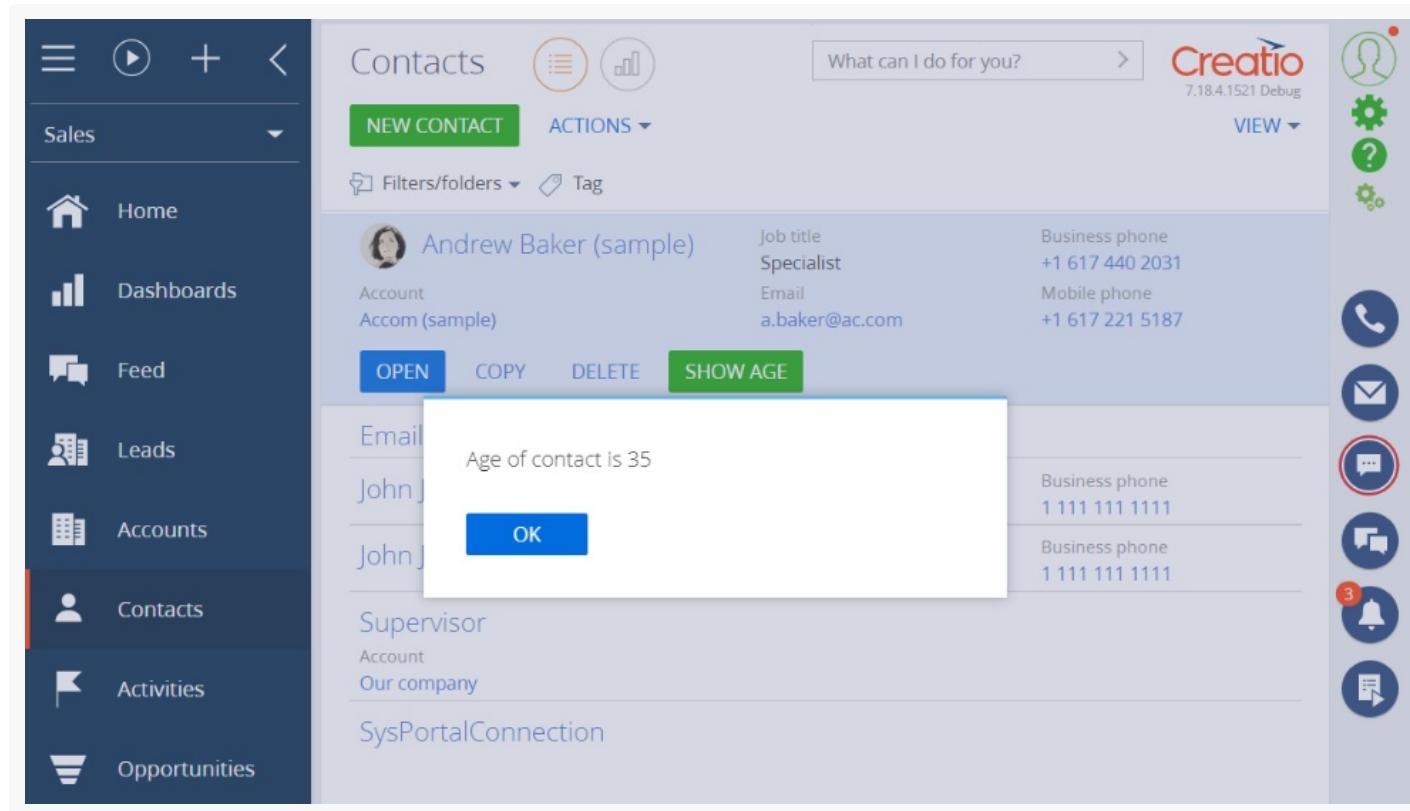
6. На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

## 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**

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

# Деталь



Основы

**Деталь** — элемент интерфейса на странице записи, который отображает записи определенного объекта, связанного с текущей записью. Например, на странице контакта детали используются для хранения информации о связанных с ним активностях, адресах, документах, и т. д. Большинство деталей имеют собственный реестр. Отдельные детали, например, [ Средства связи ] ([ *Communication options* ]), отображаются не в виде реестра. Визуально деталь отличается от [группы полей](#) наличием панели инструментов для управления данными (добавления и изменения записей, сортировки, фильтрации, настройки детали и других действий).

**Назначение** детали — отображение дополнительных данных для основного объекта раздела. Детали раздела отображаются во вкладках страницы записи раздела в контейнере вкладок.

## Структура и типы деталей

**Составляющие** детали:

- **Схема объекта детали** связана с объектом раздела. Например, детали [ Адреса ] ([ *Addresses* ]) страницы контакта соответствует схема объекта `ContactAddress` пакета `Base`. Связь с объектом раздела выполняется по обязательной колонке [ *Contact* ] объекта детали.
- **Схема модели представления детали** позволяет настроить структуру, расположение и поведение элементов пользовательского интерфейса детали. Например, деталь [ Адреса ] ([ *Addresses* ]) страницы контакта настраивается в схеме `ContactAddressDetailV2` модели представления детали, которая наследует схему `BaseAddressDetailV2` пакета `UIV2`.
- **Схема модели представления страницы записи детали** позволяет настроить страницу детали. Например, свойства страницы детали [ Адреса ] ([ *Addresses* ]) страницы контакта настраивается в схеме `ContactAddressPageV2` модели представления страницы записи детали, которая наследует схему `BaseAddressPageV2` пакета `UIV2`.

**Типы** деталей, которые предоставляет *Creatio*:

- Деталь с редактируемым реестром.
- Деталь со страницей добавления.
- Деталь с выбором из справочника.
- Деталь с полями.
- Деталь типа [ Файлы и ссылки ] ([ *Attachments* ]).

Тип детали зависит от метода ввода и отображения данных.

## Реализовать деталь

Функциональность базовой детали реализована в схеме `BaseDetailV2` пакета `NUI`.

**Инструменты**, которые позволяют реализовать деталь:

- Мастер деталей.
- Creatio IDE.

Реализовать некоторые типы деталей невозможно исключительно в мастере деталей. В этом случае необходимо использовать комбинацию мастера деталей и Creatio IDE. Особенности использования инструментов при реализации разных типов деталей будут рассмотрены далее.

Общий алгоритм реализации детали с использованием **мастера деталей**:

- Создайте пользовательскую деталь.** Для этого воспользуйтесь инструкцией, которая приведена в статье [Создать новую деталь](#).
- Добавьте пользовательскую деталь на страницу записи.** Для этого воспользуйтесь инструкцией, которая приведена в статье [Настроить существующую деталь на странице записи](#).
- Настройте внешний вид пользовательской детали** (опционально). Для этого воспользуйтесь инструкцией, которая приведена в статье [Настроить существующую деталь на странице записи](#).

Общий алгоритм реализации детали с использованием **Creatio IDE**:

- Создайте пользовательскую деталь.**
  - Создайте схему объекта детали. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
  - Создайте схему модели представления детали. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
  - Добавьте пользовательские стили детали (опционально).
  - Зарегистрируйте деталь в базе данных (опционально).
- Добавьте пользовательскую деталь на страницу записи.**  
Создайте схему замещающей модели представления страницы записи, на которой будет размещена деталь. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
- Настройте внешний вид пользовательской детали** (опционально). Для этого воспользуйтесь инструкцией, которая приведена в статье [Настроить существующую деталь на странице записи](#).

## Реализовать деталь с редактируемым реестром

**Деталь с редактируемым реестром** позволяет вводить и редактировать данные в реестре детали. Данные отображаются в списочном представлении. Деталь с редактируемым реестром является подвидом детали с реестром. Функциональность базовой детали с реестром реализована в схеме `BaseGridDetailV2` пакета `mui`. Примером детали с редактируемым реестром является деталь [Продукты] ([Products]) страницы заказа. Данные каждого продукта редактируются на странице заказа.

Products						ORDER DETAILS	DELIVERY	SUMMARY	HISTORY	APPROVALS	GENERAL INFORMATION	>
Products + :						Items: 3 Total: \$ 13,000.00						
Product	Price	Quantity	Unit of measure	Discount, %	Total							
Asus R9280X-DC2T-3GD5	450.00	25.000	pieces	0.00	11,250.00							
Installing software	100.00	10.000	hours	0.00	1000.00							
Windows 10 Pro	150.00	5.000	pieces	0.00	750.00							

Использовать мастер деталей для реализации детали с редактируемым реестром

## 1. Создайте пользовательскую деталь.

- Настройте редактируемый реестр. Для этого установите признак [ Сделать реестр редактируемым ] ([ Make the list editable ]). В другом случае будет создана деталь со страницей добавления.
- Настройте многострочный текст (опционально). Для отображения данных в несколько строк установите признак [ Многострочный текст ] ([ Multi-line text ]). Многострочный текст доступен к использованию только для колонок типа [ Стока ] ([ String ]).

## 2. Выполните шаг 2 общего [алгоритма реализации детали с использованием мастера деталей](#).

Использовать Creatio IDE для реализации детали с редактируемым реестром

## 1. Создайте пользовательскую деталь.

### a. Создайте схему объекта детали.

- В качестве родительского объекта выберите  `BaseEntity` .
- В схему объекта добавьте колонку типа [ Стока ] ([ String ]) и другие необходимые колонки.

### d. Создайте схему модели представления детали с редактируемым реестром.

- В качестве родительского объекта выберите  `BaseGridDetailV2` .
- На панели инструментов в контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) выберите локализуемую строку [ Caption ] и в свойстве [ Значение ] ([ Value ]) задайте название детали.
- Реализуйте редактируемый реестр.
  - В зависимости добавьте схемы модулей  `ConfigurationGrid` ,  `ConfigurationGridGenerator` ,  `ConfigurationGridUtilities` .
  - В свойство  `mixins`  добавьте МИКСИН  `ConfigurationGridUtilities` .
  - В свойство  `attributes`  добавьте атрибут  `IsEditable`  со значением  `true`  свойства  `value` .
- Реализуйте многострочный текст (опционально). Для этого в свойство  `attributes`  добавьте колонку типа [ Стока ] ([ String ]) со значением  `Terrasoft.ContentType.LONG_TEXT`  свойства

```
contentType .
```

Пример схемы `ContactPageV2` модели представления детали с редактируемым реестром `UsrCourierServiceDetail`, у которой для колонки [ *UsrDescription* ] используется многострочный текст, приведен ниже.

### Пример схемы замещающей модели представления

```
/* Определение схемы и установка ее зависимостей от других модулей. */
define("UsrCourierServiceDetail", ["ConfigurationGrid", "ConfigurationGridGenerator",
    "ConfigurationGridUtilities"], function() {
    return {
        /* Название схемы объекта детали. */
        entitySchemaName: "UsrCourierService",
        /* Перечень атрибутов схемы. */
        attributes: {
            /* Признак возможности редактирования. */
            "IsEditable": {
                /* Тип данных – логический. */
                dataType: Terrasoft.DataValueType.BOOLEAN,
                /* Тип атрибута – виртуальная колонка модели представления. */
                type: Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
                /* Устанавливаемое значение. */
                value: true
            }
        },
        /* Используемые миксины. */
        mixins: {
            ConfigurationGridUtilities: "Terrasoft.ConfigurationGridUtilities"
        },
        /* Массив модификаций модели представления. */
        diff: /**SCHEMA_DIFF*/[
            {
                /* Тип операции – слияние. */
                "operation": "merge",
                /* Название элемента схемы, над которым производится действие. */
                "name": "DataGrid",
                /* Объект, свойства которого будут объединены со свойствами элемента схемы. */
                "values": {
                    /* Имя класса. */
                    "className": "Terrasoft.ConfigurationGrid",
                    /* Генератор представления должен генерировать только часть представления. */
                    "generator": "ConfigurationGridGenerator.generatePartial",
                    /* Привязка события получения конфигурации элементов редактирования активной записи к методу-обработчику. */
                    "generateControlsConfig": {"bindTo": "generateActiveRowControlsConfig"},
                    /* Привязка события смены активной записи к методу-обработчику. */
                    "changeRow": {"bindTo": "changeRow"},
                    /* Привязка события отмены выбора записи к методу-обработчику. */
                    "cancelSelect": {"bindTo": "cancelSelect"}
                }
            }
        ]
    }
},
```

```

    "unSelectRow": {"bindTo": "unSelectRow"},  

    /* Привязка события клика на реестре к методу-обработчику. */  

    "onGridClick": {"bindTo": "onGridClick"},  

    /* Действия, производимые с активной записью. */  

    "activeRowActions": [  

        /* Настройка действия Сохранить. */  

        {  

            /* Имя класса элемента управления, с которым связано действие.  

            "className": "Terrasoft.Button",  

            /* Стиль отображения – прозрачная кнопка. */  

            "style": this.Terrasoft.controls.ButtonEnums.style.TRANSPARENT,  

            /* Тег. */  

            "tag": "save",  

            /* Значение маркера. */  

            "markerValue": "save",  

            /* Привязка к изображению кнопки. */  

            "imageConfig": {"bindTo": "Resources.Images.SaveIcon"}  

        },  

        /* Настройка действия Отменить. */  

        {  

            /* className": "Terrasoft.Button",  

            "style": this.Terrasoft.controls.ButtonEnums.style.TRANSPARENT,  

            "tag": "cancel",  

            "markerValue": "cancel",  

            "imageConfig": {"bindTo": "Resources.Images.CancelIcon"}  

        },  

        /* Настройка действия Удалить. */  

        {  

            /* className": "Terrasoft.Button",  

            "style": this.Terrasoft.controls.ButtonEnums.style.TRANSPARENT,  

            "tag": "remove",  

            "markerValue": "remove",  

            "imageConfig": {"bindTo": "Resources.Images.RemoveIcon"}  

        }  

    ],  

    /* Привязка к методу, который инициализирует подписку на события нажатия клавиш. */  

    "initActiveRowKeyMap": {"bindTo": "initActiveRowKeyMap"},  

    /* Привязка события выполнения действия активной записи к методу-обработчику. */  

    "activeRowAction": {"bindTo": "onActiveRowAction"},  

    /* Признак возможности выбора нескольких записей. */  

    "multiSelect": {"bindTo": "MultiSelect"},  

    /* Колонка описания. */  

    "UsrDescription": {  

        /* Тип отображения - длинный текст. */  

        "contentType": Terrasoft.ContentType.LONG_TEXT  

    }  

}
]/**SCHEMA_DIFF*/

```

```
    };
});
```

- i. Зарегистрируйте деталь в базе данных. Для этого выполните SQL-запрос к таблице [SysDetails] базы данных.

### SQL-запрос

```
DECLARE
    -- Название схемы детали.
    @ClientUnitSchemaName NVARCHAR(100) = 'UsrИмяСхемыДетали',
    -- Название схемы объекта детали.
    @EntitySchemaName NVARCHAR(100) = 'UsrИмяСхемыОбъектаДетали',
    -- Название детали.
    @DetailCaption NVARCHAR(100) = 'ИмяДетали'

    INSERT INTO SysDetail(
        Caption,
        DetailSchemaUId,
        EntitySchemaUId
    )
    VALUES(
        @DetailCaption,
        (SELECT TOP 1 UIId
        from SysSchema
        WHERE Name = @ClientUnitSchemaName),
        (SELECT TOP 1 UIId
        from SysSchema
        WHERE Name = @EntitySchemaName)
    )
```

Регистрация детали выполняется, чтобы деталь стала видимой для мастера разделов и мастера деталей.

## 2. Добавьте пользовательскую деталь на страницу записи.

Создайте схему замещающей модели представления страницы записи, на которой будет размещена деталь с редактируемым реестром.

- В качестве родительского объекта выберите схему модели представления, которую необходимо заменить.
- В свойство `details` добавьте деталь.
- В массив модификаций `diff` добавьте конфигурационный объект модели представления детали.

Пример схемы `ContactPageV2` замещающей модели представления страницы записи, на которой размещена деталь с редактируемым реестром `UsrRegDocumentFieldsDetail` приведен ниже.

## Пример схемы замещающей модели представления

```

define("ContactPageV2", [], function() {
    return {
        entitySchemaName: "Contact",
        details: /**SCHEMA_DETAILS*/ {
            /* Добавление детали с полями. */
            "UsrRegDocumentFieldsDetail": {
                /* Название клиентской схемы детали. */
                "schemaName": "UsrRegDocumentFieldsDetail",
                /* Фильтрация записей детали текущего контакта (физ. лица). */
                "filter": {
                    /* Колонка объекта детали. */
                    "detailColumn": "UsrContact",
                    /* Колонка идентификатора контакта. */
                    "masterColumn": "Id"
                }
            }
        } /**SCHEMA_DETAILS*/ ,
        diff: /**SCHEMA_DIFF*/ [
            /* Добавление нового элемента. */
            {"operation": "insert",
             /* Название элемента. */
             "name": "UsrRegDocumentFieldsDetail",
             /* Конфигурационный объект значений. */
             "values": {
                 /* Тип элемента. */
                 "itemType": Terrasoft.ViewItemType.DETAIL
             },
             /* Имя элемента-контейнера. */
             "parentName": "HistoryTab",
             /* Имя свойства элемента-контейнера, который содержит коллекцию вложенных элементов. */
             "propertyName": "items",
             /* Индекс добавляемого в коллекцию элемента. */
             "index": 0
            }] /**SCHEMA_DIFF*/
    };
});

```

## Реализовать деталь со страницей добавления

**Деталь со страницей добавления** позволяет вводить и редактировать данные на странице детали. Деталь со страницей добавления является подвидом детали с реестром. Функциональность базовой детали с реестром реализована в схеме `BaseGridDetailV2` пакета `NUI`. Примером детали со страницей добавления является деталь `[ Адреса ] ([ Addresses ])` страницы контакта. Данные каждого адреса вводятся и редактируются на странице `[ Адрес контакта ] ([ Contact address ])`.

Чтобы реализовать деталь со страницей добавления с использованием **мастера деталей**, используйте общий алгоритм реализации детали с использованием мастера деталей.

Чтобы реализовать деталь со страницей добавления с использованием **Creatio IDE**:

#### 1. Создайте пользовательскую деталь.

a. Создайте схему объекта детали.

- В качестве родительского объекта выберите  `BaseEntity`.
  - В схему объекта добавьте необходимые колонки.
- d. Создайте схему модели представления детали со страницей добавления.
- В качестве родительского объекта выберите  `BaseGridDetailV2`.
  - На панели инструментов в контекстном меню узла [ *Локализуемые строки* ] ([ *Localizable strings* ]) выберите локализуемую строку [ *Caption* ] и в свойстве [ *Значение* ] ([ *Value* ]) задайте название детали.

g. Создайте схему модели представления страницы записи детали. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).

- В качестве родительского объекта выберите  `BasePageV2`.
- В массив модификаций `diff` добавьте конфигурационный объект модели представления детали.

Пример схемы  `UsrCourierDetailPage` модели представления страницы записи детали  `UsrCourierInOrder` приведен ниже.

#### Пример схемы модели представления страницы записи детали

```
define("UsrCourierDetailPage", [], function() {
    return {
        /* Название схемы объекта детали. */
        entitySchemaName: "UsrCourierInOrder",
        details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
```

```

/* Массив модификаций. */
diff: /**SCHEMA_DIFF*/[
    /* Метаданные для добавления поля [Заказ]. */
    {
        "operation": "insert",
        /* Название поля. */
        "name": "Order",
        "values": {
            /* Настройка расположения поля на странице записи. */
            "layout": {
                "colSpan": 12,
                "rowSpan": 1,
                "column": 0,
                "row": 0,
                "layoutName": "Header"
            },
            /* Привязка к колонке [Order] схемы объекта. */
            "bindTo": "UsrOrder"
        },
        "parentName": "Header",
        "propertyName": "items",
        "index": 0
    },
    /* Метаданные для добавления поля [Контакт]. */
    {
        "operation": "insert",
        /* Название поля. */
        "name": "Contact",
        "values": {
            /* Настройка расположения поля на странице записи. */
            "layout": {
                "colSpan": 12,
                "rowSpan": 1,
                "column": 12,
                "row": 0,
                "layoutName": "Header"
            },
            /* Привязка к колонке [Contact] схемы объекта. */
            "bindTo": "UsrContact"
        },
        "parentName": "Header",
        "propertyName": "items",
        "index": 1
    }
]/**SCHEMA_DIFF*/,
methods: {},
rules: {}
};

});

```

j. Зарегистрируйте деталь в базе данных.

а. Зарегистрируйте связь между схемой объекта детали и схемой реестра детали. Для этого выполните SQL-запрос к таблице [SysDetails] базы данных.

### SQL-запрос

```

DECLARE
    -- Название схемы детали.
    @DetailSchemaName NCHAR(100) = 'UsrИмяСхемыДетали',
    -- Название схемы объекта детали.
    @EntitySchemaName NVARCHAR(100) = 'UsrИмяСхемыОбъектаДетали',
    -- Название детали.
    @DetailCaption NVARCHAR(100) = 'ИмяДетали'

INSERT INTO SysDetail(
    ProcessListeners,
    Caption,
    DetailSchemaUID,
    EntitySchemaUID
)
VALUES (
    0,
    @DetailCaption,
    (SELECT TOP 1 UID
    FROM SysSchema
    WHERE name = @DetailSchemaName),
    (SELECT TOP 1 UID
    FROM SysSchema
    WHERE name = @EntitySchemaName)
)

```

b. Зарегистрируйте связь между схемой объекта детали и схемой страницы записи детали. Для этого выполните SQL-запрос к таблицам [SysModuleEntity] и [SysModuleEdit] базы данных.

### SQL-запрос

```

DECLARE
    -- Название схемы страницы детали.
    @CardSchemaName NCHAR(100) = 'UsrИмяСхемыСтраницыДетали',
    -- Название схемы объекта детали.
    @EntitySchemaName NVARCHAR(100) = 'UsrИмяСхемыОбъектаДетали',
    -- Название страницы детали.
    @PageCaption NVARCHAR(100) = 'ИмяСтраницыДетали',
    -- Пустая строка.
    @Blank NCHAR(100) = ''

```

```

INSERT INTO SysModuleEntity(
    ProcessListeners,
    SysEntitySchemaUID
)
VALUES (
    0,
    (SELECT TOP 1 UId
    FROM SysSchema
    WHERE Name = @EntitySchemaName
    )
)

INSERT INTO SysModuleEdit(
    SysModuleEntityId,
    UseModuleDetails,
    Position,
    HelpContextId,
    ProcessListeners,
    CardSchemaUID,
    ActionKindCaption,
    ActionKindName,
    PageCaption
)
VALUES (
    (SELECT TOP 1 Id
    FROM SysModuleEntity
    WHERE SysEntitySchemaUID = (
        SELECT TOP 1 UId
        FROM SysSchema
        WHERE Name = @EntitySchemaName
        )
    ),
    1,
    0,
    @Blank,
    0,
    (SELECT TOP 1 UId
    FROM SysSchema
    WHERE name = @CardSchemaName
    ),
    @Blank,
    @Blank,
    @PageCaption
)

```

Регистрация детали выполняется, чтобы деталь стала видимой для мастера разделов и мастера деталей.

k. Для применения изменений перезапустите приложение в IIS.

## 2. Добавьте пользовательскую деталь на страницу записи.

Создайте схему замещающей модели представления страницы записи, на которой будет размещена деталь со страницей добавления.

- В качестве родительского объекта выберите схему модели представления, которую необходимо заменить.
- В свойство `details` добавьте деталь.
- В массив модификаций `diff` добавьте конфигурационный объект модели представления детали.

3. Выполните шаг 4 общего [алгоритма реализации детали с использованием Creatio IDE](#). Без выполнения этого шага деталь отображается на странице записи, но не содержит записей, поскольку не указаны колонки для отображения.

## Реализовать деталь с выбором из справочника

**Деталь с выбором из справочника** позволяет выбирать данные из справочника, который отображается в модальном окне. Деталь с выбором из справочника является подвидом детали с реестром. Функциональность базовой детали с реестром реализована в схеме `BaseGridDetailV2` пакета `nui`. Примером детали с выбором из справочника является деталь [ Продукты ] ([ `Products` ]) страницы листа. Выбор продукта выполняется в модальном окне [ Выбор: Продукты ] ([ `Select: Products` ]).

The screenshot shows the Creatio application interface. On the left is a dark sidebar with various icons: a right arrow, three horizontal lines, a play button, a plus sign, a house, a bar chart, a speech bubble, a user icon, a grid icon, a document icon, a person icon, and a flag icon. The main area displays a lead record for 'Ronald Gitte...'. The top navigation bar includes 'What can I do for you?' with a dropdown, the 'Creatio' logo (version 7.18.3.1241), and 'VIEW' with a dropdown. Below the navigation is a toolbar with 'CLOSE', 'ACTIONS', and 'QUALIFY' buttons. The main content area has tabs: 'LEAD INFO', 'CUSTOMER NEED DETAILS' (which is selected and highlighted in red), 'LEAD ENGAGEMENT', and 'WEBSITE EVENTS'. Under 'CUSTOMER NEED DETAILS', there are sections for 'Need maturity' (set to 'Suspected') and 'Notes'. Below these are sections for 'Features' and 'Products'. A modal window titled 'Select: Products' is open, showing a list of products with three yellow stars under 'Predictive score'. The modal has buttons for 'New account' and 'Search'. The right side of the interface features a vertical column of circular icons with icons for phone, email, messaging, and other communication channels, with a '99+' notification badge above the messaging icon.

Использовать мастер деталей для реализации детали с выбором из справочника

### 1. Создайте пользовательскую деталь.

- Добавьте на деталь колонку типа [ Справочник ] ([ *Lookup* ]).
  - Настройте вид справочника. Для этого в свойстве [ Вид справочника ] ([ *Lookup view* ]) выберите значение "Всплывающее окно" ("Selection window").
2. Выполните шаг 2 общего [алгоритма реализации детали с использованием мастера деталей](#).

Использовать Creatio IDE для реализации детали с выбором из справочника

### 1. Создайте пользовательскую деталь.

a. Создайте схему объекта детали.

- В качестве родительского объекта выберите  `BaseEntity` .
  - В схему объекта добавьте колонку типа [ Справочник ] ([ *Lookup* ]) и другие необходимые колонки.
- d. Создайте схему модели представления детали с выбором из справочника.
- В качестве родительского объекта выберите  `BaseGridDetailV2` .
  - На панели инструментов в контекстном меню узла [ Локализуемые строки ] ([ *Localizable strings* ]) выберите локализуемую строку [ *Caption* ] и в свойстве [ Значение ] ([ *Value* ]) задайте название детали.

### 2. Добавьте пользовательскую деталь на страницу записи.

Создайте схему замещающей модели представления страницы записи, на которой будет размещена деталь с выбором из справочника.

- В качестве родительского объекта выберите схему модели представления, которую необходимо заместить.
- В зависимости схемы модели представления страницы записи добавьте схему модуля  `ConfigurationEnums` .
- В свойство  `methods` добавьте методы:
  - `onDocumentInsert()` — обрабатывает событие добавления записей в реестр детали.
  - `onCardSaved()` — обрабатывает событие сохранения страницы записи с деталью.
  - `openDocumentLookup()` — вызывает модальное окно справочника.
  - Вспомогательные методы управления данными.
- В массив модификаций  `diff` добавьте конфигурационный объект модели представления детали.

Пример схемы  `UsrCourierCertDetail` модели представления страницы записи, на которой размещена деталь с выбором из справочника  `UsrCourierCertInOrder` приведен ниже.

#### Пример схемы замещающей модели представления

```

/* Определение схемы и установка ее зависимостей от других модулей. */
define("UsrCourierCertDetail", ["ConfigurationEnums"],
    function(configurationEnums) {
        return {
            /* Название схемы объекта детали. */
            entitySchemaName: "UsrCourierCertInOrder",
            /* Методы схемы детали. */
            methods: {
                /* Возвращает колонки, которые выбираются запросом. */
                getGridDataColumns: function() {
                    return {
                        "Id": {path: "Id"},
                        "Document": {path: "UsrDocument"},
                        "Document.Number": {path: "UsrDocument.Number"}
                    };
                },
                /* Конфигурирует и отображает модальное окно справочника. */
                openDocumentLookup: function() {
                    /* Конфигурационный объект. */
                    var config = {
                        /* Название схемы объекта, записи которого будут отображены в справочнике. */
                        entitySchemaName: "Document",
                        /* Возможность множественного выбора. */
                        multiSelect: true,
                        /* Колонки, которые будут использованы в справочнике, например, для сортировки. */
                        columns: ["Number", "Date", "Type"]
                    };
                    var OrderId = this.get("MasterRecordId");
                    if (this.Ext.isEmpty(OrderId)) {
                        return;
                    }
                    /* Экземпляр класса EntitySchemaQuery. */
                    var esq = this.Ext.create("Terrasoft.EntitySchemaQuery", {
                        /* Установка корневой схемы. */
                        rootSchemaName: this.entitySchemaName
                    });
                    /* Добавление колонки Id. */
                    esq.addColumn("Id");
                    /* Добавление колонки Id из схемы Document. */
                    esq.addColumn("Document.Id", "DocumentId");
                    /* Создание и добавление фильтров в коллекцию запроса. */
                    esq.filters.add("filterOrder", this.Terrasoft.createColumnFilterWithParam(
                        this.Terrasoft.ComparisonType.EQUAL, "UsrOrder", OrderId));
                    /* Получение всей коллекции записей и отображение ее в модальном окне справочника. */
                    esq.getEntityCollection(function(result) {
                        var existsDocumentsCollection = [];
                        if (result.success) {

```

```

        result.collection.each(function(item) {
            existsDocumentsCollection.push(item.get("DocumentId"));
        });
    }
    /* Добавление фильтра в конфигурационный объект. */
    if (existsDocumentsCollection.length > 0) {
        var existsFilter = this.Terrasoft.createColumnInFilterWithParamet
            existsDocumentsCollection);
        existsFilter.comparisonType = this.Terrasoft.ComparisonType.NOT_E
        existsFilter.Name = "existsFilter";
        config.filters = existsFilter;
    }
    /* Вызов модального окна справочника. */
    this.openLookup(config, this.addCallBack, this);
}, this);
},

/* Обработчик события сохранения страницы записи. */
onCardSaved: function() {
    this.openDocumentLookup();
},

/* Открывает справочник документов в случае если страница заказа была ранее с
addRecord: function() {
    var masterCardState = this.sandbox.publish("GetCardState", null, [this.sa
    var isNewRecord = (masterCardState.state === configurationEnums.CardState
    masterCardState.state === configurationEnums.CardStateV2.COPY);
    if (isNewRecord === true) {
        var args = {
            isSilent: true,
            messageTags: [this.sandbox.id]
        };
        this.sandbox.publish("SaveRecord", args, [this.sandbox.id]);
        return;
    }
    this.openDocumentLookup();
},
}

/* Добавление выбранных продуктов. */
addCallBack: function(args) {
    /* Экземпляр класса пакетного запроса BatchQuery. */
    var bq = this.Ext.create("Terrasoft.BatchQuery");
    var OrderId = this.get("MasterRecordId");
    /* Коллекция выбранных в справочнике документов. */
    this.selectedRows = args.selectedRows.getItems();
    /* Коллекция, передаваемая в запрос. */
    this.selectedItems = [];
    /* Копирование необходимых данных. */

```

```

        this.selectedRows.forEach(function(item) {
            item.OrderId = OrderId;
            item.DocumentId = item.value;
            bq.add(this.getDocumentInsertQuery(item));
            this.selectedItems.push(item.value);
        }, this);
        /* Выполнение пакетного запроса, если он не пустой. */
        if (bq.queries.length) {
            this.showBodyMask.call(this);
            bq.execute(this.onDocumentInsert, this);
        }
    },
    /* Возвращает запрос на добавление текущего объекта. */
    getDocumentInsertQuery: function(item) {
        var insert = Ext.create("Terrasoft.InsertQuery", {
            rootSchemaName: this.entitySchemaName
        });
        insert.setParameterValue("UsrOrder", item.OrderId, this.Terrasoft.DataVal);
        insert.setParameterValue("UsrDocument", item.DocumentId, this.Terrasoft.D
        return insert;
    },
    /* Метод, вызываемый при добавлении записей в реестр детали. */
    onDocumentInsert: function(response) {
        this.hideBodyMask.call(this);
        this.beforeLoadGridData();
        var filterCollection = [];
        response.queryResults.forEach(function(item) {
            filterCollection.push(item.id);
        });
        var esq = Ext.create("Terrasoft.EntitySchemaQuery", {
            rootSchemaName: this.entitySchemaName
        });
        this.initQueryColumns(esq);
        esq.filters.add("recordId", Terrasoft.createColumnInFilterWithParameters(
        /* Создание модели представления. */
        esq.on("createviewmodel", this.createViewModel, this);
        esq.getEntityCollection(function(response) {
            this.afterLoadGridData();
            if (response.success) {
                var responseCollection = response.collection;
                this.prepareResponseCollection(responseCollection);
                this.getGridData().loadAll(responseCollection);
            }
        }, this);
    },
    /* Метод, вызываемый при удалении выбранных записей детали. */

```

```

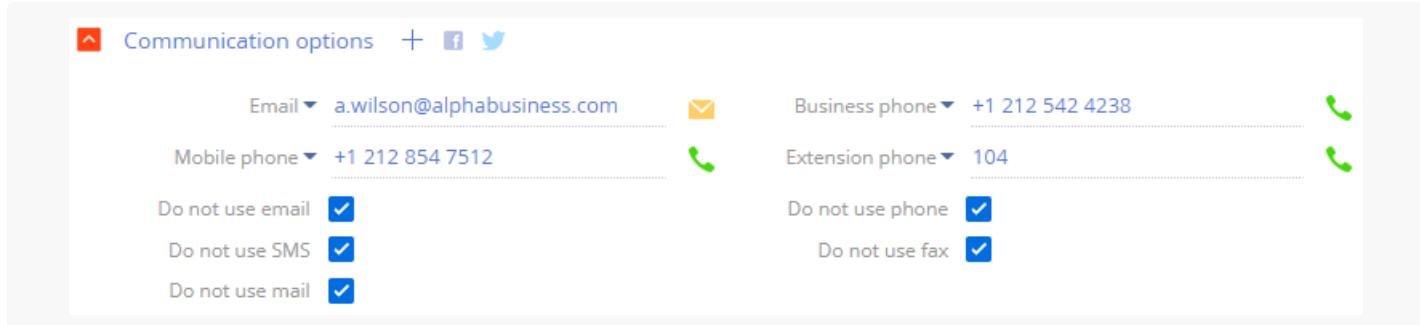
        deleteRecords: function() {
            var selectedRows = this.getSelectedItems();
            if (selectedRows.length > 0) {
                this.set("SelectedRows", selectedRows);
                this.callParent(arguments);
            }
        },
        /* Скрыть пункт меню Копировать. */
        getCopyRecordMenuItem: Terrasoft.emptyFn,
        /* Скрыть пункт меню Изменить. */
        getEditRecordMenuItem: Terrasoft.emptyFn,
        /* Возвращает имя колонки по умолчанию для фильтра. */
        getFilterDefaultColumnName: function() {
            return "UsrDocument";
        }
    },
    /* Массив модификаций. */
    diff: /**SCHEMA_DIFF*/[
        {
            /* Тип операции – слияние. */
            "operation": "merge",
            /* Название элемента схемы, над которым производится действие. */
            "name": "DataGrid",
            /* Объект, свойства которого будут объединены со свойствами элемента схемы */
            "values": {
                "rowDataItemMarkerColumnName": "UsrDocument"
            }
        },
        {
            /* Тип операции – слияние. */
            "operation": "merge",
            /* Название элемента схемы, над которым производится действие. */
            "name": "AddRecordButton",
            /* Объект, свойства которого будут объединены со свойствами элемента схемы */
            "values": {
                "visible": {"bindTo": "getToolsVisible"}
            }
        }
    ]/**SCHEMA_DIFF*/
];
}
);

```

3. Выполните шаг 4 общего [алгоритма реализации детали с использованием Creatio IDE](#). Без выполнения этого шага деталь отображается на странице записи, но не содержит записей, поскольку не указаны колонки для отображения.

## Реализовать деталь с полями

**Деталь с полями** позволяет вводить и редактировать данные непосредственно в полях детали. Может содержать несколько групп полей. Примером детали с полями является деталь [Средства связи] ([Communication options]) страницы контакта.



**Действия**, которые позволяет выполнять деталь с полями:

- Добавить записи на деталь без сохранения страницы, которая содержит текущую деталь.
- Работать с деталью, как со страницей записи.
- Использовать базовую валидацию полей.
- Реализовать пользовательскую валидацию полей.
- Добавить виртуальную запись.
- Расширить логику поведения записей.

Реализовать деталь с полями невозможно исключительно в мастере деталей, поскольку по умолчанию через мастер деталей создается деталь с реестром. Для реализации необходимо использовать комбинацию мастера деталей и Creatio IDE.

Реализация детали с полями для продуктов линейки Financial Services Creatio имеет свои особенности. Функциональность базовой детали с полями продуктов линейки Financial Services Creatio реализована в схеме `BaseFieldsDetail` пакета `BaseFinance`. Модель представления записи детали с полями реализована в схеме `BaseFieldRowViewModel` пакета `BaseFinance`.

Использовать комбинацию мастера деталей и Creatio IDE для реализации детали с полями

- Для реализации детали с полями в CRM продуктах Creatio [скачайте пакет](#) `sdkFieldsDetailPackage`.
- Для реализации детали с полями в CRM продуктах Creatio импортируйте пакет в пользовательское приложение. Для этого воспользуйтесь инструкцией, которая приведена в статье [Перенести пакеты](#).
- Для реализации детали с полями в CRM продуктах Creatio добавьте пакет `sdkFieldsDetailPackage` в зависимости пользователяского пакета. Для этого воспользуйтесь инструкцией, которая приведена в статье [Создать пользовательский пакет](#).
- Выполните шаг 1 общего [алгоритма реализации детали с использованием мастера деталей](#). При необходимости, выполните настройку колонки детали, используя Creatio IDE.
- Используя Creatio IDE, замените родительский объект детали на `BaseFieldsDetail`.

6. Выполните шаг 2 общего [алгоритма реализации детали с использованием мастера деталей](#).

Использовать Creatio IDE для реализации детали с полями

1. Для реализации детали с полями в CRM продуктах Creatio выполните [шаги 1-3](#) алгоритма реализации детали с полями с помощью комбинации мастера деталей и Creatio IDE.

**2. Создайте пользовательскую деталь.**

a. Создайте схему объекта детали.

- В качестве родительского объекта выберите  `BaseEntity`.
- В схему объекта добавьте необходимые колонки.

d. Создайте схему модели представления детали с полями.

- В качестве родительского объекта выберите  `BaseFieldsDetail`.
- На панели инструментов в контекстном меню узла [ *Локализуемые строки* ] ([ *Localizable strings* ]) выберите локализуемую строку [ *Caption* ] и в свойстве [ *Значение* ] ([ *Value* ]) задайте название детали.
- В свойство  `methods` добавьте метод  `getDisplayColumns`, который возвращает массив с названиями колонок, отображающихся как поля в детали.

h. Добавьте пользовательские стили детали (опционально).

- a. Создайте схему модуля, в которой определите стили. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).

b. Укажите наследуемый класс.

- Для реализации детали с полями в CRM продуктах Creatio укажите  `UsrBaseFieldRowViewModel`.
- Для реализации детали с полями в продуктах линейки Financial Services Creatio укажите  `BaseFieldRowViewModel`.

e. В зависимости схемы модели представления реестра детали добавьте схему модуля с реализацией стилей.

f. В свойство  `methods` добавьте методы переопределения базовых CSS-классов стилей:

- `getRowViewModelClassName()` — возвращает имя класса модели представления записи на детали.
- `getLeftRowContainerWrapClass()` — возвращает массив строк с названиями CSS-классов, используемых для генерации представления контейнеров, содержащих подписи полей записей.

i. Зарегистрируйте деталь в базе данных. Для этого выполните SQL-запрос к таблице  `[SysDetails]` базы данных.

**SQL-запрос**

```
DECLARE
```

```
-- Название схемы детали.
@ClientUnitSchemaName NVARCHAR(100) = 'UsrИмяСхемыДетали',
-- Название схемы объекта детали.
@EntitySchemaName NVARCHAR(100) = 'UsrИмяСхемыОбъектадетали',
-- Название детали.
@DetailCaption NVARCHAR(100) = 'ИмяДетали'

INSERT INTO SysDetail(
    Caption,
    DetailSchemaUId,
    EntitySchemaUId
)
VALUES(
    @DetailCaption,
    (SELECT TOP 1 UId
    from SysSchema
    WHERE Name = @ClientUnitSchemaName),
    (SELECT TOP 1 UId
    from SysSchema
    WHERE Name = @EntitySchemaName)
)
```

Регистрация детали выполняется, чтобы деталь стала видимой для мастера разделов и мастера деталей.

### 3. Добавьте пользовательскую деталь на страницу записи.

Создайте схему замещающей модели представления страницы записи, на которой будет размещена деталь с полями.

- В качестве родительского объекта выберите схему модели представления, которую необходимо заместить.
- В свойство `details` добавьте деталь.
- В массив модификаций `diff` добавьте конфигурационный объект модели представления детали.

Пример схемы `ContactPageV2` замещающей модели представления страницы записи, на которой размещена деталь с полями `UsrRegDocumentFieldsDetail` приведен ниже.

#### Пример схемы замещающей модели представления

```
define("ContactPageV2", [], function() {
    return {
        entitySchemaName: "Contact",
        details: /**SCHEMA_DETAILS*{
            /* Добавление детали с полями. */
            "UsrRegDocumentFieldsDetail": {
                /* Название клиентской схемы детали. */
                "schemaName": "UsrRegDocumentFieldsDetail",
```

```

        /* Фильтрация записей детали текущего контакта (физ. лица). */
        "filter": {
            /* Колонка объекта детали. */
            "detailColumn": "UsrContact",
            /* Колонка идентификатора контакта. */
            "masterColumn": "Id"
        }
    }
}

} /**SCHEMA_DETAILS*/ ,
diff: /**SCHEMA_DIFF*/ [
    /* Добавление нового элемента. */
    "operation": "insert",
    /* Название элемента. */
    "name": "UsrRegDocumentFieldsDetail",
    /* Конфигурационный объект значений. */
    "values": {
        /* Тип элемента. */
        "itemType": Terrasoft.ViewItemType.DETAIL
    },
    /* Имя элемента-контейнера. */
    "parentName": "HistoryTab",
    /* Имя свойства элемента-контейнера, который содержит коллекцию вложенных элементов. */
    "propertyName": "items",
    /* Индекс добавляемого в коллекцию элемента. */
    "index": 0
}] /**SCHEMA_DIFF*/
};

});
);

```

## Реализовать деталь типа [ Файлы и ссылки ] ([ *Attachments* ])

**Деталь типа [ Файлы и ссылки ] ([ *Attachments* ])** позволяет хранить внешние файлы, ссылки на веб-ресурсы и статьи базы знаний. Доступна во всех разделах приложения. Функциональность базовой детали типа [ Файлы и ссылки ] ([ *Attachments* ]) реализована в схеме `FileDetailV2` пакета `uiV2`. Деталь типа [ Файлы и ссылки ] ([ *Attachments* ]) описана в статье [Файлы и примечания](#). Примером детали типа [ Файлы и ссылки ] ([ *Attachments* ]) является деталь [ Файлы и ссылки ] ([ *Attachments* ]) страницы контакта.

The screenshot shows the 'Attachments' detail view. At the top, there is a toolbar with icons for back, forward, search, and other navigation functions. Below the toolbar is a table with columns: Name, Description, Type, Created on, and Created by. Two files are listed: 'Contact\_Summary\_Alexander\_Wilson.docx' and 'Account\_Summary\_Alpha\_Business.docx'. Both files are categorized as 'File' and were created on '2/12/2020 8:56 AM' by 'John Best'. Below the table is a dashed rectangular area with the placeholder text 'Drag file here'.

Name	Description	Type	Created on	Created by
Contact_Summary_Alexander_Wilson.docx		File	2/12/2020 8:56 AM	John Best
Account_Summary_Alpha_Business.docx		File	2/12/2020 8:56 AM	John Best

Drag file here

Реализовать деталь типа [ *Файлы и ссылки* ] ([ *Attachments* ]) невозможно исключительно в мастере деталей, поскольку по умолчанию через мастер деталей создается деталь с реестром. Для реализации необходимо использовать комбинацию мастера деталей и Creatio IDE.

Чтобы реализовать деталь типа [ *Файлы и ссылки* ] ([ *Attachments* ]) с использованием **комбинации мастера деталей и Creatio IDE**:

## 1. Создайте пользовательскую деталь.

- Настройте объект детали типа [ *Файлы и ссылки* ] ([ *Attachments* ]).
  - Для этого в свойстве [ *По какому объекту создать деталь?* ] ([ *How to create detail?* ]) выберите "Существующему объекту" ("Based on existing object").
  - Для этого в свойстве [ *Объект* ] ([ *Object* ]) выберите "Файл и ссылка объекта [ИмяПользовательскогоРаздела]" ("[CustomSectionName] attachment").
- Используя Creatio IDE, замените родительский объект страницы записи детали на `FileDetailV2`.
- Добавьте пользовательские стили детали (опционально).
  - Создайте схему модуля, в которой определите стили. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
  - В зависимости схемы модели представления детали добавьте схему модуля с реализацией стилей.

## 2. Выполните шаг 2 общего [алгоритма реализации детали с использованием мастера деталей](#).

# Реализовать множественное добавление записей на деталь

По умолчанию деталь позволяет добавлять только одну запись. **Назначение** миксина `LookupMultiAddMixin` — расширение действия по добавлению записи на деталь. Использование миксина позволяет пользователю выбирать несколько записей из справочника за один раз.

Чтобы **реализовать множественное добавление записей на деталь**:

- Создайте схему замещающей модели представления детали. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
- В свойство `mixins` добавьте миксин `LookupMultiAddMixin`.
- В свойстве `methods`:
  - Переопределите методы.
    - `init()` — реализует логику, выполняемую при загрузке модуля. В методе выполните инициализацию миксина `LookupMultiAddMixin`. Метод `init()` описан в статье [Виды модулей](#).
    - `getAddRecordButtonVisible()` — отвечает за отображение кнопки добавления.
    - `onCardSaved()` — отвечает за сохранение страницы детали. В переопределенном методе используйте метод `openLookupWithMultiSelect()`, который вызывает справочное окно для множественного выбора.

- `addRecord()` — отвечает за добавление записи на деталь. Как и для метода `onCardSaved()`, в переопределенном методе используйте метод `openLookupWithMultiSelect()`. Значение `true` указывает на необходимость выполнения проверки является ли запись новой.
- Реализуйте метод `getMultiSelectLookupConfig()`, который связан с методом `openLookupWithMultiSelect()`. Метод `getMultiSelectLookupConfig()` выполняет конфигурирование справочного окна и возвращает объект конфигурации для справочного окна.

## Удалить деталь

**Важно.** Удаление детали невозможно без доступа к конфигурации системы и базе данных.

Чтобы **удалить деталь**:

1. В SVN-хранилище снимите блокировку с файлов детали, которую необходимо удалить.
2. Удалите записи из базы данных. Для этого выполните SQL-запрос в базу данных.

### SQL-запрос

```
DECLARE @Caption nvarchar(max);
SET @Caption = 'UsrИмяСхемыДетали';
DECLARE @UIid UNIQUEIDENTIFIER;
select @UIid = EntitySchemaUID from SysDetail
where Caption = @Caption
delete from SysDetail where EntitySchemaUID = @UIid
```

3. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и удалите схему модели представления детали и схему объекта детали.

## Настроить деталь с полями

 Сложный

**Важно.** При разработке детали с полями для продуктов линейки Financial Services Creatio используется схема `BaseFieldsDetail` пакета `BaseFinance`. Этот пакет присутствует только в продуктах линейки Financial Services Creatio.

Чтобы **использовать деталь с полями в CRM продуктах Creatio**:

1. [Скачайте пакет](#) `sdkFieldsDetailPackage`.
2. Импортируйте пакет в пользовательское приложение. Для этого воспользуйтесь инструкцией, которая приведена в статье [Перенести пакеты](#).

3. Добавьте пакет `sdkFieldsDetailPackage` в зависимости пользовательского пакета.

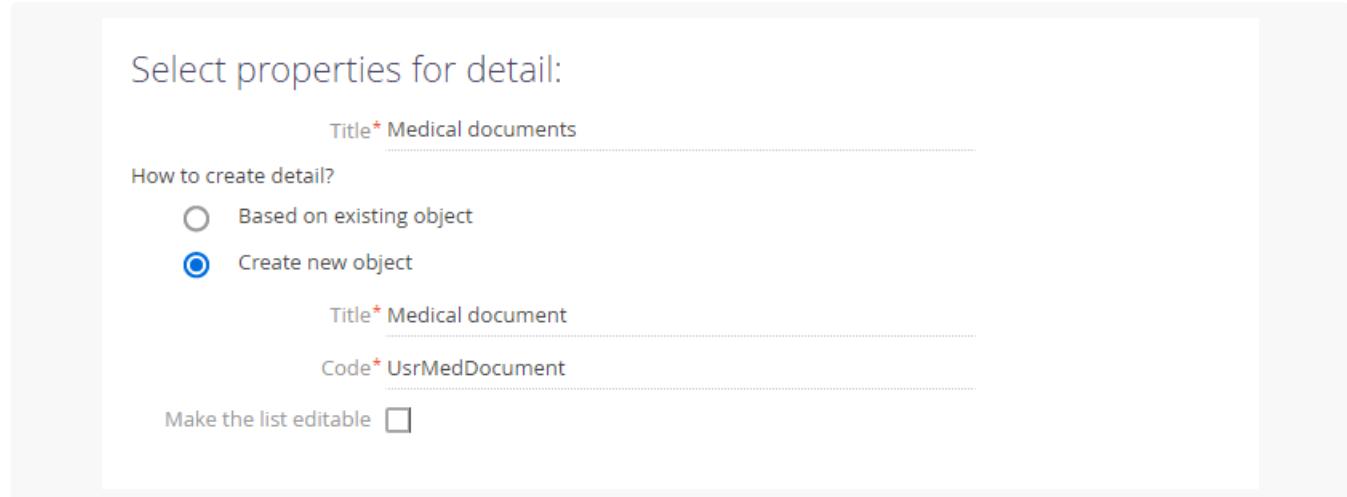
**Пример.** Реализовать пользовательскую деталь [ *Медицинские документы* ] ([ *Medical documents* ]), которая содержит виртуальные поля [ *Номер* ] ([ *Number* ]) и [ *Серия* ] ([ *Series* ]). Добавить деталь на вкладку [ *История* ] ([ *History* ]) страницы физического лица. Значение, введенное в поле [ *Номер* ] ([ *Number* ]), не должно быть отрицательным. Названия полей детали отображать синим цветом.

## 1. Создать пользовательскую деталь

1. Создайте пользовательский пакет и установите его в качестве текущего. Подробнее читайте в статье [Общие принципы работы с пакетами](#).
2. Перейдите в дизайнер системы по кнопке
3. В блоке [ *Настройка системы* ] ([ *System setup* ]) перейдите по ссылке [ *Мастер деталей* ] ([ *Detail wizard* ]).
4. Заполните свойства детали:
  - [ *Заголовок* ] ([ *Title* ]) — "Медицинские документы" ("Medical documents").
  - [ *По какому объекту создать деталь?* ] ([ *How to create detail?* ]) — выберите "Новому объекту" ("Create new object").

Заполните свойства объекта:

- [ *Заголовок* ] ([ *Title* ]) — "Медицинский документ" ("Medical document").
- [ *Код* ] ([ *Code* ]) — "UsrMedDocument".



После сохранения в конфигурации будут созданы:

- Схема `UsrMedDocument` объекта детали.
- Схема `UsrSchemac6fd3fd0Detail` модели представления реестра детали [ *Медицинские документы* ] ([ *Medical documents* ]).
- Схема `UsrUsrMedDocument4988cee4Page` модели представления страницы записи детали [ *Медицинские*

документы ] ([ *Medical documents* ]).

5. Перейдите на вкладку [ Страница ] ([ *Page* ]) для настройки страницы записи детали.

6. Настройте поля детали.

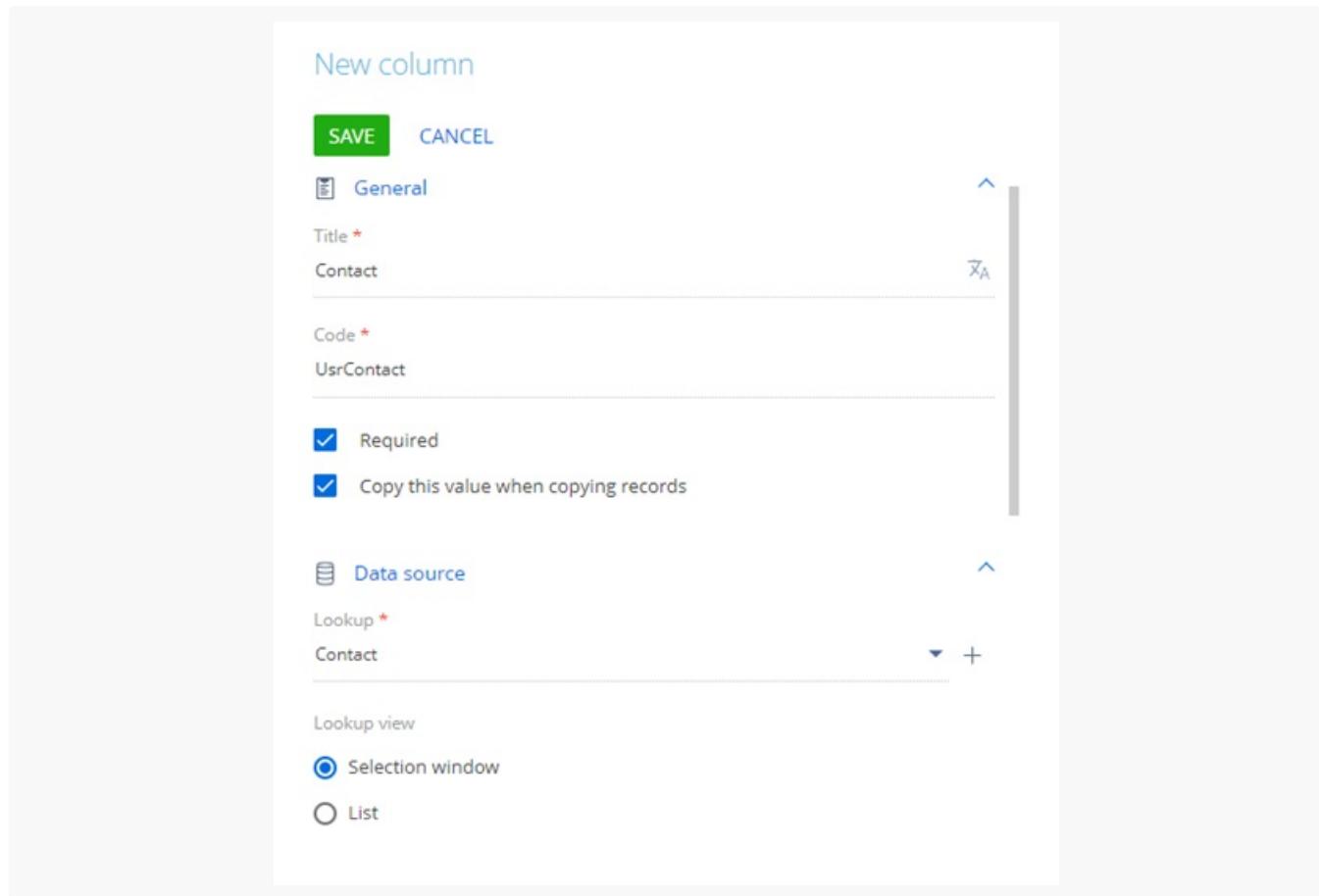
a. Заполните свойства поля [ *Contact* ] типа [ Справочник ] ([ *Lookup* ]).

a. [ Заголовок ] ([ *Title* ]) — "Физ. лицо" ("Contact").

b. [ Код ] ([ *Code* ]) — "UsrContact".

c. Установите признак [ Обязательное ] ([ *Required* ]).

d. [ Справочник ] ([ *Lookup* ]) — выберите "Контакт" ("Contact").



b. Заполните свойства поля [ *Series* ] типа [ Стока ] ([ *String* ]).

a. [ Заголовок ] ([ *Title* ]) — "Серия" ("Series").

b. [ Код ] ([ *Code* ]) — "UsrSeries".

c. [ Длина строки ] ([ *Text length* ]) — выберите "Строка (50 символов)" ("Text (50 characters)").

d. Установите признак [ Обязательное ] ([ *Required* ]).

New column

**SAVE**   **CANCEL**

**General**

Title **\***  
Series

Code **\***  
UsrSeries

Text length  
Text (50 characters)

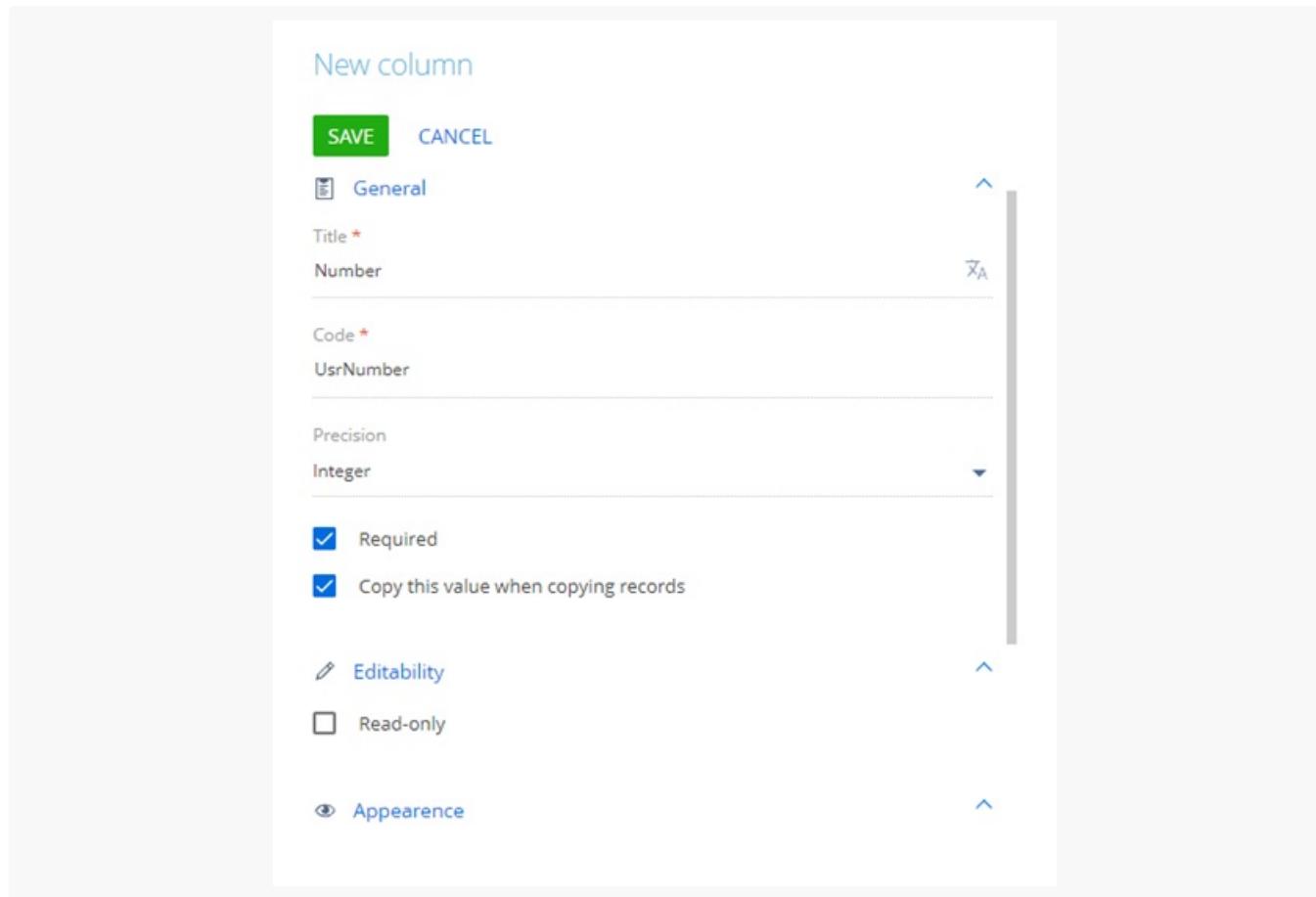
Required  
 Copy this value when copying records

**Editability**

Read-only

**Appearence**

- c. Заполните свойства поля [ *Number* ] типа [ *Целое число* ] ([ *Integer* ]).
  - a. [ *Заголовок* ] ([ *Title* ]) — "Номер" ("Number").
  - b. [ *Код* ] ([ *Code* ]) — "UsrNumber".
  - c. Установите признак [ *Обязательное* ] ([ *Required* ]).

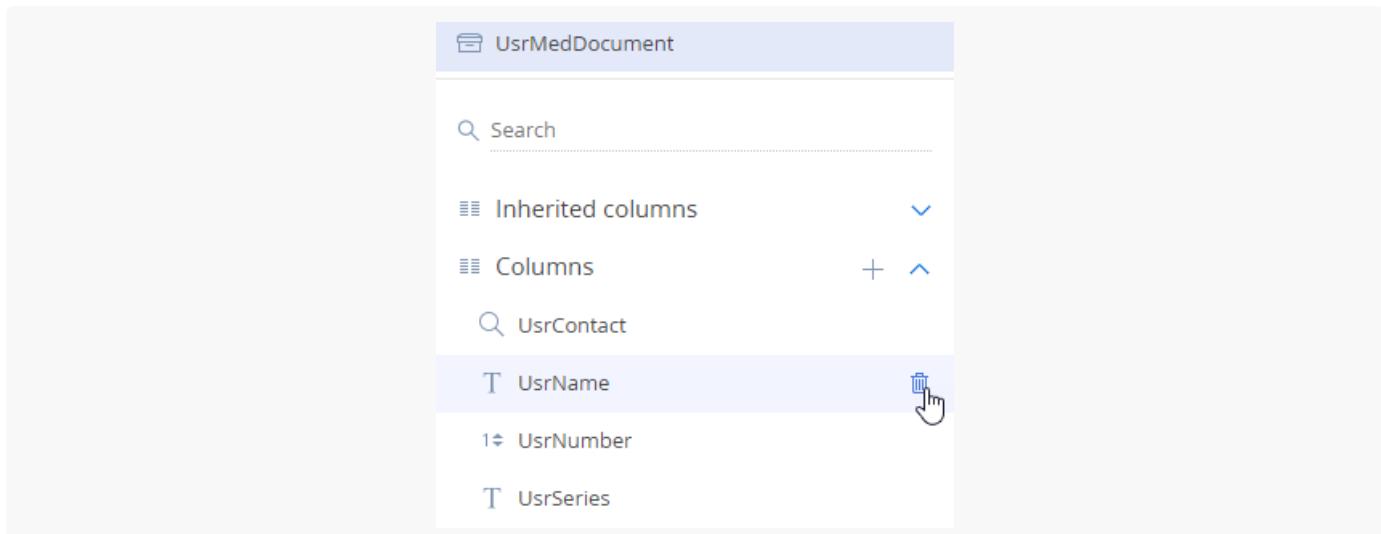


7. При необходимости, измените расположение полей детали.
8. На панели инструментов мастера деталей нажмите [ Сохранить ] ([ Save ]).

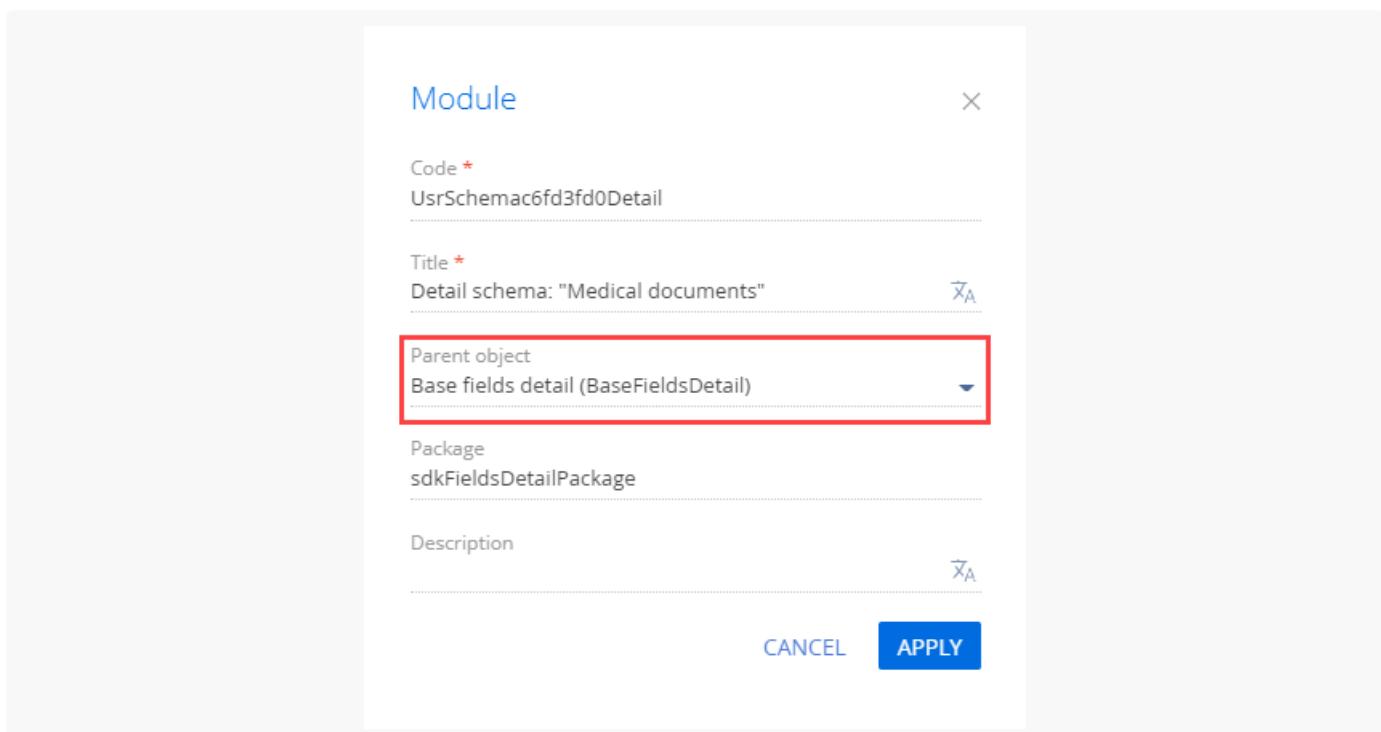
После сохранения в конфигурации будет модифицирована схема `UsrUsrMedDocument4988cee4Page` модели представления страницы записи детали [ Медицинские документы ] ([ Medical documents ]).

## 2. Настроить пользовательскую деталь

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), который был установлен в качестве текущего.
2. Откройте схему `UsrMedDocument` объекта детали.
3. В контекстном меню узла [ Колонки ] ([ Columns ]) структуры объекта удалите обязательную колонку `[UsrName]`.



4. На панели инструментов дизайнера объектов нажмите [ Опубликовать ] ([ Publish ]).
5. Откройте схему `UsrSchemac6fd3fd0Detail` модели представления реестра детали [ Медицинские документы ] ([ Medical documents ]).
6. На панели свойств нажмите кнопку и измените значение поля [ Родительский объект ] ([ Parent object ]) на `BaseFieldsDetail`. Схема `BaseFieldsDetail` реализует деталь с полями. По умолчанию в мастере деталей в качестве родительского объекта устанавливается базовая схема детали с реестром.



7. В свойство `methods` схемы модели представления детали добавьте метод `getDisplayColumns`, который возвращает массив с названиями колонок, отображающихся как поля в детали.

Исходный код схемы `UsrSchemac6fd3fd0Detail` представлен ниже.

```
UsrSchemac6fd3fd0Detail
```

```

define("UsrSchemac6fd3fd0Detail", [], function() {
    return {
        entitySchemaName: "UsrMedDocument",
        details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
        diff: /**SCHEMA_DIFF*/[]/**SCHEMA_DIFF*/,
        methods: {
            getDisplayColumns: function() {
                return ["UsrSeries", "UsrNumber"];
            }
        }
    };
});

```

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

### 3. Добавить деталь на страницу записи раздела

1. Перейдите в раздел [ Физ. лица ] ([ Contacts ]) и откройте страницу физического лица.
2. На панели инструментов кликните [ Вид ] —> [ Открыть мастер раздела ] ([ View ] —> [ Open section wizard ]).
3. В рабочей области мастера разделов перейдите на вкладку [ История ] ([ History ]) и нажмите кнопку [ Добавить деталь ] ([ New detail ]).
4. Заполните настройки детали.
  - [ Деталь ] ([ Detail ]) — выберите "Медицинские документы" ("Medical documents"). Поля [ Заголовок ] ([ Title ]) и [ Код (на английском) ] ([ Code ]) заполняются автоматически.
  - [ У которых колонка детали ] ([ Where detail column ]) — выберите "Физ. лицо" ("Contact").

Значения остальных колонок оставьте без изменений.

**Detail settings**

**SAVE**   **CANCEL**

Detail *	Medical documents	▼
Title *	Medical documents	A
Code *	UsrSchemaC6fd3fd0Detail7b52c652	
<b>What records to show on the page?</b>		
Where detail column *	Contact	▼
Equals to page column *	Id	▼

5. Нажмите [ Сохранить ] —> [ Мастер раздела ] —> [ Сохранить ] ([ Save ] —> [ Section wizard ] —> [ Save ]).

В результате деталь [ Медицинские документы ] ([ Medical documents ]) будет добавлена на вкладку [ История ] ([ History ]) страницы физического лица.



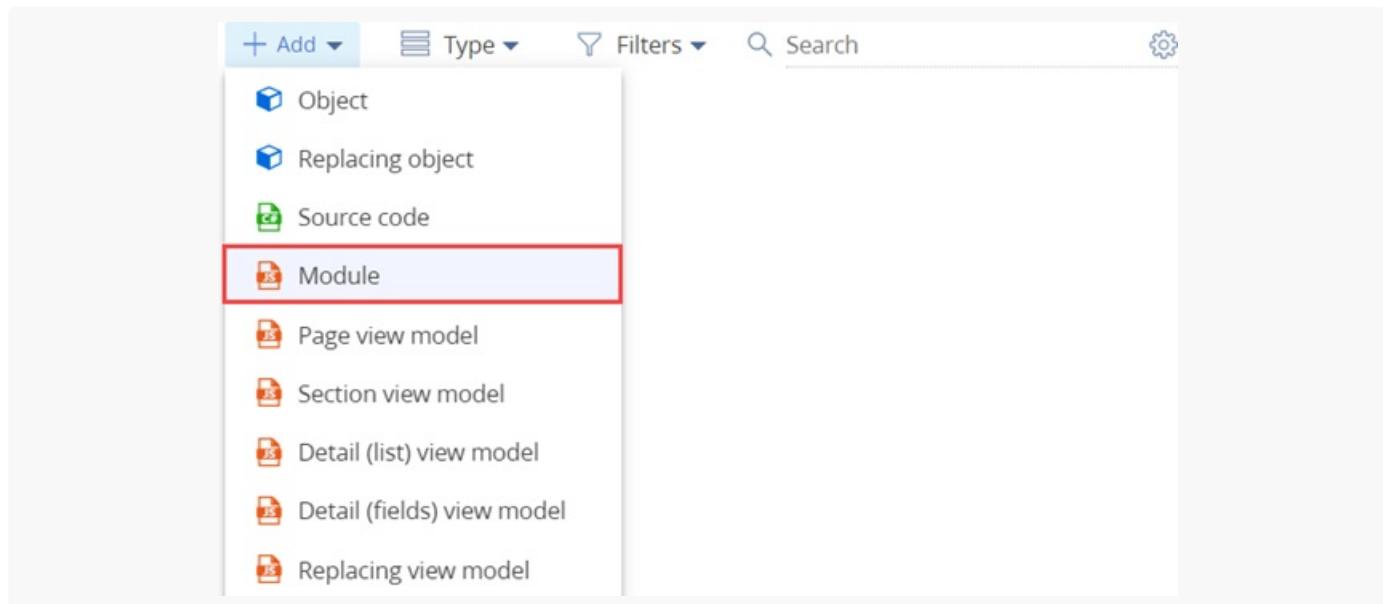
## 4. Добавить пользовательские стили детали

Поскольку в схеме модели представления страницы детали невозможно задать стили для отображения, необходимо:

1. Создать схему модуля, в которой определить стили.
2. Добавить модуль со стилями в зависимости модуля детали.

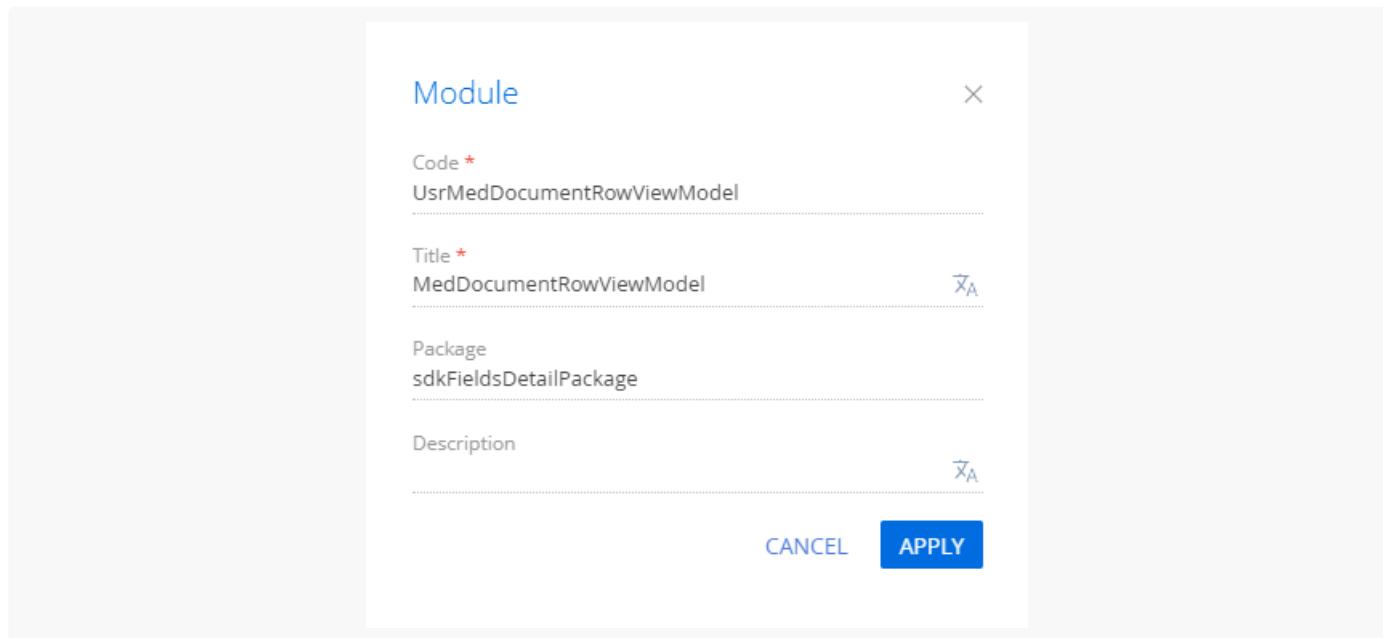
### 1. Создать схему модуля

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



3. Заполните свойства схемы:

- [Код] ([Code]) — "UsrMedDocumentRowViewModel".
- [Заголовок] ([Title]) — "MedDocumentRowViewModel".



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

4. В дизайнере схем добавьте исходный код. В исходном коде схемы создайте описание модуля и определите в нем класс `Terrasoft.configuration.UsrMedDocumentRowViewModel`, унаследованный от класса `Terrasoft.BaseFieldRowViewModel`.

```
UsrMedDocumentRowViewModel
```

```
define("UsrMedDocumentRowViewModel", ["BaseFieldRowViewModel"], function() {
    Ext.define("Terrasoft.configuration.UsrMedDocumentRowViewModel", {
        extend: "Terrasoft.BaseFieldRowViewModel",
        alternateClassName: "Terrasoft.UsrMedDocumentRowViewModel"
    });
    return Terrasoft.UsrMedDocumentRowViewModel;
});
```

- Перейдите в узел [ *LESS* ] структуры объекта и задайте необходимые стили отображения детали.

### Настройка стилей отображения детали

```
.med-document-left-row-container {
    .t-label {
        color: blue;
    }
}
.field-detail-row {
    width: 100;
    display: inline-flex;
    margin-bottom: 10px;

    .field-detail-row-left {
        display: flex;
        flex-wrap: wrap;

        .control-width-15 {
            min-width: 300px;
            width: 50;
            margin-bottom: 5px;
        }
    }
    .field-detail-row-left.singlecolumn {
        width: 50%;
    }
}
```

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

## 2. Модифицировать схему модели представления детали

Чтобы **использовать созданный модуль и его стили** в схеме детали:

- Откройте схему `UsrSchemac6fd3fd0Detail` модели представления реестра детали [ *Медицинские документы* ] ([ *Medical documents* ]).
- В зависимости схемы `UsrSchemac6fd3fd0Detail` добавьте модуль `UsrMedDocumentRowViewModel`.
- В определение модуля схемы детали добавьте методы переопределения базовых CSS-классов стилей:

- `getRowViewModelClassName()` — метод, который возвращает имя класса модели представления записи на детали.
- `getLeftRowContainerWrapClass()` — метод, который возвращает массив строк с названиями CSS-классов, используемых для генерации представления контейнеров, содержащих подписи полей записей.

Исходный код модифицированной схемы представлен ниже.

```
UserSchemaC6fd3fd0Detail

define("UserSchemaC6fd3fd0Detail", ["UserMedDocumentRowViewModel", "css!UserMedDocumentRowViewMo
return {
    entitySchemaName: "UserMedDocument",
    details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
    diff: /**SCHEMA_DIFF*/[], /**SCHEMA_DIFF*/
    methods: {
        getDisplayColumns: function() {
            return ["UserSeries", "UserNumber"];
        },
        getRowViewModelClassName: function() {
            return "Terrasoft.UserMedDocumentRowViewModel";
        },
        getLeftRowContainerWrapClass: function() {
            return ["med-document-left-row-container", "field-detail-row"];
        }
    }
};
});
```

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

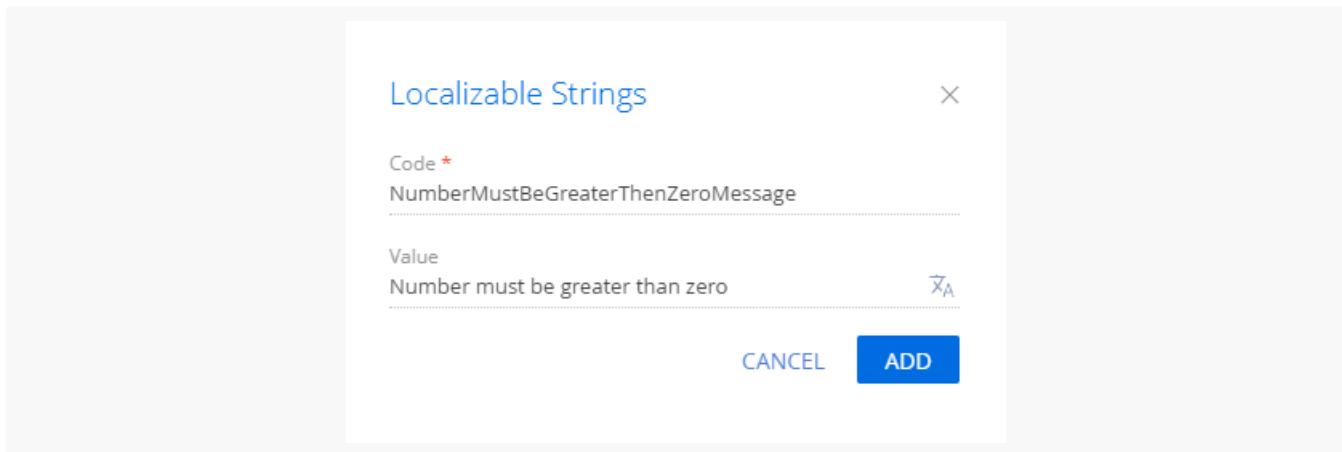
В результате названия полей детали [ Медицинские документы ] ([ Medical documents ]), которая была добавлена на вкладку [ История ] ([ History ]) страницы физического лица, отображаются синим цветом.



## 5. Добавить валидацию к полю детали

1. Откройте схему `UserMedDocumentRowViewModel` модуля.
2. Добавьте локализуемую строку с сообщением о неверном значении поля [ Номер ] ([ Number ]).
  - a. В контекстном меню узла [ Локализуемые строки ] ([ Localizable strings ]) нажмите кнопку
  - b. Заполните свойства локализуемой строки:

- [ Код ] ([ Code ]) — "NumberMustBeGreaterThenZeroMessage".
- [ Значение ] ([ Value ]) — "Number must be greater than zero" ("Введите номер больше нуля").



- Для добавления локализуемой строки нажмите [ Добавить ] ([ Add ]).
  - В зависимости модуля `UsrMedDocumentRowViewModel` добавьте модуль ресурсов `UsrMedDocumentRowViewModelResources`. Это необходимо, чтобы значение локализуемой строки отобразилось во front-end части приложения.
3. Добавьте логику работы валидации значения поля [ Номер ] ([ Number ]). Для этого реализуйте методы:
- `validateNumberMoreThanZero()` — метод, который содержит логику валидации значения поля.
  - `setValidationConfig()` — метод, который связывает колонку [Number] и метод-валидатор `validateNumberMoreThanZero()`.
  - `init()` — переопределенный базовый метод, в котором выполняется вызов базовой логики и метода `setValidationConfig()`.

Исходный код модифицированной схемы представлен ниже.

```
UserMedDocumentFieldsDetail

define("UsrMedDocumentRowViewModel", ["BaseFieldRowViewModel", "UsrMedDocumentRowViewModelResources"], function(resources) {
    Ext.define("Terrasoft.configuration.UsrMedDocumentRowViewModel", {
        extend: "Terrasoft.BaseFieldRowViewModel",
        alternateClassName: "Terrasoft.UsrMedDocumentRowViewModel",
        validateNumberMoreThanZero: function(columnValue) {
            var invalidMessage = "";
            if (columnValue < 0) {
                invalidMessage = resources.localizableStrings.NumberMustBeGreaterThenZeroMessage;
            }
            return {
                fullInvalidMessage: invalidMessage,
                invalidMessage: invalidMessage
            };
        }
    });
});
```

```

        };
    },
    setValidationConfig: function() {
        this.addColumnValidator("UsrNumber", this.validateNumberMoreThenZero);
    },
    init: function() {
        this.callParent(arguments);
        this.setValidationConfig();
    }
});
return Terrasoft.UsrMedDocumentRowViewModel;
});

```

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

В результате при вводе отрицательного значения в поле [ Номер ] ([ Number ]) отображается соответствующее предупреждение.



## 6. Сделать виртуальными поля детали

- Откройте схему `UsrSchemac6fd3fd0Detail` модели представления реестра детали [ Медицинские документы ] ([ Medical documents ]).
- Добавьте реализацию метода `useVirtualRecord()`.

Исходный код модифицированной схемы представлен ниже.

```

UsrSchemac6fd3fd0Detail

define("UsrSchemac6fd3fd0Detail", ["UsrMedDocumentRowViewModel", "css!UsrMedDocumentRowViewMo
return {
    entitySchemaName: "UsrMedDocument",
    details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
    diff: /**SCHEMA_DIFF*/ [], /**SCHEMA_DIFF*/
    methods: {
        getDisplayColumns: function() {
            return ["UsrSeries", "UsrNumber"];
        },
        getRowViewModelClassName: function() {
            return "Terrasoft.UsrMedDocumentRowViewModel";
        }
    }
};

```

```

},
getLeftRowContainerWrapClass: function() {
    return ["med-document-left-row-container", "field-detail-row"];
},
useVirtualRecord: function() {
    return true;
}
}
);
});

```

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

В результате при открытии вкладки [ История ] ([ History ]), которая содержит деталь [ Медицинские документы ] ([ Medical documents ]), отображается виртуальная запись.



## Добавить редактируемый реестр в деталь



**Пример.** На странице добавления продукта (деталь [ Продукты ] ([ Products ])) в разделе [ Заказы ] ([ Orders ]) в колонку [ Скидка, % ] ([ Discount, % ]), которая уже присутствует в схеме объекта продукта, добавить редактируемый реестр. Также добавить редактируемый реестр в пользовательскую колонку [ Пользовательская цена ] ([ Custom price ]).

### 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 top navigation bar with a 'Type' dropdown menu. The 'Replacing object' option is highlighted with a red box. Other options in the menu include 'Object', 'Source code', and 'Module'. To the right of the menu is a table with columns 'Status' and 'Type', showing entries for 'Object' and 'Client module'.

### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "OrderProduct".
- [ Заголовок ] ([ Title ]) — "Продукт в заказе" ("Product in order").
- [ Родительский объект ] ([ Parent object ]) — выберите "OrderProduct".

The screenshot shows the 'General' and 'Inheritance' sections of a schema configuration form.

**General:**

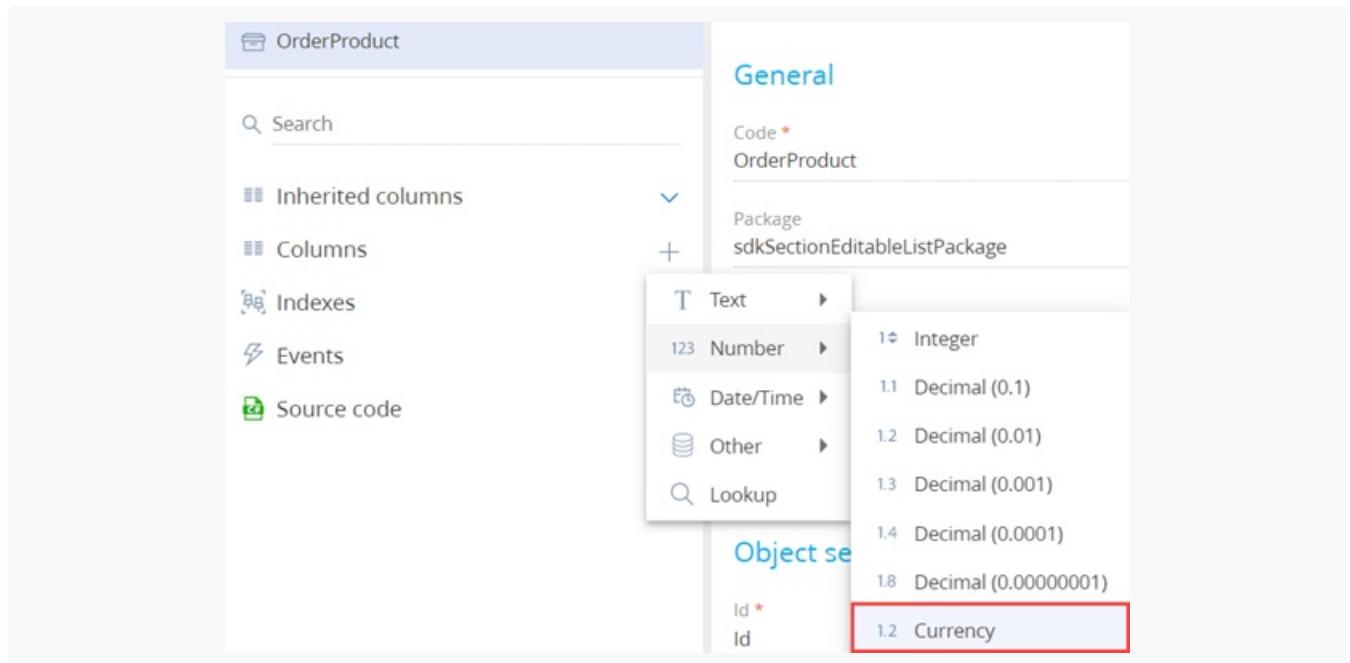
- Code \*: OrderProduct
- Title \*: Product in order
- Package: sdkSectionEditableListPackage
- Description

**Inheritance:**

- Parent object \*: OrderProduct
- Replace parent

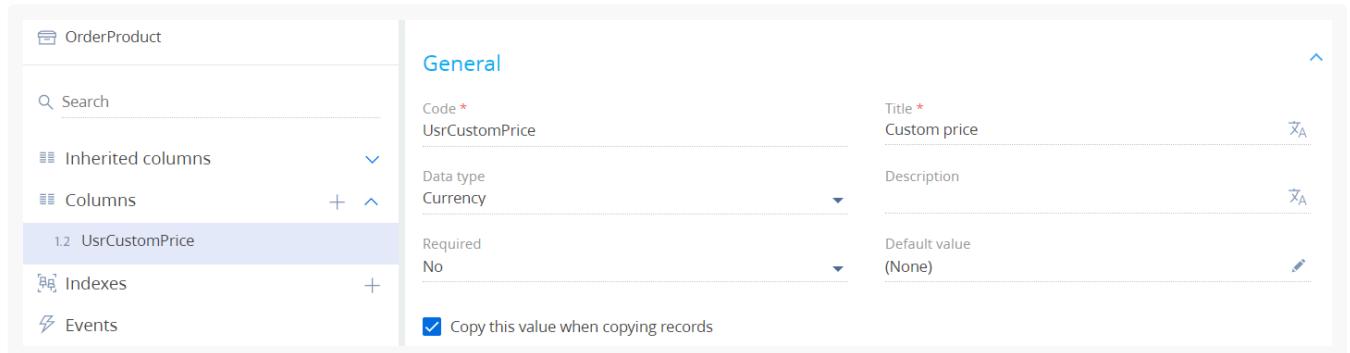
### 4. В схему добавьте **колонку**.

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



c. Заполните **свойства добавляемой колонки**.

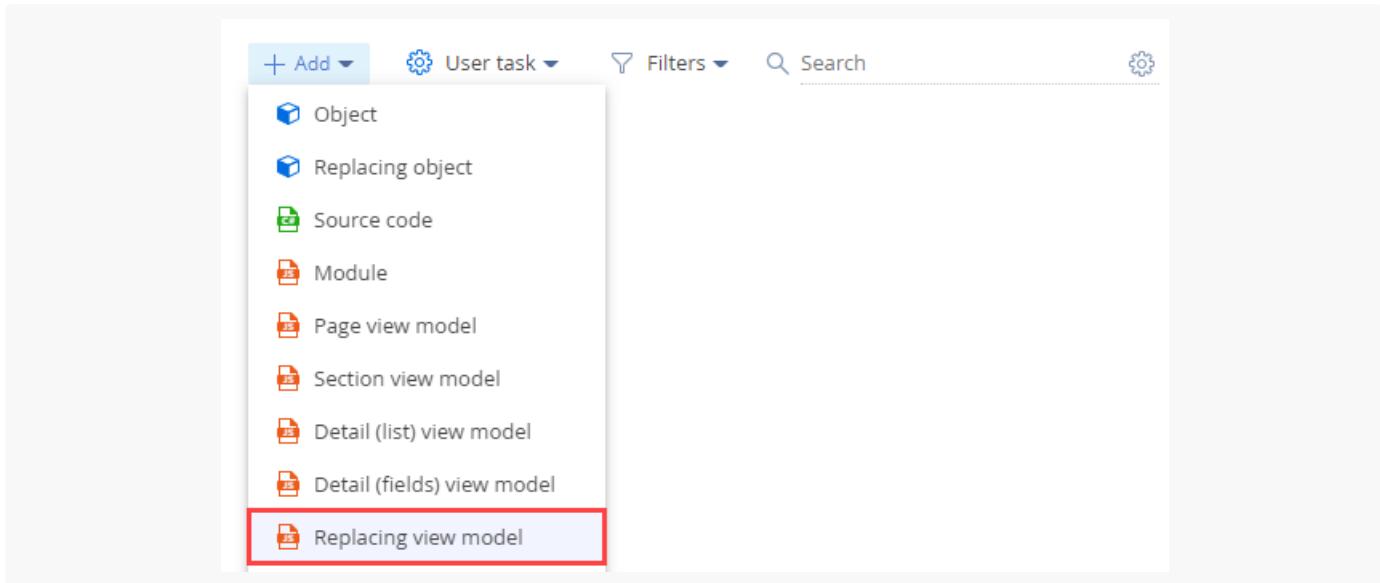
- [ Код ] ([ Code ]) — "UsrCustomPrice".
- [ Заголовок ] ([ Title ]) — "Пользовательская цена" ("Custom price").



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

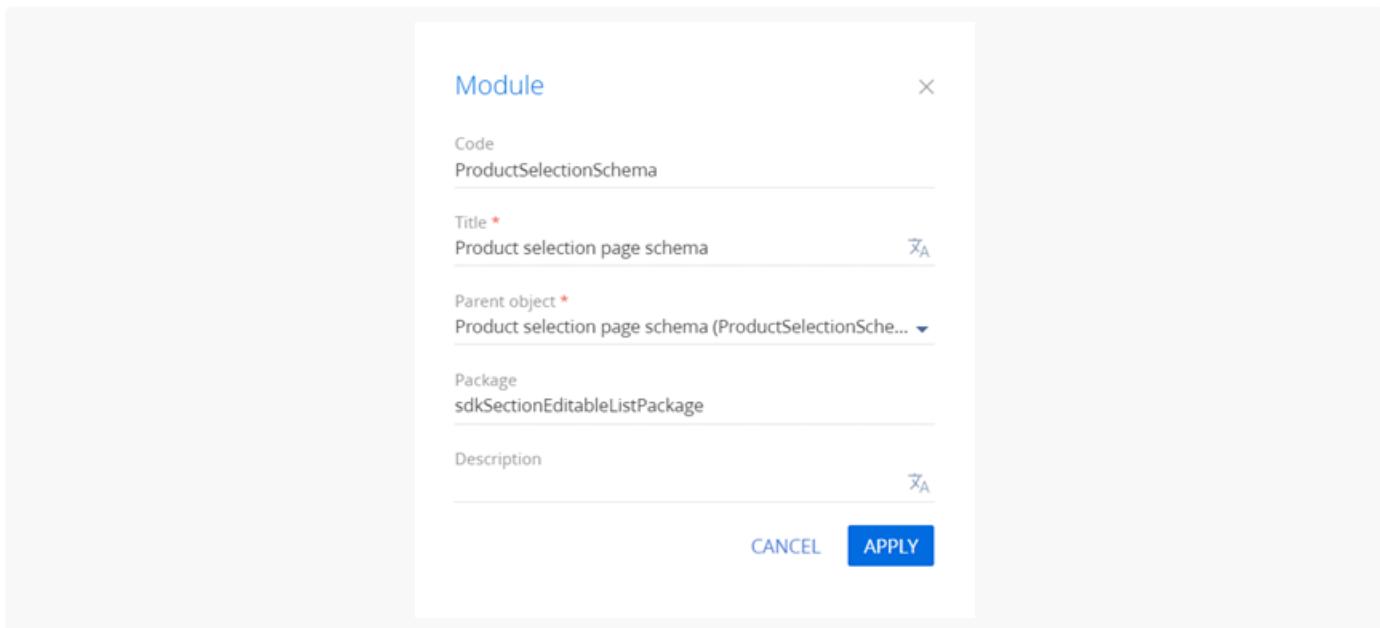
## 2. Создать схему замещающей модели представления раздела

- 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* ]) — "ProductSelectionSchema".
- [ Заголовок ] ([ *Title* ]) — "Схема страницы подбора продуктов" ("Product selection page schema").
- [ Родительский объект ] ([ *Parent object* ]) — выберите "ProductSelectionSchema".



### 4. Реализуйте **редактируемый реестр**. Для этого в свойстве `methods` реализуйте **методы**:

- `getEditableColumns()` — получает массив редактируемых колонок и добавляет пользовательскую колонку в массив.
- `setColumnHandlers()` — привязывает обработчик события изменения пользовательской колонки.
- `onCustomPriceChanged()` — метод-обработчик, который вызывается при изменении значения поля.

Исходный код схемы замещающей модели представления раздела представлен ниже.

**ProductSelectionSchema**

```

define("ProductSelectionSchema", [], function() {
    return {
        methods: {
            getEditableColumns: function() {
                /* Получает массив редактируемых колонок. */
                var columns = this.callParent(arguments);
                /* Добавляет колонку [Скидка, %] в массив редактируемых колонок. */
                columns.push("DiscountPercent");
                /* Добавляет пользовательскую колонку. */
                columns.push("UsrCustomPrice");
                return columns;
            },
            setColumnHandlers: function(item) {
                this.callParent(arguments);
                /* Привязка обработчика события изменения пользовательской колонки. */
                item.on("change:UsrCustomPrice", this.onCustomPriceChanged, this);
            },
            /* Метод-обработчик, который вызывается при изменении значения поля. */
            onCustomPriceChanged: function(item, value) {
                window.console.log("Changed: ", item, value);
            }
        }
    };
});

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Обновите страницу раздела [ Заказы ] ([ Orders ]).
2. Настройте **колонки страницы добавления продукта**.
  - a. На панели инструментов раздела нажмите [ Вид ] —> [ Настроить колонки ] ([ View ] —> [ Select fields to display ]) и на странице настройки колонок перейдите в режим настройки плиточного представления реестра раздела ([ Плиточное представление ] ([ Tile view ])).
  - b. Добавьте колонку в реестр раздела. Для этого нажмите на кнопку  . Затем нажмите на кнопку  и в поле [ Выберите объект ] ([ Select object ]) выберите объект [ Продукт в заказе ] ([ Product in order ]).
  - c. В поле [ Колонка ] ([ Column ]) выберите колонку [ Скидка, % ] ([ Discount, % ]).
  - d. Аналогично добавьте колонку [ Пользовательская цена ] ([ Custom price ]).

В результате выполнения примера на странице добавления продукта (деталь [ Продукты ] ([ Products ]))

в разделе [ Заказы ] ([ Orders ]) в реестр добавлены редактируемые колонки [ Скидка, % ] ([ Discount, % ]) и [ Пользовательская цена ] ([ Custom price ]).

Product	Price	Quantity	Discount, %	Total	Custom price
Website development	40.00	150.000	0.08	5,995.20	0.00
Preparing software docum...	40.00	80.000	5.00	3,040.00	
Installing software	12.00	60.000	0.69	715.03	0.00

## Скрыть пункты меню детали с реестром

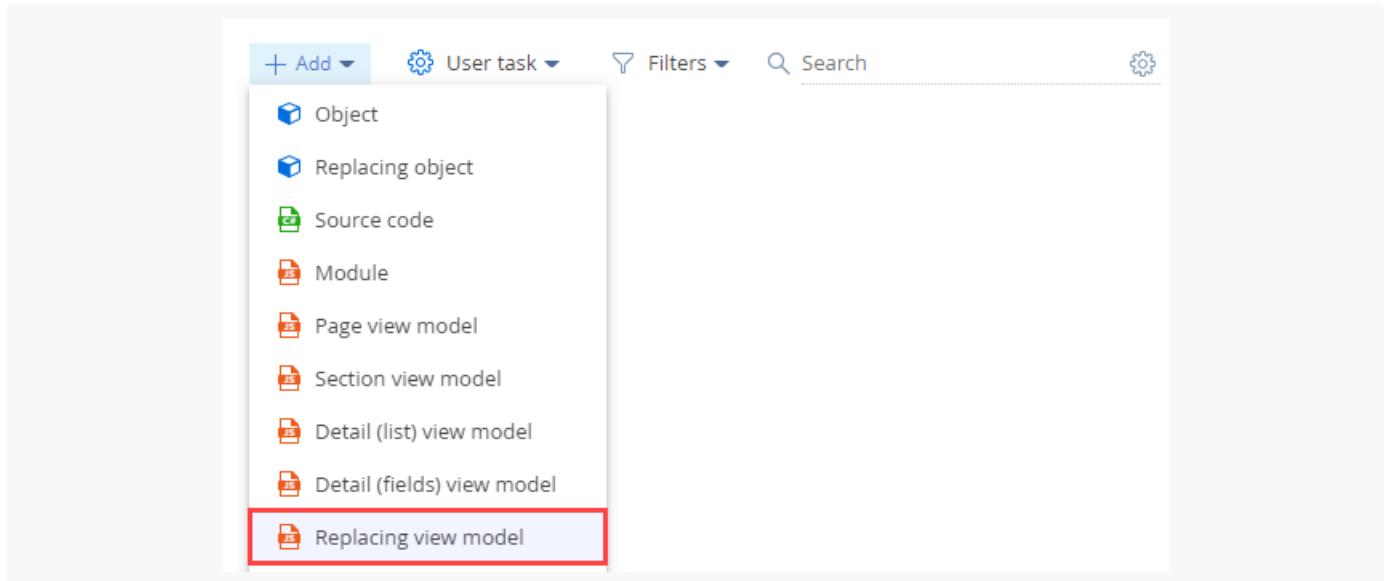
Средний

**Пример.** Для детали [ Адреса ] ([ Addresses ]), которая находится на вкладке [ Основная информация ] ([ Contact info ]) страницы контакта, скрыть пункты [ Копировать ] ([ Copy ]), [ Изменить ] ([ Edit ]), [ Удалить ] ([ Delete ]) меню.

**Назначение** пунктов [ Копировать ] ([ Copy ]), [ Изменить ] ([ Edit ]), [ Удалить ] ([ Delete ]) меню — управление записями реестра детали.

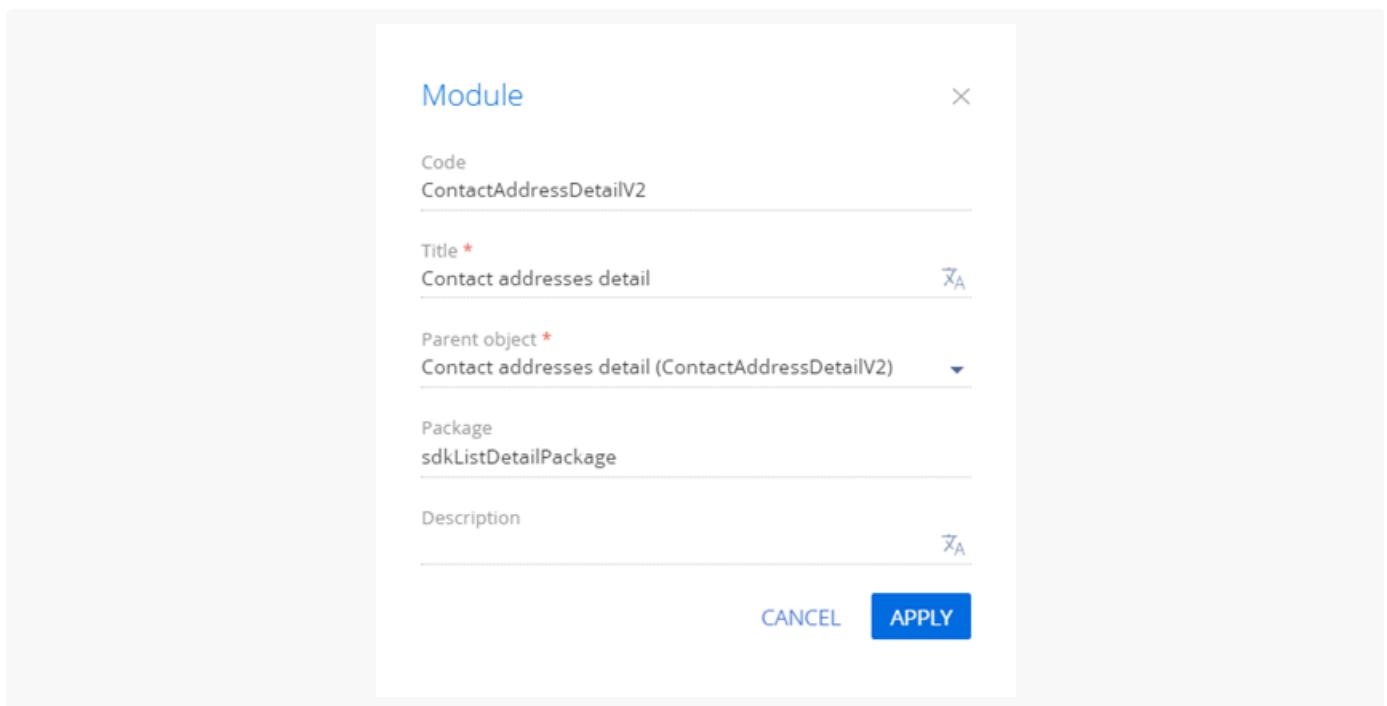
## Создать схему замещающей модели представления реестра детали

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 ]) — "ContactAddressDetailV2".
- [ Заголовок ] ([ Title ]) — "Деталь адресов контакта" ("Contact addresses detail").
- [ Родительский объект ] ([ Parent object ]) — выберите "ContactAddressDetailV2".



4. В дизайнере модуля добавьте исходный код.

```
ContactAddressDetailV2

define("ContactAddressDetailV2", [], function() {
    return {
        entitySchemaName: "AccountAddress",
```

```

methods: {
    /* Удаление пункта [Копировать] ([Copy]) меню. */
    getCopyRecordMenuItem: Terrasoft.emptyFn,
    /* Удаление пункта [Редактировать] ([Edit]) меню. */
    getEditRecordMenuItem: Terrasoft.emptyFn,
    /* Удаление пункта [Удалить] ([Delete]) меню. */
    getDeleteRecordMenuItem: Terrasoft.emptyFn
},
diff: /**SCHEMA_DIFF*/[]/**SCHEMA_DIFF*/
};

});

```

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

## Результат выполнения примера

В результате выполнения примера пункты [ Копировать ] ([ Copy ]), [ Изменить ] ([ Edit ]), [ Удалить ] ([ Delete ]) скрыты из меню детали [ Адреса ] ([ Addresses ]).

The screenshot shows the 'CONTACT INFO' tab selected in the top navigation bar. Below it, there's a list of contact details: Type (Show on map), Title (Select multiple records), Recipient's name (Export to Excel), Age (Data import), and Communication (Apply filter, Sort by, Columns setup, Change log setup, Detail setup). The 'Communication' section is expanded, showing options for Email, Mobile phone, and various 'Do not use' checkboxes. To the right, there are fields for Owner (Marina Kysla), Gender (Male), Preferred language (English (United States)), Business phone (+1 212 542 4238), Extension phone (104), and several 'Do not use' checkboxes for phone and fax. At the bottom, there's a table for Addresses with columns for Address type, Address, City, Country, and ZIP/postal code, showing one entry for Home address in Denver, United States.

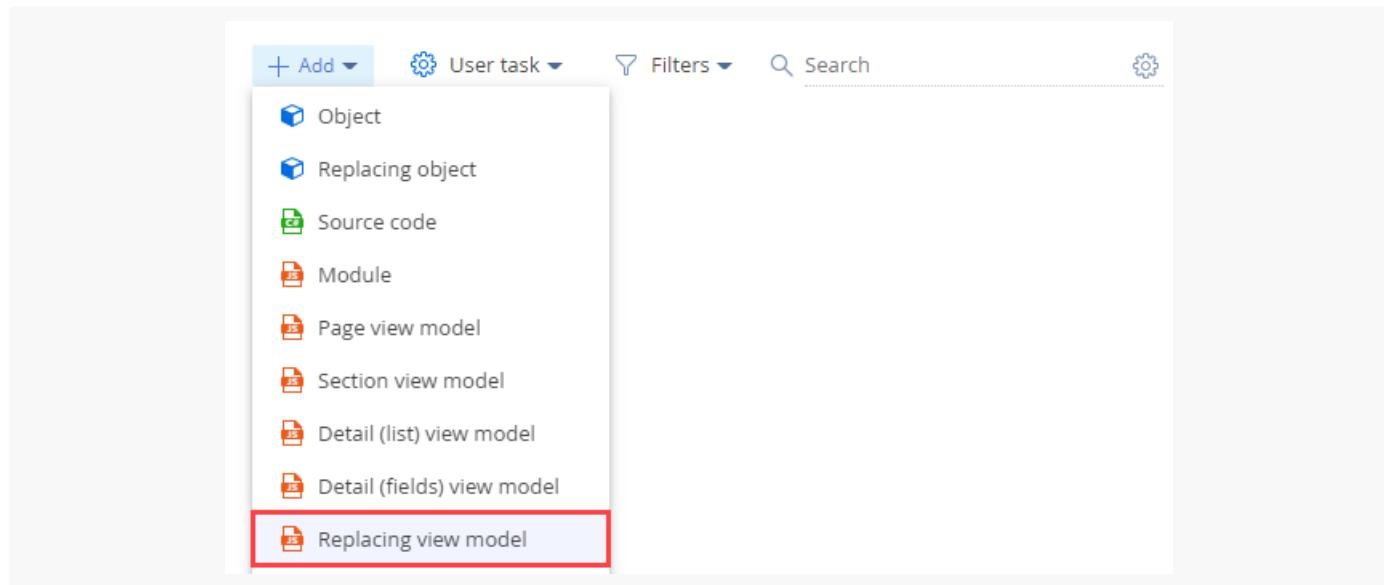
# Реализовать множественное добавление записей на деталь



**Пример.** Реализовать множественное добавление записей на деталь [ Контакты ] ([ *Contacts* ]) страницы записи раздела [ Продажи ] ([ *Opportunities* ]).

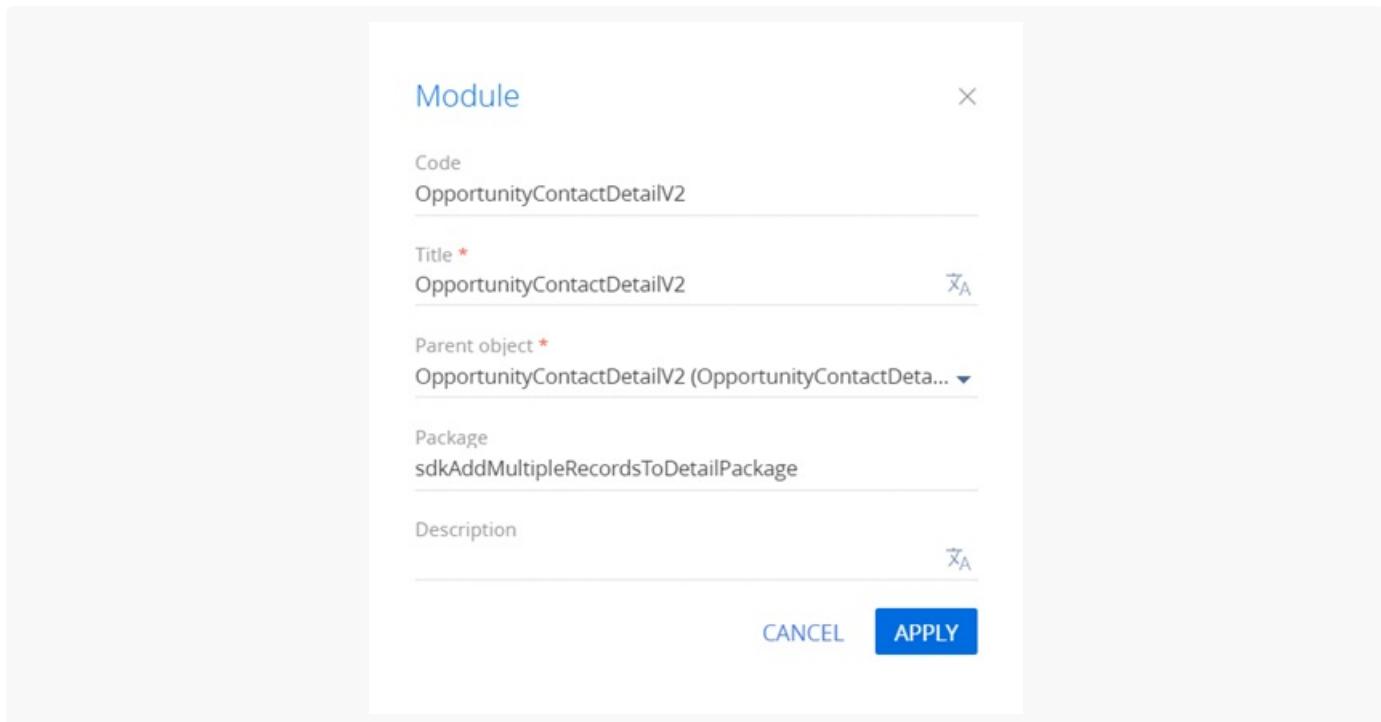
## 1. Создать схему замещающей модели представления детали

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. В дизайнере модуля заполните свойства схемы:

- [ Код ] ([ *Code* ]) — "OpportunityContactDetailV2".
- [ Заголовок ] ([ *Title* ]) — "OpportunityContactDetailV2".
- [ Родительский объект ] ([ *Parent object* ]) — выберите "OpportunityContactDetailV2".



## 2. Реализуйте бизнес-логику детали

1. В свойство `mixins` схемы детали добавьте миксин `LookupMultiAddMixin`.
2. В переопределенном методе `init()` схемы детали инициализируйте миксин `LookupMultiAddMixin`. Метод `init()` описан в статье [Виды модулей](#).
3. Переопределите метод `getAddRecordButtonVisible()`, который отвечает за отображение кнопки добавления.
4. Переопределите метод `onCardSaved()`, который отвечает за сохранение страницы детали. Используйте метод `openLookupWithMultiSelect()`, который вызывает справочное окно для множественного выбора.
5. Реализуйте метод `getMultiSelectLookupConfig()`, который выполняет конфигурирование справочного окна. Связанный с методом `openLookupWithMultiSelect()`. Возвращает объект конфигурации для справочного окна.

**Свойства** объекта:

- `rootEntitySchemaName` — корневая схема объекта.
  - `rootColumnName` — связующая колонка, которая указывает на запись корневой схемы.
  - `relatedEntitySchemaName` — связанная схема.
  - `relatedColumnName` — колонка, которая указывает на запись связанной схемы.
6. Переопределите метод `addRecord()`, который отвечает за добавление записи на деталь. Как и для метода `onCardSaved()`, используйте метод `openLookupWithMultiSelect()`. Значение `true` указывает на необходимость выполнения проверки является ли запись новой.

В текущем примере справочное окно использует данные из таблицы `[OpportunityContact]`, которая

связана с колонкой [Opportunity] корневой схемы [Opportunity] и колонкой [Contact] связанной схемы [Contact].

Исходный код схемы детали представлен ниже.

#### OpportunityContactDetailV2

```
define("OpportunityContactDetailV2", ["LookupMultiAddMixin"], function() {
    return {
        mixins: {
            /* Подключение миксина к схеме. */
            LookupMultiAddMixin: "Terrasoft.LookupMultiAddMixin"
        },
        methods: {
            /* Переопределение базового метода инициализации схемы. */
            init: function() {
                this.callParent(arguments);
                /* Инициализация миксина. */
                this.mixins.LookupMultiAddMixin.init.call(this);
            },
            /* Переопределение базового метода отображения кнопки добавления. */
            getAddRecordButtonVisible: function() {
                /* Отображать кнопку добавления если деталь развернута, даже если для детали не
                return this.getToolsVisible();
            },
            /* Переопределение базового метода.
            Обработчик события сохранения страницы записи детали. */
            onCardSaved: function() {
                /* Открывает справочное окно с множественным выбором записей. */
                this.openLookupWithMultiSelect();
            },
            /* Переопределение базового метода добавления записи на деталь. */
            addRecord: function() {
                /* Открывает справочное окно с множественным выбором записей. */
                this.openLookupWithMultiSelect(true);
            },
            /* Метод, который возвращает конфигурационный объект для справочного окна. */
            getMultiSelectLookupConfig: function() {
                return {
                    /* Корневая схема – [Продажа]. */
                    rootEntitySchemaName: "Opportunity",
                    /* Колонка корневой схемы. */
                    rootColumnName: "Opportunity",
                    /* Связанная схема – [Контакт]. */
                    relatedEntitySchemaName: "Contact",
                    /* Колонка связанной схемы. */
                    relatedColumnName: "Contact"
                };
            }
        }
    }
});
```

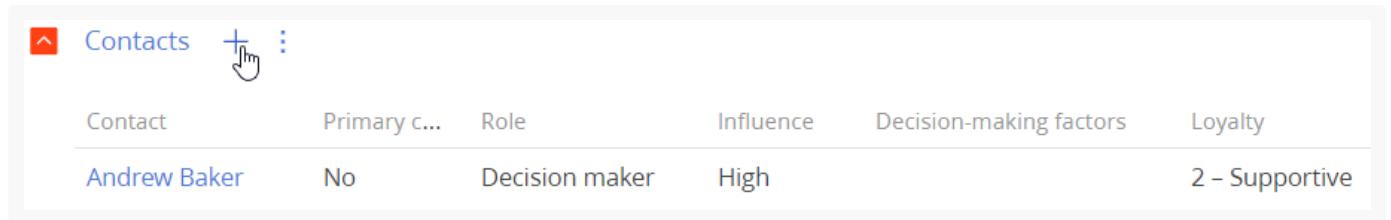
```
        }
    }
};

});
```

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

## Результат выполнения примера

1. Обновите страницу приложения.
2. На детали [ Контакты ] ([ Contacts ]) страницы записи раздела [ Продажи ] ([ Opportunities ]) нажмите кнопку .



Contact	Primary c...	Role	Influence	Decision-making factors	Loyalty
Andrew Baker	No	Decision maker	High		2 – Supportive

В результате выполнения примера приложение позволяет выбрать несколько записей из справочника.

Select: Contact

**SELECT** CANCEL NEW ACTIONS ▾ Records selected: 4 VIEW ▾

Full name	Email	Account
<input checked="" type="checkbox"/> James Smith	smith@gateway-invest.co.uk	Gateway
<input type="checkbox"/> Tran Manzo	TranManzo@gmail.com	
<input checked="" type="checkbox"/> Symon Clarke	symon-clarke@yahoo.com	Our company
<input type="checkbox"/> Youlonda Mcwhorter	YoulondaMcwhorter@gmail.com	
<input checked="" type="checkbox"/> Jason Robinson	jason_r@gmail.com	Our company
<input type="checkbox"/> Stasia Henrickson	StasiaHenrickson@hotmail.com	
<input type="checkbox"/> Stephen Washington	StephenWashington@gmail.com	
<input type="checkbox"/> Stewart Crispin	StewartCrispin@hotmail.com	
<input checked="" type="checkbox"/> Nora Wesley	TyroneRigg@hotmail.com	Gtech
<input type="checkbox"/> Winter Hodge	winterhodge@gmail.com	Console Solutions
<input type="checkbox"/> Zachariah Kershner	ZachariahKershner@gmail.com	
<input type="checkbox"/> Wilbur Brochures	WilburBrochures@gmail.com	

После подтверждения выбранные записи добавляются на деталь [ Контакты ] ([ Contacts ]) на странице записи раздела [ Продажи ] ([ Opportunities ]).

Contact	Primary contact	Role	Influen...	Decision-making factors	Loyalty
Jason Robinson	No	Contact person	Low		1 – Interested
James Smith	No	Decision maker	High		2 – Supportive
Nora Wesley	No	Decision maker	Medium		1 – Interested
Symon Clarke	No	Influencer	High		3 – Active supporter

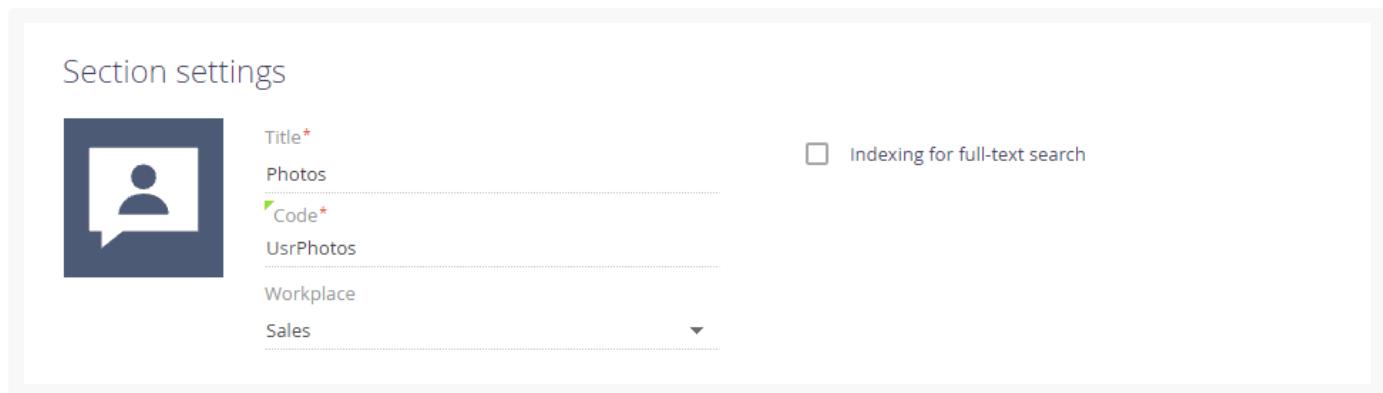
## Реализовать деталь типа [Файлы и ссылки]



**Пример.** На страницу записи пользовательского раздела [ Фотографии ] ([ Photos ]) добавить деталь [ Прикрепленные фотографии ] ([ Photos attachment ]) типа [ Файлы и ссылки ] ([ Attachments ]).

## 1. Создать пользовательский раздел

1. Создайте пользовательский пакет и установите его в качестве текущего. Подробнее читайте в статье [Общие принципы работы с пакетами](#).
2. Перейдите в дизайнер системы по кнопке .
3. В блоке [ Настройка системы ] ([ System setup ]) перейдите по ссылке [ Мастер разделов ] ([ Section wizard ]).
4. Заполните настройки раздела:
  - [ Заголовок ] ([ Title ]) — "Фотографии" ("Photos").
  - [ Код (на английском) ] ([ Code ]) — "UsrPhotos".
  - [ Рабочее место ] ([ Workplace ]) — выберите "Продажи" ("Sales").



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

В результате:

- Пользовательский раздел [ Фотографии ] ([ Photos ]) отображается в рабочем месте [ Продажи ] ([ Sales ]).

This screenshot shows the Creatio application interface. On the left, there's a sidebar with various menu items: Sales, Invoices, Documents, Products, Projects, Knowledge base, Forecasts, Chat, Partnership, and Photos. The 'Photos' item is highlighted with a red box. The main content area is titled 'Photos' and displays a large blue speech bubble icon with a white 'i'. Below it, the text 'This section has no records.' is shown. To the right, there's a vertical sidebar with icons for phone, email, messaging, notifications (with a red '99+' badge), and other system functions.

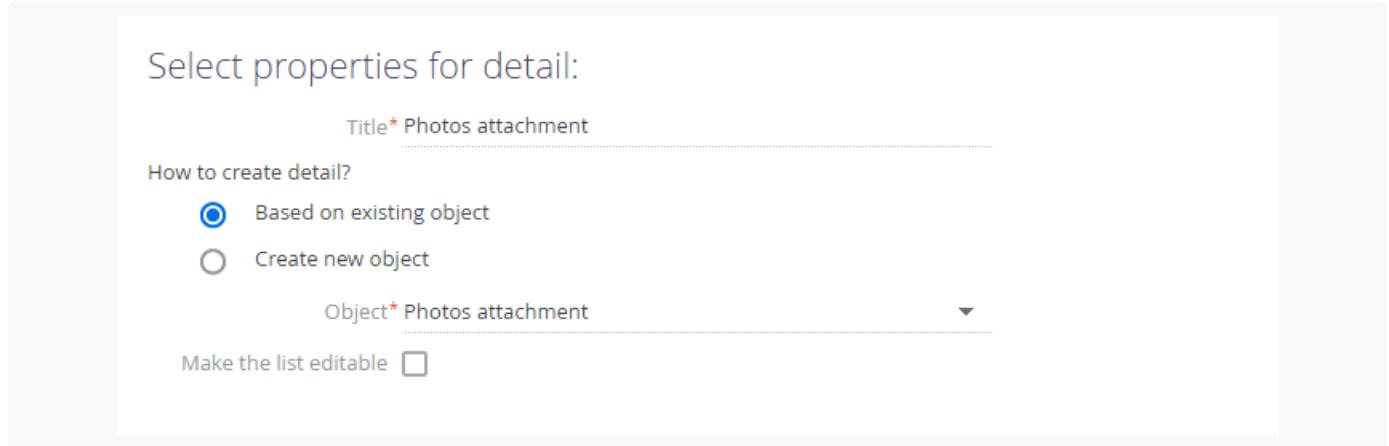
- В конфигурации созданы схемы раздела [ Фотографии ] ([ Photos ]).

<input type="checkbox"/> UsrPhotos *	Photos	Object	10/11/2021, 7:43:02 AM	sdkDetailAttachmentPackage	
<input type="checkbox"/> UsrPhotos1Page *	Edit page: "Photos"	Client module	10/11/2021, 7:42:47 AM	sdkDetailAttachmentPackage	
<input type="checkbox"/> UsrPhotos6ec3f85Section *	Section schema: "Photos"	Client module	10/11/2021, 7:42:36 AM	sdkDetailAttachmentPackage	
<input type="checkbox"/> UsrPhotosFile *	Photos attachment	Object	10/11/2021, 7:43:11 AM	sdkDetailAttachmentPackage	
<input type="checkbox"/> UsrPhotosFolder *	Photos folder	Object	10/11/2021, 7:43:27 AM	sdkDetailAttachmentPackage	
<input type="checkbox"/> UsrPhotosInFolder *	Photos in Folder	Object	10/11/2021, 7:43:36 AM	sdkDetailAttachmentPackage	
<input type="checkbox"/> UsrPhotosInTag *	Photos section record tag	Object	10/11/2021, 7:43:57 AM	sdkDetailAttachmentPackage	
<input type="checkbox"/> UsrPhotosTag *	Photos section tag	Object	10/11/2021, 7:43:48 AM	sdkDetailAttachmentPackage	

## 2. Создать пользовательскую деталь

1. Перейдите в дизайнер системы по кнопке .
2. В блоке [ Настройка системы ] ([ System setup ]) перейдите по ссылке [ Мастер деталей ] ([ Detail wizard ]).
3. Заполните свойства детали:
  - [ Заголовок ] ([ Title ]) — "Прикрепленные фотографии" ("Photos attachment").
  - [ По какому объекту создать деталь? ] ([ How to create detail? ]) — выберите "Существующему объекту" ("Based on existing object").

- [ Объект ] ([ Object ]) — выберите "Файл и ссылка объекта Фотографии" ("Photos attachment").



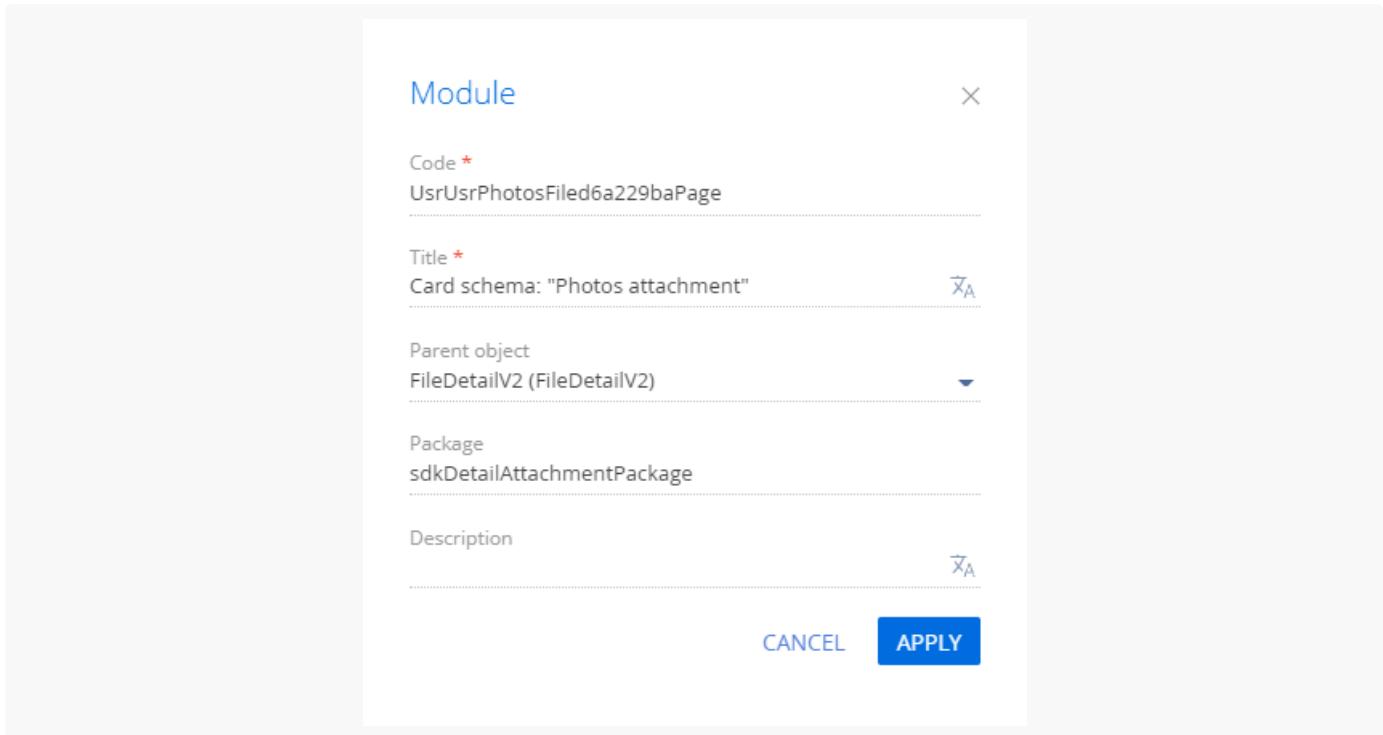
4. На панели инструментов мастера деталей нажмите [ Сохранить ] ([ Save ]).

После сохранения в конфигурации созданы:

- Схема `UsrSchemae9733d1bDetail` модели представления детали [ Прикрепленные фотографии ] ([ Photos attachment ]).
- Схема `UsrUsrPhotosFiled6a229baPage` страницы записи детали [ Прикрепленные фотографии ] ([ Photos attachment ]).

### 3. Настроить пользовательскую деталь

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ Configuration ]) и выберите пользовательский [пакет](#), который был установлен в качестве текущего.
2. Откройте схему `UsrUsrPhotosFiled6a229baPage` страницы записи детали [ Прикрепленные фотографии ] ([ Photos attachment ]).
3. На панели свойств нажмите кнопку и измените значение поля [ Родительский объект ] ([ Parent object ]) на `FileDetailV2`. Схема `FileDetailV2` пакета `UIv2` реализует деталь [ Файлы и ссылки ] ([ Attachments ]). По умолчанию в мастере деталей в качестве родительского объекта устанавливается базовая схема детали с реестром.



4. Для применения заданных свойств нажмите [ Применить ] ([ *Apply* ]).
5. На панели инструментов дизайнера нажмите [ Сохранить ] ([ *Save* ]).

## 4. Добавить деталь в раздел

1. Перейдите в раздел [ Фотографии ] ([ *Photos* ]).
2. На панели инструментов кликните [ Вид ] —> [ Открыть мастер раздела ] ([ *View* ] —> [ *Open section wizard* ]).
3. В блоке [ Страницы раздела ] ([ *Section pages* ]) нажмите кнопку [ Редактировать страницу ] ([ *Edit page* ]).
4. В рабочей области мастера разделов нажмите кнопку [ Добавить деталь ] ([ *New detail* ]).
5. Заполните настройки детали.
  - [ Деталь ] ([ *Detail* ]) — выберите "Прикрепленные фотографии" ("Photos attachment"). Поля [ Заголовок ] ([ *Title* ]) и [ Код (на английском) ] ([ *Code* ]) заполняются автоматически.
  - [ Заголовок ] ([ *Title* ]) — измените на "Прикрепленные фотографии" ("Photos attachment").

Detail settings

**SAVE** **CANCEL**

**General**

**Detail \***  
Photos attachment

**Title \***  
Photos attachment

**Code \***  
FileDetailV2c4be19b899

**What records to show on the page?**

**Where detail column \***  
Photos

**Equals to page column \***  
Id

6. Нажмите [ Сохранить ] —> [ Мастер раздела ] —> [ Сохранить ] ([ Save ] —> [ Section wizard ] —> [ Save ]).

В результате деталь [ Прикрепленные фотографии ] ([ Photos attachment ]) будет добавлена на страницу записи раздела [ Фотографии ] ([ Photos ]).



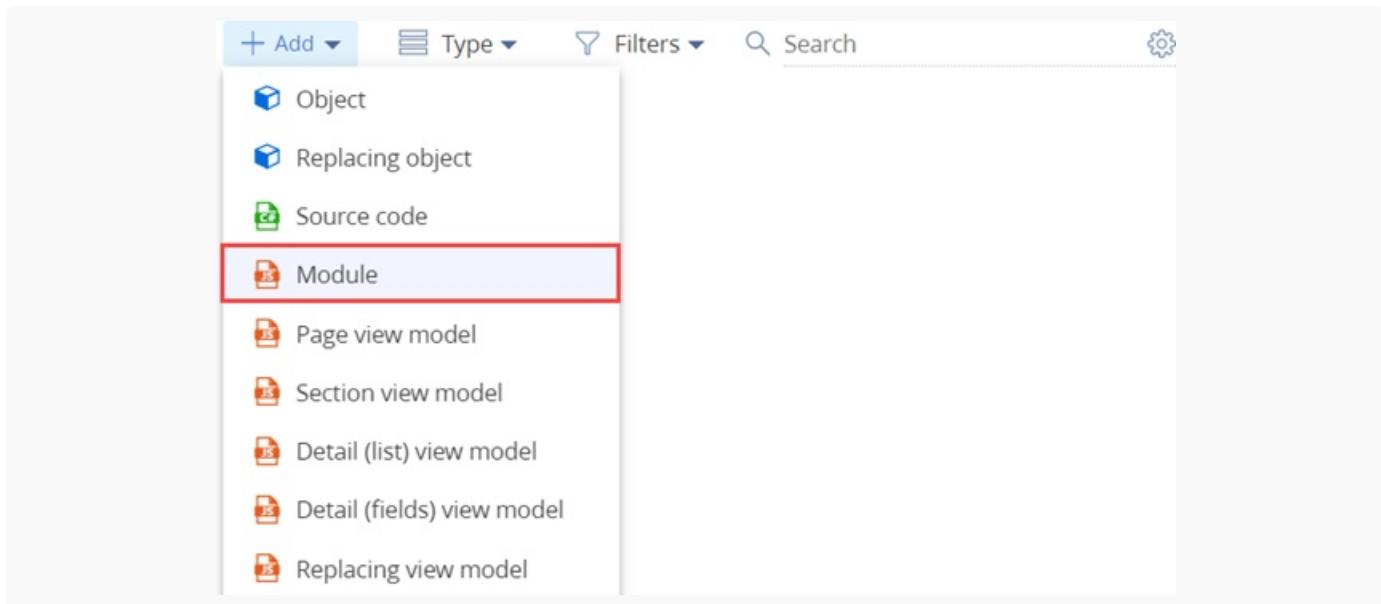
## 5. Добавить пользовательские стили детали

Поскольку в схеме модели представления страницы детали невозможно задать стили для отображения, необходимо:

1. Создать схему модуля, в которой определить стили.
2. Добавить модуль со стилями в зависимости модуля детали.

### 1. Создать схему модуля

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



3. Заполните свойства схемы:

- [Код] ([Code]) — "UsrSchemaDetailCSS".
- [Заголовок] ([Title]) — "SchemaDetailCSS".

### Module

X

---

Code *	UsrSchemaDetailCSS
Title *	SchemaDetailCSS <span style="float: right;">X<sub>A</sub></span>
Package	sdkDetailAttachmentPackage
Description	<span style="color: #ccc;">X<sub>A</sub></span>

CANCEL APPLY

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

4. Перейдите в узел [LESS] структуры объекта и задайте необходимые стили отображения детали.

#### Настройка стилей отображения детали

```
div[id*="UsrSchemae9733d1bDetail"] {
    .grid-status-message-empty {
        display: none;
    }
}
```

```
}

.grid-empty > .grid-bottom-spinner-space {
    height: 5px;
}

.dropzone {
    height: 35px;
    width: 100%;
    border: 1px dashed #999999;
    text-align: center;
    line-height: 35px;
}

.dropzone-hover {
    border: 1px dashed #4b7fc7;
}

.DragAndDropLabel {
    font-size: 1.8em;
    color: rgb(110, 110, 112);
}

}

div[data-item-marker*="added-detail"] {
    div[data-item-marker*="tiled"], div[data-item-marker*="listed"] {
        .entity-image-class {
            width: 165px;
        }

        .entity-image-container-class {
            float: right;
            width: 128px;
            height: 128px;
            text-align: center;
            line-height: 128px;
        }

        .entity-image-view-class {
            max-width: 128px;
            max-height: 128px;
            vertical-align: middle;
        }

        .images-list-class {
            min-height: 0.5em;
        }

        .images-list-class > .selectable {
            margin-right: 10px;
            display: inline-block;
        }

        .entity-label {
            display: block;
            max-width: 128px;
            margin-bottom: 10px;
            text-align: center;
        }
    }
}
```

```
}

.entity-link-container-class > a {
    font-size: 1.4em;
    line-height: 1.5em;
    display: block;
    max-width: 128px;
    margin-bottom: 10px;
    color: #444;
    text-decoration: none;
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: nowrap;
}
.entity-link-container-class > a:hover {
    color: #0e84cf;
}
.entity-link-container-class {
    float: right;
    width: 128px;
    text-align: center;
}
.select-entity-container-class {
    float: left;
    width: 2em;
}
.listed-mode-button {
    border-top-right-radius: 1px;
    border-bottom-right-radius: 1px;
}
.tiled-mode-button {
    border-top-left-radius: 1px;
    border-bottom-left-radius: 1px;
}
.tiled-mode-button, .listed-mode-button {
    padding-left: 0.308em;
    padding-right: 0.462em;
}
.button-pressed {
    background: #fff;

    .t-btn-image {
        background-position: 0 16px !important;
    }
}
div[data-item-marker*="tiled"] {
    .tiled-mode-button {
        .button-pressed;
```

```

        }
    }
    div[data-item-marker*="listed"] {
        .listed-mode-button {
            .button-pressed;
        }
    }
}

```

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

## 2. Модифицировать схему модели представления детали

Чтобы **использовать созданный модуль и его стили** в схеме детали:

- Откройте схему `UsrSchemae9733d1bDetail` модели представления детали [ *Прикрепленные фотографии* ] ([ *Photos attachment* ]).
- В зависимости схемы `UsrSchemae9733d1bDetail` добавьте модуль `UsrSchemaDetailCSS`.

Исходный код модифицированной схемы представлен ниже.

### UsrSchemae9733d1bDetail

```

define("UsrSchemae9733d1bDetail", ["css!UsrSchemaDetailCSS"], function() {
    return {
        entitySchemaName: "UsrPhotosFile",
        details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
        methods: {},
        diff: /**SCHEMA_DIFF*/[]/**SCHEMA_DIFF*/
    };
});

```

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

## Результат выполнения примера

В результате на страницу записи пользовательского раздела [ *Фотографии* ] ([ *Photos* ]) добавлена деталь [ *Прикрепленные фотографии* ] ([ *Photos attachment* ]).

Name	Description	Type	Created on	Created by
scr_section_props_in_wizard.png		File	10/11/2021 8:54 AM	Marina Kysla

Drag file here

# Схема BaseDetailV2 js



Сложный

`BaseDetailV2` — базовая схема детали. Предоставляет базовую логику инициализации данных детали и взаимодействия детали со страницей записи. Реализована в пакете `nui`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Все схемы деталей должны наследовать схему `BaseDetailV2`.

## Сообщения

Сообщения базовой детали

Название	Режим	Направление	Описание
<code>GetCardState</code>	Адресное	Публикация	Возвращает состояние страницы записи.
<code>IsCardChanged</code>	Адресное	Публикация	Сообщает об изменении страницы записи.
<code>SaveRecord</code>	Адресное	Публикация	Сообщает странице записи о необходимости сохранить данные.
<code>DetailChanged</code>	Адресное	Публикация	Сообщает странице записи об изменении данных детали.
<code>UpdateDetail</code>	Адресное	Подписка	Подписка на обновление страницы записи.
<code>OpenCard</code>	Адресное	Публикация	Открывает страницу записи.
<code>GetColumnsValues</code>	Адресное	Публикация	Возвращает запрашиваемые значения колонок.
<code>UpdateCardProperty</code>	Адресное	Публикация	Изменяет значение модели страницы записи.
<code>GetEntityInfo</code>	Адресное	Публикация	Запрашивает информацию о сущности основной записи.

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

---

**CanAdd** BOOLEAN

Признак возможности добавления данных.

---

**CanEdit** BOOLEAN

Признак возможности редактирования данных.

---

**CanDelete** BOOLEAN

Признак возможности удаления данных.

---

**Collection** COLLECTION

Коллекция данных детали.

---

**Filter** CUSTOM\_OBJECT

Фильтр детали. Используется для фильтрации данных в детали.

---

**DetailColumnName** STRING

Имя колонки, по которой выполняется фильтрация.

---

**MasterRecordId** GUID

Значение ключа родительской записи.

---

**IsDetailCollapsed** BOOLEAN

Признак свернутости детали.

---

**DefaultValues** CUSTOM\_OBJECT

Значения колонок модели по умолчанию.

---

**Caption** STRING

Заголовок детали.

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

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

## Методы

---

`init(callback, scope)`

Инициализирует страницу детали.

### Параметры

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

---

`initProfile`

Инициализирует профиль схемы. По умолчанию содержит значение `Terrasoft.emptyFn`.

`initDefaultCaption()`

Устанавливает заголовок детали по умолчанию.

`initDetailOptions()`

Инициализирует коллекцию данных представления реестра.

`subscribeSandboxEvents()`

Подписывается на сообщения, необходимые для работы детали.

`getUpdateDetailSandboxTags()`

Генерирует массив тегов для сообщения `UpdateDetail`.

`updateDetail`

Обновляет деталь согласно переданным параметрам. По умолчанию содержит значение `Terrasoft.emptyFn`.

### Параметры

<code>{Object} config</code>	Конфигурационный объект, содержащий свойства детали.
------------------------------	--

---

```
initWith(callback, scope)
```

Инициализирует коллекцию данных представления реестра.

#### Параметры

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

---

```
getEditPageName()
```

Возвращает имя страницы записи в зависимости от типа выбранной записи (при редактировании) или от выбранного типа записи для добавления (при добавлении).

---

```
onDetailCollapsedChanged(isCollapsed)
```

Обработчик сворачивания или разворачивания детали.

#### Параметры

{Boolean} isCollapsed	Признак свернутости детали.
-----------------------	-----------------------------

---

```
getToolsVisible()
```

Возвращает значение свернутости детали.

---

```
getDetailInfo()
```

Публикует сообщение для получения информации о детали.

## Массив модификаций

В массиве модификаций `diff` базовой детали определен только базовый контейнер для представления детали.

### Массив модификаций `diff`

```
diff: /**SCHEMA_DIFF*/[
    /* Базовый контейнер для представления детали. */
    {
        "operation": "insert",
        "name": "Detail",
        "values": {
```

```

    ...
}
}/*SCHEMA_DIFF*/

```

# Схема BaseGridDetailV2 js



Сложный

`BaseGridDetailV2` — базовая схема детали с реестром. Предоставляет базовую логику работы с реестром (загрузка, фильтрация), удаление, добавление и редактирование записей на детали. Реализована в пакете `nui`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#). Является наследником схемы `BaseDetailV2`. Все схемы деталей с реестром должны наследовать схему `BaseGridDetailV2`.

## Сообщения

Сообщения базовой детали с реестром

Название	Режим	Направление	Описание
<code>getCardInfo</code>	Адресное	Подписка	Возвращает информацию о странице записи — значения по умолчанию, название колонки типизации, значение колонки типизации.
<code>CardSaved</code>	Широковещательное	Подписка	Обрабатывает сообщение сохранения страницы записи.
<code>RerenderQuickFilterModule</code>	Адресное	Публикация	Публикует сообщение с применением фильтра.
<code>GetExtendedFilterConfig</code>	Адресное	Подписка	Публикует конфиг пользовательского фильтра.
<code>GetModuleSchema</code>	Адресное	Подписка	Возвращает информацию о

			сущности, которая работает с фильтром.
<code>UpdateFilter</code>	Широковещательное	Подписка	Обновляет фильтры в детали.
<code>LoadedFiltersFromStorage</code>	Широковещательное	Публикация	Фильтры, которые загружены с хранилища.
<code>InitFilterFromStorage</code>	Широковещательное	Подписка	Инициализирует фильтры, которые загружены с хранилища.
<code>GetColumnsValues</code>	Адресное	Публикация	Получает значения колонок модели страницы записи.
<code>IsCardChanged</code>	Адресное	Публикация	Сообщает об изменениях страницы записи.
<code>ValidateCard</code>	Адресное	Публикация	Запрос на валидацию страницы записи.

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](#).

## Миксины

`GridUtilities` `Terrasoft.GridUtilities`

Миксин для работы с реестром.

`WizardUtilities` `Terrasoft.WizardUtilities`

Миксин для работы с мастером деталей.

## Атрибуты

`ActiveRow` `GUID`

Значение первичной колонки активной записи реестра.

---

**IsEmpty** BOOLEAN

Признак пустого реестра.

---

**MultiSelect** BOOLEAN

Признак наличия разрешения ли множественный выбор.

---

**SelectedRows** COLLECTION

Массив выбранных записей.

---

**RowCount** INTEGER

Количество строк в реестре.

---

**IsPageable** BOOLEAN

Признак активности постраничной загрузки.

---

**SortColumnIndex** INTEGER

Индекс колонки сортировки.

---

**CardState** TEXT

Режим открытия страницы записи.

---

**EditPageUId** GUID

Уникальный идентификатор страницы записи.

---

**DetailFilters** COLLECTION

Коллекция фильтров детали.

---

**IsDetailWizardAvailable** BOOLEAN

Признак доступности мастера деталей.

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

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

## Методы

---

`init(callback, scope)`

Замещает метод класса `BaseDetailV2`. Вызывает логику метода `init` родителя, регистрирует сообщения, инициализирует фильтры.

### Параметры

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

---

`initData(callback, scope)`

Замещение метода класса `BaseDetailV2`. Вызывает логику метода `initData` родительского класса, инициализирует коллекцию данных представления реестра.

### Параметры

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

---

`loadGridData()`

Выполняет загрузку данных реестра.

---

`initGridData()`

Выполняет инициализацию значений по умолчанию для работы со списком.

---

`getGridData()`

Возвращает коллекцию реестра.

---

`getFilters()`

Возвращает коллекцию фильтров детали.

---

```
getActiveRow()
```

Возвращает идентификатор выбранной записи в реестре.

---

```
addRecord(editPageUID)
```

Добавляет новую запись в реестр. Сохраняет страницу записи в случае необходимости.

---

#### Параметры

{String} editPageUID	Идентификатор типизированной страницы записи.
----------------------	---

---

```
copyRecord(editPageUID)
```

Копирует запись и открывает страницу записи.

---

#### Параметры

{String} editPageUID	Идентификатор типизированной страницы записи.
----------------------	---

---

```
editRecord(record)
```

Открывает страницу выбранной записи.

---

#### Параметры

{Object} record	Модель записи для редактирования.
-----------------	-----------------------------------

---

```
subscribeSandboxEvents()
```

Подписывается на сообщения, которые необходимы для работы детали.

---

```
updateDetail(config)
```

Замещение метода класса `BaseDetailV2`. Вызывает логику метода `updateDetail` родителя, обновляет деталь.

---

#### Параметры

{Object} config	Конфигурационный объект, который содержит свойства детали.
-----------------	--

```
openCard(operation, typeColumnName, recordId)
```

Открывает страницу записи.

#### Параметры

{String} operation	Тип операции (добавление/редактирование).
{String} typeColumnName	Значение колонки типизации записи.
{String} recordId	Идентификатор записи.

---

```
onCardSaved()
```

Обрабатывает событие сохранения страницы записи, в которой находится деталь.

---

```
addToolsButtonMenuItems(toolsButtonMenu)
```

Добавляет элементы в коллекцию выпадающего списка функциональной кнопки.

#### Параметры

{Terrasoft. BaseViewModelCollection} toolsButtonMenu	Коллекция выпадающего списка функциональной кнопки.
--	---

---

```
initDetailFilterCollection()
```

Инициализирует фильтр детали.

---

```
setFilter(key, value)
```

Устанавливает значение фильтров детали.

#### Параметры

{String} key	Тип фильтров.
{Object} value	Значение фильтров.

---

```
loadQuickFilter(config)
```

Загружает быстрый фильтр.

## Параметры

{Object} config

Параметры загрузки модуля фильтров.

`destroy()`

Очищает данные, выгружает деталь.

## Массив модификаций

В массиве модификаций `diff` базовой детали с реестром определен только базовый контейнер для представления детали.

### Массив модификаций `diff`

```
diff: /**SCHEMA_DIFF*/ [
  {
    /* Элемент для отображения реестра. */
    "operation": "insert",
    "name": "DataGrid",
    "parentName": "Detail",
    "propertyName": "items",
    "values": {
      "itemType": Terrasoft.ViewItemType.GRID,
      ...
    }
  },
  {
    /* Кнопка дозагрузки реестра. */
    "operation": "insert",
    "parentName": "Detail",
    "propertyName": "items",
    "name": "loadMore",
    "values": {
      "itemType": Terrasoft.ViewItemType.BUTTON,
      ...
    }
  },
  {
    /* Кнопка добавления записи. */
    "operation": "insert",
    "name": "AddRecordButton",
    "parentName": "Detail",
    "propertyName": "tools",
    "values": {
      "itemType": Terrasoft.ViewItemType.BUTTON,
      ...
    }
  }
]
```

```

        }
    },
{
    /* Кнопка добавления типизированной записи. */
    "operation": "insert",
    "name": "AddTypedRecordButton",
    "parentName": "Detail",
    "propertyName": "tools",
    "values": {
        "itemType": Terrasoft.ViewItemType.BUTTON,
        ...
    }
},
{
    /* Меню детали. */
    "operation": "insert",
    "name": "ToolsButton",
    "parentName": "Detail",
    "propertyName": "tools",
    "values": {
        "itemType": Terrasoft.ViewItemType.BUTTON,
        ...
    }
}
] /**SCHEMA_DIFF*/

```

## Миксин GridUtilitiesV2



Сложный

`GridUtilitiesV2` — миксин, который предоставляет логику работы с элементом управления "Реестр". Реализован в классе `Terrasoft.configuration.mixins.GridUtilities` пакета `NUI`. Элемент управления "Реестр" описан в статье [Реестр раздела](#).

**Миксин позволяет:**

- Подписываться на сообщения.
- Загружать данные.
- Работать с реестром:
  - Выбирать записи (выполнять поиск активных записей).
  - Добавлять, удалять, редактировать записи.
  - Задавать фильтры.
  - Сортировать записи.
  - Экспортировать записи в файл.
  - Проверять права доступа к записям реестра.

## Методы

---

`init()`

Выполняет подписку на события.

---

`destroy()`

Очищает подписки на события.

---

`loadGridData()`

Выполняет загрузку данных реестра.

---

`beforeLoadGridData()`

Подготавливает модель представления перед загрузкой данных.

---

`afterLoadGridData()`

Подготавливает модель представления после загрузки данных.

---

`onGridDataLoaded(response)`

Обрабатывает события загрузки данных. Выполняется, когда сервер возвращает данные.

### Параметры

<code>{Object} response</code>	Результат выборки данных из базы данных.
--------------------------------	--

`addItemsToGridData(dataCollection, options)`

Добавляет коллекцию новых элементов в коллекцию реестра.

### Параметры

<code>{Object} dataCollection</code>	Коллекция новых элементов.
<code>{Object} options</code>	Параметры добавления.

`initQueryOptions(esq)`

Инициализирует настройки (постраничность, иерархичность) экземпляра запроса.

#### Параметры

{Terrasoft.data.queries. EntitySchemaQuery} esq	Запрос, в котором будут инициализированы необходимые настройки.
--	--

---

`initQuerySorting(esq)`

Инициализирует колонки сортировки.

#### Параметры

{Terrasoft.data.queries. EntitySchemaQuery} esq	Запрос, в котором будут инициализированы необходимые настройки.
--	--

---

`prepareResponseCollection(collection)`

Модифицирует коллекцию данных перед загрузкой в реестр.

#### Параметры

{Object} collection	Коллекция элементов реестра.
---------------------	------------------------------

---

`getFilters()`

Возвращает примененные в данной схеме фильтры. Переопределяется в наследниках.

---

`exportToFile()`

Экспортирует содержимое реестра в файл.

---

`sortGrid(tag)`

Выполняет сортировку в реестре.

#### Параметры

{String} tag	Ключ, который указывает, как пересортировать реестр.
--------------	---

---

```
deleteRecords()
```

Инициирует удаление выбранных записей.

---

```
checkCanDelete(items, callback, scope)
```

Проверяет возможность удаления записи.

#### Параметры

{Array} items	Идентификаторы выбранных записей.
{Function} callback	Функция обратного вызова.
{Object} scope	Контекст выполнения метода.

---

```
onDeleteAccept()
```

Выполняет удаление после подтверждения пользователем.

---

```
getSelectedItems()
```

Возвращает выбранные записи в реестре.

---

```
removeGridRecords(records)
```

Убирает из реестра удаленные записи.

#### Параметры

{Array} records	Удаленные записи.
-----------------	-------------------

---

```
reloadGridData()
```

Выполняет перезагрузку реестра.

## Модуль ConfigurationGrid



Сложный

ConfigurationGrid — модуль, который предоставляет логику работы с элементом управления "Конфигурационный реестр". Реализован в классе `Terrasoft.controls.ConfigurationGrid` пакета `UIv2`.

Класс `Terrasoft.controls.ConfigurationGrid` является наследником класса `Terrasoft.Grid`.

## Методы

`init()`

Инициализирует компонент. Осуществляет подписку на события.

`activateRow(id)`

Выделяет строку и добавляет элементы редактирования.

### Параметры

<code>{String Number} id</code>	Идентификатор строки реестра.
---------------------------------	-------------------------------

`deactivateRow(id)`

Снимает выделение строки и удаляет элементы редактирования.

### Параметры

<code>{String Number} id</code>	Идентификатор строки реестра.
---------------------------------	-------------------------------

`formatCellContent(cell, data, column, link)`

Форматирует данные ячейки строки.

### Параметры

<code>{Object} cell</code>	Ячейка.
----------------------------	---------

<code>{Object} data</code>	Данные.
----------------------------	---------

<code>{Object} column</code>	Конфигурация ячейки.
------------------------------	----------------------

<code>{Object} link</code>	Ссылка.
----------------------------	---------

`onUpdateItem(item)`

Обработчик события обновления записи.

### Параметры

{Terrasoft.BaseViewModel} item

Элемент коллекции.

onDestroy(clear)

Уничтожает реестр и его компоненты.

#### Параметры

{Boolean} clear

Очистка реестра и компонентов.

## Модуль ConfigurationGridGenerator js



Сложный

ConfigurationGridGenerator — модуль, который генерирует конфигурацию реестра. Реализован в классе Terrasoft.configuration.ConfigurationGridGenerator пакета UIv2 . Класс Terrasoft.configuration.ConfigurationGridGenerator является наследником класса Terrasoft.ViewGenerator .

### Методы

addLinks

Переопределенный метод класса Terrasoft.ViewGenerator . По умолчанию содержит значение Terrasoft.emptyFn . В редактируемый реестр не будут добавлены ссылки.

generateGridCellValue(config)

Переопределенный метод класса Terrasoft.ViewGenerator . Генерирует конфигурацию значения в ячейке.

#### Параметры

{Object} config

Конфигурация колонки.

## Модуль ConfigurationGridUtilities js



Сложный

ConfigurationGridUtilities — модуль, который содержит методы инициализации модели представления строки реестра, обработки действий активной записи и обработки горячих клавиш. Реализован в классе Terrasoft.configuration.mixins.ConfigurationGridUtilities пакета UIv2 .

## Свойства

`currentActiveColumnName String`

Имя текущей выделенной колонки.

`columnsConfig String`

Конфигурация колонок.

`systemColumns Array`

Коллекция названий системных колонок.

## Методы

`onActiveRowAction(buttonTag, primaryColumnName)`

Обрабатывает нажатие действия активной записи.

### Параметры

<code>{String} buttonTag</code>	Тег выбранного действия.
<code>{String} primaryColumnName</code>	Идентификатор активной записи.

`activeRowSaved(row, callback, scope)`

Обрабатывает результат сохранения записи.

### Параметры

<code>{Object} row</code>	Строка реестра.
<code>{Function} [callback]</code>	Функция обратного вызова.
<code>{Object} scope</code>	Контекст вызова функции обратного вызова.

`initActiveRowKeyMap(keyMap)`

Инициализирует подписку на события нажатия кнопок в активной строке.

### Параметры

{Array} keyMap

Описание событий.

`getCellControlsConfig(entitySchemaColumn)`

Возвращает конфигурацию элементов редактирования ячейки реестра.

**Параметры**

{Terrasoft.EntitySchemaColumn} entitySchemaColumn

Колонка ячейки реестра.

`copyRow(recordId)`

Копирует и добавляет запись в реестр.

**Параметры**

{String} recordId

Идентификатор копируемой записи.

`initEditableGridRowViewModel(callback, scope)`

Инициализирует классы элементов коллекции редактируемого реестра.

**Параметры**

{Function} callback

Функция обратного вызова.

{Object} scope

Контекст вызова функции обратного вызова.

`saveRowChanges(row, callback, scope)`

Сохраняет запись.

**Параметры**

{Object} row

Строка реестра.

{Function} [callback]

Функция обратного вызова.

{Object} [scope]

Контекст вызова функции обратного вызова.

# Схема BasePageV2 js

 Средний

`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
<code>UpdatePageHeaderCaption</code>	Адресное	Публикация	Обновляет заголовок страницы.
<code>GridRowChanged</code>	Адресное	Подписка	Получает идентификатор выбранной в реестре записи при ее изменении.
<code>UpdateCardProperty</code>	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>ReInitializeActionsDashboard</code>	Адресное	Подписка	Обновляет конфиг панели действий.
<code>ReloadDashboardItems</code>	Широковещательное	Публикация	Перезагружает элементы дашбордов.
<code>ReloadDashboardItemsPTP</code>	Адресное	Публикация	Перезагружает элементы дашбордов для текущей страницы.
<code>CanChangeHistoryState</code>	Широковещательное	Подписка	Разрешает или запрещает изменение текущего состояния истории.
<code>IsEntityChanged</code>	Адресное	Подписка	Возвращает измененную сущность.
<code>IsDcmFilterColumnChanged</code>	Адресное	Подписка	Возвращает <code>true</code> , если изменены отфильтрованные колонки.
<code>UpdateParentLookupDisplayValue</code>	Широковещательное	Двунаправленное	Обновляет значение родительской записи справочника по конфигу.
<code>UpdateParentLookupDisplayValue</code>	Широковещательное	Двунаправленное	Указывает на

Value	ненуждимость 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.

`addChangeDataViewOptions(viewOptions)`

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

#### Параметры

{Terrasoft.BaseViewModelCollection} viewOptions	Пункты выпадающего списка кнопки [ Вид ] ([ View ]).
--	--

---

```
addSectionDesignerViewOptions(viewOptions)
```

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

#### Параметры

{Terrasoft.BaseViewModelCollection} viewOptions	Пункты выпадающего списка кнопки [ Вид ] ([ View ]).
--	--

---

```
getReportFilters()
```

Возвращает коллекцию фильтров для запроса.

---

```
initPageHeaderColumnNames()
```

Инициализировать имена колонок заголовка страницы.

---

```
getParameters(parameters)
```

Возвращает значения параметров `ViewModel`.

#### Параметры

{Array} parameters	Имена параметров.
--------------------	-------------------

---

```
setParameters(parameters)
```

Задает параметры `ViewModel`.

#### Параметры

{Object} parameters	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

<code>{String} reportId</code>	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 BaseFieldsDetail



Средний

`BaseFieldsDetail` — базовая схема детали с полями. Реализована в пакете `baseFinance` продуктов линейки Financial Services Creatio. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#).

## Сообщения

Сообщения базовой детали с полями

Название	Режим	Направление	Описание
LookupInfo	Адресное	Подписка	Возвращает информацию о справочнике.
UpdateCardProperty	Адресное	Публикация	Изменяет значение модели страницы записи.
CardSaved	Широковещательное	Подписка	Получает информацию о сохранении родительской страницы.
IsCardChanged	Адресное	Публикация	Сообщает об изменении карточки.

## Схема FileDetailV2 js



Средний

`FileDetailV2` — базовая схема детали типа [ Файлы и ссылки ] ([ *Attachments* ]). Реализована в пакете `uiV2`. Схема является схемой модели представления. Описание свойств схемы содержится в статье [Клиентская схема](#).

## Атрибуты

`SchemaCardName` TEXT

Сохранить имя страницы записи.

`parentEntity` CUSTOM\_OBJECT

Родительская сущность.

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

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

## Методы

`getShowPreviewSettingsValue()`

Получить значение системных настроек `ShowPreview`.

`initParentEntity()`

Инициализировать родительскую сущность.

`itemsRendered()`

Обрабатывает событие `itemsRendered`, которое сработало в компоненте `ContainerList`.

## Мини-карточка



Основы

**Мини-карточка** — сокращенная версия страницы записи с ограниченным количеством полей. Мини-карточки позволяют быстро получить или отредактировать информацию о записи, не открывая отдельную страницу. **Назначение** мини-карточки — ускорение добавления, редактирования и просмотра записей. Набор полей в каждом из типов мини-карточек настраивается отдельно и будет различаться.

Мини-карточка контакта

Field	Value
Job title	CFO
Email	alexander.wilson@alphabusiness.com
Business phone	+1 212 542 4238
Mobile phone	+1 212 854 7512
Business phone	+1 212 1440 5222
Mobile phone	+1 212 1204 5477
Business phone	+1 617 440 2031
Mobile phone	+1 617 221 5187

Мини-карточку можно создать для любого объекта системы.

Подробности о работе с мини-карточками описаны в статье [Мини-карточки](#).

## Схема модели представления мини-карточки

В Creatio IDE мини-карточка реализована с помощью [схемы модели представления](#).

Схема модели представления мини-карточки **позволяет настроить**:

- Состав мини-карточки.
- Расположение элементов пользовательского интерфейса мини-карточки.

- Поведение элементов пользовательского интерфейса мини-карточки.

Например, мини-карточка контакта конфигурируется схемой `ContactMiniPage`, а мини-карточка контрагента — схемой `AccountMiniPage`. Родительской схемой для схем-миникарточек является схема `BaseMiniPage` пакета `NUI`.

Структура схемы модели представления мини-карточки не отличается от общей структуры [клиентской схемы модели представления](#).

**Обязательные свойства** в структуре схемы мини-карточки:

- `entitySchemaName` — имя схемы объекта, к которому будет привязана мини-карточка,
- `diff` — массив модификаций визуальных элементов мини-карточки.

**Дополнительные свойства** в структуре схемы мини-карточки:

- `attributes` — атрибуты схемы.
- `methods` — методы схемы.
- `mixins` — миксины схемы.
- `messages` — сообщения схемы.

Дополнительные свойства **позволяют**:

- Добавлять пользовательские элементы управления.
- Регистрировать сообщения.
- Формировать бизнес-логику работы мини-карточки.

Существует возможность изменения внешнего вида визуальных элементов мини-карточки с помощью пользовательских стилей.

**Важно.** В мини-карточках не поддерживается механизм настройки бизнес-логики с помощью бизнес-правил.

## Операции с мини-карточками

### Добавить мини-карточку в раздел

- В пользовательский [пакет](#) добавьте [схему модели представления](#) карточки.
- В качестве родительского объекта выберите схему `BaseMiniPage`.
- Добавьте необходимую функциональность мини-карточки в исходный код схемы. При этом в элементе `entitySchemaName` обязательно укажите имя схемы объекта, к которому будет привязана мини-карточка, и внесите хотя бы одну модификацию в массив `diff`.
- Используя SQL-запрос, внесите изменения в системную таблицу `[SysModuleEdit]` базы данных.
- Добавьте системную настройку `[HasCodeРазделаMiniPageAddMode]`. Добавление системной настройки

описано в статье [Управление системными настройками](#).

**Важно.** Выполнение SQL-запроса, который содержит ошибку, может привести к повреждению существующих данных и неработоспособности приложения.

## Добавить мини-карточку к произвольному модулю

Для некоторых бизнес-задач возникает необходимость подключения мини-карточки к произвольному модулю Creatio. **Произвольные модули** позволяют создавать ссылки в системе на определенный объект, поэтому подключение отображения мини-карточки при наведении на ссылку позволяет получить информацию об этом объекте, не переходя в раздел этого объекта.

В базовой версии приложения мини-карточка объекта подключена к следующим модулям:

- телефония в коммуникационной панели;
- email в коммуникационной панели;
- центр уведомлений в коммуникационной панели;
- раздел [Лента] в коммуникационной панели;
- графика-списка в разделе итогов.

Чтобы **добавить мини-карточку к произвольному модулю**:

1. Создайте схему модуля.
2. Создайте представление и модель представления модуля. Подключите в свойство `mixins` модели представления утилитный класс `Terrasoft.MiniPageUtilities`. Класс позволит использовать методы вызова мини-карточки.
3. Добавьте стили модуля.
4. Создайте контейнер отображения представления модуля.

## Создать пользовательскую мини-карточку

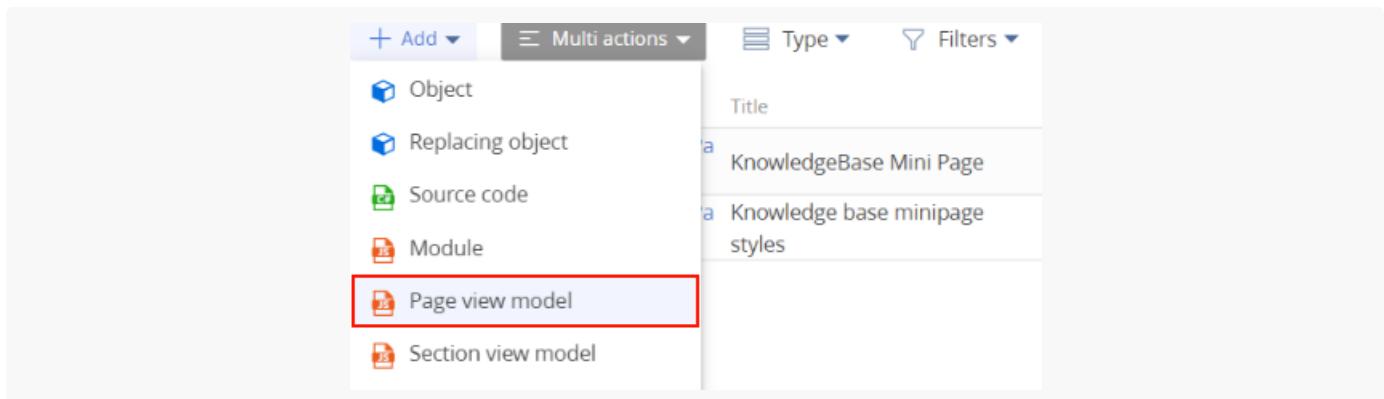
 Сложный

**Пример.** Создать пользовательскую мини-карточку для раздела [База знаний] ([Knowledge base]). Мини-карточка будет служить для просмотра базового набора полей [Название] ([Name]) и [Теги] ([Tags]) с возможностью скачивания прикрепленных файлов.

### 1. Создать схему модели представления мини-карточки

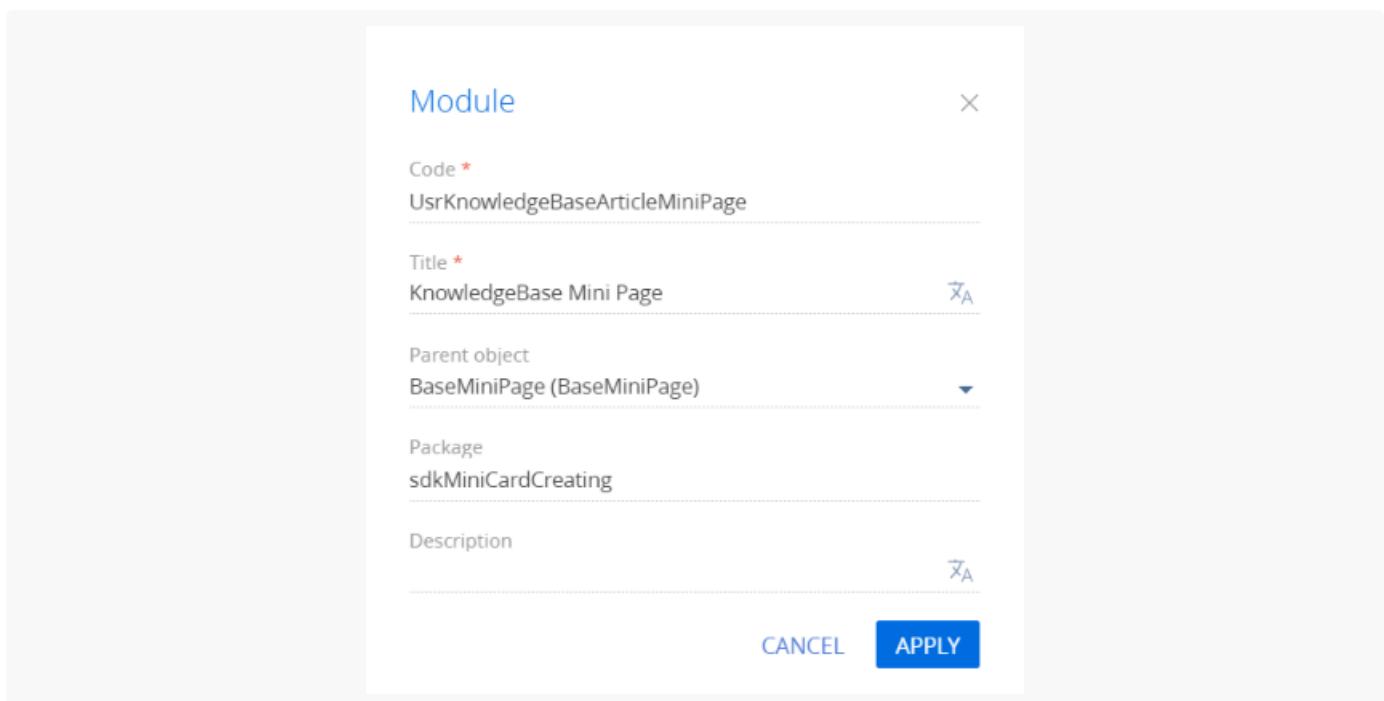
1. [Перейдите в раздел \[Конфигурация\]](#) ([Configuration]) и выберите пользовательский [пакет](#), в который будет добавлена схема.

2. На панели инструментов реестра раздела нажмите [Добавить] —> [Модель представления страницы] ([Add] —> [Page view model]).



3. В дизайнере схем заполните свойства схемы:

- [Код] ([Code]) — "UsrKnowledgeBaseArticleMiniPage".
- [Заголовок] ([Title]) — "Миникарточка базы знаний" ("KnowledgeBase Mini Page").
- [Родительский объект] ([Parent object]) — выберите "BaseMiniPage".



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

## 2. Отобразить поля основного объекта

В дизайнере схем добавьте необходимый исходный код.

1. В качестве схемы объекта укажите схему `KnowledgeBase`.
2. Добавьте необходимые модификации в массив модификаций модели представления `diff`.

## Элементы модели представления базовой мини-карточки:

- `MiniPage` — поле карточки.
- `HeaderContainer` — заголовок карточки (по умолчанию размещается в первом ряду поля карточки).

В примере в массив модификаций `diff` добавлены два объекта, которые конфигурируют поля [ *Name* ] и [ *Keywords* ].

Исходный код схемы модели представления приведен ниже.

### UsrKnowledgeBaseArticleMiniPage.js

```
define("UsrKnowledgeBaseArticleMiniPage", [], function() {
    return {
        entitySchemaName: "KnowledgeBase",
        attributes: {
            "MiniPageModes": {
                "value": [this.Terrasoft.ConfigurationEnums.CardOperation.VIEW]
            }
        },
        diff: /**SCHEMA_DIFF*/[
            {
                "operation": "insert",
                "name": "Name",
                "parentName": "HeaderContainer",
                "propertyName": "items",
                "index": 0,
                "values": {
                    "labelConfig": {
                        "visible": false
                    },
                    "isMiniPageModelItem": true
                }
            },
            {
                "operation": "insert",
                "name": "Keywords",
                "parentName": "MiniPage",
                "propertyName": "items",
                "values": {
                    "labelConfig": {
                        "visible": false
                    },
                    "isMiniPageModelItem": true,
                    "layout": {
                        "column": 0,
                        "row": 1,
                        "colSpan": 24
                    }
                }
            }
        ]
    }
});
```

```

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

});

```

### 3. Добавить функциональную кнопку в мини-карточку

По условию примера карточка должна обеспечивать скачивание файлов, связанных со статьей базы знаний.

Работа с дополнительными данными обеспечивается с помощью механизма их отображения в виде выпадающего списка преднастроенной кнопки.

В дизайнере схем измените исходный код модели представления.

Для **добавления кнопки** выбора файлов статьи базы знаний:

- Добавьте в массив `diff` элемент `FilesButton`, который является описанием кнопки
- Добавьте в свойство `attributes` виртуальную колонку `Article`, которая связывает основную и дополнительные записи.
- Добавьте в свойство `attributes` трибут `MiniPageModes` — массив, который содержит коллекцию необходимых операций, выполняемых мини-карточкой.
- Добавьте в ресурсы схемы изображение кнопки. Например, можно использовать это изображение — . Добавление изображения в ресурсы описано в статье [Добавить поле с изображением](#).
- Добавьте в свойство `methods` методы работы с выпадающим списком кнопки выбора файла:
  - `init()` — переопределенный базовый метод.
  - `onEntityInitialized()` — переопределенный базовый метод.
  - `setArticleInfo()` — устанавливает значение атрибута `Article`.
  - `getFiles(callback, scope)` — получает информацию о файлах текущей статьи базы знаний.
  - `initFilesMenu(files)` — наполняет коллекцию выпадающего списка кнопки выбора файлов.
  - `fillFilesExtendedMenuData()` — инициирует загрузку файлов и их добавление в выпадающий список кнопки выбора файлов.
  - `downloadFile()` — инициирует скачивание выбранного файла.

Исходный код схемы модели представления, который добавляет функциональную кнопку, приведен ниже.

#### UserKnowledgeBaseArticleMiniPage.js

```

define("UserKnowledgeBaseArticleMiniPage",
["terrasoft", "KnowledgeBaseFile", "ConfigurationConstants"],
function(Terrasoft, KnowledgeBaseFile, ConfigurationConstants) {

```

```

return {
    entitySchemaName: "KnowledgeBase",
    attributes: {
        "MiniPageModes": {
            "value": [this.Terrasoft.ConfigurationEnums.CardOperation.VIEW]
        },
        "Article": {
            "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
            "referenceSchemaName": "KnowledgeBase"
        }
    },
    methods: {
        /* Инициализирует коллекцию выпадающего списка кнопки выбора файлов.*/
        init: function() {
            this.callParent(arguments);
            this.initExtendedMenuButtonCollections("File", ["Article"], this.close);
        },
        /* Инициализирует значение атрибута, связывающего основную и дополнительные записи.
Наполняет коллекцию выпадающего списка кнопки выбора файлов.*/
        onEntityInitialized: function() {
            this.callParent(arguments);
            this.setArticleInfo();
            this.fillFilesExtendedMenuData();
        },
        /* Инициирует загрузку файлов и их добавление в выпадающий список кнопки выбора*/
        fillFilesExtendedMenuData: function() {
            this.getFiles(this.initFilesMenu, this);
        },
        /* Устанавливает значение атрибута, связывающего основную и дополнительные записи*/
        setArticleInfo: function() {
            this.set("Article", {
                value: this.get(this.primaryColumnName),
                displayValue: this.get(this.primaryDisplayColumnName)
            });
        },
        /* Получает информацию о файлах текущей статьи базы знаний.*/
        getFiles: function(callback, scope) {
            var esq = this.Ext.create("Terrasoft.EntitySchemaQuery", {
                rootSchema: KnowledgeBaseFile
            });
            esq.addColumn("Name");
            var articleFilter = >this.Terrasoft.createColumnFilterWithParameter(
                this.Terrasoft.ComparisonType.EQUAL, "KnowledgeBase", this.get(this.primaryColumnName));
            var typeFilter = this.Terrasoft.createColumnFilterWithParameter(
                this.Terrasoft.ComparisonType.EQUAL, "Type", ConfigurationConstants.FileType);
            esq.filters.addItem(articleFilter);
            esq.filters.addItem(typeFilter);
            esq.getEntityCollection(function(response) {
                if (!response.success) {

```

```

        return;
    }
    callback.call(scope, response.collection);
}, this);
},
/* Наполняет коллекцию выпадающего списка кнопки выбора файлов.*/
initFilesMenu: function(files) {
    if (files.isEmpty()) {
        return;
    }
    var data = [];
    files.each(function(file) {
        data.push({
            caption: file.get("Name"),
            tag: file.get("Id")
        });
    }, this);
    var recipientInfo = this.fillExtendedMenuItems("File", ["Article"]);
    this.fillExtendedMenuData(data, recipientInfo, this.downloadFile);
},
/* Инициирует скачивание выбранного файла.*/
downloadFile: function(id) {
    var element = document.createElement("a");
    element.href = "../rest/FileService/GetFile/" + KnowledgeBaseFile.uId + "/";
    document.body.appendChild(element);
    element.click();
    document.body.removeChild(element);
}
},
diff: /**SCHEMA_DIFF*/[
{
    "operation": "insert",
    "name": "Name",
    "parentName": "HeaderContainer",
    "propertyName": "items",
    "index": 0,
    "values": {
        "labelConfig": {
            "visible": true
        },
        "isMiniPageModelItem": true
    }
},
{
    "operation": "insert",
    "name": "Keywords",
    "parentName": "MiniPage",
    "propertyName": "items",
}
]
}

```

```

    "values": {
        "labelConfig": {
            "visible": true
        },
        "isMiniPageModelItem": true,
        "layout": {
            "column": 0,
            "row": 1,
            "colSpan": 24
        }
    }
},
{
    "operation": "insert",
    "parentName": "HeaderContainer",
    "propertyName": "items",
    "name": "FilesButton",
    "values": {
        "itemType": Terrasoft.ViewItemType.BUTTON,
        /* Настройка изображения кнопки.*/
        "imageConfig": {
            /* Изображение предварительно необходимо добавить в ресурсы мини-карточки.*/
            "bindTo": "Resources.Images.FilesImage"
        },
        /* Настройка выпадающего списка.*/
        "extendedMenu": {
            /* Название элемента выпадающего списка.*/
            "Name": "File",
            /* Название атрибута миникарточки, связывающего основную и дополнительную информацию.*/
            "PropertyName": "Article",
            /* Настройка обработчика нажатия на кнопку.*/
            "Click": {
                "bindTo": "fillFilesExtendedMenuData"
            }
        }
    },
    "index": 1
}
]/**SCHEMA_DIFF*/
};

});
});

```

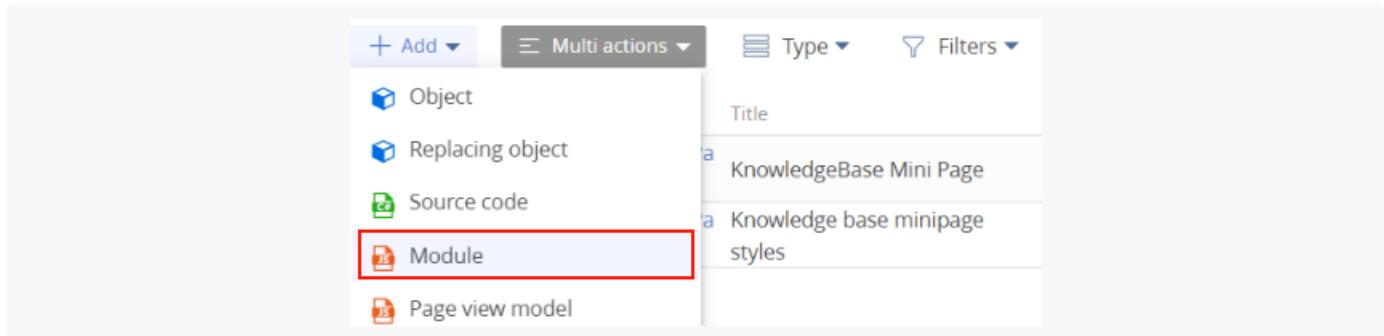
## 4. Выполнить стилизацию мини-карточки

Для добавления стилей в модель представления необходимо создать отдельный модуль со стилями и подключить этот модуль к схеме модели представления.

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

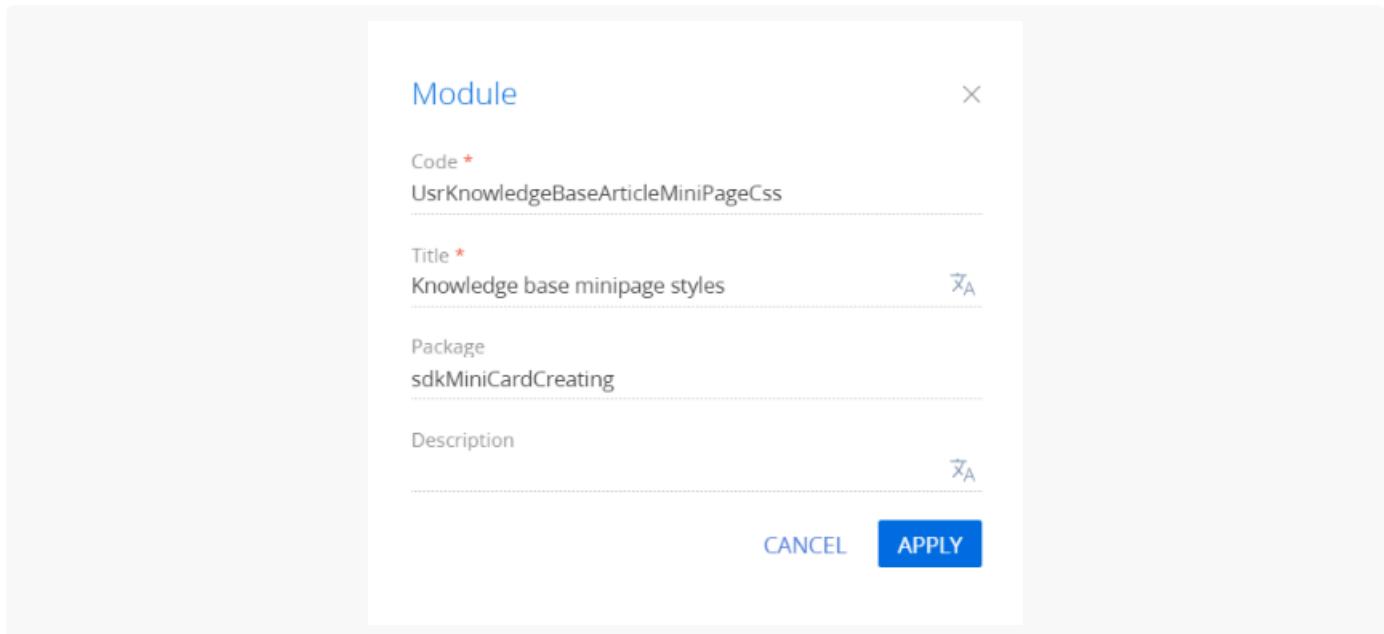
будет добавлена схема.

- На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Модуль ] ([ Add ] —> [ Module ]).



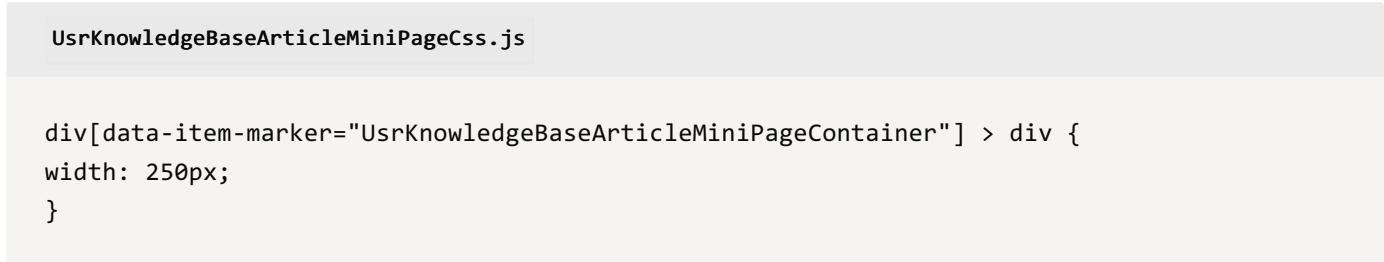
- В дизайнере схем заполните свойства схемы:

- [ Код ] ([ Code ]) — "UsrKnowledgeBaseArticleMiniPageCss".
- [ Заголовок ] ([ Title ]) — "Стили миникарточки базы знаний" ("Knowledge base minipage styles").



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

- В контекстном меню узла LESS укажите необходимые стили.



- В дизайнере схем измените код схемы модели представления: добавьте загрузку этого модуля в исходном коде.

Ниже приведен полный исходный код мини-карточки:

```
UserKnowledgeBaseArticleMiniPage.js

define("UserKnowledgeBaseArticleMiniPage",
["terrasoft", "KnowledgeBaseFile", "ConfigurationConstants", "css!UserKnowledgeBaseArticleMiniPage"],
function(Terrasoft, KnowledgeBaseFile, ConfigurationConstants) {
    return {
        entitySchemaName: "KnowledgeBase",
        attributes: {
            "MiniPageModes": {
                "value": [this.Terrasoft.ConfigurationEnums.CardOperation.VIEW]
            },
            "Article": {
                "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
                "referenceSchemaName": "KnowledgeBase"
            }
        },
        methods: {
            init: function() {
                this.callParent(arguments);
                this.initExtendedMenuButtonCollections("File", ["Article"], this.close);
            },
            onEntityInitialized: function() {
                this.callParent(arguments);
                this.setArticleInfo();
                this.fillFilesExtendedMenuData();
            },
            fillFilesExtendedMenuData: function() {
                this.getFiles(this.initFilesMenu, this);
            },
            setArticleInfo: function() {
                this.set("Article", {
                    value: this.get(this.primaryColumnName),
                    displayValue: this.get(this.primaryDisplayColumnName)
                });
            },
            getFiles: function(callback, scope) {
                var esq = this.Ext.create("Terrasoft.EntitySchemaQuery", {
                    rootSchema: KnowledgeBaseFile
                });
                esq.addColumn("Name");
                var articleFilter = this.Terrasoft.createColumnFilterWithParameter(
                    this.Terrasoft.ComparisonType.EQUAL, "KnowledgeBase", this.get(this.primaryColumnName));
                var typeFilter = this.Terrasoft.createColumnFilterWithParameter(
                    this.Terrasoft.ComparisonType.EQUAL, "Type", ConfigurationConstants.FILTER_TYPE_ARTICLE);
                esq.filters.addItem(articleFilter);
                esq.filters.addItem(typeFilter);
            }
        }
    };
});
```

```

        esq.getEntityCollection(function(response) {
            if (!response.success) {
                return;
            }
            callback.call(scope, response.collection);
        }, this);
    },
    initFilesMenu: function(files) {
        if (files.isEmpty()) {
            return;
        }
        var data = [];
        files.each(function(file) {
            data.push({
                caption: file.get("Name"),
                tag: file.get("Id")
            });
        }, this);
        var recipientInfo = this.fillExtendedMenuItems("File", ["Article"]);
        this.fillExtendedMenuData(data, recipientInfo, this.downloadFile);
    },
    downloadFile: function(id) {
        var element = document.createElement("a");
        element.href = "../rest/FileService/GetFile/" + KnowledgeBaseFile.uId + "document.body.appendChild(element)";
        element.click();
        document.body.removeChild(element);
    }
},
diff: /**SCHEMA_DIFF*/[
{
    "operation": "insert",
    "name": "Name",
    "parentName": "HeaderContainer",
    "propertyName": "items",
    "index": 0,
    "values": {
        "labelConfig": {
            "visible": true
        },
        "isMiniPageModelItem": true
    }
},
{
    "operation": "insert",
    "name": "Keywords",
    "parentName": "MiniPage",
    "propertyName": "items",
    "values": {

```

```

        "labelConfig": {
            "visible": true
        },
        "isMiniPageModelItem": true,
        "layout": {
            "column": 0,
            "row": 1,
            "colSpan": 24
        }
    }
},
{
    "operation": "insert",
    "parentName": "HeaderContainer",
    "propertyName": "items",
    "name": "FilesButton",
    "values": {
        "itemType": Terrasoft.ViewItemType.BUTTON,
        "imageConfig": {
            "bindTo": "Resources.Images.FilesImage"
        },
        "extendedMenu": {
            "Name": "File",
            "PropertyName": "Article",
            "Click": {
                "bindTo": "fillFilesExtendedMenuData"
            }
        }
    },
    "index": 1
}
]
/**SCHEMA_DIFF*/
);
});

```

## 5. Зарегистрировать мини-карточку в базе данных

Создание мини-карточки предполагает ее обязательную регистрацию в базе данных. Для внесения изменений в базу данных выполните следующий SQL-запрос.

### Запрос на создание мини-карточки

```

DECLARE
    -- Название схемы представления создаваемой мини-карточки.
    @ClientUnitSchemaName NVARCHAR(100) = 'UsrKnowledgeBaseArticleMiniPage',
    -- Название схемы объекта, к которому привязывается мини-карточка.
    @EntitySchemaName NVARCHAR(100) = 'KnowledgeBase'

```

```

UPDATE SysModuleEdit
SET MiniPageSchemaUID = (
    SELECT TOP 1 UID
    FROM SysSchema
    WHERE Name = @ClientUnitSchemaName
)
WHERE SysModuleEntityId = (
    SELECT TOP 1 Id
    FROM SysModuleEntity
    WHERE SysEntitySchemaUID = (
        SELECT TOP 1 UID
        FROM SysSchema
        WHERE Name = @EntitySchemaName
        AND ExtendParent = 0
    )
);

```

В результате выполнения запроса уникальный идентификатор мини-карточки будет добавлен в таблицу [SysModuleEdit] в поле [MiniPageSchemaUID] записи, соответствующей разделу [База знаний] ([Knowledge base]).

	CardSchemaUId	ActionKindCaption	ActionKindName	PageCaption	MiniPageSchemaUId
50	9DBD0611-FA52-4A90-9542-E5FD997B4AFD	New article	KnowledgeBase	Knowledge base article	9C072BC6-93C7-488A-87A6-A41B79CC67...

## 6. Добавить системную настройку

В разделе [Системные настройки] ([System settings]) дизайнера системы добавьте системную настройку со следующими свойствами:

- [Название] ([Name]) — "HasKnowledgeBaseMiniPageAddMode".
- [Код] ([Code]) — "HasKnowledgeBaseMiniPageAddMode".
- [Тип] ([Type]) — "Логическое" ("Boolean").
- [Значение по умолчанию] ([Default value]) — признак установлен.

HasKnowledgeBaseMiniPageAddMode

What can I do for you? >

**SAVE** **CANCEL**

Name*	HasKnowledgeBaseMiniPageAddMode	Code*	HasKnowledgeBaseMiniPageAddMode
Type*	Boolean	Cached	<input checked="" type="checkbox"/>
Default value	<input checked="" type="checkbox"/>	Personal	<input type="checkbox"/>
		Allow for portal users	<input type="checkbox"/>
Description			

## Результат выполнения примера

После сохранения схемы и обновления веб-страницы приложения в разделе [ База знаний ] ([ Knowledge base ]) при наведении курсора на название будет отображаться пользовательская мини-карточка, в которой будут отображены связанные с записью файлы и реализована возможность скачать их.

New Presentation Style

Name\* New Presentation Style

Author John Best

Tags corporate style, presentation, new

Presentation.pptx

## Создать мини-карточку добавления

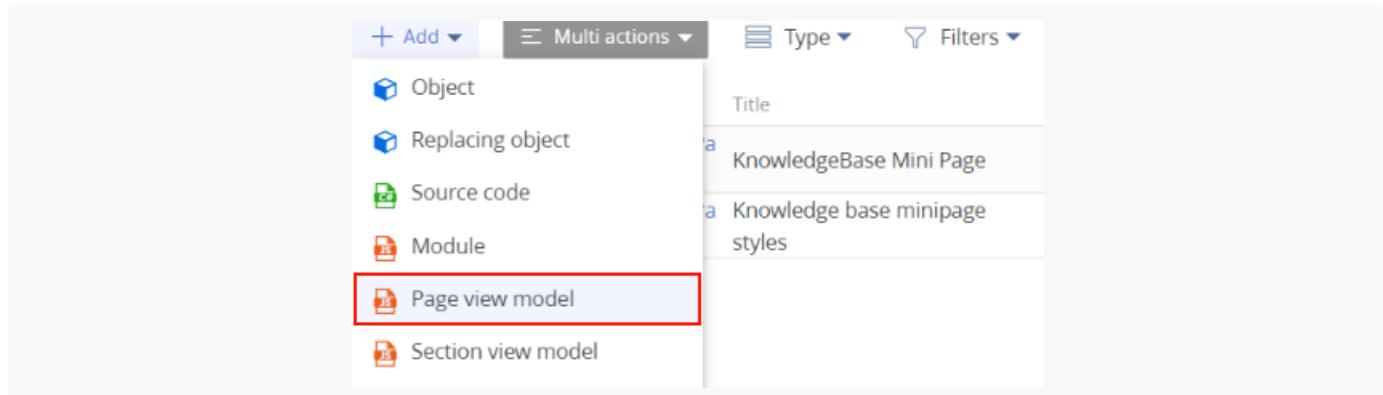


**Пример.** Создать пользовательскую мини-карточку добавления новой записи в раздел [ Продукты ] ([ Products ]). Мини-карточка должна добавлять базовый набор полей [ Название ] ([ Name ]) и [ Код ] ([ Code ]).

### 1. Создать схему модели представления мини-карточки

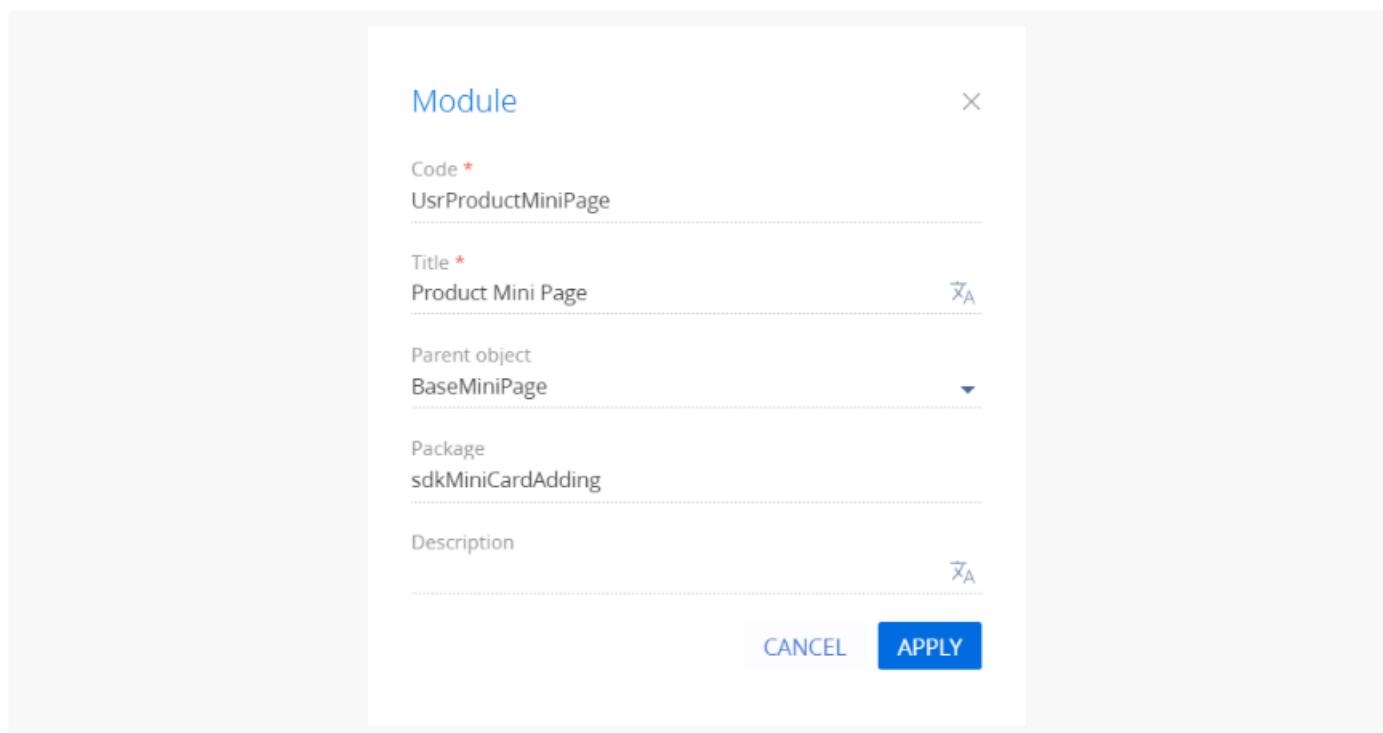
- Перейдите в раздел [ Конфигурация ] ([ Configuration ]) и выберите пользовательский пакет, в который будет добавлена схема.
- На панели инструментов реестра раздела нажмите [ Добавить ] —> [ Модель представления страницы ]

] ([ Add ] —> [ Page view model ]).



3. В дизайнере схем заполните свойства схемы:

- [ Код ] ([ Code ]) — "UsrProductMiniPage".
- [ Заголовок ] ([ Title ]) — "Мини-карточка продукта" ("Product Mini Page").
- [ Родительский объект ] ([ Parent object ]) — выберите "BaseMiniPage".



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

## 2. Отобразить поля основного объекта

В дизайнере схем добавьте необходимый исходный код.

1. В качестве схемы объекта укажите схему `Product`.
2. Объявите атрибут `MiniPageModes` и присвойте ему массив, содержащий коллекцию необходимых

операций, выполняемых мини-карточкой.

**На заметку.** Если кроме операции добавления новой записи требуется отображение мини-карточки на странице раздела (см. [Создать пользовательскую мини-карточку](#)), то в массив, присваиваемый атрибуту `MiniPageModes`, также необходимо добавить значение `this.Terrasoft.ConfigurationEnums.CardOperation.VIEW`.

3. Добавьте необходимые модификации в массив модификаций `diff` модели представления .

**Элементы модели представления** базовой мини-карточки:

- `MiniPage` — поле карточки.
- `HeaderContainer` — заголовок карточки (по умолчанию размещается в первом ряду поля карточки).

В примере в массив модификаций `diff` добавлены два объекта, которые конфигурируют поля `[ Name ]` и `[ Code ]`.

Исходный код схемы модели представления приведен ниже.

#### UsrProductMiniPage.js — отобразить поля объекта

```
define("UsrProductMiniPage", ["UsrProductMiniPageResources"],
    function(resources) {
        return {
            entitySchemaName: "Product",
            details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
            attributes: {
                "MiniPageModes": {
                    "value": [this.Terrasoft.ConfigurationEnums.CardOperation.ADD]
                }
            },
            diff: /**SCHEMA_DIFF*/[
                {
                    "operation": "insert",
                    "parentName": "MiniPage",
                    "propertyName": "items",
                    "name": "Name",
                    "values": {
                        "isMiniPageModelItem": true,
                        "layout": {
                            "column": 0,
                            "row": 1,
                            "colSpan": 24
                        },
                        "controlConfig": {
                            "focused": true
                        }
                    }
                }
            ]
        }
    }
);
```

```

},
{
    "operation": "insert",
    "parentName": "MiniPage",
    "propertyName": "items",
    "name": "Code",
    "values": {
        "isMiniPageModelItem": true,
        "layout": {
            "column": 0,
            "row": 2,
            "colSpan": 24
        }
    }
}
]/**SCHEMA_DIFF*/
);
});
});
```

### 3. Зарегистрировать мини-карточку в базе данных

Создание мини-карточки предполагает ее обязательную регистрацию в базе данных. Для внесения изменений в базу данных выполните следующий SQL-запрос.

#### Запрос на создание мини-карточки

```

DECLARE
    -- Название схемы представления создаваемой мини-карточки.
    @ClientUnitSchemaName NVARCHAR(100) = 'UsrProductMiniPage',
    -- Название схемы объекта, к которому привязывается мини-карточка.
    @EntitySchemaName NVARCHAR(100) = 'Product'

UPDATE SysModuleEdit
SET MiniPageSchemaUID = (
    SELECT TOP 1 UID
    FROM SysSchema
    WHERE Name = @ClientUnitSchemaName
)
WHERE SysModuleEntityId = (
    SELECT TOP 1 Id
    FROM SysModuleEntity
    WHERE SysEntitySchemaUID = (
        SELECT TOP 1 UID
        FROM SysSchema
        WHERE Name = @EntitySchemaName
        AND ExtendParent = 0
    )
)
```

```
);
```

В результате выполнения запроса уникальный идентификатор мини-карточки будет добавлен в таблицу `[SysModuleEdit]` в поле `[MiniPageSchemaUid]` записи, соответствующей разделу [Продукты] ([Products]).

CardSchemaUid	ActionKindCaption	ActionKindName	PageCaption	MiniPageSchemaUid	SearchRowSchemaUid
0DAEC87E-A84D-44BC-9DD0-C90B8D1BAA33	New product	Product	Product	15D85C82-D78F-400F-B10C-44BB13C72282	NULL

## 4. Добавить системную настройку

В разделе [Системные настройки] ([System settings]) дизайнера системы добавьте системную настройку со следующими свойствами:

- [Название] ([Name]) — "HasProductMiniPageAddMode".
- [Код] ([Code]) — "HasProductMiniPageAddMode".
- [Тип] ([Type]) — "Логическое" ("Boolean").
- [Значение по умолчанию] ([Default value]) — признак установлен.

The screenshot shows the 'HasProductMiniPageAddM...' configuration page in the Creatio system settings. The page has a header with 'What can I do for you?' and a 'Creatio' logo. Below the header, there are 'SAVE' and 'CANCEL' buttons. The main area contains two sets of configuration fields. The first set on the left includes 'Name\*' (HasProductMiniPageAddMode), 'Type\*' (Boolean), and 'Default value' (checkbox checked). The second set on the right includes 'Code\*' (HasProductMiniPageAddMode), 'Cached' (checkbox checked), 'Personal' (checkbox unchecked), and 'Allow for portal users' (checkbox unchecked). There is also a 'Description' section at the bottom.

## Результат выполнения примера

В результате выполнения примера при добавлении нового продукта будет отображаться мини-карточка с двумя полями.

The screenshot shows the 'Products' module interface. A modal window is open for creating a new product. The modal title is 'Product'. It contains fields for 'Name\*' (with placeholder 'My Product') and 'Code' (with value 'Code12345'). At the bottom of the modal are two buttons: 'SAVE' (in blue) and 'CANCEL'.

После сохранения мини-карточки соответствующая запись появится в реестре раздела.

The screenshot shows the 'Products' module interface after saving a new product. The list now includes two items: 'My Product' and 'Motherboard UT165LZ-32P1 (sample)'. The 'My Product' entry has a price of 0.00 and USD currency. The 'Motherboard UT165LZ-32P1 (sample)' entry has a price of 900.00 and USD currency. The sidebar on the left shows navigation links for Sales, Dashboards, Feed, Leads, Accounts, Contacts, and Activities. The vertical toolbar on the right provides quick access to various functions.

**Важно.** Запись в реестре раздела будет отображена только после обновления страницы браузера. Чтобы запись отображалась сразу же после сохранения мини-карточки, необходимо добавить соответствующую функциональность в схему мини-карточки и страницы раздела, используя механизм сообщений (см. статью [Sandbox](#)).

## Добавить мини-карточку к произвольному модулю



**Пример.** Отобразить текущего пользователя в правом верхнем углу приложения возле иконки с профилем пользователя. При наведении на ссылку текущего пользователя системы открыть мини-карточку.

## 1. Создать схему модуля

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

- В дизайнере схем заполните свойства схемы:

- [[Код](#)] ([ Code ]) — "UsrCurrentUserModule".
- [[Заголовок](#)] ([ Title ]) — "Модуль "Текущий пользователь" ("Current user module").

Code *	UsrCurrentUserModule
Title *	Current user module
Package	sdkMiniPageAnyModule
Description	
<a href="#">CANCEL</a> <a href="#">APPLY</a>	

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

## 2. Создать представление и модель представления модуля

В дизайнере схем добавьте в модуль `UsrCurrentUserModule` необходимый исходный код.

1. Для создания модели представления реализуйте класс, унаследованный от `Terrasoft.BaseViewModel`.
2. Подключите утилитный класс `Terrasoft.MiniPageUtilities` в свойство `mixins` модели представления. Класс позволит использовать методы вызова мини-карточки.
3. Для создания представления реализуйте класс, унаследованный от `Terrasoft.BaseModule`.
4. В классе переопределите методы базового класса `Terrasoft.BaseModule`:
  - `init()` — инициализирует модель представления модуля.
  - `render()` — связывает модель представления с отображением представления в контейнере, передаваемом в параметре `renderTo`.
  - `getViewModel()` — используется для создания модели представления.
  - `getView()` — используется для получения представления для его дальнейшего отображения. Представление должно отображать ФИО текущего пользователя с гиперссылкой на страницу контакта. При построении гиперссылки определите обработчик события наведения курсора мыши.
5. Определите свойство `viewModel` — используется для хранения ссылки на полученную модель представления.

Исходный код модуля приведен ниже.

### UsrCurrentUserModule.js

```
/* Определение модуля. */
define("UsrCurrentUserModule", ["MiniPageUtilities"], function() {
    /* Определение класса CurrentUserViewModel. */
    Ext.define("Terrasoft.configuration.CurrentUserViewModel", {
        /* Имя родительского класса. */
        extend: "Terrasoft.BaseViewModel",
        /* Сокращенное название класса. */
        alternateClassName: "Terrasoft.CurrentUserViewModel",
        /* Используемые миксины. */
        mixins: {
            MiniPageUtilitiesMixin: "Terrasoft.MiniPageUtilities"
        }
    });
    /* Определение класса UsrCurrentUserModule. */
    Ext.define("Terrasoft.configuration.UsrCurrentUserModule", {
        /* Сокращенное название класса. */
        alternateClassName: "Terrasoft.UsrCurrentUserModule",
        /* Имя родительского класса. */
        extend: "Terrasoft.BaseModule",
        /* Объект Ext. */
    })
});
```

```

Ext: null,
/* Объект sandbox. */
sandbox: null,
/* Объект Terrasoft. */
Terrasoft: null,
/* Модель представления. */
viewModel: null,
/* Создает представления модуля. */
getView: function() {
    /* Получение контакта текущего пользователя. */
    var currentUser = Terrasoft.SysValue.CURRENT_USER_CONTACT;
    /* Представление – экземпляр класса Terrasoft.Hyperlink. */
    return Ext.create("Terrasoft.Hyperlink", {
        /* Заполнение заголовка ссылки именем контакта. */
        "caption": currentUser.displayValue,
        /* Обработчик события наведения на ссылку. */
        "linkMouseOver": {"bindTo": "linkMouseOver"},
        /* Свойство, содержащее дополнительные параметры объекта. */
        "tag": {
            /* Идентификатор текущего пользователя. */
            "recordId": currentUser.value,
            /* Название схемы объекта. */
            "referenceSchemaName": "Contact"
        }
    });
},
/* Создает модель представления модуля. */
getViewModel: function() {
    return Ext.create("Terrasoft.CurrentUserViewModel");
},
/* Инициализация модуля. */
init: function() {
    this.viewModel = this.getViewModel();
},
/* Отображает представление модуля. */
render: function(renderTo) {
    /* Получение объекта представления. */
    var view = this.getView();
    /* Связывание представления с моделью представления. */
    view.bind(this.viewModel);
    /* Отображение представления в элементе renderTo. */
    view.render(renderTo);
}
});
return Terrasoft.UsrCurrentUserModule;
});

```

## 3. Добавить стили модуля

Для настройки отображения гиперссылки добавьте стили для созданного модуля.

Чтобы **добавить стили модуля**:

1. В дизайнере схем выберите узел [ *LESS* ].
2. Добавьте следующий исходный код.

### Стили модуля

```
.current-user-class a {
    font-weight: bold;
    font-size: 2.0em;
    margin: 6px 20px;
}

.current-user-class a:hover {
    text-decoration: none;
}
```

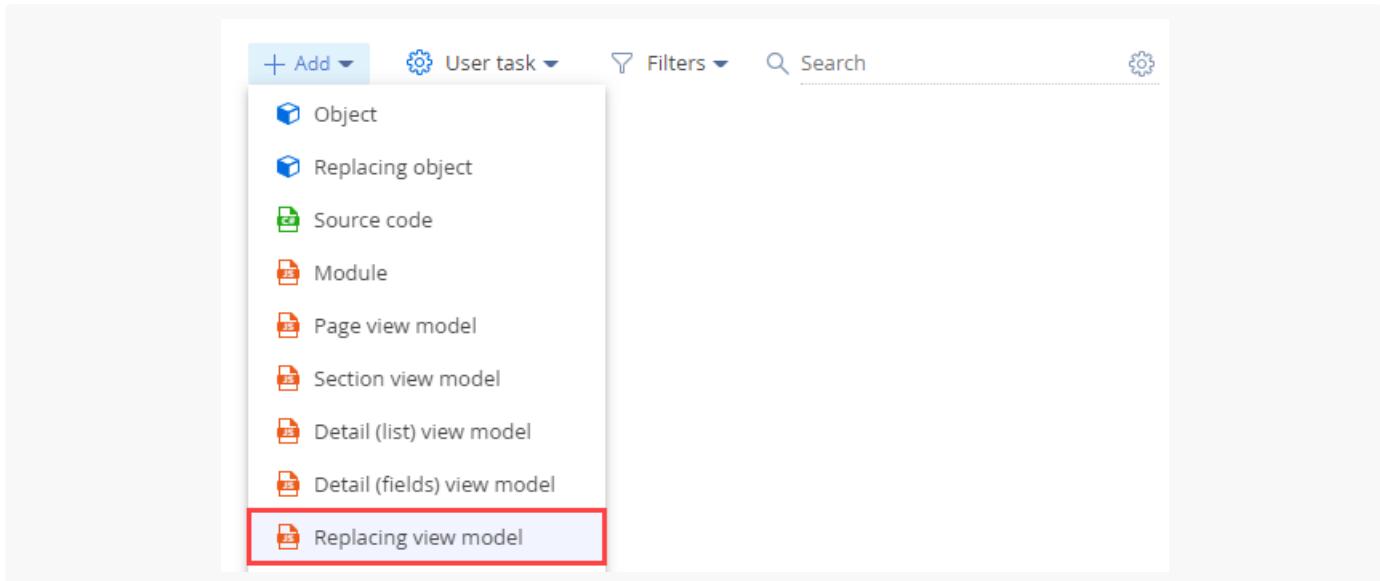
Сохраните созданный модуль.

## 4. Создать контейнер отображения представления

Для вывода ссылки на профиль пользователя в правом верхнем углу приложения необходимо разместить контейнер и загрузить в него представление созданного модуля.

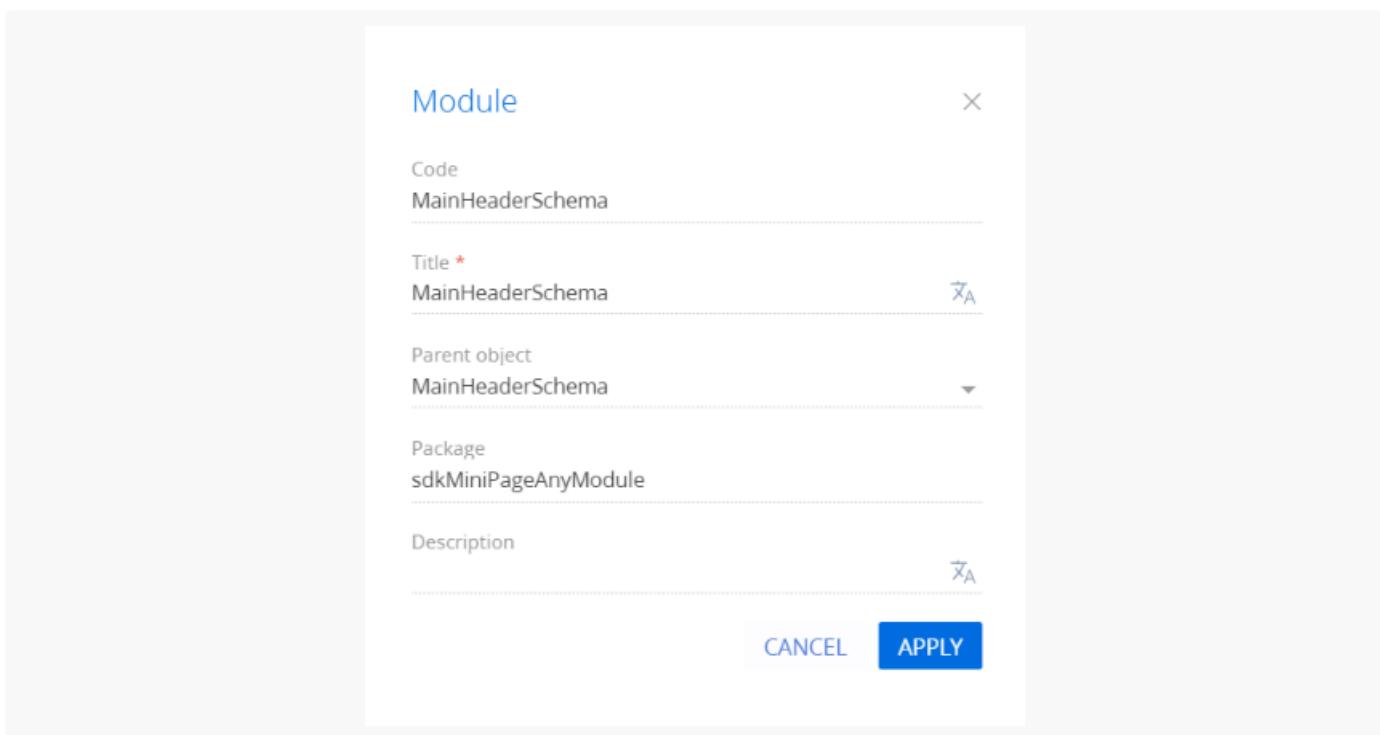
Для этого создайте схему замещающей модели представления, которая расширит функциональность схемы `MainHeaderSchema`.

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. В дизайнере модуля выберите родительский объект `MainHeaderSchema`.

После подтверждения выбранного родительского объекта остальные свойства будут заполнены автоматически.



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

4. В дизайнере модуля добавьте исходный код.

Для отображения представления в исходном коде схемы замещающей модели представления используйте свойство `diff`. Чтобы контейнер отобразился в правом верхнем углу страницы, в качестве родительского элемента создаваемого контейнера установите элемент `RightHeaderContainer`. Далее переопределите метод `onRender()`, в котором выполните загрузку созданного модуля.

Ниже приведен исходный код схемы замещающей модели представления.

**MainHeaderSchema.js**

```

/* Определение модуля. */
define("MainHeaderSchema", [], function() {
    return {
        methods: {
            /* Выполняет действия после отображения представления. */
            onRender: function() {
                /* Вызов родительского метода. */
                this.callParent(arguments);
                /* Загрузка модуля текущего пользователя. */
                this.loadCurrentUserModule();
            },
            /* Загружает модуль текущего пользователя. */
            loadCurrentUserModule: function() {
                /* Получение контейнера, в который будет загружен модуль. */
                var currentUserContainer = this.Ext.getCmp("current-user-container");
                /* Проверка существования контейнера. */
                if (currentUserContainer && currentUserContainer.rendered) {
                    /* Загрузка модуля в контейнер. */
                    this.sandbox.loadModule("UsrCurrentUserModule", {
                        /* Название контейнера. */
                        renderTo: "current-user-container"
                    });
                }
            },
            diff: [
                {
                    /* Операция вставки элемента. */
                    "operation": "insert",
                    /* Название элемента. */
                    "name": "CurrentUserContainer",
                    /* Название родительского контейнера. */
                    "parentName": "RightHeaderContainer",
                    /* Название свойства. */
                    "propertyName": "items",
                    /* Значения элемента. */
                    "values": {
                        /* Идентификатор контейнера. */
                        "id": "current-user-container",
                        /* Тип элемента. */
                        "itemType": Terrasoft.ViewItemType.CONTAINER,
                        /* Классы контейнера. */
                        "wrapClass": ["current-user-class"],
                        /* Элементы контейнера. */
                        "items": []
                    }
                }
            ]
        }
    }
});

```

```

        }
    ]
};

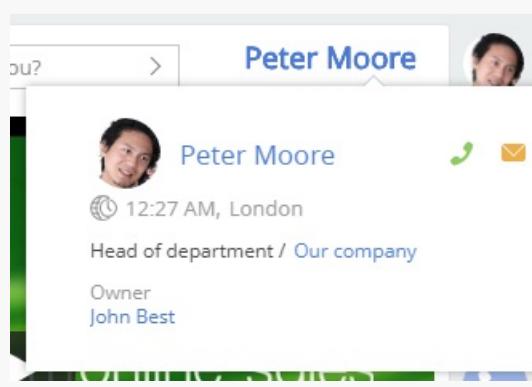
});

```

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

## Результат выполнения примера

После обновления страницы приложения в правом верхнем углу отобразится ФИО текущего пользователя с гиперссылкой на страницу его контакта. При наведении курсора на гиперссылку появится мини-карточка с данными о текущем пользователе.



## 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
<code>Id</code>	Идентификатор записи.
<code>Key</code>	Ключ — название схемы страницы раздела. Например, <code>AccountPageV2</code> , <code>ContactPageV2</code> и т. д.
<code>Data</code>	Настройки хронологии для раздела в формате JSON.

Основные колонки таблицы `TimelineTileSetting`

Колонка	Описание
<code>Id</code>	Идентификатор записи.
<code>Name</code>	Заголовок плитки, который будет отображаться в меню фильтра. Должен быть во множественном числе, например "Задачи" ("Tasks"). Локализация осуществляется с помощью таблицы <code>SysTimelineTileSettingLcz</code> . Если данное поле не будет указано, тогда заголовок плитки будет взят из названия сущности или типа.
<code>Data</code>	Настройки хронологии для раздела в формате JSON.
<code>Image</code>	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	Да	Account

...  
происходить отбор  
записей.

<code>masterRecordColumnName</code>	Название колонки родительской записи, по которой будет происходить отбор записей.	Да	<code>Id</code>
<code>typeColumnName</code>	Название колонки типа.	Нет	<code>Type</code>
<code>typeColumnValue</code>	Значение колонки типа.	Указывается только при указании <code>typeColumnName</code>	<code>fbe0acdc-cfc0-df11-</code>
<code>viewModelClassName</code>	Название класса модели представления преднастроенной плитки.	Нет. Если значение отсутствует, то будет применен базовый класс <code>BaseTimelineItemViewModel</code>	<code>Terrasoft.Activity</code>
<code>viewClassName</code>	Название класса представления преднастроенной плитки.	Нет. Если значение отсутствует, то будет применен базовый класс <code>BaseTimelineItemView</code>	<code>Terrasoft.Activity</code>
<code>orderColumnName</code>	Колонка для сортировки.	Да	<code>StartDate</code>
<code>authorColumnName</code>	Колонка для автора.	Да	<code>Owner</code>
<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	Нет	<code>My Activity</code>

"Задачи" ("Tasks").  
 Используется для задания заголовка плитки, отличного от указанного в поле `Name` настройки соответствующей плитки в `TimelinePageSetting`.

<code>columns</code>	Массив настроек дополнительных колонок для плитки.	Нет	
<code>columnName</code>	Путь к колонке в объекте сущности.	Да	Result
<code>columnAlias</code>	Псевдоним колонки в представлении модели плитки.	Да	ResultMessage
<code>isSearchEnabled</code>	Указывает на возможность текстового поиска по значению в колонке (только для текстовых колонок).	Нет	true

## 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. На экране видна карта хронологии для позиции '201 (sample)', указывающая на дату '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 ]).

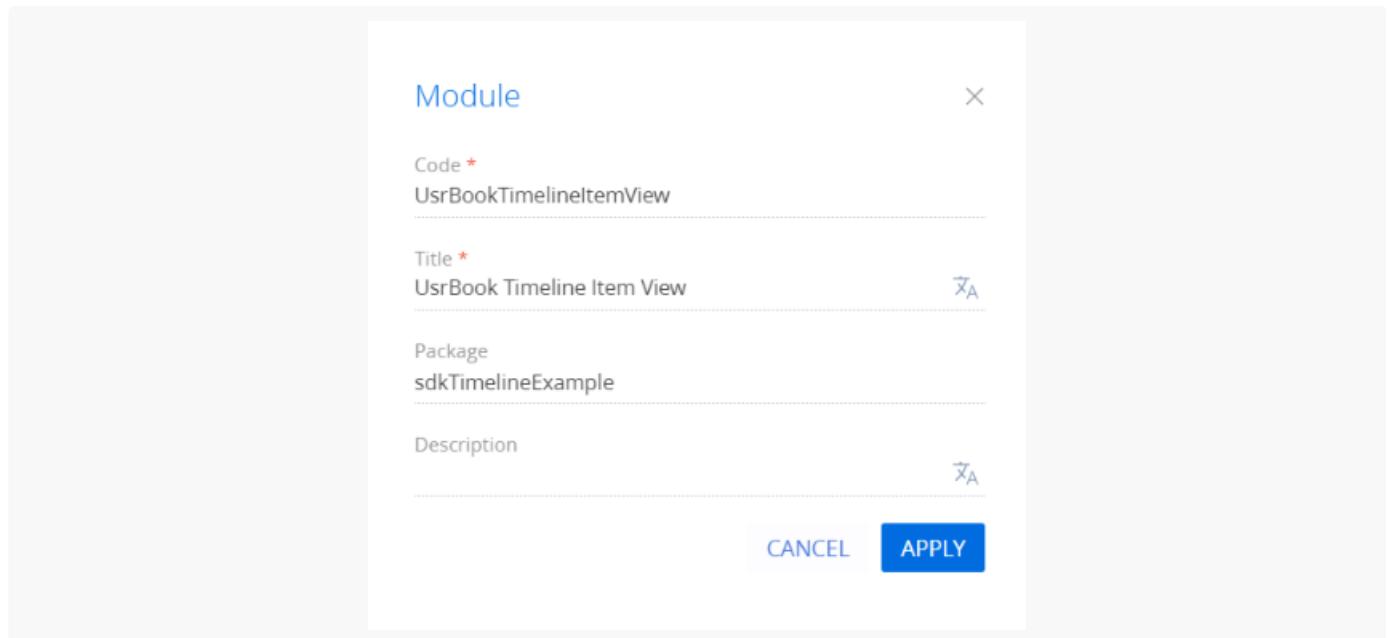
Name	ISBN	Author	Publisher
JavaScript: The Definitive Guide: Activate Your Web Pages	978-0596805524	David Flanagan	Apress
Pro C# 7: With .NET and .NET Core	978-1484230176	Andrew Troelsen	Apress

## 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.

```
UserBookTimelineItemView

/* Определение модуля и его зависимостей.*/
define("UserBookTimelineItemView", ["UserBookTimelineItemViewResources", "BaseTimelineItemView"]
    /* Определение класса представления плитки.*/
    Ext.define("Terrasoft.configuration.UserBookTimelineItemView", {
        extend: "Terrasoft.BaseTimelineItemView",
        alternateClassName: "Terrasoft.UserBookTimelineItemView",
        /* 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;
}
});
});
});

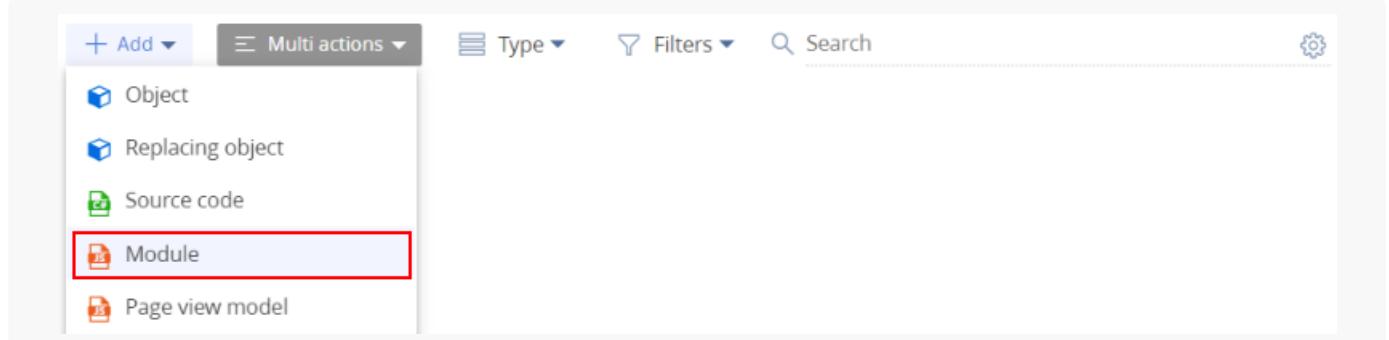
```

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

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

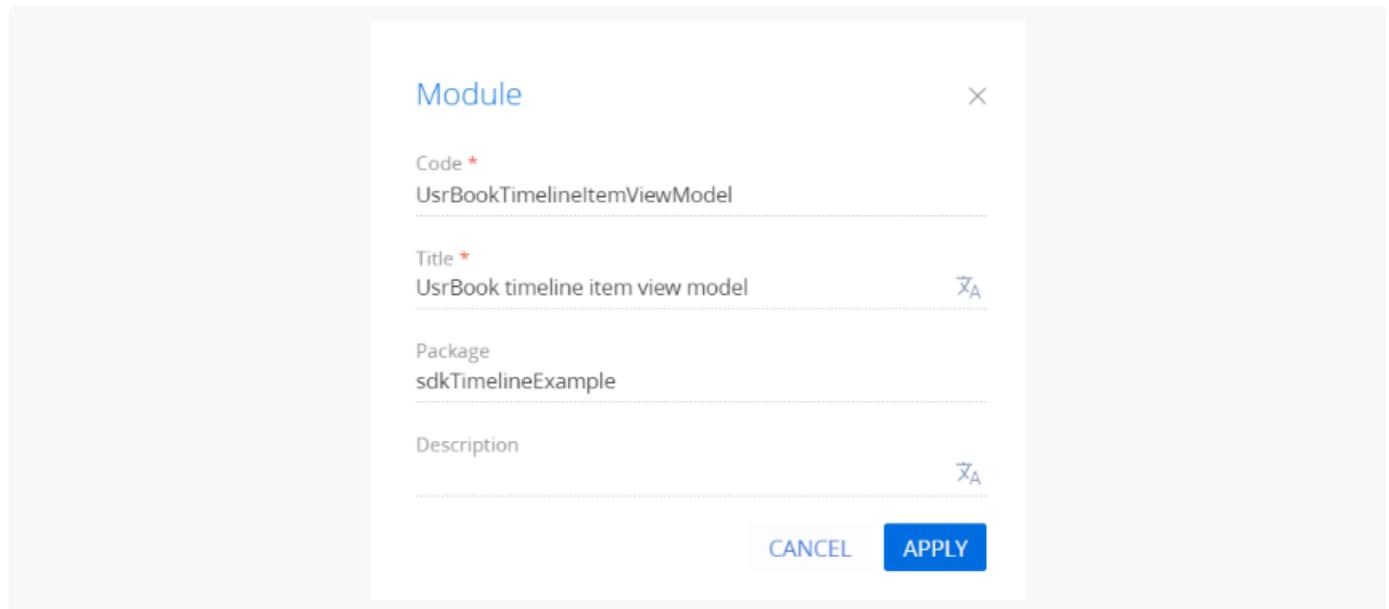
### 3. Создать модуль модели представления плитки

- 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);

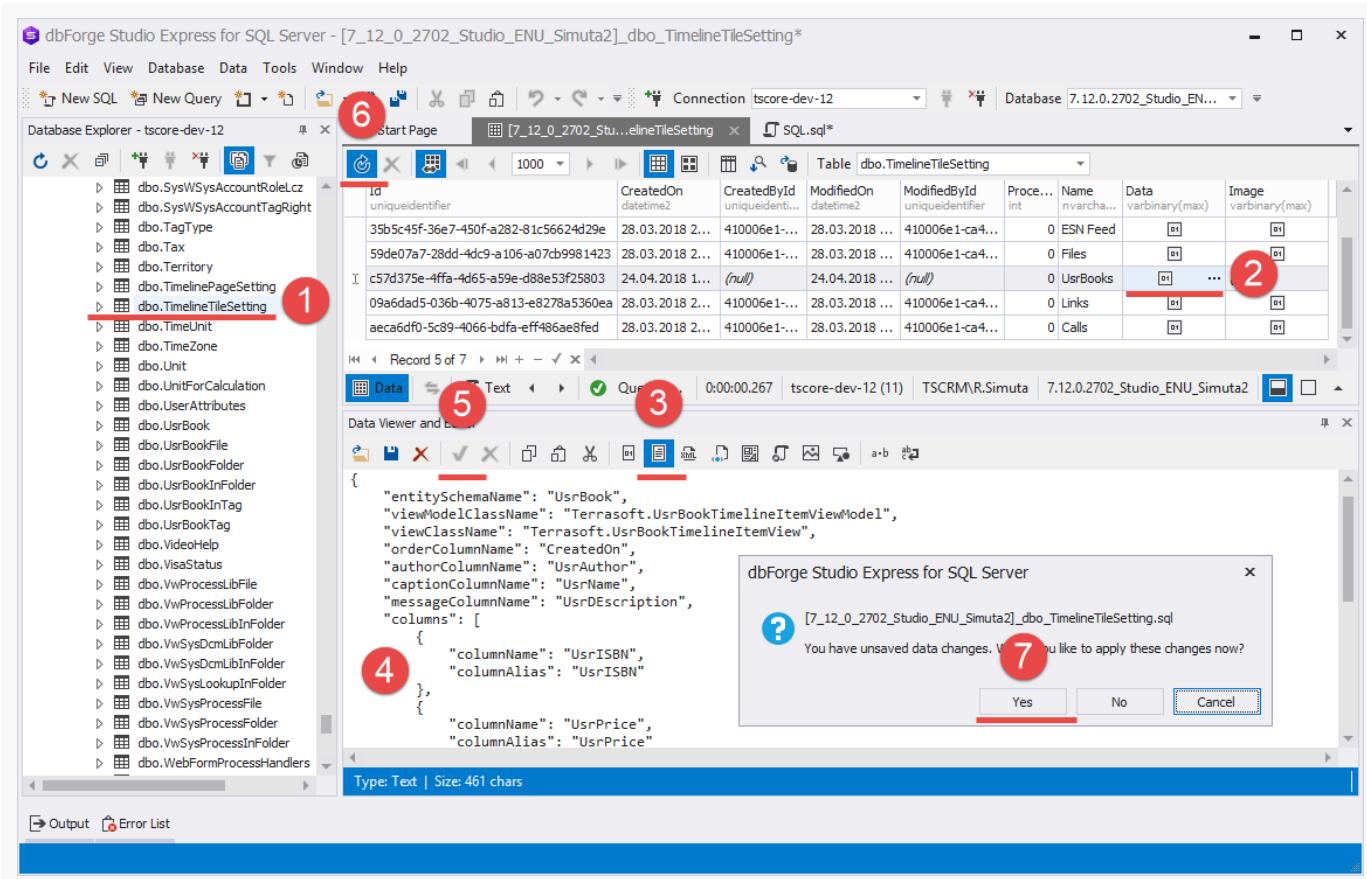
```

- 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` ]

```

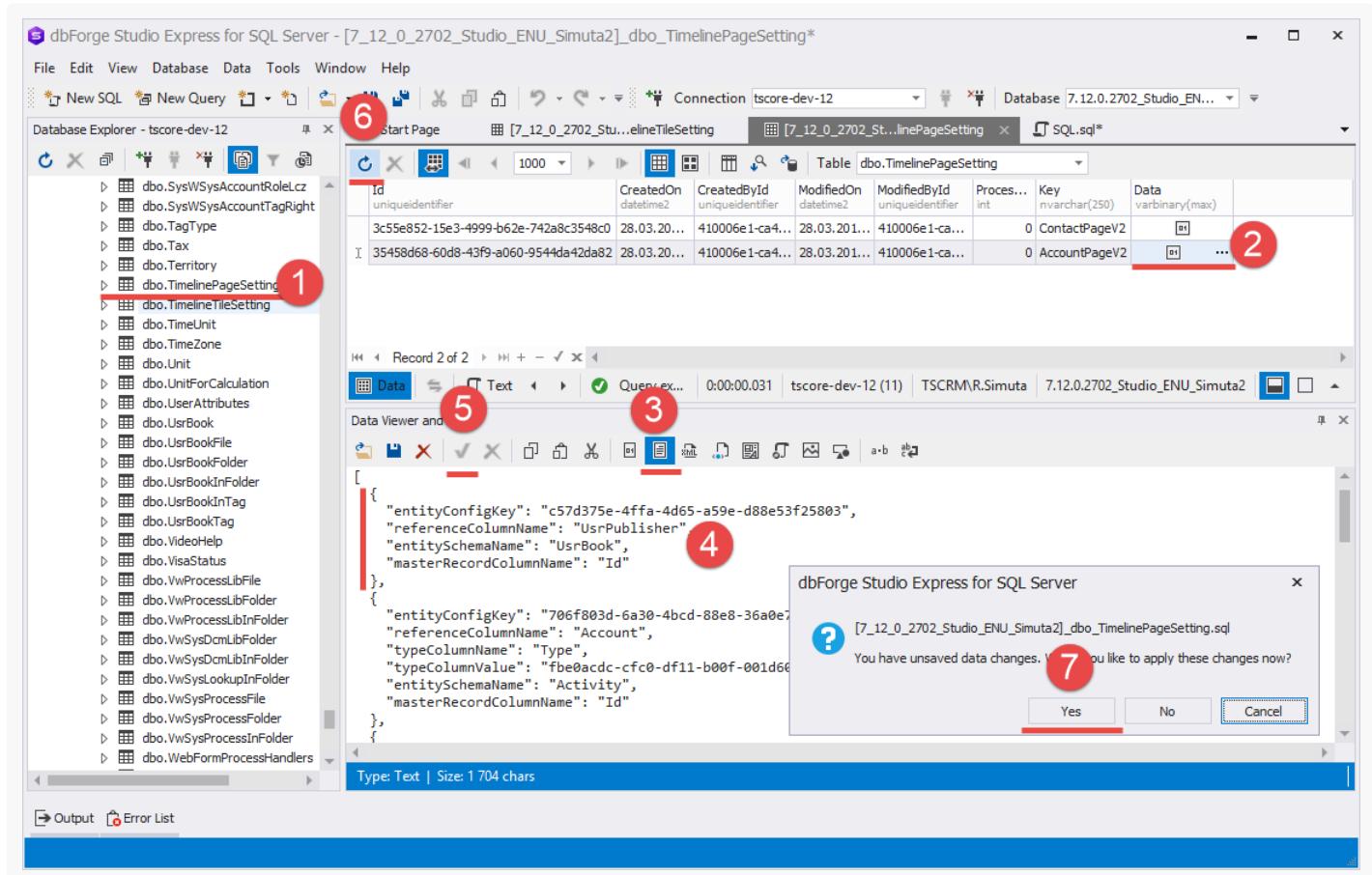
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 52 52" enable-background="new 0 0 52 52"
<path d="M46.072,31.384c-0.011-0.026-0.025-0.048-0.039-0.073c-0.036-0.064-0.077-0.125-0.123-0
c-0.018-0.022-0.034-0.044-0.053-0.064c-0.034-0.036-0.068-0.07-0.105-0.104c-0.062-0.055-0.
c-1.958-1.307-7.465-4.978-9.424-6.284c-0.388-0.258-0.703-0.845-0.703-1.312V3.938c0-0.401-
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. On the left, a modal window for the entity 'Apress' is open. It contains fields for Name (Apress), Type (Web), Owner (Supervisor), Primary phone, Category, and Industry. A progress bar indicates 10% completion. On the right, the main workspace shows a timeline for December 2021. The 'Timeline' tab is selected. Two items are listed:

- 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.

## Модальное окно

**Назначение** модального окна — отображение данных во всплывающем диалоговом окне.

**Логика работы** модального окна:

- Остается открытой страница приложения, из которой было вызвано модальное окно.
- Не выполняется переход на новую страницу приложения.
- Страница, которая отображается в модальном окне, не учитывается в истории переходов по страницам браузера.

Модальное окно **позволяет**:

- Отображать произвольную информацию (например, текст, кнопки и т. д.).
- Выбирать данные из [справочника](#). Например, выбор ответственного за активность выполняется из справочника контактов, который отображается в модальном окне.

Full name	Email	Account
Symon Clarke	symon-clarke@yahoo.com	Our company
Jordan Anderson	j.anderson@yahoo.com	Alpha Business
William Walker	william.walker.work@gmail.com	Our company
Megan Lewis	megan.lewis.business@gmail.com	Our company
Supervisor		Our company
Andrew Wayne	a.wayne@apex.co.uk	Apex Solutions
Email Supervisor		
John Best	john_best_business@yahoo.com	Our company
Peter Moore	peter.moore@yahoo.com	Our company
Caleb Jones	c.jones@yahoo.co.uk	Our company
Alexander Wilson	a.wilson@alphabusiness.com	Alpha Business
William Clarke	w.clarke@alphabusiness.com	Alpha Business

**Составляющие** модального окна:

- Функциональность модального окна — схемы `ModalBox` И `ModalBoxSchemaModule` пакета `NUI`.
- Вызов модального окна для выбора данных из справочника — схема `LookupUtilitiesV2` пакета `NUI`.

## Реализовать модальное окно

 Сложный

**Пример.** Реализовать действие процесса для отображения модального окна в бизнес-процессе. Модальное окно содержит произвольный текст и кнопки [ Yes ], [ No ], [ Cancel ].

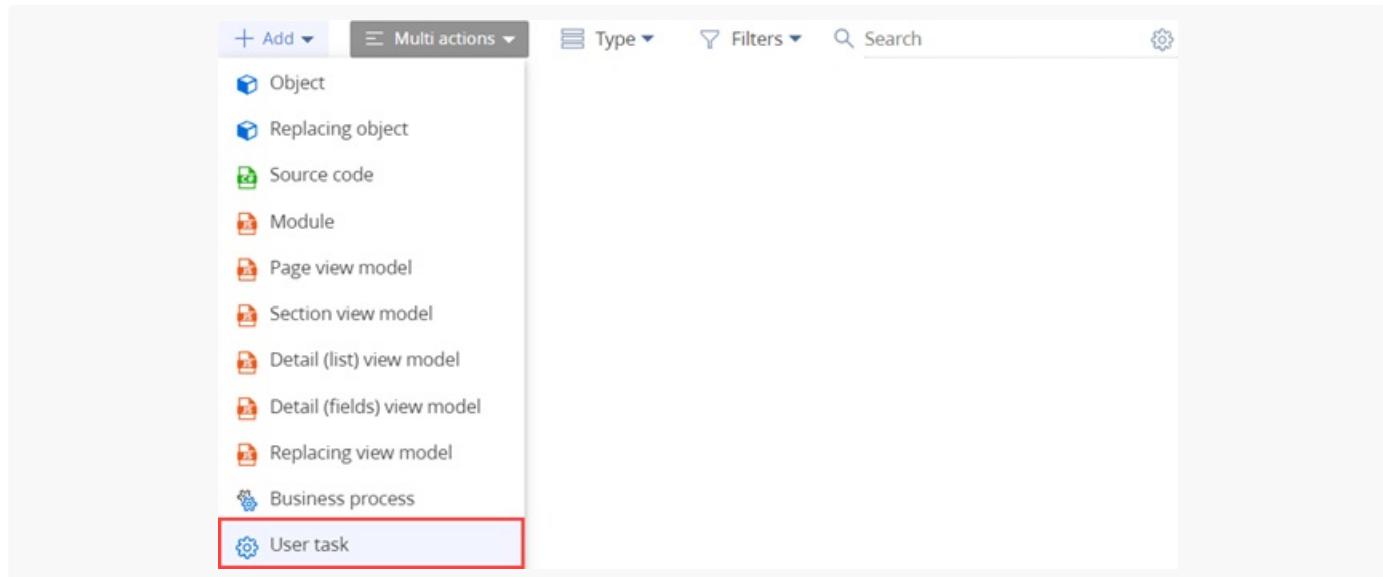
Логика работы кнопок:

- По нажатию на кнопки [ Yes ] и [ No ] отображаются соответствующие autogenerated страницы.
- По нажатию на кнопку [ Cancel ] закрывается модальное окно.

Для действия процесса настройте логирование.

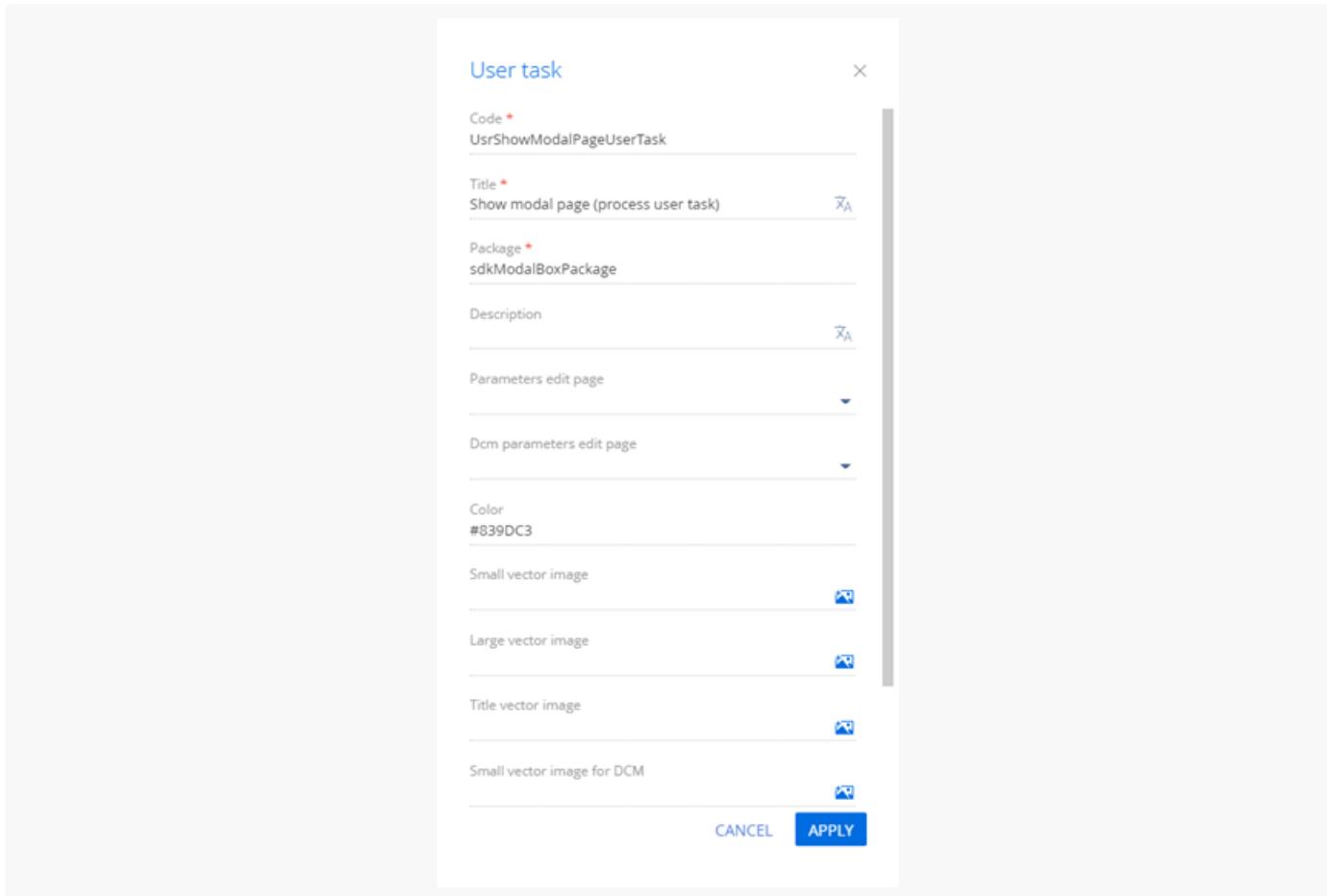
## 1. Создать действие процесса

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



### 3. Заполните **свойства действия процесса**.

- [ Код ] ([ Code ]) — "UsrShowModalPageUserTask".
- [ Заголовок ] ([ Title ]) — "Показать модальное окно (действие процесса)" ("Show modal page (process user task)").



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

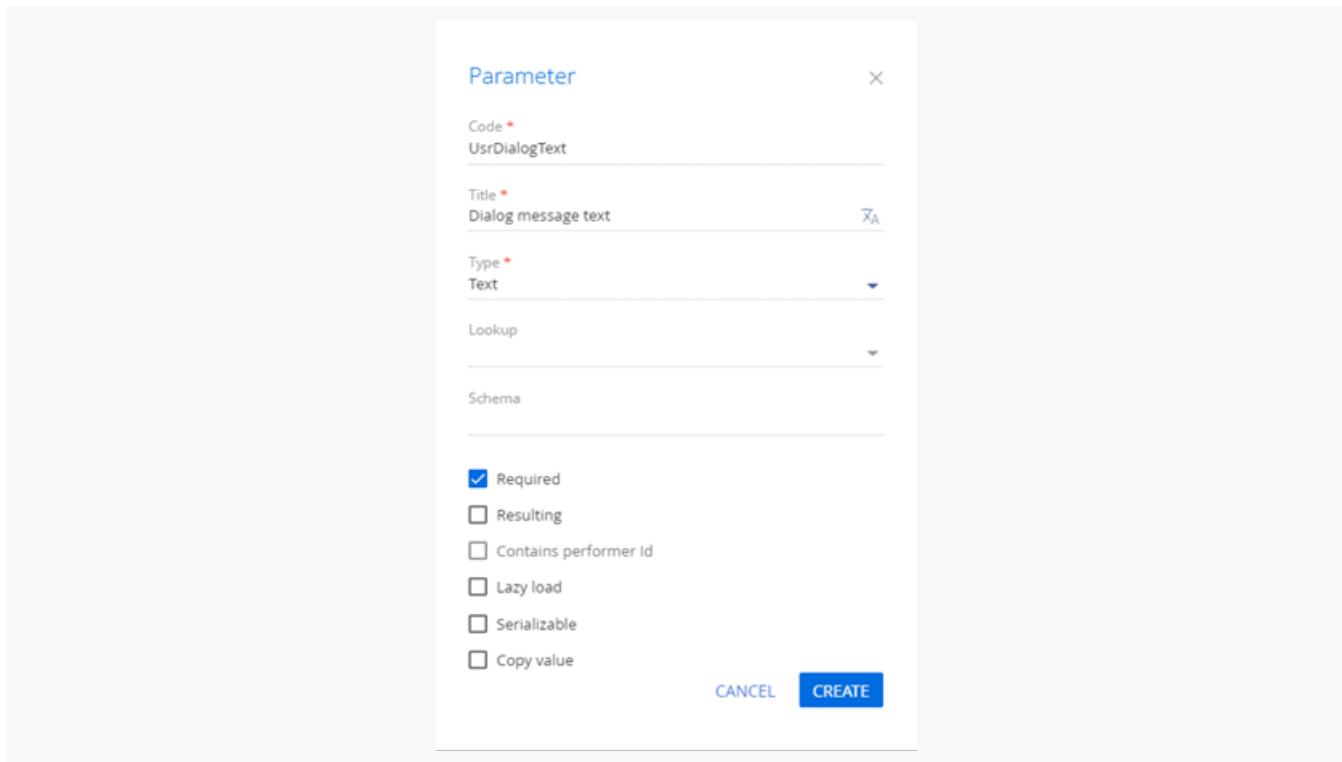
## 2. Добавить параметры действия процесса

1. Добавьте параметр, который выводит пользователю сообщение.

a. В узле [ Параметры ] ([ *Parameters* ]) нажмите кнопку .

b. Заполните **свойства параметра**.

- [ Код ] ([ *Code* ]) — "UsrDialogText".
- [ Название ] ([ *Title* ]) — "Текст в диалоговом окне" ("Dialog message text").
- [ Тип ] ([ *Type* ]) — выберите "Строка" ("Text").
- Установите признак [ Обязательный для заполнения ] ([ *Required* ]).

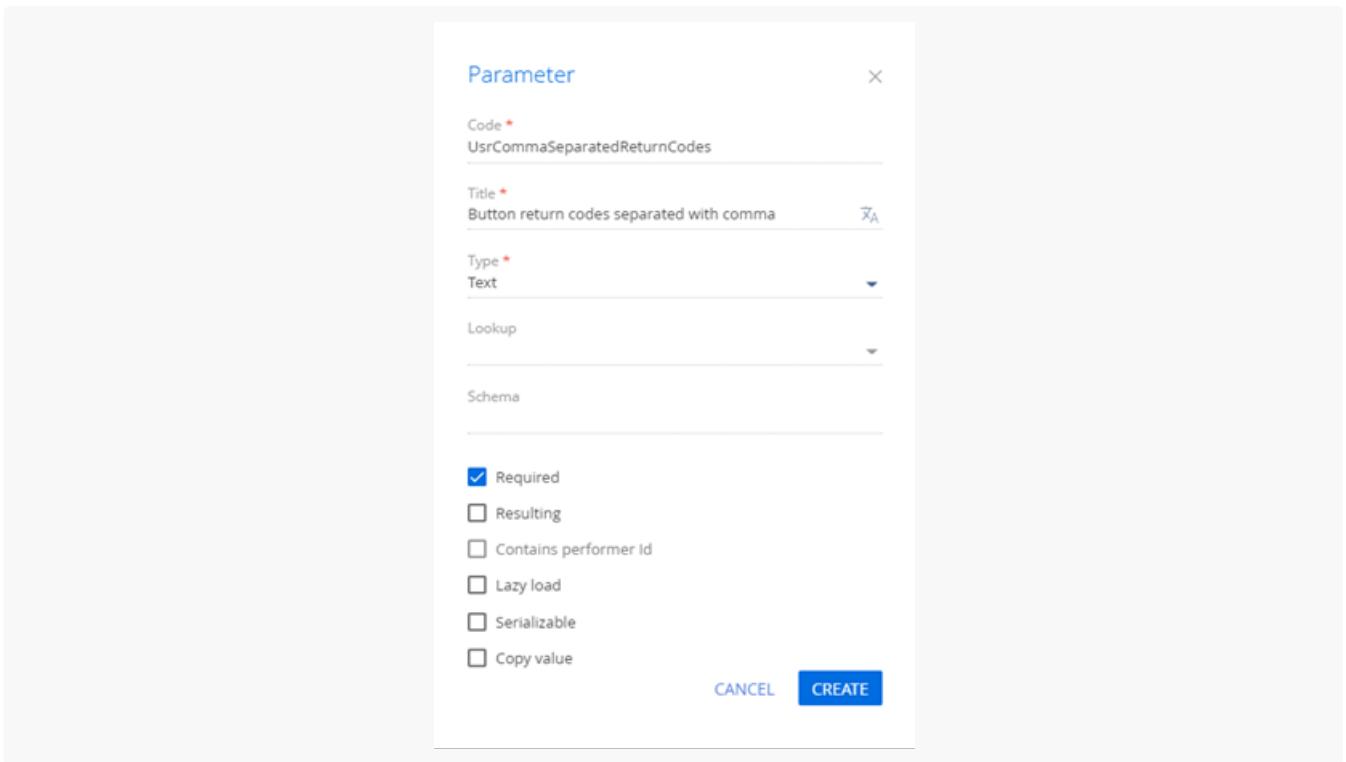


2. Добавьте параметр, который из процесса в элемент передает названия кнопок, которые необходимо отобразить пользователю. Параметр в строке принимает коды кнопок, которые разделены запятой.

a. В узле [ Параметры ] ([ Parameters ]) нажмите кнопку .

b. Заполните **свойства параметра**.

- [ Код ] ([ Code ]) — "UsrCommaSeparatedReturnCodes".
- [ Название ] ([ Title ]) — "Коды возврата кнопок через запятую" ("Button return codes separated with comma").
- [ Тип ] ([ Type ]) — выберите "Строка" ("Text").
- Установите признак [ Обязательный для заполнения ] ([ Required ]).

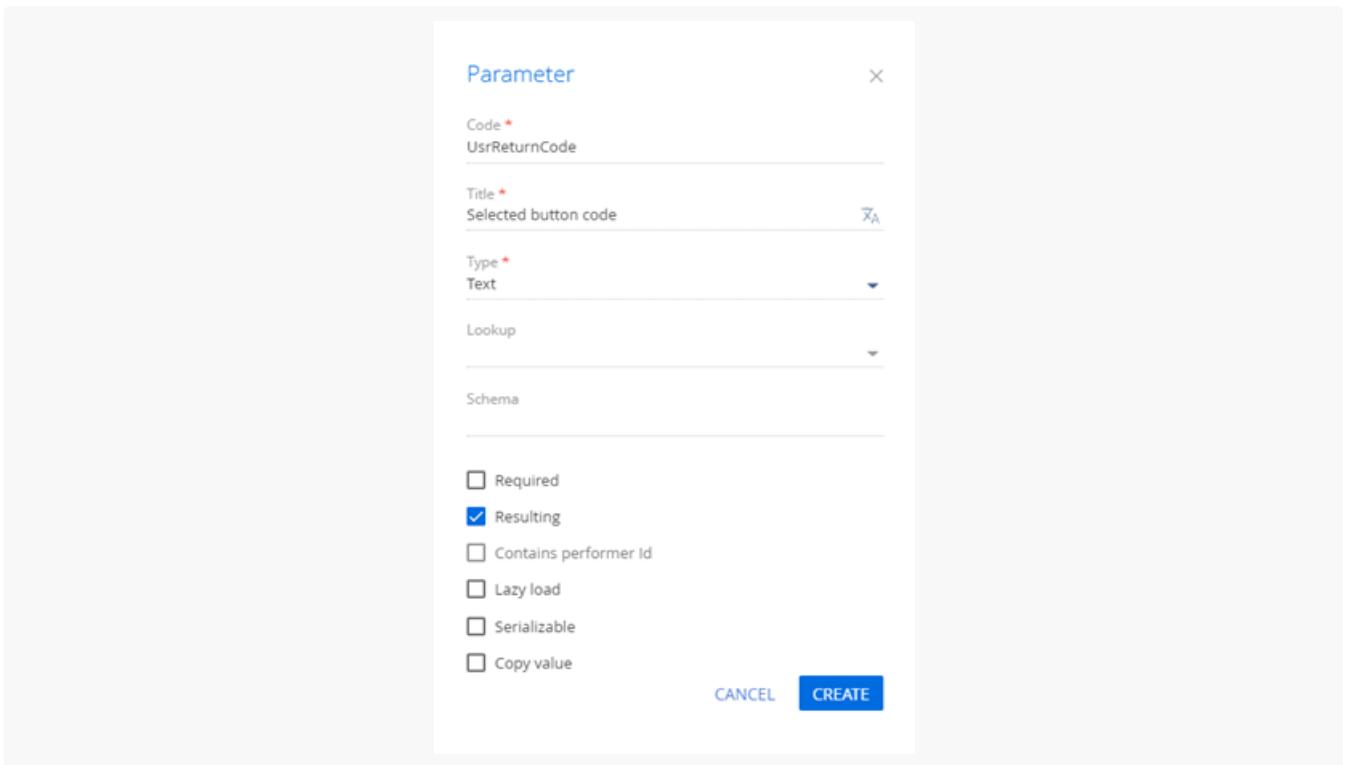


3. Добавьте параметр, который содержит код нажатой кнопки.

a. В узле [ Параметры ] ([ Parameters ]) нажмите кнопку .

b. Заполните **свойства параметра**.

- [ Код ] ([ Code ]) — "UsrReturnCode".
- [ Название ] ([ Title ]) — "Код нажатой кнопки" ("Selected button code").
- [ Тип ] ([ Type ]) — выберите "Строка" ("Text").
- Установите признак [ Результирующий ] ([ Resulting ]).



### 3. Реализовать действие процесса на back-end стороне

#### 1. Настройте логирование процесса.

- С помощью директивы `using` добавьте необходимые пространства имен.

```
using global::Common.Logging;
using Terrasoft.Configuration;
```

- Реализуйте логику работы логирования и действия процесса.

#### UsrShowModalPageUserTask

```
namespace Terrasoft.Core.Process.Configuration
{
    using Newtonsoft.Json;
    using Newtonsoft.Json.Linq;
    using System;
    using System.Collections.Generic;
    using System.Collections.ObjectModel;
    using System.Globalization;
    using Terrasoft.Common;
    using Terrasoft.Core;
    using Terrasoft.Core.Configuration;
    using Terrasoft.Core.DB;
```

```

using Terrasoft.Core.Entities;
using Terrasoft.Core.Process;
using Terrasoft.UI.WebControls.Controls;

/* Для работы логирования.*/
using global::Common.Logging;

/* Для работы инструментов отправки сообщений.*/
using Terrasoft.Configuration;

#region Class: UsrShowModalPageUserTask

/// <exclude>
public partial class UsrShowModalPageUserTask
{
    /* Настройка логирования.
    Создание отдельного логгера. Результаты логирования записываются в файл Common.log.
    private static readonly ILog _log = LogManager.GetLogger("UsrShowModalPageUserTask");

    /* Определение отправителя со стороны back-end.*/
    private const string MessageSender = "UsrShowModalPageUserTask";

    #region Methods: Protected

    /* Бизнес-логика действия процесса.*/
    protected override bool InternalExecute(ProcessExecutingContext context) {

        /* Вывод информационного сообщения в логи.*/
        _log.InfoFormat("UserTask works well. UsrDialogText = {0}, UsrCommaSeparatedReturnCodes = {1}", UsrDialogText, UsrCommaSeparatedReturnCodes);
        /* Формирование сообщения.*/
        var messageData = new
        {
            /* Текст в диалоговом окне.*/
            UsrDialogText = UsrDialogText,
            /* Коды возврата кнопок через запятую.*/
            UsrCommaSeparatedReturnCodes = UsrCommaSeparatedReturnCodes,
            /* Служебный параметр. Уникальный идентификатор экземпляра элемента внутри procElUID = UIId*/
            procElUID = UIId
        };
        /* Сериализация объекта тела сообщения.*/
        string messageBody = JsonConvert.SerializeObject(messageData);
        /* Тело сообщения, которое отправляется из back-end во front-end часть.*/
        MsgChannelUtilities.PostMessage(UserConnection, MessageSender, messageBody);
        return false;
    }

    #endregion
}

```

```

#region Methods: Public

    public override bool CompleteExecuting(params object[] parameters) {
        return base.CompleteExecuting(parameters);
    }

    public override void CancelExecuting(params object[] parameters) {
        base.CancelExecuting(parameters);
    }

    public override string GetExecutionData() {
        return string.Empty;
    }

    public override ProcessElementNotification GetNotificationData() {
        return base.GetNotificationData();
    }

#endregion

}

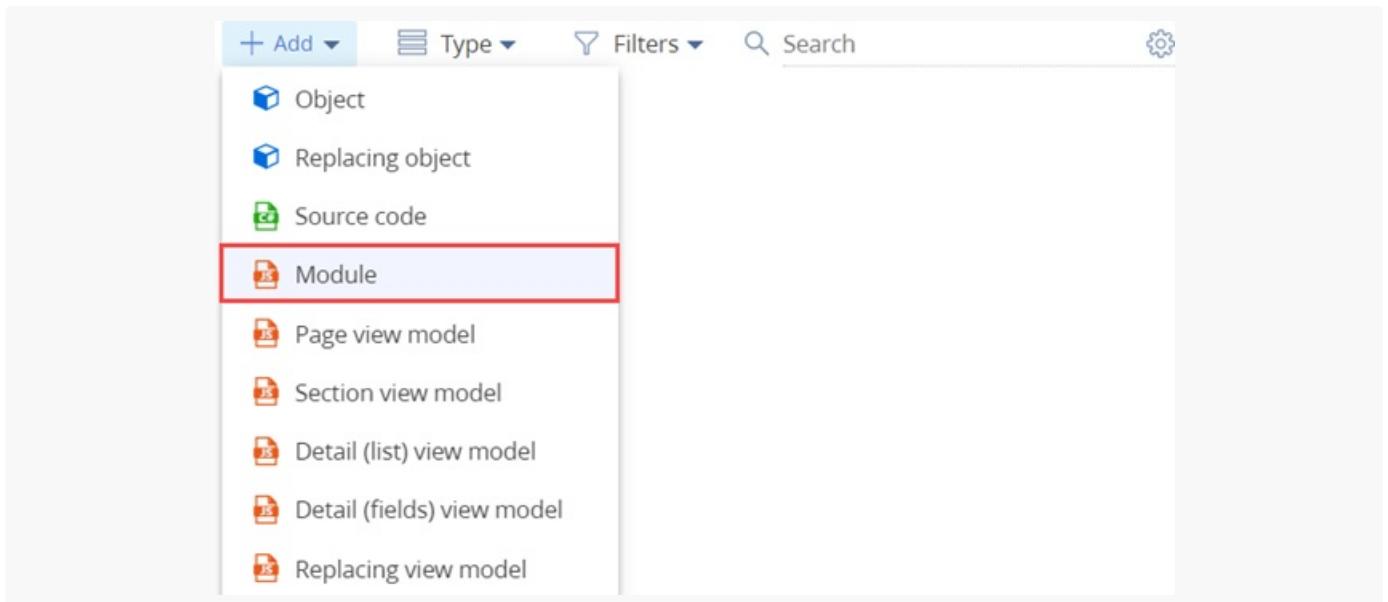
#endregion
}

```

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

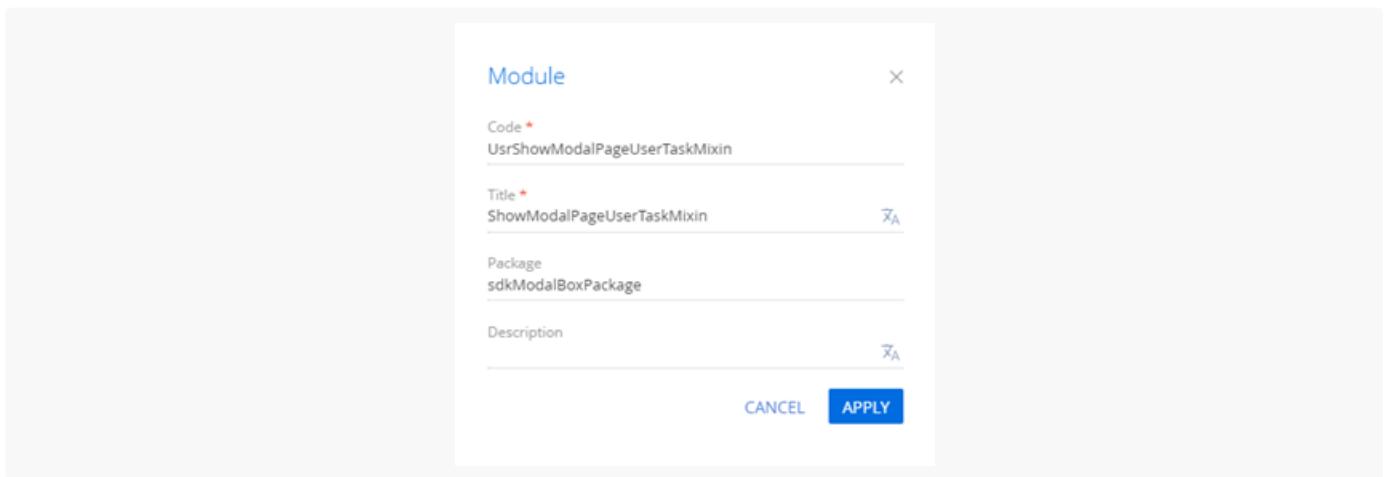
## 4. Реализовать обработку действия процесса на front-end стороне

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



### 3. Заполните **свойства модуля**.

- [ Код ] ([ Code ]) — "UsrShowModalPageUserTaskMixin".
- [ Заголовок ] ([ Title ]) — "ShowModalPageUserTaskMixin".



### 4. В дизайнере модуля добавьте исходный код.

```
UsrShowModalPageUserTaskMixin

define("UsrShowModalPageUserTaskMixin", [],
    function() {
        Ext.define("Terrasoft.configuration.mixins.ShowModalPageUserTaskMixin", {
            alternateClassName: "Terrasoft.ShowModalPageUserTaskMixin",

            /* Подписаться на сообщения канала WebSocket. */
            UsrSubscribeForShowModalWindowServerChannelMessages: function() {
                this.Terrasoft.ServerChannel.on(this.Terrasoft.EventName.ON_MESSAGE, this.Usr
            },
            /* Отписаться от сообщений канала WebSocket. */
        })
    }
)
```

```

    UsrUnsubscribeForShowModalWindowServerChannelMessages: function() {
        this.Terrasoft.ServerChannel.un(this.Terrasoft.EventName.ON_MESSAGE, this.Usr
    },

    UsrShowModalPageUserTask_procElUID: "",

    /* Функция, которая обрабатывает WebSocket сообщения. */
    UsrServerChannelMessageHandler: function(scope, message) {
        if (!message) {
            return;
        }
        /* Определение сообщения от отправителя UsrShowModalPageUserTask. */
        if (message.Header && message.Header.Sender !== "UsrShowModalPageUserTask") {
            return;
        }
        if (message.Body) {
            var bodyData = this.Ext.decode(message.Body);
            var UsrDialogText = bodyData.UsrDialogText;
            var UsrCommaSeparatedReturnCodes = bodyData.UsrCommaSeparatedReturnCodes;

            /* Получение и сохранение кодов кнопок в виде массива. */
            var returnCodesArray = UsrCommaSeparatedReturnCodes.split(",");
            var procElUID = bodyData.procElUID;
            UsrShowModalPageUserTask_procElUID = procElUID;

            /* Отобразить диалоговое окно. */
            this.showConfirmationDialog(UsrDialogText, this.UsrGetConfirmationResult,
        }
    },
    /* Обработка результата выбора кода кнопки. */
    UsrGetConfirmationResult: function(returnCode) {
        this.console.log("UsrGetConfirmationResult: returnCode = " + returnCode + " U

        var procElUID = UsrShowModalPageUserTask_procElUID;

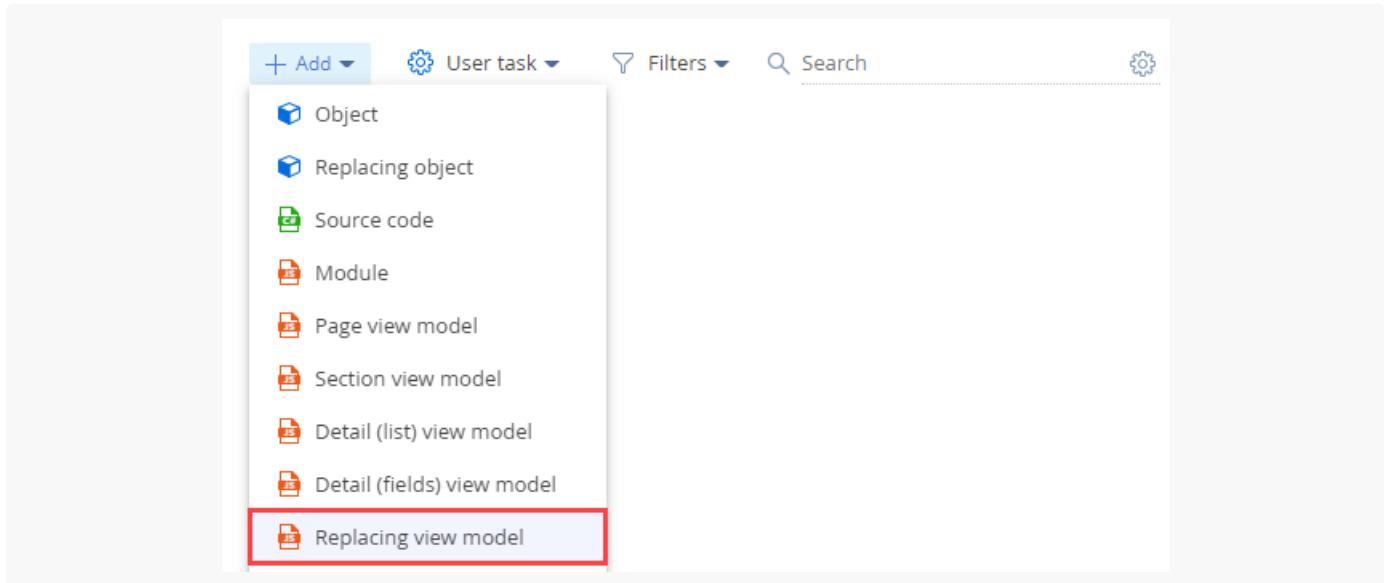
        /* Создание POST-запроса. */
        this.Terrasoft.AjaxProvider.request({
            url: "../ServiceModel/ProcessEngineService.svc/" + procElUID + "/Complete
            method: "POST",
            scope: this,
            callback: function(request, success, response) {
            }
        });
    }
});
});

```

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

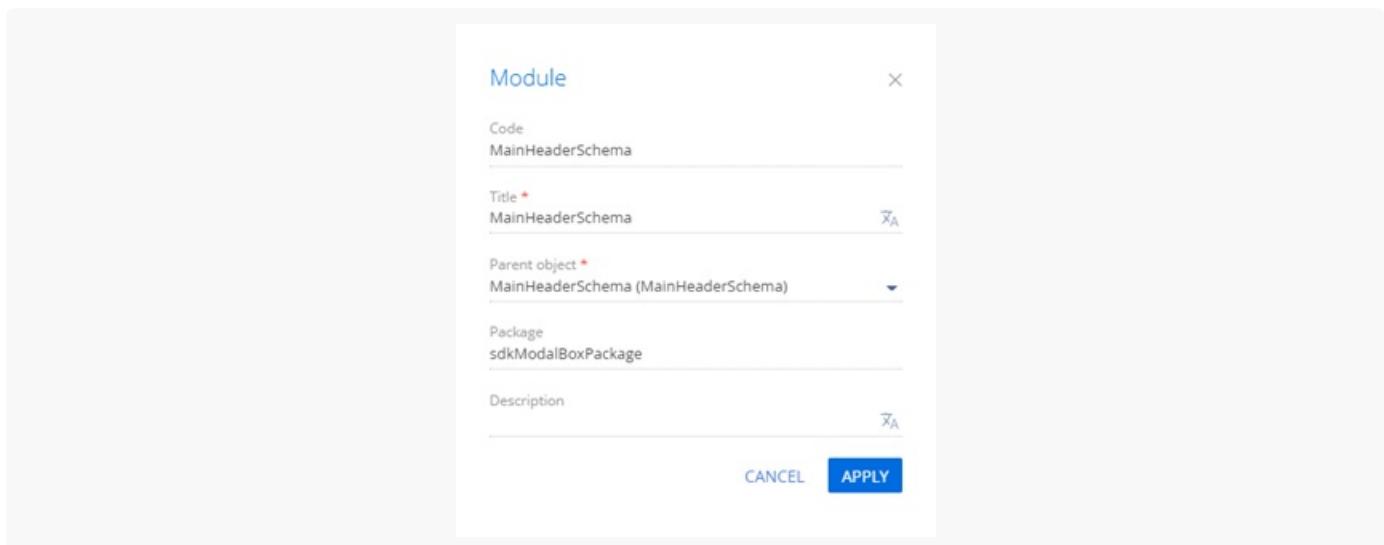
## 5. Создать замещающую модель представления контейнера

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 модуля**.

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



### 4. В дизайнере модуля добавьте исходный код.

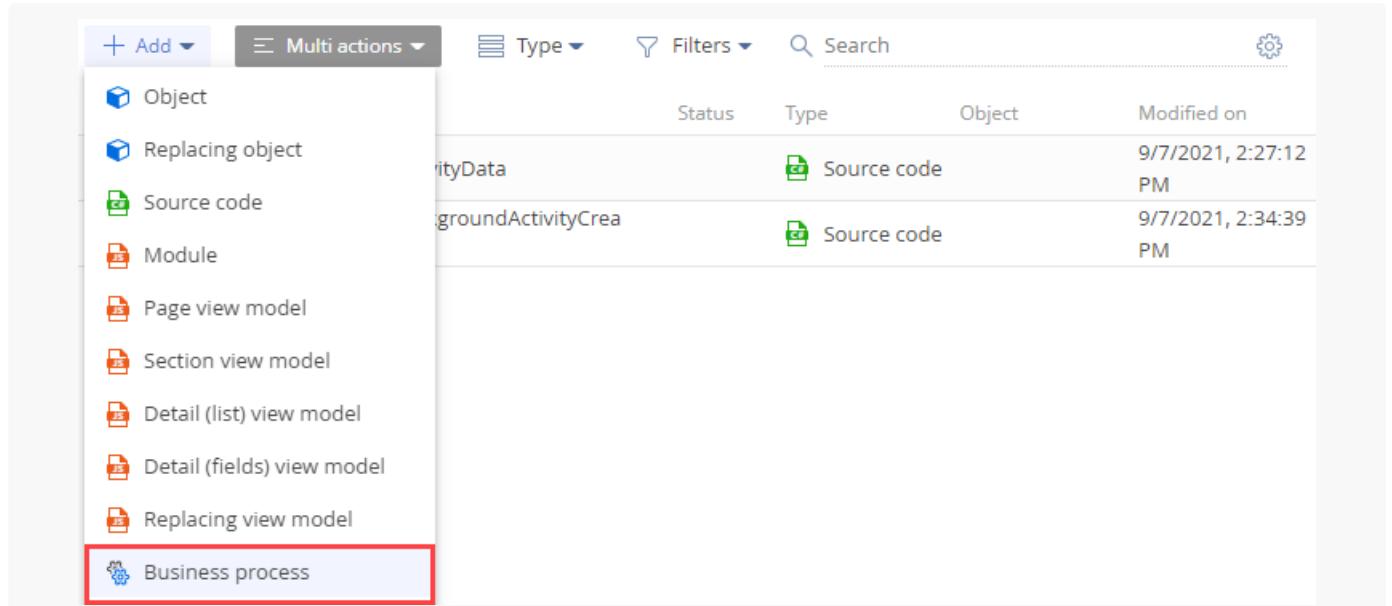
```
MainHeaderSchema

define("MainHeaderSchema", [ "UsrShowModalPageUserTaskMixin"], function(){
    return {
        mixins: {
            ShowModalPageUserTaskMixin: "Terrasoft.ShowModalPageUserTaskMixin"
        },
        methods: {
            init: function() {
                this.callParent(arguments);
                this.UsrSubscribeForShowModalWindowServerChannelMessages();
            },
            destroy: function() {
                this.callParent(arguments);
                this.UsrUnsubscribeForShowModalWindowServerChannelMessages();
            }
        }
    };
});
```

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

## 6. Создать бизнес-процесс отображения модального окна

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

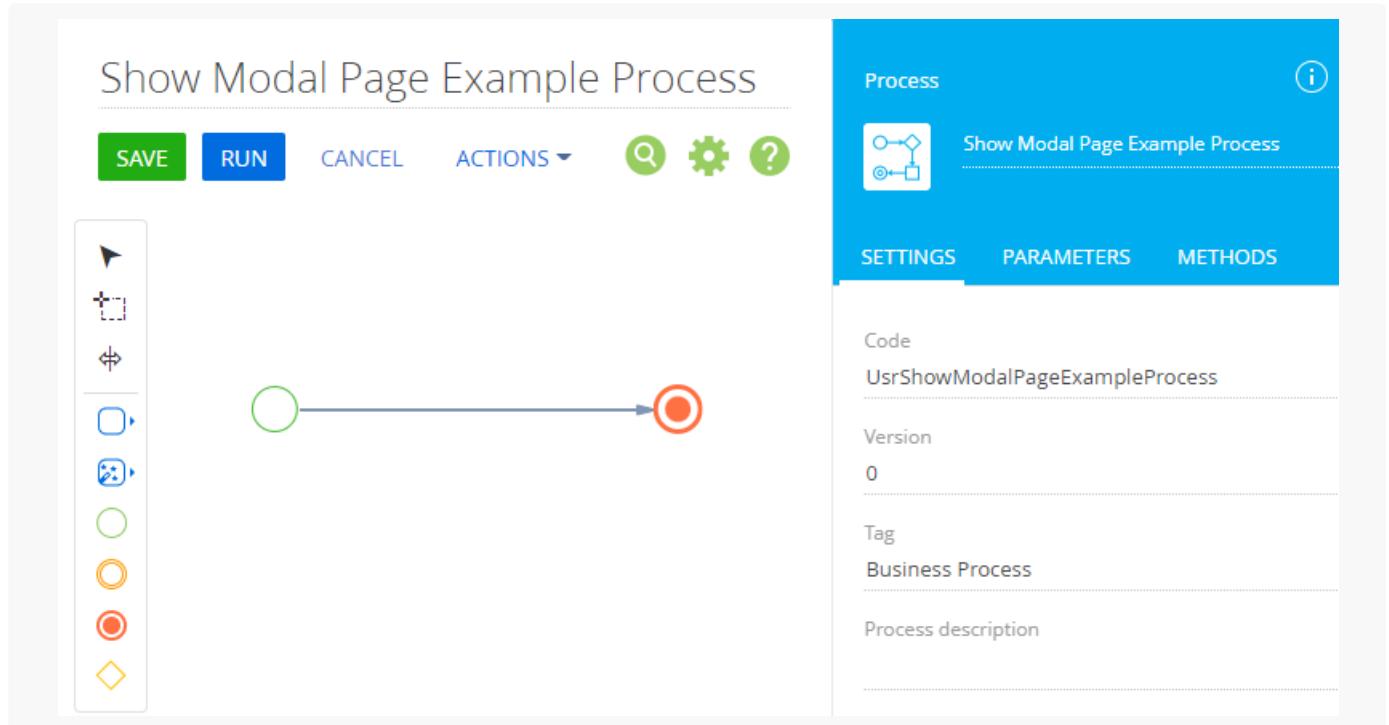


The screenshot shows a list of objects in a configuration interface. On the left is a sidebar with icons for various object types: Object, Replacing object, Source code, Module, Page view model, Section view model, Detail (list) view model, Detail (fields) view model, and Replacing view model. Below this is a main list area with columns: Status, Type, Object, and Modified on. Two items are listed: 'cityData' (Status: Draft, Type: Source code, Modified on: 9/7/2021, 2:27:12 PM) and 'groundActivityCreation' (Status: Draft, Type: Source code, Modified on: 9/7/2021, 2:34:39 PM). At the bottom of the sidebar, the 'Business process' item is highlighted with a red box. The top of the screen has a toolbar with buttons for 'Add', 'Multi actions', 'Type', 'Filters', 'Search', and a gear icon.

	Status	Type	Object	Modified on
cityData	Draft	Source code	9/7/2021, 2:27:12 PM	
groundActivityCreation	Draft	Source code	9/7/2021, 2:34:39 PM	

### 3. Заполните **свойства процесса**.

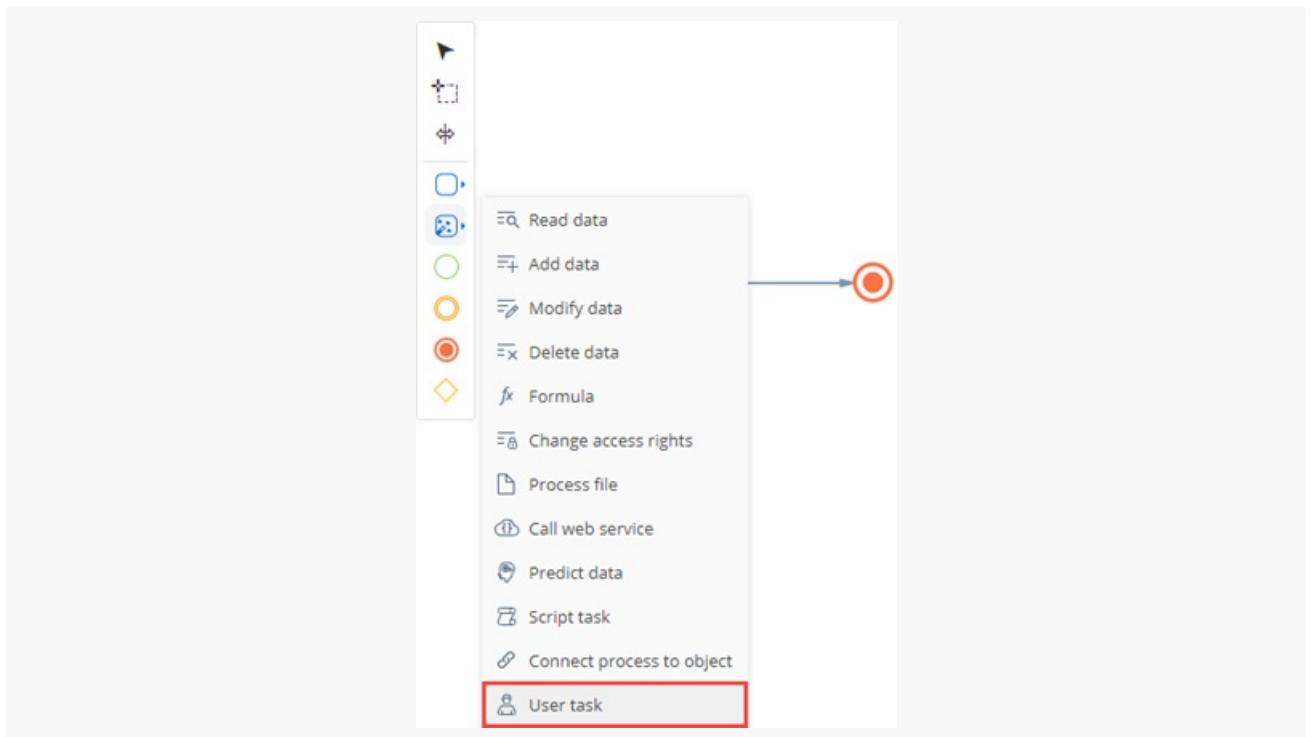
- На панели настройки элементов заполните свойство [ **Заголовок** ] ([ *Title* ]) — "Show Modal Page Example Process".
- На вкладке [ **Настройки** ] ([ *Settings* ]) панели настройки элементов заполните свойство [ **Имя** ] ([ *Code* ]) — "UsrShowModalPageExampleProcess".



### 4. Реализуйте бизнес-процесс.

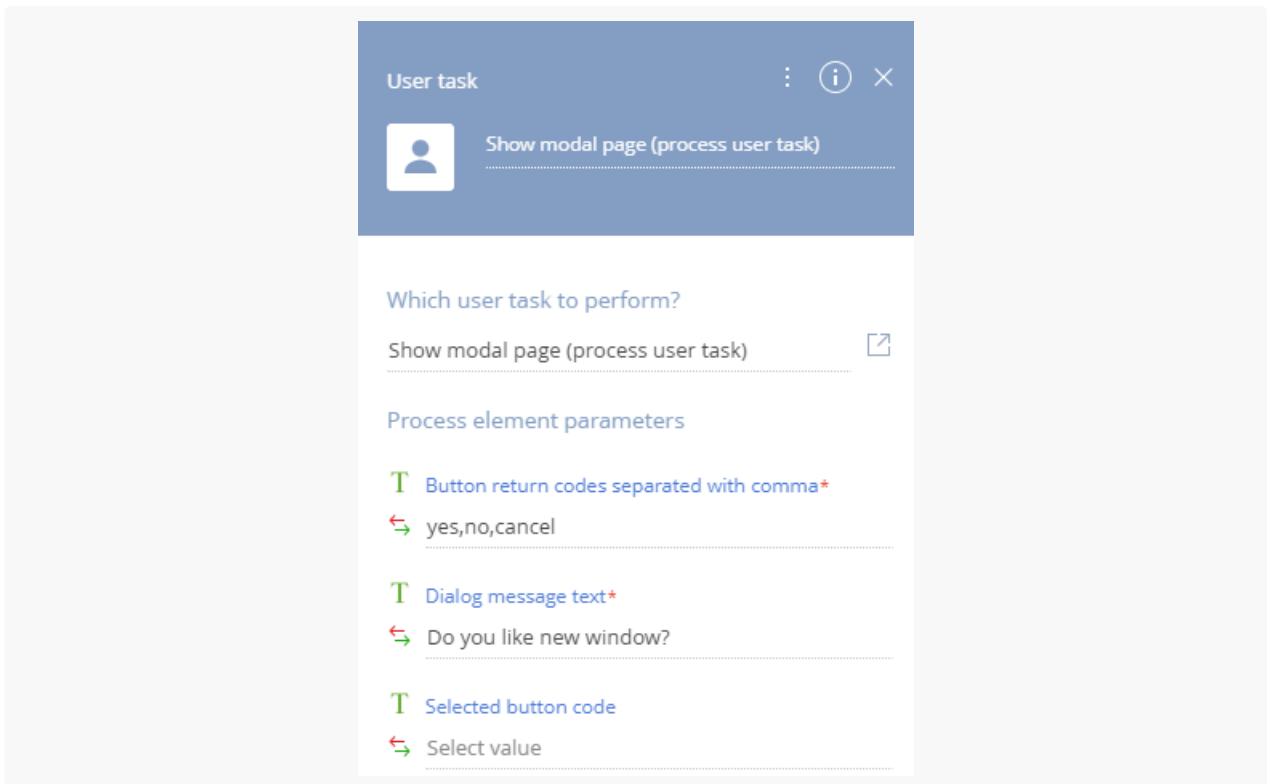
#### a. Добавьте действие процесса.

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



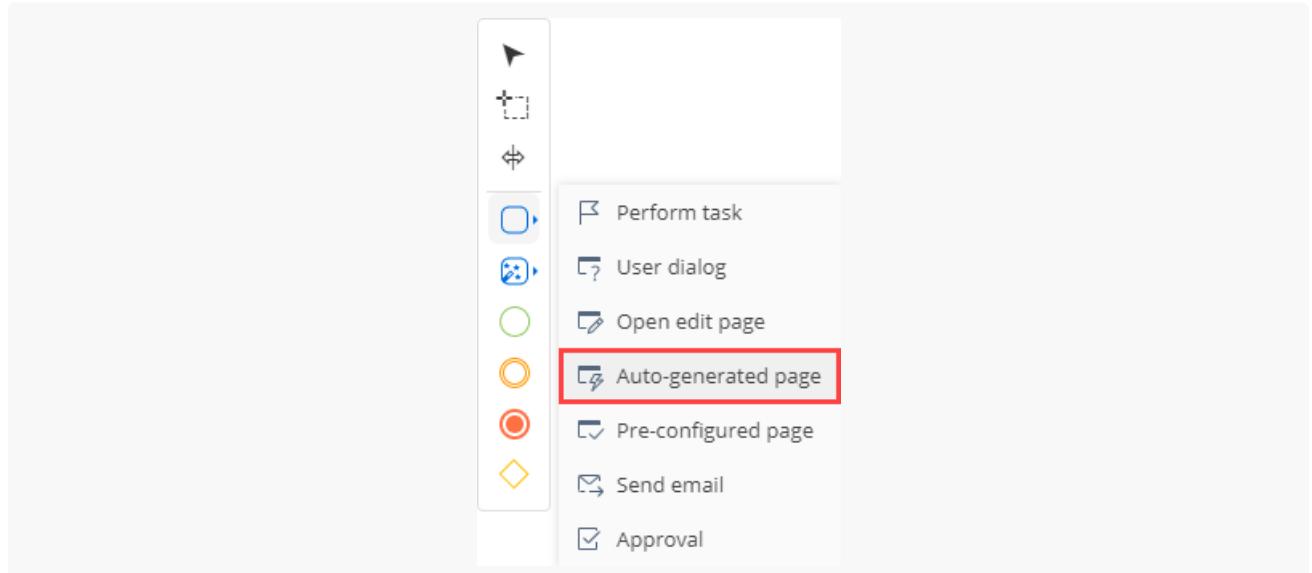
b. Заполните **свойства действия процесса**.

- [ Какое пользовательское действие выполнить? ] ([ Which user task to perform? ]) — выберите "Показать модальное окно (действие процесса)" ("Show modal page (process user task)").
- Заполните значения параметров действия процесса.
  - [ Текст в диалоговом окне ] ([ Dialog message text ]) — "Вам нравится новое окно?" ("Do you like new window?").
  - [ Коды возврата кнопок через запятую ] ([ Button return codes separated with comma ]) — "yes,no,cancel".



б. Добавьте автогенерируемую страницу.

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



б. Заполните **свойства автогенерируемой страницы**.

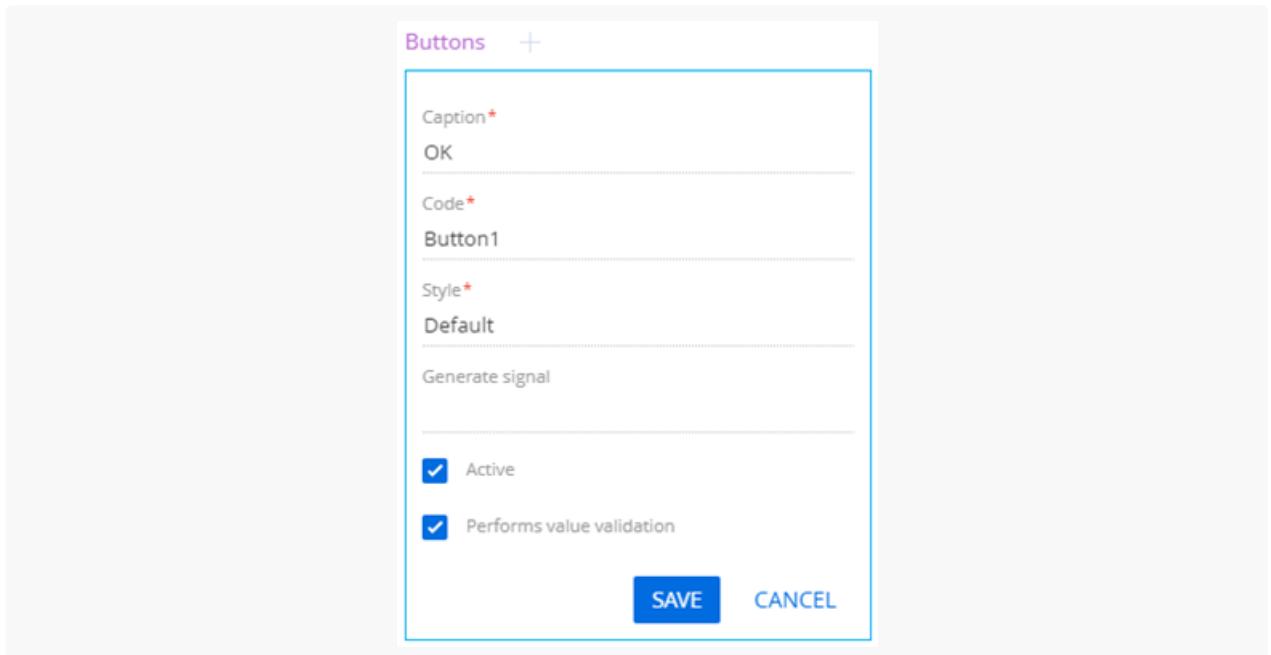
- [Заголовок] ([Title]) — "YES".
- [Название страницы] ([Page title]) — "Нажато YES" ("Pressed YES").

е. Добавьте кнопку.

a. В блоке [ Кнопки ] ([ Buttons ]) нажмите кнопку **+**.

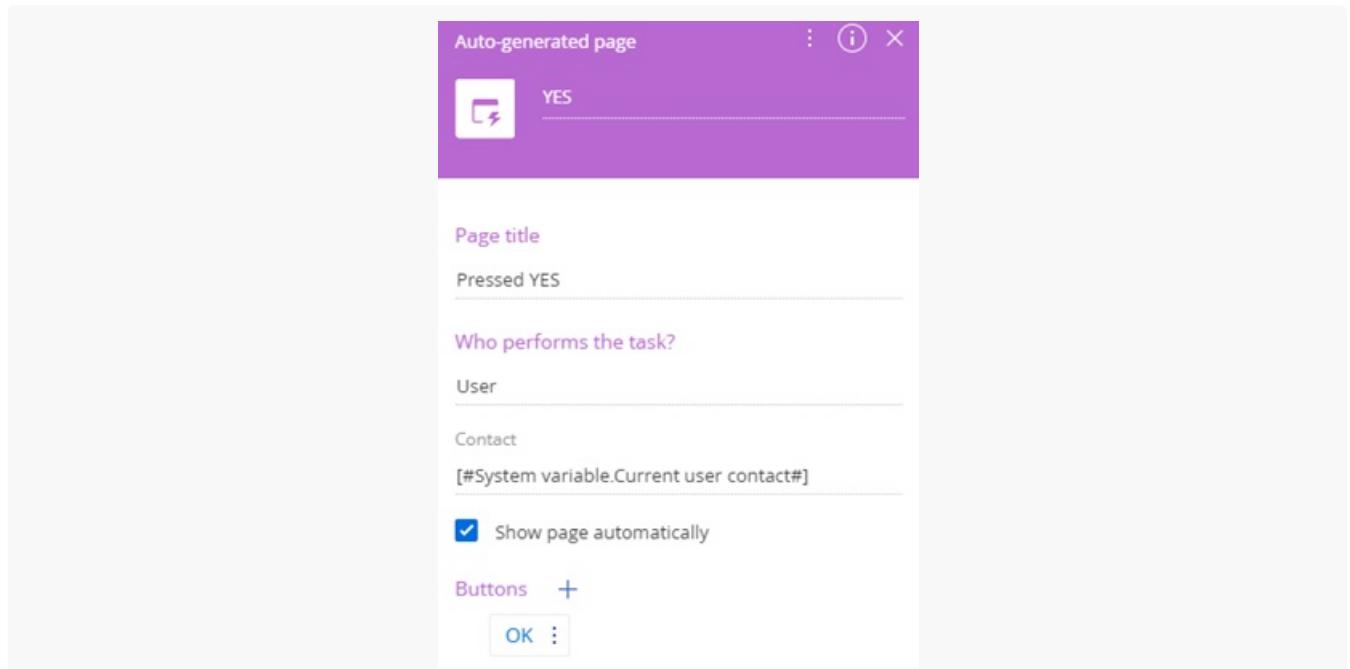
b. Заполните **свойства кнопки**.

- [ Название ] ([ Caption ]) — "OK".



d. Сохраните изменения.

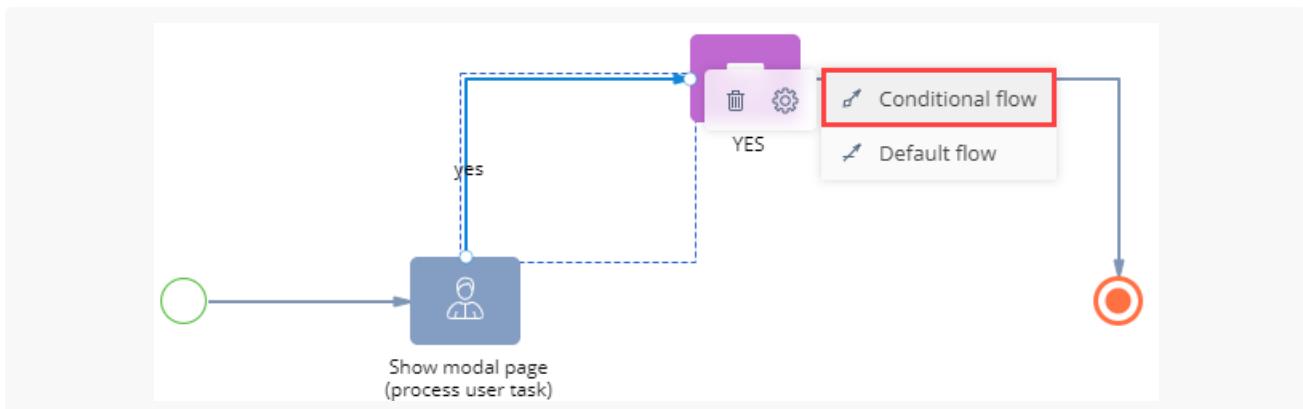
Настройки автогенерируемой страницы представлены на рисунке ниже.



c. Настройте условный поток.

a. В меню действия процесса выберите [ Добавить поток ] ([ Add flow ]) и соедините действие процесса с автогенерируемой страницей.

- b. Трансформируйте поток управления в условный поток. Для этого нажмите кнопку  и в меню потока выберите [ Условный поток ] ([ Conditional flow ]).



- c. Заполните **свойства условного потока**.

- [ Заголовок ] ([ Title ]) — "yes".

- e. Настройте условия перехода.

- На панели настройки элементов в свойстве [ Условие перехода ] ([ Condition to move down the flow ]) нажмите кнопку .
- Выберите элемент процесса "Показать модальное окно (действие процесса)" ("Show modal page (process user task)").
- Двойным кликом выберите параметр процесса "Код нажатой кнопки" ("Selected button code").
- Задайте формулу для параметра.

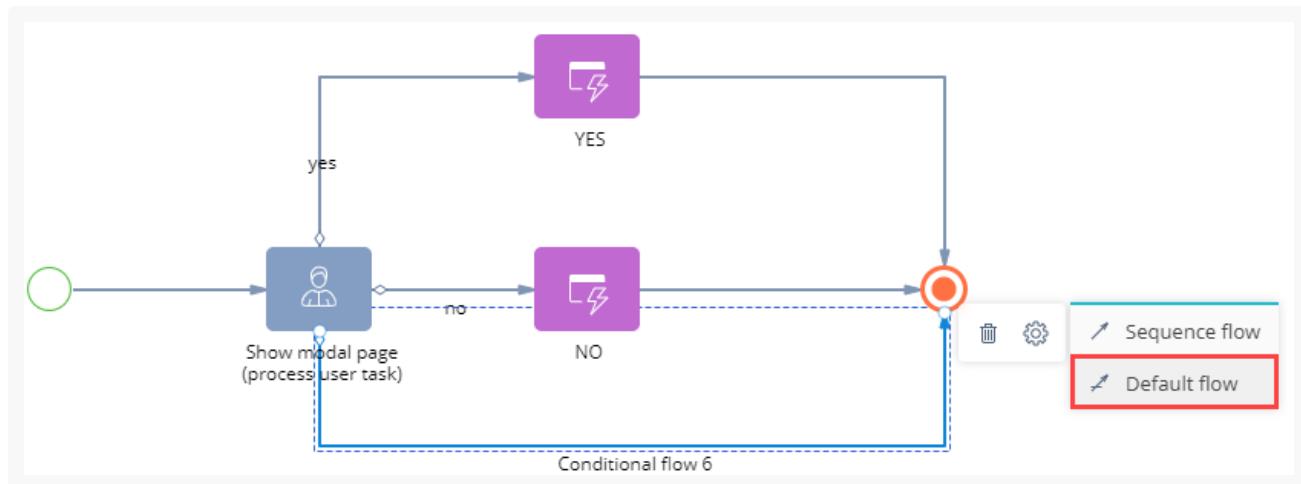
```
[#Show modal page (process user task).Selected button code#] == "yes"
```

- e. Сохраните изменения.

- d. Аналогично добавьте autogenerated страницу **no** с соответствующим условным потоком **no**.

- e. Настройте поток по умолчанию.

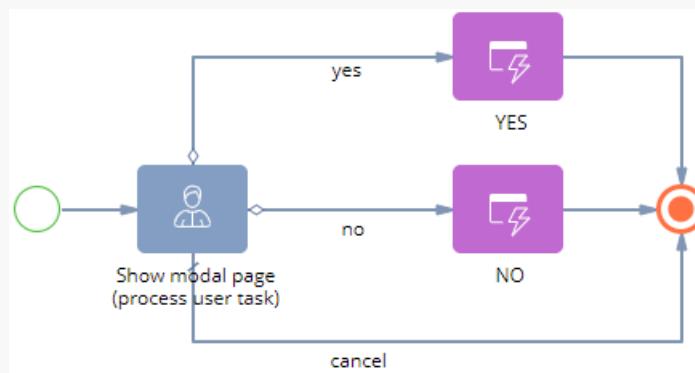
- В меню действия процесса выберите [ Добавить поток ] ([ Add flow ]) и соедините действие процесса с завершающим событием [ Останов ] ([ Terminate ]).
- Трансформируйте условный поток в поток по умолчанию. Для этого нажмите кнопку  и в меню потока выберите [ Поток по умолчанию ] ([ Default flow ]).



с. Заполните **свойства потока по умолчанию**.

- [Заголовок] ([Title]) — "cancel".

Бизнес-процесс представлен на рисунке ниже.



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

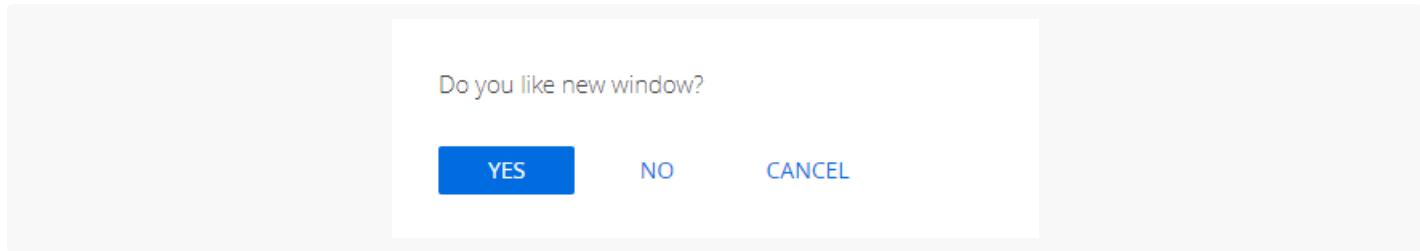
## Результат выполнения примера

Бизнес-процесс `Show Modal Page Example Process` доступен для запуска в любом разделе приложения Creatio. Например, запустим бизнес-процесс из раздела [Контакты] ([Contacts]).

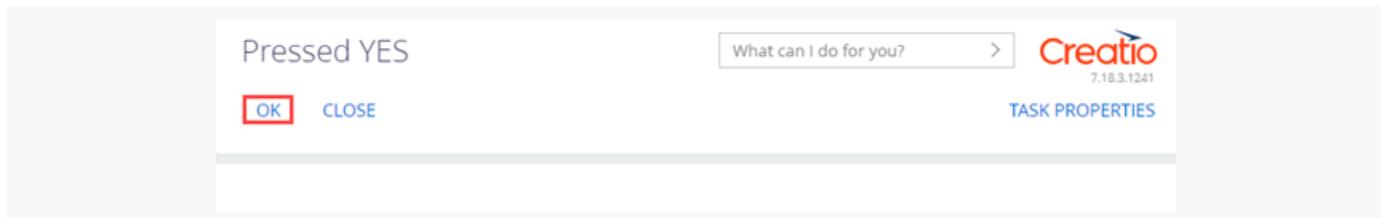
Чтобы **запустить бизнес-процесс** `Show Modal Page Example Process` из раздела [Контакты] ([Contacts]):

1. Перейдите в раздел [Контакты] ([Contacts]).
2. На панели разделов нажмите кнопку .
3. Выберите бизнес-процесс `Show Modal Page Example Process` и нажмите [Запустить] ([Run]).

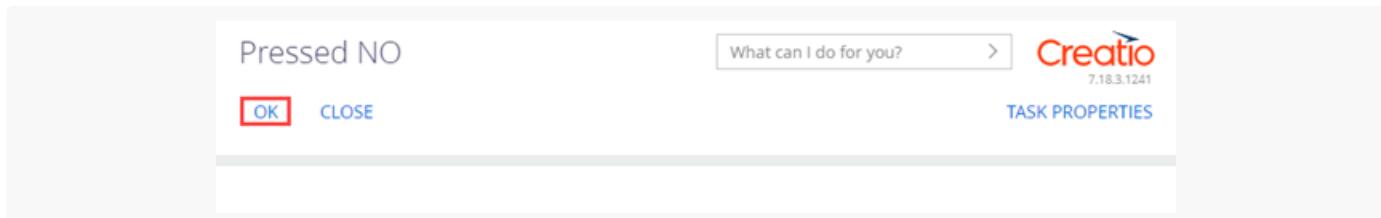
В результате выполнения бизнес-процесса `Show Modal Page Example Process` в разделе [Контакты] ([Contacts]) отображается модальное окно.



- При нажатии на кнопку [ Yes ] отображается страница [ Нажато YES ] ([ Pressed YES ]) с кнопкой [ OK ].



- При нажатии на кнопку [ No ] отображается страница [ Нажато NO ] ([ Pressed NO ]) с кнопкой [ OK ].



- При нажатии на кнопку [ Cancel ] модальное окно закрывается.

Результаты логирования запросов отображаются в файле `Common.log` корневой папки приложения. Используя инструменты разработчика в браузере, можно отследить WebSocket-сообщения, которые переданы из back-end части во front-end часть.

## 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 связанных с ним контрагентах. 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 связанный 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 связанный сущности

1. Создайте схему модели представления профиля связанный сущности. Для этого воспользуйтесь инструкцией, которая приведена в статье [Разработка конфигурационных элементов](#).
2. В схеме модели представления настройте **профиль связанный сущности**.
  - a. В свойство `mixins` добавьте миксин `ProfileSchemaMixin`.
  - b. В массив модификаций `diff` добавьте конфигурационный объект с настройками связанныго 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 связанныго пользовательского профиля контрагента. В свойство `masterColumnName` свойства `viewModelConfig` добавьте название колонки, по которой выполняется связь связанныго 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 связанныго 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 связанныго 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<sup>(New account)</sup> ] — 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.

**CONTACT INFO**

Full name\* Caleb Jones

Full job title Managing Director

Mobile phone +44 782 223 4967

Business phone 3010

Email c.jones@yahoo.co.uk

**Account**  
Our company

Type Our company

Owner John Best

**CONNECTED TO**

Discuss contract 102-01 with Caleb  
11/11/2021 | Symon Clarke

Prepare new customer report  
11/15/2021 | John Best

**MAINTENANCE**

Type Employee Owner John Best

Title Mr. Gender Male

Recipient's name Jones Preferred language

Age 42

**COMMUNICATION OPTIONS**

Business phone 3010 Mobile phone +44 782 223 4967

Email c.jones@yahoo.co.uk Extension phone 105

Do not use fax

**ADDRESSES**

Address type	Primary	Address	City	Country	ZIP/postal...
Home	Yes	153 Sunshine Street	London	United Kingdom	

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

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 () {
```

```

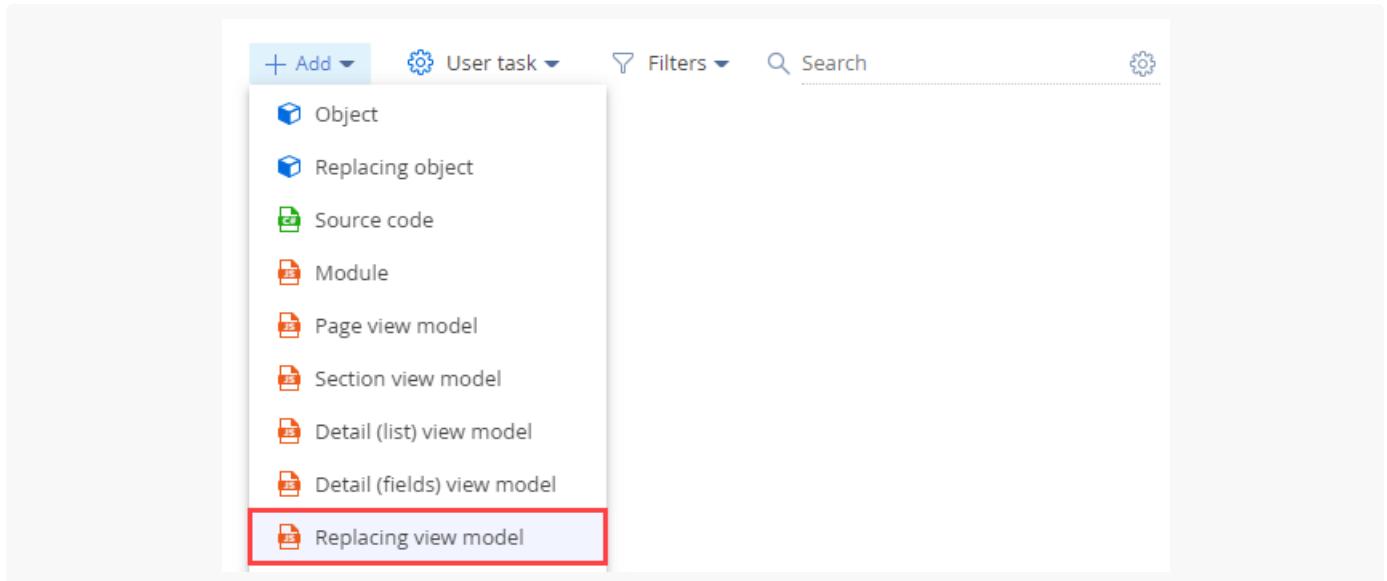
return {
    /* Название схемы объекта. */
    entitySchemaName: "Account",
    /* Миксины. */
    mixins: {
        /* Миксин, который содержит функции для получения иконок и картинок профиля. */
        ProfileSchemaMixin: "Terrasoft.ProfileSchemaMixin"
    },
    /* Массив модификаций diff. */
    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 ]).

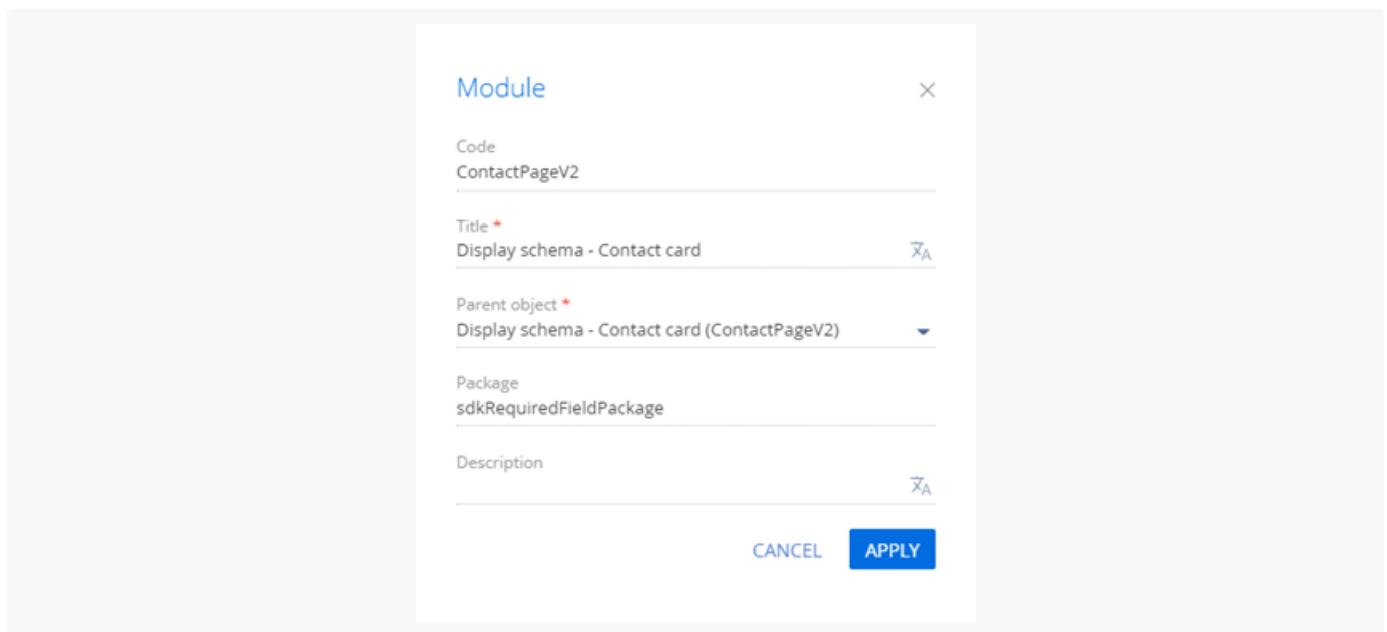
## 2. Создать схему замещающей модели представления страницы контакта

- [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**.

- а. В свойство `modules` добавьте модуль связанного пользовательского профиля контрагента.
- б. 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*/
};

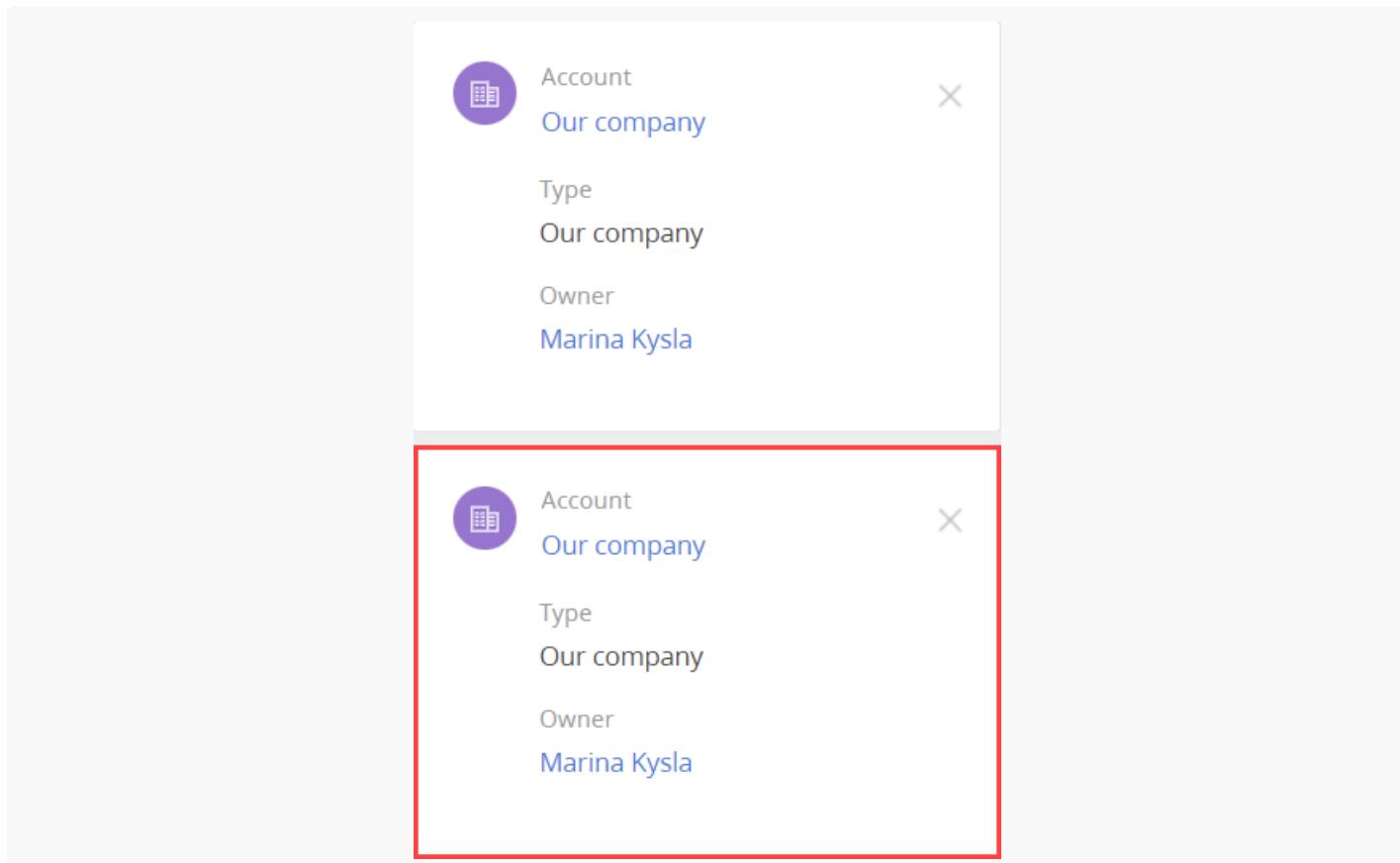
});
```

- 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>ProfileEntity ColumnChanges</code>	Широковещательное	Подписка	Обрабатывает изменения колонки сущности профиля.
<code>GetProfileEntity ColumnChanges</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](#).

# Коммуникационная панель



Основы

**Коммуникационная панель** — инструмент ведения коммуникаций с клиентами и коллегами, а также получения уведомлений в Creatio. Отображается в правой части экрана.

## Структура коммуникационной панели

**Вкладки** коммуникационной панели:

- [ *Единое окно оператора* ] ([ *Operation single window* ]) — проведение менеджером банка консультаций клиенту. Вкладка доступна в продукте Financial Services Creatio, customer journey edition. Позволяет воспользоваться различными критериями для поиска клиента банка, начать проведение консультации, отложить консультацию.
- [ *Звонки* ] ([ *CTI panel* ]) — прием входящих и выполнение исходящих звонков непосредственно в приложении. Один из инструментов телефонии в Creatio. Подробнее о телефонии читайте в блоке статей [Коннекторы к телефонии](#) и статье [Работа со звонками](#).
- [ *Email* ] — отправка и получение email-сообщений. Позволяет связывать email-сообщения с другими объектами приложения. Подробнее об управлении почтой читайте в статье [Работа с почтой](#).
- [ *Лента* ] ([ *Feed* ]) — отображение сообщений раздела [ *Лента* ] ([ *Лента* ]). Позволяет просматривать

сообщения тех каналов, на которые вы подписаны, а также добавлять новые сообщений и комментарии. Функциональность вкладки аналогична функциональности раздела [ Лента ] ([ Feed ]).

- [ Центр уведомлений ] ([ Notification center ]) — отображает уведомления о различных событиях в приложении. Подробнее об управлении уведомлениями читайте в статье [Работа с уведомлениями](#).
- [ Задачи по бизнес-процессам ] ([ Business process tasks ]) — отображает невыполненные шаги по запущенным бизнес-процессам. Подробнее об управлении запущенными бизнес-процессами читайте в статье [Работа с уведомлениями](#).

## Контейнеры коммуникационной панели

Элементы пользовательского интерфейса приложения, которые относятся к коммуникационной панели, размещены в соответствующих контейнерах. Контейнеры коммуникационной панели конфигурируются в базовой схеме `CommunicationPanel` пакета `uiV2`. Вкладка [ Звонки ] ([ CTI panel ]) конфигурируется в дочерней схеме `CtiPanel` пакета `CTIBase`.

**На заметку.** В приложении используются мета-имена html-контейнеров. На основании мета-имен приложение формирует фактические идентификаторы соответствующих html-элементов коммуникационной панели.

Основные **контейнеры коммуникационной панели** представлены на рисунке ниже.

The screenshot shows two main sections of the Communication Panel:

- Contacts**: A list of contacts with their details (Name, Job title, Account, Email, Phone numbers). Buttons for "NEW CONTACT" and "ACTIONS" are at the top.
- Active chats**: A list of active conversations with users Alexander Wilson and Alice Phillips. Each conversation shows the user's profile picture, name, time, and a message preview. To the right is a sidebar with various communication-related icons (gear, question mark, phone, envelope, speech bubble, etc.). A red box highlights this sidebar area, and a red arrow points to the "CommunicationPanelContainer" label. Another red label "rightPanel" is placed near the bottom of the sidebar.

- Контейнер кнопок коммуникационной панели (`CommunicationPanelContainer`) — содержит кнопки, которые позволяют открыть вкладки коммуникационной панели.
- Контейнер вкладок коммуникационной панели (`rightPanel`) — содержит контент текущей вкладки коммуникационной панели.

# Создать обращение по сообщению из внутренней ленты другого обращения

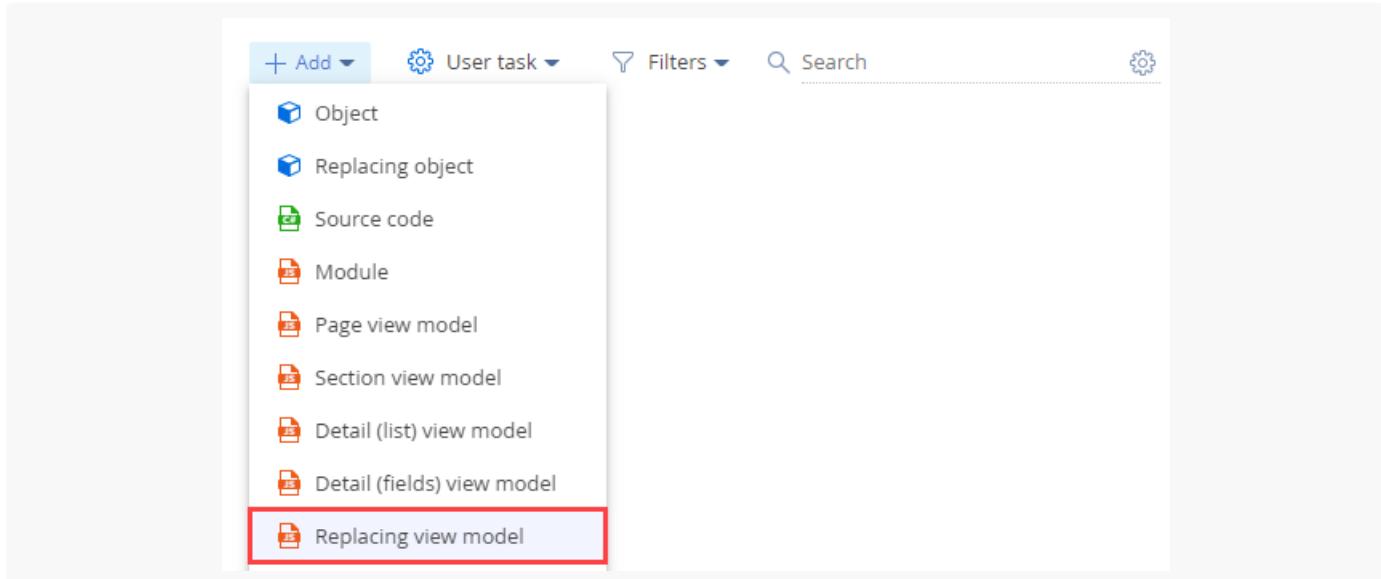
 Сложный

Пример реализован для продуктов линейки Service Creatio.

**Пример.** На вкладку [ Обработка ] ([ Processing ]) страницы обращения добавить всплывающую кнопку. Кнопка отображается при выделении текста в сообщениях с почты и портала самообслуживания, отправленных с внутренней ленты обращения. При нажатии на кнопку создается новое обращение. Поля [ Тема ] ([ Subject ]) и [ Описание ] ([ Description ]) заполняются автоматически. Значения полей — выделенный текст.

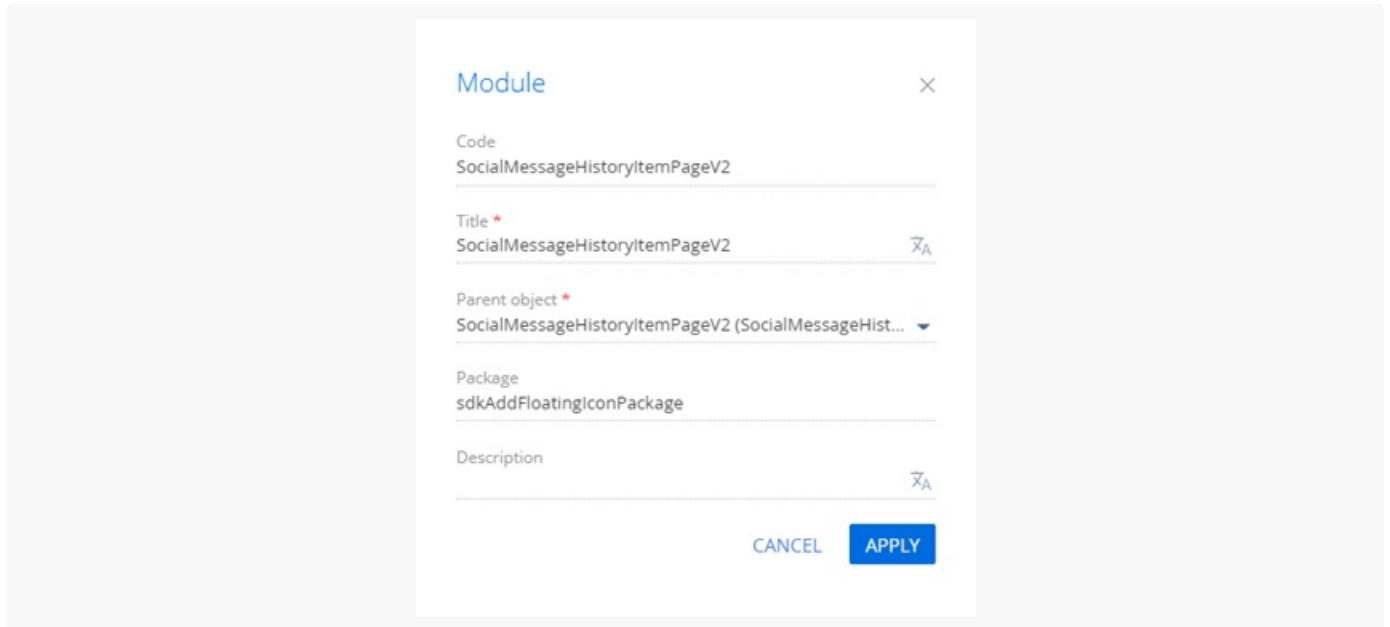
## Создать схему замещающей модели представления страницы обращения

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 ]) — "SocialMessageHistoryItemPageV2".
- [ Заголовок ] ([ Title ]) — "SocialMessageHistoryItemPageV2".
- [ Родительский объект ] ([ Parent object ]) — выберите "SocialMessageHistoryItemPageV2".



#### 4. Реализуйте логику работы всплывающей кнопки.

- В свойстве `methods` реализуйте **методы**:
  - `onSelectedTextChanged()` — передает значение выделенного текста в атрибут `HighlightedHistoryMessage`. Срабатывает при выделении текста.
  - `onSelectedTextButtonClick()` — создает обращение, тему которого получает из атрибута `HighlightedHistoryMessage`. Логика создания обращения определена в родительской схеме `BaseMessageHistory`. Срабатывает при нажатии на всплывающую кнопку.
  - `getMessageFromHistory()` — переопределенный метод родительской схемы, который получает тему выделенного сообщения.
- В массив модификаций `diff` добавьте конфигурационный объект с настройками элемента `SelectionHandlerMultiLineLabel` пакета `Message`, который реализует логику создания нового обращения на основе выделенного текста.

Исходный код схемы замещающей модели представления страницы обращения представлен ниже.

##### SocialMessageHistoryItemPageV2

```
define("SocialMessageHistoryItemPageV2", ["SocialMessageConstants", "css!SocialMessageHistory"
  return {
    /* Название схемы объекта страницы записи. */
    entitySchemaName: "BaseMessageHistory",
    /* Детали модели представления страницы записи. */
    details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
    /* Методы модели представления страницы записи. */
    methods: {
      /* Переопределение базового метода. Получает тему для выделенного сообщения. */
      getMessageFromHistory: function() {
        var message = this.get("HighlightedHistoryMessage");
      }
    }
  }
]
```

```

        if (this.isHistoryMessageEmpty(message)) {
            message = this.get("[Activity:Id:RecordId].Body");
        }
        return message;
    },
    /* Обработчик события выделения текста. */
    onSelectedTextChanged: function(text) {
        this.set("HighlightedHistoryMessage", text);
    },
    /* Обработчик нажатия всплывающей кнопки. */
    onSelectedTextButtonClick: function() {
        /* Подготовка данных по обращению из истории. */
        this.prepareCaseDataFromHistory();
    }
},
/* Отображение кнопки на странице записи. */
diff: /**SCHEMA_DIFF*/[
    /* Метаданные для добавления на страницу всплывающей кнопки. */
    {
        /* Выполняется операция изменения существующего элемента. */
        "operation": "merge",
        /* Мета-имя изменяемого компонента. */
        "name": "MessageText",
        /* Свойства, передаваемые в конструктор элемента. */
        "values": {
            /* Свойства генератора представления. */
            "generator": function() {
                return {
                    /* Значение HTML-тега id. */
                    "id": "MessageText",
                    /* Значение маркера. */
                    "markerValue": "MessageText",
                    /* Название класса компонента. */
                    "className": "Terrasoft.SelectionHandlerMultilineLabel",
                    /* Настройка CSS-стилей. */
                    "classes": {
                        "multilineLabelClass": ["messageText"]
                    },
                    /* Заголовок. */
                    "caption": {"bindTo": "Message"},
                    "showLinks": true,
                    /* Привязка события изменения выделенного текста к методу-обработку. */
                    "selectedTextChanged": {"bindTo": "onSelectedTextChanged"},
                    /* Привязка события нажатия всплывающей кнопки выделенного текста. */
                    "selectedTextHandlerButtonClick": {"bindTo": "onSelectedTextButtonClicked"},
                    /* Признак видимости всплывающей кнопки. */
                    "showFloatButton": true
                };
            }
        }
    }
];

```

```

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

```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

- Обновите страницу раздела [ Обращения ] ([ Cases ]).
- Откройте страницу обращения.

В результате выполнения примера на вкладку [ Обработка ] ([ Processing ]) страницы обращения добавлена всплывающая кнопка. Кнопка отображается при выделении текста в сообщениях с почты и портала самообслуживания, отправленных с внутренней ленты обращения. При нажатии на кнопку создается новое обращение. Поля [ Тема ] ([ Subject ]) и [ Описание ] ([ Description ]) заполняются автоматически. Значения полей — выделенный текст.

Number	Subject	Category	Assignee	Status	Resolution time
SR00000048	Consultation on functionality	Service request	Marina Kysla	Waiting for response	11/25/2021 10:00 PM
SR00000047	Unable to create an account on the site	Service request	William Walker	Resolved	11/27/2021 12:05 PM
SR00000046	What's your e-mail address?	Service request	Jason Robinson	New	11/25/2021 3:00 PM
SR00000044	Recovering password to login to the system	Service request	Marina Kysla	In progress	11/25/2021 1:14 AM
SR00000041	Missing data on hard drive	Service request	Marina Kysla	In progress	11/23/2021 1:00 AM
SR00000038	Color laserjet pro CP1025nw - printing issues	Service request	Marina Kysla	In progress	11/24/2021 12:21 PM
SR00000037	Consultation on settings	Service request	Marina Kysla	Waiting for response	11/23/2021 10:00 PM
SR00000036	An unknown error when the computer starts	Incident	Megan Lewis	In progress	11/26/2021 3:00 AM
SR00000030	The customer cannot create an account on the site.	Service request	Marina Kysla	In progress	11/27/2021 5:00 AM

## Скрыть область ленты в едином окне

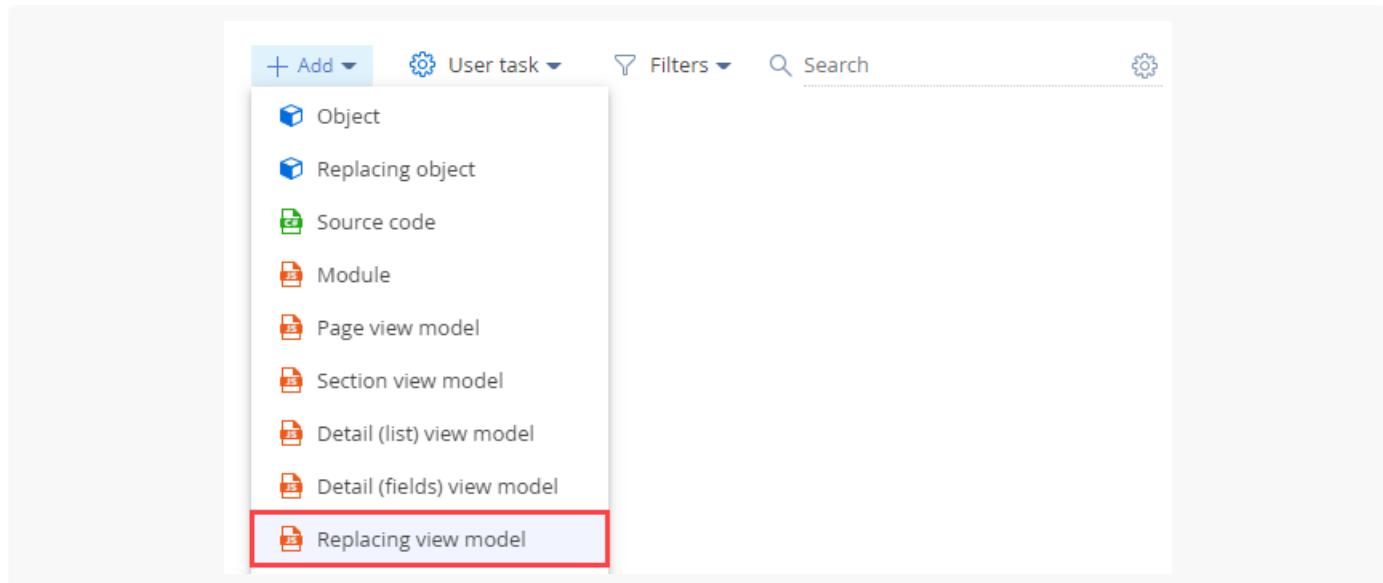
Средний

Пример реализован для рабочего места [ Контакт центр ] ([ Contact center ]).

**Пример.** В разделе [ Единое окно ] ([ Agent desktop ]) скрыть область ленты.

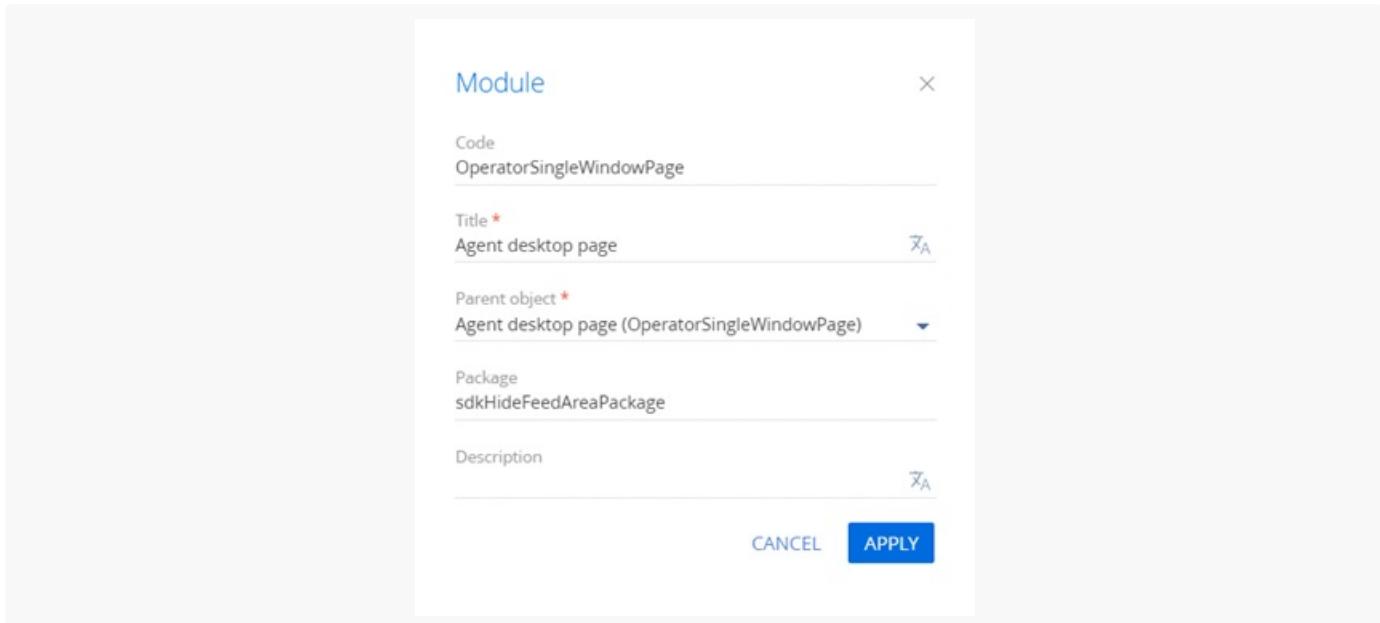
## Создать схему замещающей модели представления раздела

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 ]) — "OperatorSingleWindowPage".
- [ Заголовок ] ([ Title ]) — "Страница единого окна оператора" ("Agent desktop page").
- [ Родительский объект ] ([ Parent object ]) — выберите "OperatorSingleWindowPage".



#### 4. Реализуйте логику скрытия области ленты.

- В свойстве `methods` реализуйте метод `loadContent()` — переопределенный базовый метод, который исключает из перечня загружаемых модулей модуль ленты `ESNFeedModule`.
- В массив модификаций `diff` добавьте конфигурационный объект, который удаляет элемент со страницы.

Исходный код схемы замещающей модели представления раздела представлен ниже.

##### OperatorSingleWindowPage

```
define("OperatorSingleWindowPage", [], function() {
    return {
        /* Методы модели представления раздела. */
        methods: {
            /* Замещает базовый метод для исключения из состава загружаемых модулей модуля ленты */
            loadContent: function() {
                /* Поскольку контейнер centerContainer удален, то модуль ESNFeedModule загружается в контейнер rightContainer */
                this.loadModule("ESNFeedModule", "centerContainer");
                /* Загрузка модулей. */
                this.loadModule("SectionDashboardsModule", "rightContainer");
                this.loadModule("OperatorQueuesModule", "leftContainer");
            }
        },
        /* Отображение контейнера в разделе. */
        diff: /**SCHEMA_DIFF*/[
            /* Метаданные для удаления контейнера из раздела. */
            {
                /* Выполняется операция удаления существующего элемента. */
                "operation": "remove",
                /* Мета-имя удаляемого компонента. */
                "metaName": "centerContainer"
            }
        ]
    }
});
```

```

        "name": "centerContainer"
    }
]/**SCHEMA_DIFF*/
;
});

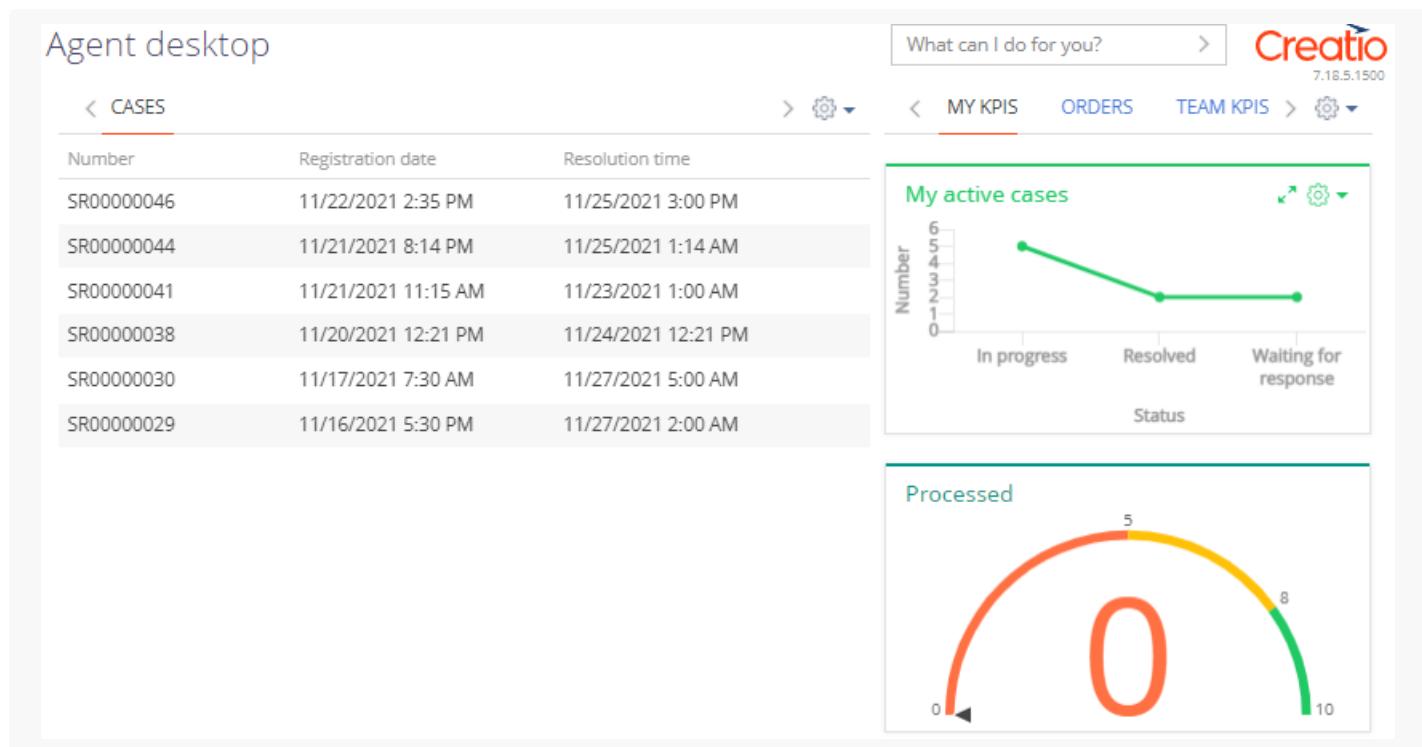
```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**, обновите страницу раздела [ Единое окно ] ([ Agent desktop ]).

В результате выполнения примера в разделе [ Единое окно ] ([ Agent desktop ]) скрыта область ленты.



## Отобразить часовой пояс контакта на вкладку коммуникационной панели

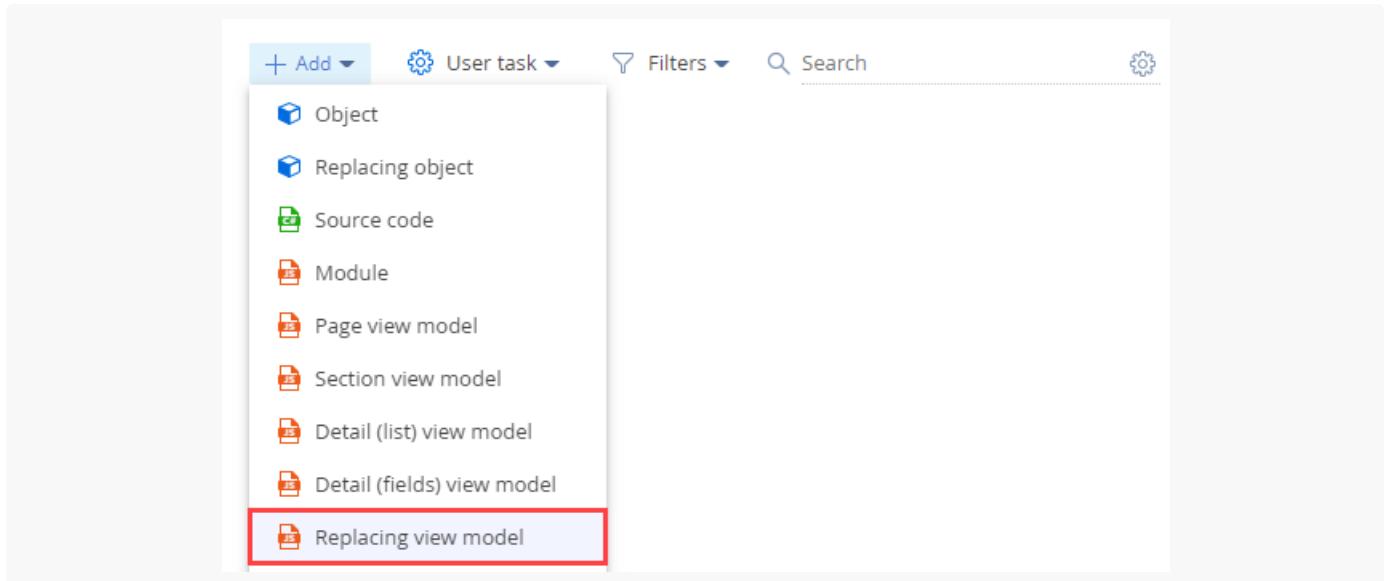
Сложный

**Пример.** При поиске контакта на вкладке [ Звонки ] ([ CTI panel ]) коммуникационной панели отображать его часовой пояс. Отображать текущее время контакта.

### 1. Создать схему замещающей модели представления

## страницы найденного абонента

- 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 ]) — "SubscriberSearchResultItem".
- [ Заголовок ] ([ Title ]) — "Схема найденного абонента" ("Found subscriber schema").
- [ Родительский объект ] ([ Parent object ]) — выберите "SubscriberSearchResultItem".

The screenshot shows a 'Module' configuration dialog. The fields are:

- Code**: SubscriberSearchResultItem
- Title \***: Found subscriber schema
- Parent object \***: Found subscriber schema (SubscriberSearchResultItem)
- Package**: sdkTimeZonePackage
- Description**: (empty)

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

- В объявлении класса модели представления в качестве зависимостей добавьте модули `TimezoneGenerator` И `TimezoneMixin`. Модуль `TimezoneGenerator` формирует элемент отображения

часового пояса контакта. Миксин `TimezoneMixin` выполняет поиск часового пояса контакта.

## 5. Реализуйте логику отображения часового пояса.

- В свойстве `attributes` добавьте атрибут `IsShowTimeZone`, который отвечает за состояние отображения элемента часового пояса.
- В свойстве `mixins` добавьте миксин `TimezoneMixin`. Для запуска поиска часового пояса контакта в метод `init()` миксина `TimezoneMixin` передайте уникальный идентификатор контакта. **Атрибуты**, которые будут установлены в результате выполнения:
  - `TimeZoneCaption` — название временной зоны контакта и текущее время.
  - `TimeZoneCity` — название города, для которого определена временная зона.
- В свойстве `methods` реализуйте **методы**:
  - `constructor()` — конструктор класса.
  - `isContactType()` — возвращает признак, который указывает, что абонент является контактом.
- В массив модификаций `diff` добавьте конфигурационный объект с настройками отображения часового пояса контакта.
  - Свойство `index` — настройка позиционирования элемента.

**Элементы** контейнера `SubscriberSearchResultItemContainer`:

- Индекс 0 — фотография абонента.
- Индекс 1 — информация об абоненте.
- Индекс 2 — телефоны абонента.

Свойству `index` массива модификаций присвойте значение 2, чтобы отобразить часовой пояс контакта между данными абонента и списком телефонных номеров.

- Свойство `wrapClass` — управление стилями. Свойство предоставляет генератор элемента. Стили текстовых элементов в схеме определяются CSS-классом `subscriber-data`.

Исходный код схемы замещающей модели представления страницы найденного абонента представлен ниже.

### SubscriberSearchResultItem

```
define("SubscriberSearchResultItem", ["TimezoneGenerator", "TimezoneMixin"], function() {
  return {
    /* Атрибуты модели представления страницы. */
    attributes: {
      /* Название атрибута, который отвечает за состояние отображения элемента часового
       * пояса контакта. */
      "IsShowTimeZone": {
        /* Тип данных колонки модели представления. */
        "dataValueType": Terrasoft.DataValueType.BOOLEAN,
        /* Тип атрибута. */
        "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
      }
    }
  }
})
```

```

    /* Значение по умолчанию. */
    "value": true
  }
},
/* Миксины модели представления страницы. */
mixins: {
  /* Подключение миксина. */
  TimezoneMixin: "Terrasoft.TimezoneMixin"
},
/* Методы модели представления страницы. */
methods: {
  /* Конструктор класса. */
  constructor: function() {
    /* Вызов базового конструктора. */
    this.callParent(arguments);
    /* Признак того, что абонент является контактом. */
    var isContact = this.isContactType();
    /* Если абонент – контакт, то элемент отображается. */
    this.set("IsShowTimeZone", isContact);
    /* Если абонент – контакт. */
    if (isContact) {
      /* Идентификатор контакта. */
      var contactId = this.get("Id");
      /* Поиск часового пояса контакта. */
      this.mixins.TimezoneMixin.init.call(this, contactId);
    }
  },
  /* Возвращает признак того, что абонент – контакт. */
  isContactType: function() {
    /* Тип абонента. */
    var type = this.get("Type");
    /* Возвращает результат сравнения. */
    return type === "Contact";
  }
},
/* Массив модификаций модели представления страницы. */
diff: [
  {
    /* Выполняется операция добавления элемента на страницу. */
    "operation": "insert",
    /* Мета-имя родительского контейнера, в который добавляется элемент. */
    "parentName": "SubscriberSearchResultItemContainer",
    /* Элемент добавляется в коллекцию элементов родительского элемента. */
    "propertyName": "items",
    /* Мета-имя добавляемого элемента. */
    "name": "TimezoneContact",
    /* Свойства, передаваемые в конструктор элемента. */
    "values": {
      /* Тип элемента. */

```

```

    "itemType": Terrasoft.ViewItemType.CONTAINER,
    /* Для формирования конфигурации представления вызывается метод генератор
    "generator": "TimezoneGenerator.generateTimeZone",
    /* Видимость контейнера привязывается к атрибуту. */
    "visible": {"bindTo": "IsShowTimeZone"},
    /* Имя CSS класса. */
    "wrapClass": ["subscriber-data", "timezone"],
    /* Привязка заголовка к атрибуту. */
    "timeZoneCaption": {"bindTo": "TimeZoneCaption"},
    /* Привязка города к атрибуту. */
    "timeZoneCity": {"bindTo": "TimeZoneCity"}
},
/* Позиция элемента в родительском контейнере. */
"index": 2
},
]
};
});

```

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

В результате в приложении отображается текущее время контакта и его город.

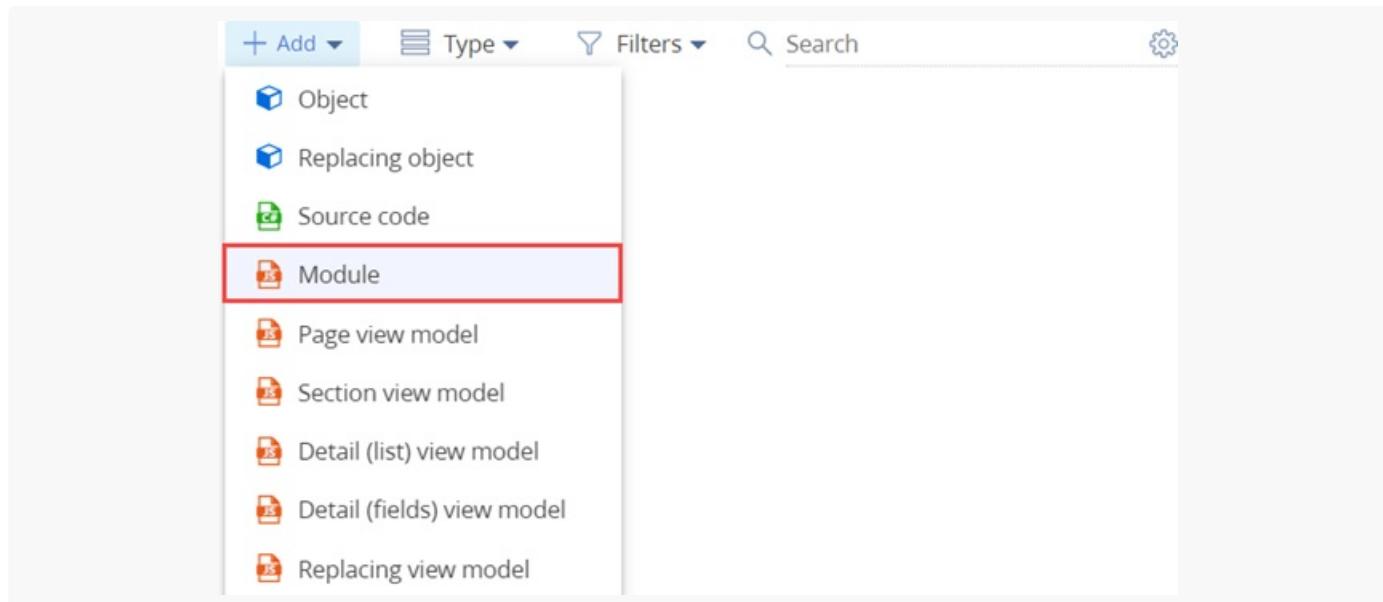
## 2. Добавить стили отображения часового пояса

Поскольку в схеме модели представления страницы найденного абонента невозможно задать стили для отображения, необходимо:

- Создать схему модуля, в которой определить стили.
- Добавить модуль со стилями в зависимости страницы найденного абонента.

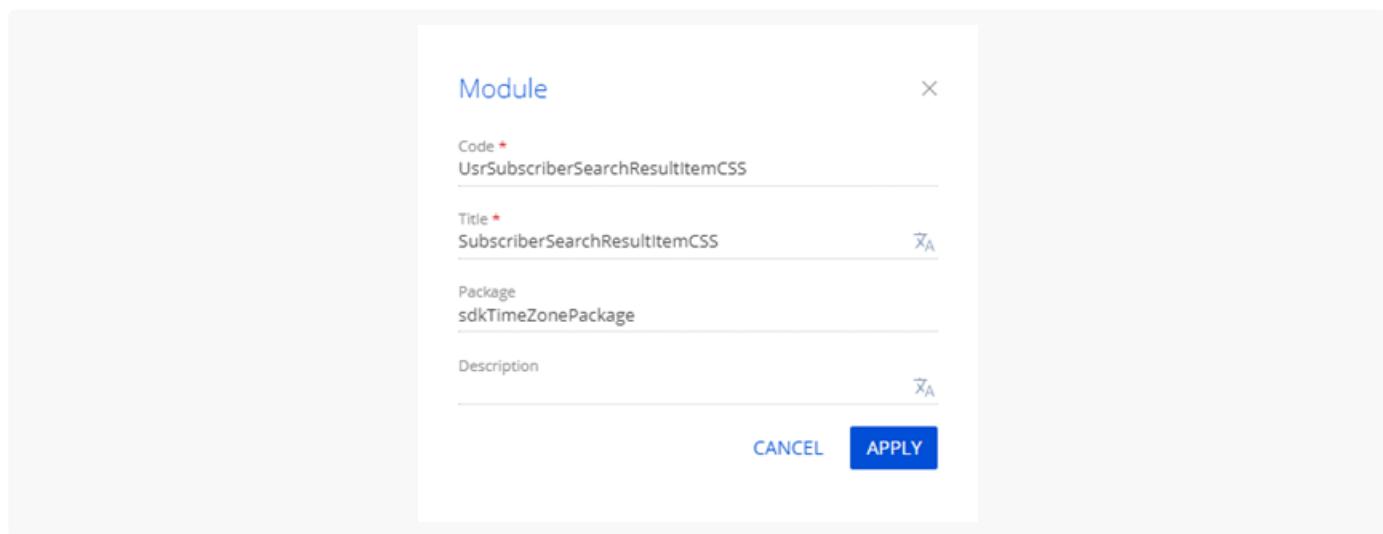
### 1. Создать схему модуля

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



### 3. Заполните **свойства схемы**:

- [Код] ([Code]) — "UsrSubscriberSearchResultItemCSS".
- [Заголовок] ([Title]) — "SubscriberSearchResultItemCSS".



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

### 4. Перейдите в узел [LESS] структуры объекта и задайте необходимые стили отображения часового пояса.

#### Настройка стилей отображения часового пояса

```
/* Настройка стилей для отображения добавляемого элемента.*/
.ctiPanelMain .search-result-items-list-container .timezone {
    /* Отступ сверху.*/
    padding-top: 13px;
    /* Смещение снизу.*/
}
```

```

margin-bottom: -10px;
}

/* Настройка стилей для отображения времени контакта.*/
.ctiPanelMain .search-result-items-list-container .timezone-caption {
    /* Отступ слева.*/
    padding-left: 10px;
    /* Цвет текста.*/
    color: rgb(255, 174, 0);
    /* Шрифт текста – жирный.*/
    font-weight: bold;
}

/* Настройка стилей для отображения города контакта.*/
.ctiPanelMain .search-result-items-list-container .timezone-city {
    /* Отступ слева.*/
    padding-left: 10px;
}

```

5. Перейдите в узел [ JS ] структуры объекта и добавьте код модуля.

#### UserSubscriberSearchResultItemCSS

```

define("UserSubscriberSearchResultItemCSS", [], function() {
    return {};
});

```

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

## 2. Модифицировать схему модели представления страницы найденного абонента

Чтобы **использовать созданный модуль и его стили** в схеме страницы найденного абонента:

1. Откройте схему `SubscriberSearchResultItem` модели представления страницы найденного абонента.
2. В зависимости схемы `SubscriberSearchResultItem` добавьте модуль `UserSubscriberSearchResultItemCSS`.

Исходный код модифицированной схемы страницы найденного абонента представлен ниже.

#### SubscriberSearchResultItem

```

define("SubscriberSearchResultItem", ["TimezoneGenerator", "TimezoneMixin", "css!UserSubscriberSearchResultItemCSS"], function() {
    return {
        /* Атрибуты модели представления страницы.*/
        attributes: {
            /* Название атрибута, который отвечает за состояние отображения элемента часового
            "IsShowTimeZone": {
                /* Тип данных колонки модели представления.*/
            }
        }
    }
});

```

```

    "dataValueType": Terrasoft.DataValueType.BOOLEAN,
    /* Тип атрибута. */
    "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN,
    /* Значение по умолчанию. */
    "value": true
  },
  /* Миксины модели представления страницы. */
  mixins: {
    /* Подключение миксина. */
    TimezoneMixin: "Terrasoft.TimezoneMixin"
  },
  /* Методы модели представления страницы. */
  methods: {
    /* Конструктор класса. */
    constructor() {
      /* Вызов базового конструктора. */
      this.callParent(arguments);
      /* Признак того, что абонент является контактом. */
      var isContact = this.isContactType();
      /* Если абонент – контакт, то элемент отображается. */
      this.set("IsShowTimeZone", isContact);
      /* Если абонент – контакт. */
      if (isContact) {
        /* Идентификатор контакта. */
        var contactId = this.get("Id");
        /* Поиск часового пояса контакта. */
        this.mixins.TimezoneMixin.init.call(this, contactId);
      }
    },
    /* Возвращает признак того, что абонент – контакт. */
    isContactType: function() {
      /* Тип абонента. */
      var type = this.get("Type");
      /* Возвращает результат сравнения. */
      return type === "Contact";
    }
  },
  /* Массив модификаций модели представления страницы. */
  diff: [
    {
      /* Выполняется операция добавления элемента на страницу. */
      "operation": "insert",
      /* Мета-имя родительского контейнера, в который добавляется элемент. */
      "parentName": "SubscriberSearchResultItemContainer",
      /* Элемент добавляется в коллекцию элементов родительского элемента. */
      "propertyName": "items",
      /* Мета-имя добавляемого элемента. */
      "name": "TimezoneContact",

```

```

    /* Свойства, передаваемые в конструктор элемента. */
    "values": {
        /* Тип элемента. */
        "itemType": Terrasoft.ViewItemType.CONTAINER,
        /* Для формирования конфигурации представления вызывается метод генератор
        "generator": "TimezoneGenerator.generateTimeZone",
        /* Видимость контейнера привязывается к атрибуту. */
        "visible": {"bindTo": "IsShowTimeZone"},
        /* Имя CSS класса. */
        "wrapClass": ["subscriber-data", "timezone"],
        /* Привязка заголовка к атрибуту. */
        "timeZoneCaption": {"bindTo": "TimeZoneCaption"},
        /* Привязка города к атрибуту. */
        "timeZoneCity": {"bindTo": "TimeZoneCity"}
    },
    /* Позиция элемента в родительском контейнере. */
    "index": 2
}
],
);
});
);

```

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

## Результат выполнения примера

Чтобы посмотреть результат выполнения примера:

В результате выполнения примера при поиске контакта на вкладке [ Звонки ] ([ CTI panel ]) коммуникационной панели отображаешься его часовой пояс. Отображается текущее время контакта.

The screenshot shows the Terrasoft CRM interface. On the left, the 'Accounts' module is open, displaying a list of accounts. One account, 'Vertigo Systems', is selected, showing its details: Primary contact 'Peter Moore', address '83 Ashton Street', and location 'London'. On the right, the 'Communication Panel (CTI panel)' is open, showing search results for 'Bruce Clayton'. The top result is 'Bruce Clayton' from 'Streamline Develop...', a Specialist. Below the search bar, there is a red box highlighting the time '3:30 AM, Atlanta' which is displayed next to a clock icon. Contact information for 'Bruce Clayton' is listed: Business phone '+1 404 532 3976' and Mobile phone '+1 404 389 0476'. There are also green call icons.

- Откройте вкладку [ Звонки ] ([ CTI panel ]) коммуникационной панели.
- Выполните поиск абонента.

# 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* ]).

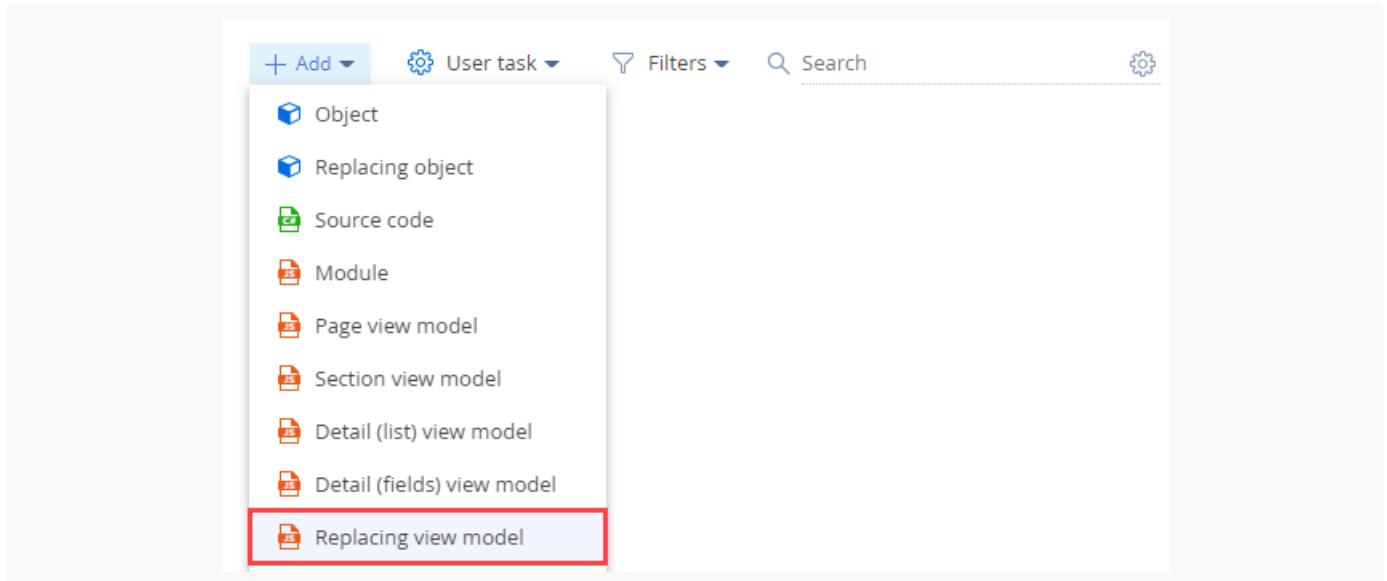
## Добавить HTML-элемент iframe

 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".

**Module**

Code  
AccountPageV2

Title \*  
Страница редактирования контрагента

Parent object  
Account edit page

Package  
sdkAddlframeIntegration

Description

CANCEL    **APPLY**

### 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");
            }
        }
    };
});
```

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

## Результат выполнения примера

Чтобы **посмотреть результат выполнения примера**:

1. Очистите кэш браузера.
2. 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.

## Командная строка



Основы

**Командная строка** — элемент интерфейса, который позволяет выполнять глобальный поиск записей в приложении, а также быстрый доступ к часто выполняемым операциям (например, открыть страницу записи, запустить бизнес-процесс и т. д.).

Командная строка представляет собой поле с текстом [ Что я могу для вас сделать? ] ([ What can I do for you? ]) и работает аналогично строке поиска в поисковых системах.

## Контейнер командной строки

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

записи.

Поле командной строки размещено в контейнере, который представлен на рисунке ниже.

The screenshot shows the Creatio Contacts application. At the top left is the 'Contacts' tab. To its right are two icons: a grid and a bar chart. In the top right corner is the 'Creatio' logo with the text '8.0.0.5434'. Below the tabs are buttons for 'NEW CONTACT' (green), 'ACTIONS ▾' (grey), and 'VIEW ▾' (grey). Underneath these are filters for 'Filters/folders ▾' and 'Tag'. The main area displays a contact record for 'Alexander Wilson'. On the left is a circular profile picture of Alexander Wilson. Next to it is his name 'Alexander Wilson' and below it, 'Account' with 'Alpha Business' and a small icon. To the right of the contact details are columns for 'Job title' (CEO), 'Business phone' (+1 212 542 4238), 'Email' (a.wilson@alphabusiness.com), and 'Mobile phone' (+1 212 854 7512). At the very top center, there is a search bar with the placeholder 'What can I do for you?' and a button with a right-pointing arrow. Below the search bar is the text 'commandLineContainer'.

Контейнер командной строки ( `commandLineContainer` ) — содержит поле командной строки.

**Составляющие**, которые реализуют работу командной строки:

- `CommandLineService` — служба, которая отслеживает введенные пользователем команды и их выполнение в приложении.
- `Command` — схема объекта, которая описывает структуру таблицы базы данных для хранения команд.
- `CommandParams` — схема объекта, которая описывает параметры команд.
- `CommandLineModule` — модуль, который содержит функции для отображения перечня доступных команд при их частичном вводе, автодополнения и другой функциональности командной строки.

## Выполнить команду из командной строки

**Команды**, которые позволяет выполнять Creatio из командной строки, приведены в таблице ниже.

## Команды командной строки

Назначение	Описание	Команда	Пример команды
<b>Навигация</b>	Перейти в любую доступную группу раздела приложения.	[ Перейти в раздел ] ([ Go to section ])	[ Перейти в раздел Контакты ] ([ Go to section Contacts ])
<b>Поиск записей</b>	Быстрый поиск, например, контакта, контрагента или записей текущего раздела. Можно использовать без ключевого слова [ Найти ] ([ Find ]).	[ Найти ] ([ Find ])	[ Найти Авдоров ] ([ Find Caleb ])
<b>Добавить запись</b>	Добавить запись в раздел приложения.	[ Добавить ] ([ Add ]) [ Создать ] ([ Create ])	[ Добавить Контакт ] ([ Add Contact ])
<b>Запустить бизнес-процесс</b>	Запустить на выполнение настроенный в приложении бизнес-процесс.	[ Запустить процесс ] ([ Run process ])	[ Запустить процесс Актуализация возраста ] ([ Run process Actualize contact age process ])
<b>Настроить пользовательскую команду</b>	Создать пользовательскую команду, которая будет запускаться из командной строки.	[ Создать пользовательскую команду ] ([ Create custom command ])	

Подробное описание команд, которые позволяет выполнять Creatio из командной строки, содержится в статье [Командная строка](#).

Чтобы **выполнить команду из командной строки**:

1. В командную строку введите команду, которую планируется выполнить.
2. Нажмите кнопку или клавишу [ Enter ].

Если вы ввели неполную команду, то в выпадающем списке приложение предлагает перечень похожих команд.

Contacts

NEW CONTACT ACTIONS ▾

Filters/folders ▾ Tag

Andrew Wayne

Account Apex Solutions

Email a.wayne@apex.co.uk

Mobile phone +44 141 258 9878

Create Account

Create Activity(Email)

Create Activity(Task)

Phone 29 1595

## Дашборды



Сложный

**Дашборд (блок итогов)** — визуальное представление разных типов аналитических данных, например, в виде графика или виджета. Для работы с аналитикой раздела необходимо перейти в представление аналитики необходимого раздела. Если же необходима работа с данными всех разделов приложения, то нужно перейти в раздел [ Итоги ].

## Структура данных для хранения информации по итогам

Раздел итогов представляет собой зависящий от прав пользователя набор элементов-вкладок. Механизм работы с итогами реализован с помощью клиентского менеджера итогов `DashboardManager` и элементов `DashboardManagerItem`, которые и представляют вкладки. За итоги в системе отвечает объект `SysDashboard`. Свойства объекта `SysDashboard` описаны в таблице.

Описание свойств объекта `SysDashboard`

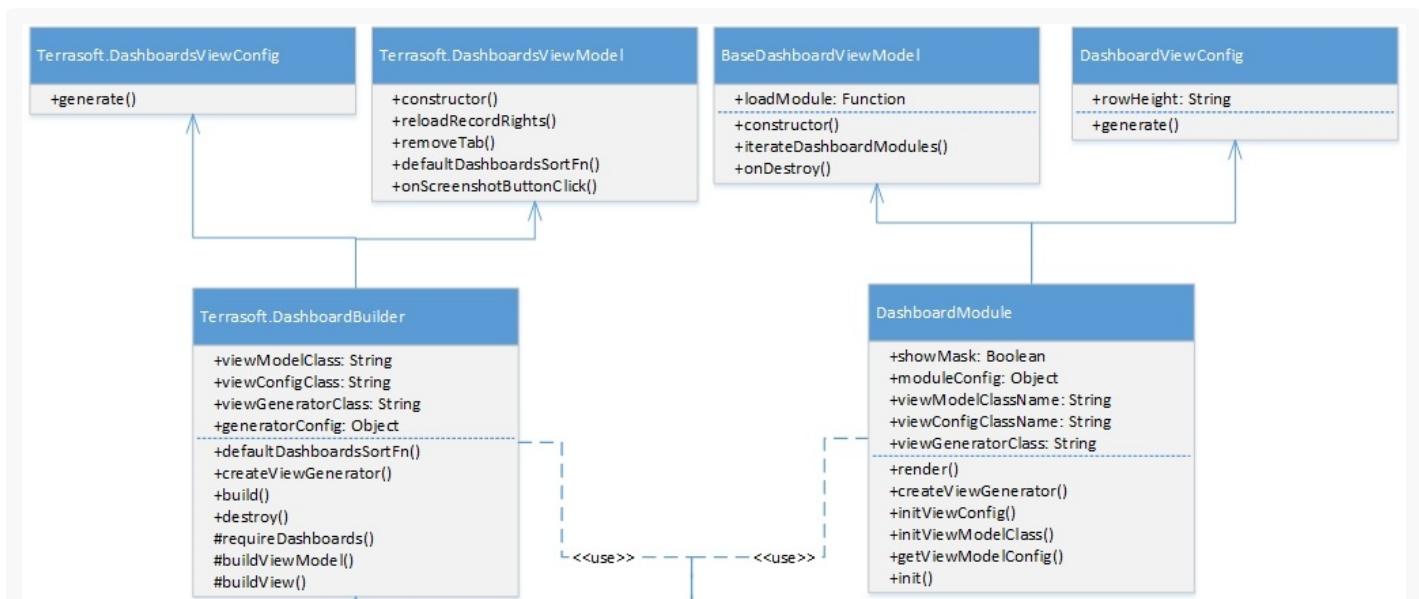
Название	Заголовок	Тип	Описание
<code>Caption</code>	Заголовок	String	Данная информация отображается в заголовке вкладки.
<code>Position</code>	Позиция	Number	Если позиция не задана, элементы отображаются в алфавитном порядке.
<code>Section</code>	Раздел	Lookup	Раздел системы.
<code>ViewConfig</code>	Конфигурация отображения элементов (дашбордов)	Array	<pre>[{     // Тип элемента (Terrasoft.ViewItemType).     itemType: "4",     // Название элемента.     name: "SomeInvoiceChart",     // Конфигурация отображения.     layout: {         ...     } }]</pre>

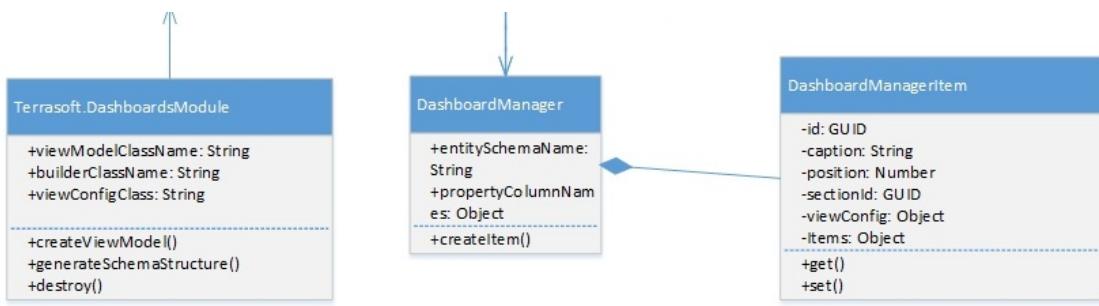
Название	Заголовок	Тип	Описание
			<pre>        columns: 4,         rows: 4,         colspan: 4,         rowspan: 4     } }, {...}]</pre>
Items	Конфигурация модулей элементов (дашбордов)	JSON Object	<pre>{     // Название элемента, для которого определяю     "SomeInvoiceChart": {         // Название модуля для отображения элемен         "widgetType": "Chart",         // Параметры, необходимые для отображения         "parameters": {             "caption": "some caption",             ...         },         {...}     } }</pre>



## Реализация функциональности в режиме просмотра итогов

Иерархия классов, реализующих функциональность в режиме просмотра итогов:





Модуль `SectionDashboardModule` :

- `SectionDashboardBuilder` — класс, инкапсулирующий в себе логику генерации представления и класса модели представления для модуля раздела итогов.
- `SectionDashboardsViewModel` — класс модели представления раздела итогов.
- `SectionDashboardsModule` — класс модуля раздела итогов.

Модуль `DashboardModule` :

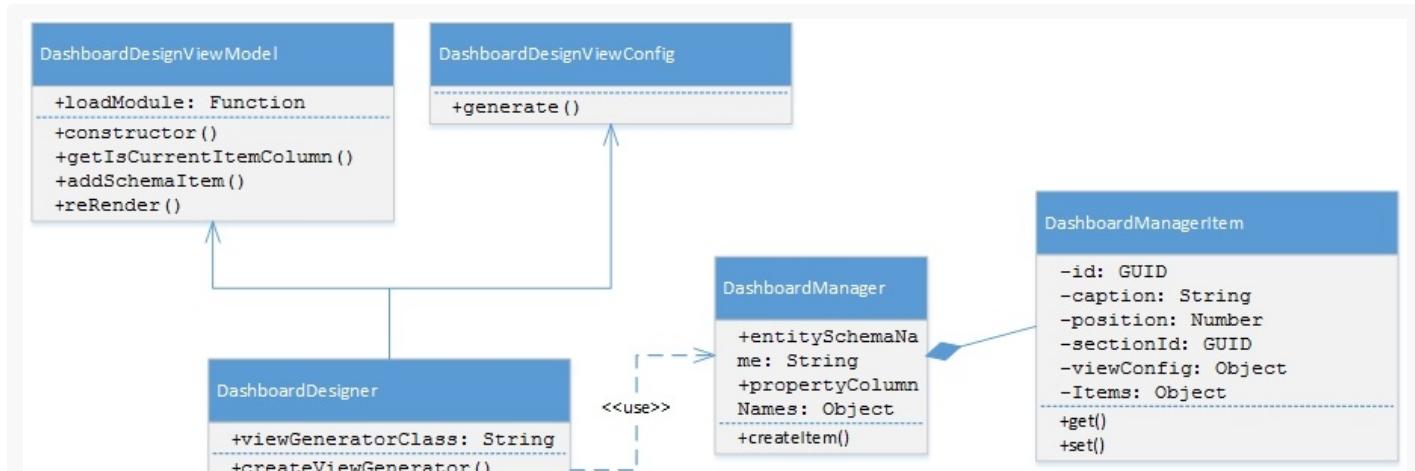
- `DashboardViewConfig` — класс, генерирующий конфигурацию представления для модуля страницы ИТОГОВ.
- `BaseDashboardViewModel` — базовый класс модели представления страницы итогов.
- `DashboardModule` — класс, содержащий функциональность по работе с модулями итогов.

Модуль `DashboardBuilder` :

- `DashboardsViewConfig` — класс, генерирующий конфигурацию представления для модуля итогов.
- `BaseDashboardsViewModel` — базовый класс модели представления итогов.
- `DashboardBuilder` — класс построения модуля итогов.

## Реализация функциональности в режиме настройки ИТОГОВ

Иерархия классов, реализующих функциональность в режиме настройки итогов:



```
+-----+
+generateSchemaStructure()
+render()
+init()
+destroy()
```

Модуль `DashboardDesigner` :

- `DashboardDesignerViewConfig` — класс, генерирующий конфигурацию представления для модуля дизайнера итогов.
- `DashboardDesignerViewModel` — класс модели представления дизайнера итогов.
- `DashboardDesigner` — класс визуального модуля итогов.

## Базовые классы, реализующие функциональность дашборда

`BaseWidgetViewModelClass` — базовый класс модели представления дашбордов. Для использования класса необходимо зарегистрировать в модуле такие сообщения:

- `GetHistoryState (publish; ptp);`
- `ReplaceHistoryState (publish; broadcast);`
- `HistoryStateChanged (subscribe; broadcast);`
- `GetWidgetParameters (subscribe; ptp);`
- `PushWidgetParameters (subscribe; ptp)` — если используется получение параметров от модулей (`useCustomParameterMethods = true`) .

`BaseWidgetDesigner` — базовая схема представления настройки дашбордов. Основные методы:

- `getWidgetConfig()` — возвращает объект актуальных настроек дашборда.
- `getWidgetConfigMessage()` — возвращает название сообщения получения настроек модуля дашборда.
- `getWidgetModuleName()` — возвращает название модуля дашборда.
- `getWidgetRefreshMessage()` — возвращает название сообщения обновления дашборда.
- `getWidgetModulePropertiesTranslator()` — возвращает объект соотношения свойств модуля дашборда и модуля настройки дашборда.

`BaseAggregationWidgetDesigner` — содержит методы для работы с агрегирующими колонками и типами агрегации.

`DashboardEnums` — содержит перечисление свойств, используемых в дашбордах.

`Terrasoft.DashboardEnums.WidgetType` — содержит конфигурацию дашбордов для режима просмотра (view) и режима настройки (design) итогов. Конфигурация определяется следующими свойствами:

- `moduleName` — название модуля дашборда.
- `configurationMessage` — название сообщения получения настроек модуля.
- `resultMessage` — название сообщения для отдачи параметров настройки модуля дизайнера дашборда.
- `stateConfig / stateConfig` — название схемы дизайнера панели.

— `staticconfig` (статичный) — настройка класса дашборда.

## Виды дашбордов

### График

[График](#) в наглядной форме отображает множественные данные из системы. С помощью графика можно отобразить, например, распределение контрагентов по типам. График может отображать информацию в виде диаграмм разных типов, либо в виде реестра данных.



### Классы, реализующие функциональность графиков

`ChartViewModel` — модель представления графиков.

`ChartViewConfig` — генерирует конфигурацию представления модуля графика.

`ChartModule` — модуль, предназначенный для работы с графиками.

`ChartDesigner` — схема представления страницы графиков.

`ChartModuleHelper` — используется для формирования запроса с помощью объекта `Terrasoft.EntitySchemaQuery`.

`ChartDrillDownProvider` — содержит методы для работы с функциональностью углубления в элементы (для работы с сериями в графиках).

### Параметры настройки графика

Для настройки графика необходимо в конфигурацию модулей дашбордов добавить конфигурационный JSON-объект с настройками свойств графика. Конфигурация модуля дашборда определяется свойством `Items` объекта `sysDashboard`. Подробнее об объекте `SysDashboard` и его свойствах можно узнать из [статьи](#).

В конфигурационном JSON-объекте с настройками дашборда необходимо установить значение "Chart" свойству `type`. Кроме того, свойству `name` нужно присвоить объект с необходимыми

свойству `widgetType`. Кроме того, свойству `parameters` нужно присвоить объект с необходимыми параметрами. Возможные параметры графика приведены в таблице.

Параметры настройки графика

Название	Тип	Описание
<code>seriesConfig</code>	<code>object</code>	Настройки вложенного графика из серии.
<code>orderBy</code>	<code>string</code>	Поле сортировки.
<code>orderDirection</code>	<code>string</code>	Направление сортировки.
<code>caption</code>	<code>string</code>	Заголовок графика.
<code>sectionId</code>	<code>string</code>	Идентификатор раздела.
<code>xAxisDefaultCaption</code>	<code>string</code>	Заголовок оси X по умолчанию.
<code>yAxisDefaultCaption</code>	<code>string</code>	Заголовок оси Y по умолчанию.
<code>primaryColumnName</code>	<code>string</code>	Название первичной колонки. По умолчанию первичной является колонка <code>Id</code> .
<code>yAxisConfig</code>	<code>object</code>	Массив настроек подписи оси Y.
<code>schemaName</code>	<code>string</code>	Объект, по которому строится график.
<code>sectionBindingColumn</code>	<code>string</code>	Колонка связи с разделом.
<code>func</code>	<code>string</code>	Агрегирующая функция.
<code>type</code>	<code>string</code>	Тип графика.
<code>XAxisCaption</code>	<code>string</code>	Заголовок оси X.
<code>YAxisCaption</code>	<code>string</code>	Заголовок оси Y.
<code>xAxisColumn</code>	<code>string</code>	Колонка группировки оси X.
<code>yAxisColumn</code>	<code>string</code>	Колонка группировки оси Y.
<code>styleColor</code>	<code>string</code>	Цвет графика.
<code>filterData</code>	<code>object</code>	Настройка фильтрации.

## I Показатель

[Дашборд “Показатель”](#) отображает расчетное числовое значение или дату по определенным данным системы, например, общее количество сотрудников отдела..



### Классы, реализующие функциональность показателей

`IndicatorViewModel` — модель представления показателя.

`IndicatorViewConfig` — генерирует конфигурацию представления модуля показателя.

`IndicatorModule` — модуль, предназначенный для работы с показателями.

`IndicatorDesigner` — схема представления страницы показателя.

### Параметры настройки показателя

Для настройки показателя необходимо в конфигурацию модулей дашбордов добавить конфигурационный JSON-объект с настройками свойств показателя. Конфигурация модуля дашборда определяется свойством `Items` объекта `sysDashboard`. Подробнее об объекте `SysDashboard` и его свойствах можно узнать из [статьи](#).

В конфигурационном JSON-объекте с настройками дашборда необходимо установить значение "Indicator" свойству `widgetType`. Кроме того, свойству `parameters` нужно присвоить объект с необходимыми параметрами. Возможные параметры показателя приведены в таблице.

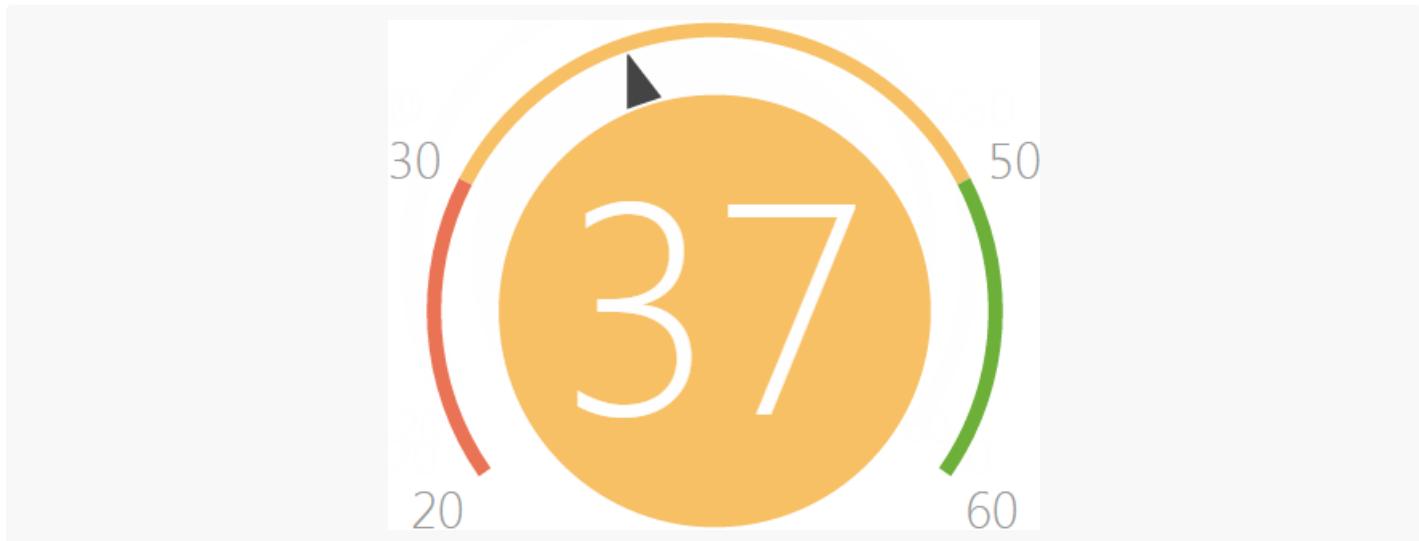
Параметры настройки показателя

Название	Тип	Описание
<code>caption</code>	<code>string</code>	Заголовок показателя.
<code>sectionId</code>	<code>string</code>	Идентификатор раздела.
<code>entitySchemaName</code>	<code>string</code>	Объект, по которому строится показатель.
<code>sectionBindingColumn</code>	<code>string</code>	Колонка связи с разделом.
<code>columnName</code>	<code>string</code>	Название агрегирующей колонки.
<code>format</code>	<code>object</code>	Формат показателя.

<code>filterData</code>	<code>object</code>	Настройка фильтрации.
<code>aggregationType</code>	<code>number</code>	Тип агрегирующей функции.
<code>style</code>	<code>string</code>	Цвет показателя.

## Шкала

[Шкала](#) отображает число, полученное в результате запроса к данным системы, относительно нормативных значений. С помощью шкалы можно отобразить, например, реальное количество выполненных активностей относительно запланированного количества.



Классы, реализующие функциональность дашборда "Шкала"

`GaugeViewModel` — модель представления шкалы.

`GaugeViewConfig` — генерирует конфигурацию представления модуля шкалы.

`GaugeModule` — модуль, предназначенный для работы со шкалой.

`GaugeChart` — реализует компонент графика типа шкала.

`GaugeDesigner` — схема представления страницы шкалы.

## Параметры настройки шкалы

Для настройки шкалы необходимо в конфигурацию модулей дашбордов добавить конфигурационный JSON-объект с настройками свойств шкалы. Конфигурация модуля дашборда определяется свойством `Items` объекта `SysDashboard`. Подробнее об объекте `SysDashboard` и его свойствах можно узнать из [статьи](#).

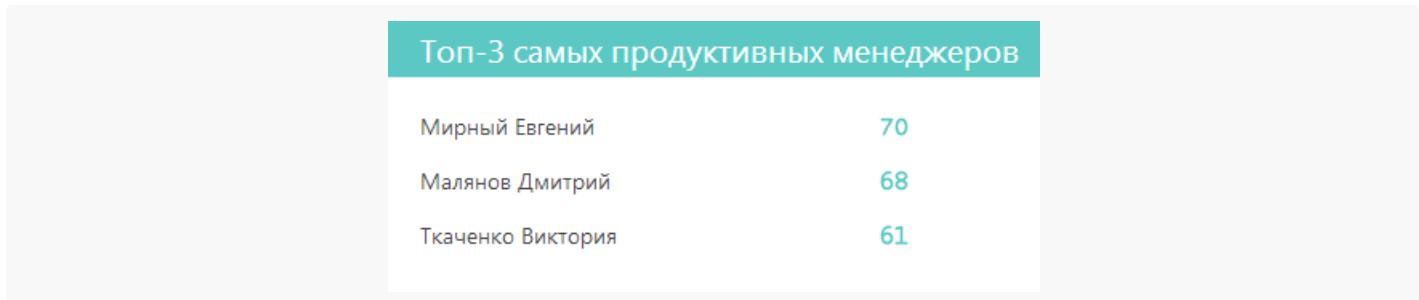
В конфигурационном JSON-объекте с настройками дашборда необходимо установить значение "Gauge" свойству `widgetType`. Кроме того, свойству `parameters` нужно присвоить объект с необходимыми параметрами. Возможные параметры шкалы приведены в таблице.

### Параметры настройки шкалы

Название	Тип	Описание
caption	string	Заголовок шкалы.
sectionId	string	Идентификатор раздела.
entitySchemaName	string	Объект, по которому строится шкала.
sectionBindingColumn	string	Колонка связи с разделом.

## Список

[Список](#) отображает информацию из системы в виде списка с заданным количеством позиций. С помощью списка можно отобразить, например, десятку самых продуктивных менеджеров по количеству закрытых сделок.



## Классы, реализующие функциональность списков

`DashboardGridViewModel` — модель представления списка.

`DashboardGridViewConfig` — генерирует конфигурацию представления модуля списка.

`DashboardGridModule` — модуль, предназначенный для работы со списками.

`DashboardGridDesigner` — схема представления страницы списка.

## Параметры настройки списка

Для настройки списка необходимо в конфигурацию модулей дашбордов добавить конфигурационный JSON-объект с настройками свойств списка. Конфигурация модуля дашборда определяется свойством `Items` объекта `SysDashboard`. Подробнее об объекте `SysDashboard` и его свойствах можно узнать из [статьи](#).

В конфигурационном JSON-объекте с настройками дашборда необходимо установить значение "DashboardGrid" свойству `widgetType`. А также свойству `parameters` нужно присвоить объект с необходимыми параметрами. Возможные параметры списка приведены в таблице.

### Параметры настройки списка

Название	Тип	Описание
caption	string	Заголовок шкалы.
sectionId	string	Идентификатор раздела.
entitySchemaName	string	Объект, по которому строится шкала.

<code>caption</code>	<code>string</code>	Заголовок списка.
<code>sectionBindingColumn</code>	<code>string</code>	Колонка связи с разделом.
<code>filterData</code>	<code>object</code>	Настройка фильтрации.
<code>sectionId</code>	<code>string</code>	Идентификатор раздела.
<code>entitySchemaName</code>	<code>string</code>	Объект, по которому строится список.
<code>style</code>	<code>string</code>	Цвет списка.
<code>orderDirection</code>	<code>number</code>	Направление сортировки (1 — по возрастанию, 2 — по убыванию).
<code>orderColumn</code>	<code>string</code>	Колонка, по которой сортируется список.
<code>rowCount</code>	<code>number</code>	Количество рядов для отображения.
<code>gridConfig</code>	<code>object</code>	Конфигурация списка.

## Web-страница

[Дашборд "Web-страница"](#) предназначен для отображения интернет-страниц на панели итогов. Например, это может быть онлайн-калькулятор валют или ваш корпоративный сайт.

### Классы, реализующие функциональность Web-страниц

`WebPageViewModel` — модель представления Web-страницы.

`WebPageViewConfig` — генерирует конфигурацию представления модуля Web-страницы.

`WebPageModule` — модуль, предназначенный для работы с Web-страницами.

`WebPageDesigner` — схема представления страницы дашборда Web-страницы.

### Параметры настройки Web-страницы

Для настройки Web-страницы необходимо в конфигурацию модулей дашбордов добавить конфигурационный JSON-объект с настройками свойств Web-страницы. Конфигурация модуля дашборда определяется свойством `Items` объекта `SysDashboard`. Подробнее об объекте `SysDashboard` и его свойствах можно узнать из [статьи](#).

В конфигурационном JSON-объекте с настройками дашборда необходимо установить значение "WebPage" свойству `widgetType`. Кроме того, свойству `parameters` нужно присвоить объект с необходимыми параметрами. Возможные параметры Web-страницы приведены в таблице.

#### Параметры настройки Web-страницы

Название	Тип	Описание
----------	-----	----------

<code>caption</code>	<code>string</code>	Заголовок дашборда Web-страницы.
<code>sectionId</code>	<code>string</code>	Идентификатор раздела.
<code>url</code>	<code>string</code>	Ссылка на Web-страницу.
<code>style</code>	<code>string</code>	CSS стили дашборда Web-страницы.

## Воронка продаж

[Дашборд "Воронка продаж"](#) используется для анализа динамики продвижения продаж по стадиям.

Классы, реализующие функциональность воронки продаж

`OpportunityFunnelChart` — класс, унаследованный от `Chart`.

### Параметры настройки воронки продаж

Для настройки воронки продаж необходимо в конфигурацию модулей дашбордов добавить конфигурационный JSON-объект с настройками свойств воронки. Конфигурация модуля дашборда определяется свойством `Items` объекта `sysDashboard`. Подробнее об объекте `SysDashboard` и его свойствах можно узнать из [статьи](#).

В конфигурационном JSON-объекте с настройками дашборда необходимо установить значение "OpportunityFunnel" свойству `widgetType`. Кроме того, свойству `parameters` нужно присвоить объект с необходимыми параметрами. Возможные параметры воронки продаж приведены в таблице.

Параметры настройки воронки продаж

Название	Тип	Описание
<code>caption</code>	<code>string</code>	Заголовок дашборда воронки.
<code>sectionId</code>	<code>string</code>	Идентификатор раздела.
<code>defPeriod</code>	<code>string</code>	Период воронки (по умолчанию последняя неделя).
<code>sectionBindingColumn</code>	<code>string</code>	Колонка связи с разделом.
<code>type</code>	<code>string</code>	Тип графика ("funnel").
<code>filterData</code>	<code>object</code>	Настройка фильтрации.

## Изменить расчеты в воронке продаж



Существует возможность изменять расчеты в воронке, которая подключена к объекту раздела [Продажи], для отображения по нему аналитики. Для этого нужно создать новый модуль для расчетов и заменить клиентскую схему отображения воронки.

## Последовательность действий для изменения расчетов в воронке:

1. Создать новый класс, который наследуется от `FunnelBaseDataProvider` и задать логику расчетов.
2. Создать схему замещающей модели представления `FunnelChartSchema` и использовать в ней новый класс расчетов.

## Пример изменения отображения расчетов в воронке продаж в срезе “по количеству”

### Описание кейса

Необходимо изменить расчеты воронки, поменяв отображение количества продаж на количество продуктов, добавленных в продажи.

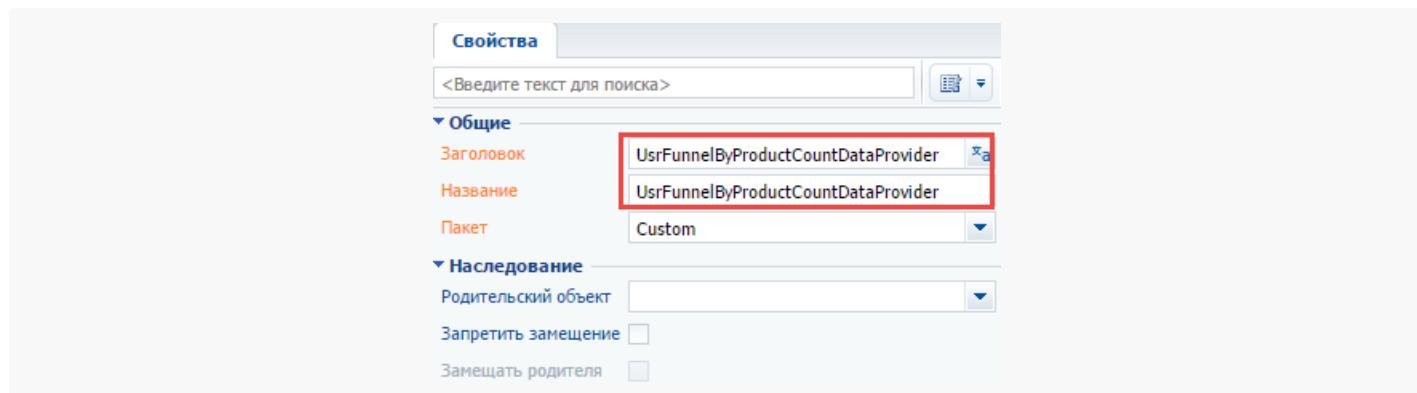
### Алгоритм реализации кейса

#### 1. В пользовательском пакете создать новый модуль

В пользовательском пакете необходимо создать новый клиентский модуль провайдера расчетов. Провайдер расчетов — это класс, который отвечает за выборку, фильтрацию и обработку данных для графика воронки.

В качестве имени и заголовка для создаваемого модуля необходимо указать, например, `UsrFunnelByProductCountDataProvider` (рис. 1).

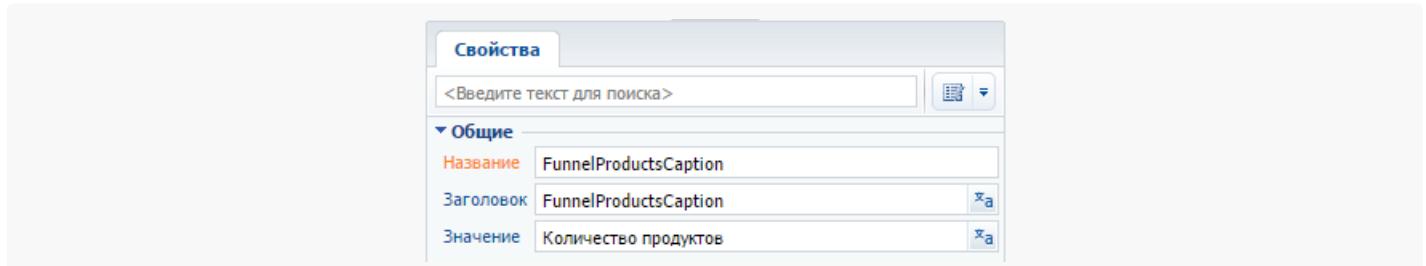
Рис. 1. — Свойства модуля провайдера расчетов



## 2. Добавить локализуемые строки

В коллекцию локализуемых строк созданного клиентского модуля необходимо добавить строку со значением *Количество продуктов*. Для этого, щелкнув правой клавишей мыши по узлу структуры [ *LocalizableStrings* ], нужно из всплывающего меню выбрать команду [ *Добавить* ]. Для созданной строки нужно установить свойства так, как показано на рисунке 2.

Рис. 2. — Свойства локализуемой строки



Также аналогичным образом необходимо добавить локализуемую строку `CntOpportunity` со значением *Количество продаж*.

## 3. Добавить реализацию в модуль провайдера

Для изменения расчетов воронки нужно переопределить:

- метод формирования колонок `addQueryColumns` для выборки данных;
- методы обработки данных выборки.

Для обработки одной записи выборки нужно определить метод `getSeriesDataConfigByItem`. Для обработки всей коллекции нужно определить метод `prepareFunnelResponseCollection`. Для желаемой фильтрации записей нужно переопределить метод `applyFunnelPeriodFilters`.

Ниже приведен исходный код нового модуля провайдера расчетов воронки продаж.

```
define("UsrFunnelByProductCountDataProvider", ["ext-base", "terrasoft", "UsrFunnelByProductCount",
    "FunnelBaseDataProvider"],
    function(Ext, Terrasoft, resources) {
        // Определение нового провайдера расчетов.
        Ext.define("Terrasoft.configuration.UsrFunnelByProductCountDataProvider", {
            // Наследование от базового провайдера.
            extend: "Terrasoft.FunnelBaseDataProvider",
            // Сокращенное имя нового провайдера
            alternateClassName: "Terrasoft.UsrFunnelByProductCountDataProvider",
            // Метод для обработки всей коллекции.
            prepareFunnelResponseCollection: function(collection) {
                this.callParent(arguments);
            },
            // Расширение метода базового модуля FunnelBaseDataProvider.
            // Устанавливает колонку количества продуктов для выборки данных.
            addQueryColumns: function(entitySchemaQuery) {
```

```

// Вызов родительского метода.
this.callParent(arguments);
// Добавляет в выборку колонку количества продуктов
entitySchemaQuery.addAggregationSchemaColumn("[OpportunityProductInterest:Opport
    Terrasoft.AggregationType.SUM, "ProductsAmount");
},
// Расширение метода базового класса FunnelBaseDataProvider.
// Устанавливает фильтрацию для выборки
applyFunnelPeriodFilters: function(filterGroup) {
    // Вызов родительского метода.
    this.callParent(arguments);
    // Создает группу фильтров.
    var endStageFilterGroup = Terrasoft.createFilterGroup();
    // Устанавливает тип оператора для группы.
    endStageFilterGroup.logicalOperation = Terrasoft.LogicalOperatorType.OR;
    // Добавляет фильтр, который указывает, что стадия на продаже еще не окончена.
    endStageFilterGroup.addItem(
        Terrasoft.createColumnIsNullFilter(this.getDetailColumnPath("DueDate")));
    // Добавляет фильтр, который указывает, что стадия на продаже является завершающей.
    endStageFilterGroup.addItem(
        Terrasoft.createColumnFilterWithParameter(Terrasoft.ComparisonType.EQUAL,
            this.getDetailColumnPath("Stage.End"), true, Terrasoft.DataValueType.BOOL));
    filterGroup.addItem(endStageFilterGroup);
},
// Расширение метода базового модуля FunnelBaseDataProvider.
// Обрабатывает данные для стадий в воронке.
getSeriesDataConfigByItem: function(responseItem) {
    // Объект, хранящий локализуемые строки.
    var lcz = resources.localizableStrings;
    // Получает объект данных стадии из родительского метода.
    var config = this.callParent(arguments);
    // Получает данные о количестве продуктов в продаже из результата выборки.
    var products = responseItem.get("ProductsAmount");
    products = Ext.isNumber(products) ? products : 0;
    // Форматирует строки.
    var name = Ext.String.format("{0}<br/>{1}: {2}<br/>{3}: {4}",
        config.menuHeaderValue, lcz.CntOpportunity, config.y, lcz.FunnelProductsCaption);
    var displayValue = Ext.String.format("<br/>{0}: {1}", lcz.FunnelProductsCaption,
    // Устанавливает новые данные в объект данных и возвращает его.
    return Ext.apply(config, {
        name: name,
        displayValue: displayValue
    });
}
});
});
});

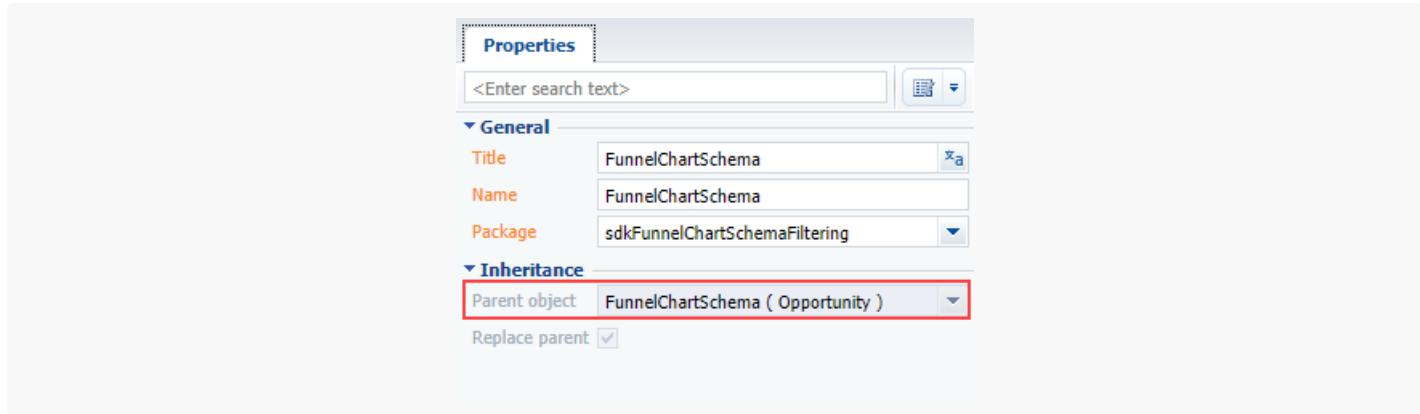
```

## 4. Создать схему замещающей модели представления для графика воронки

Для того, чтобы в расчетах использовался новый модуль провайдера, нужно переопределить метод формирования провайдеров расчетов воронки продаж.

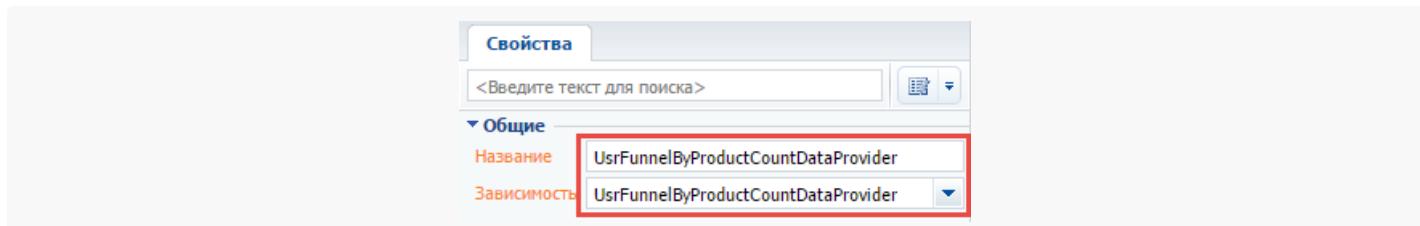
Для этого необходимо создать схему замещающей модели представления и указать ей схему `FunnelChartSchema` в качестве родительского объекта (рис. 3).

Рис. 3. — Свойства схемы замещающей модели представления



Также необходимо добавить в зависимости новый модуль для расчетов (секция `Dependencies`), указав его имя в поля [ *Зависимость* ] и [ *Название* ] значения `UsrFunnelByProductCountDataProvider` (рис. 4).

Рис. 4. — Свойства зависимости схемы воронки



## 5. Указать новый провайдер расчетов в замещенной схеме воронки

Для этого в замещенной схеме необходимо переопределить метод `getProvidersCollectionConfig`, возвращающий конфигурационный объект с коллекцией провайдеров.

```
define("FunnelChartSchema", ["UsrFunnelByProductCountDataProvider"],
  function() {
    return {
      entitySchemaName: "Opportunity",
      methods: {
        getProvidersCollectionConfig: function() {
          // Вызывает родительский метод.
          // Возвращает массив провайдеров.
        }
      }
    }
  }
)
```

```

var config = this.callParent();
// Ищет провайдер данных в срезе по количеству продаж.
var byCount = Terrasoft.findItem(config, {tag: "byNumberConversion"});
// Заменяет на новый класс.
byCount.item.className = "Terrasoft.UsrFunnelByProductCountDataProvider";
return config;
}
}
};

});
```

После сохранения схемы в воронке продаж будет использоваться новый модуль расчетов воронки, а воронка будет отображать общее количество продуктов по стадиям (рис. 5).

Рис. 5. — Воронка продаж с отображением количества продуктов, добавленных в продажи



## Подключить дополнительную фильтрацию к воронке

 Сложный

Существует возможность подключить дополнительную фильтрацию для расчетов в воронке.

Для этого необходимо реализовать следующую последовательность действий:

1. Создать унаследованный от провайдера расчетов новый класс, в котором нужно реализовать необходимую логику фильтрации.
2. Создать схему замещающей модели представления `FunnelChartSchema` и использовать в ней новый класс расчетов.

## Описание примера

Необходимо добавить фильтрацию к расчетам воронки продаж в срезе "по количеству" для выбора только тех продаж, у которых в поле [ Клиент ] указан контрагент.

## Исходный код

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

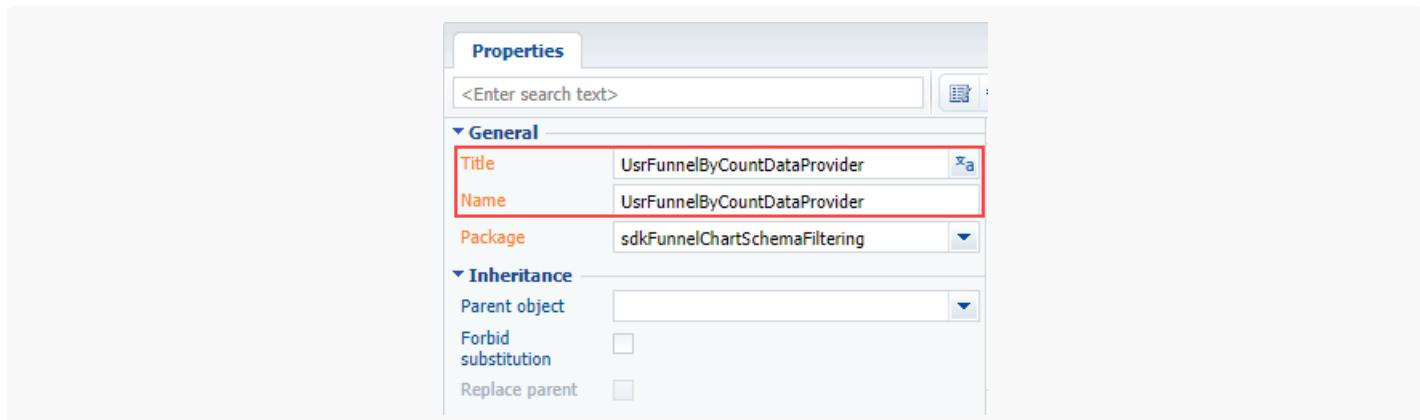
## Алгоритм реализации примера

### 1. В пользовательском пакете создать новый модуль

В пользовательском пакете создайте новый клиентский модуль провайдера расчетов. Провайдер расчетов — это класс, который отвечает за выборку, фильтрацию и обработку данных для графика воронки.

В качестве имени и заголовка для создаваемого модуля укажите, например, `UsrFunnelByCountDataProvider` (рис. 1).

Рис. 1. — Свойства модуля провайдера расчетов



### 2. Определить новый класс провайдера и задать логику фильтрации

Для добавления фильтрации к расчетам в срезе "по количеству" нужно наследовать созданный класс от

класса `FunnelByCountDataProvider` и переопределить метод `getFunnelFixedFilters`.

Исходный код модуля:

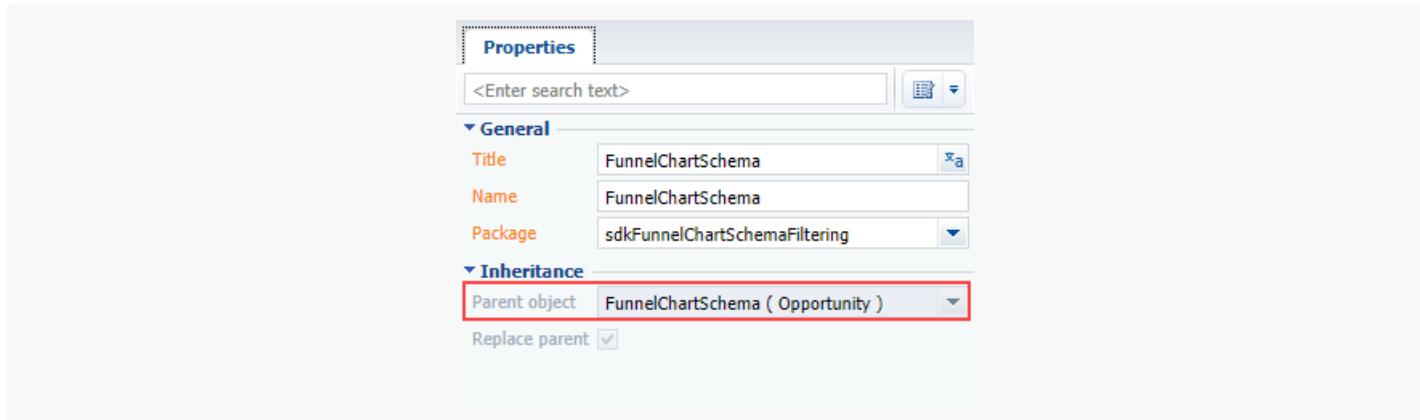
```
define("UsrFunnelByCountDataProvider", ["ext-base",
    "terrasoft", "UsrFunnelByCountDataProviderResources",
    "FunnelByCountDataProvider"],
    function(Ext, Terrasoft, resources) {
        // Определение нового провайдера расчетов.
        Ext.define("Terrasoft.configuration.UsrFunnelByCountDataProvider", {
            // Наследование от провайдера 'по количеству'.
            extend: "Terrasoft.FunnelByCountDataProvider",
            // Сокращенное имя нового провайдера.
            alternateClassName: "Terrasoft.UsrFunnelByCountDataProvider",
            // Расширение метода базового модуля FunnelByCountDataProvider.
            // Возвращает фильтры для выборки.
            getFunnelFixedFilters: function() {
                // Вызов родительского метода.
                var esqFiltersGroup = this.callParent(arguments);
                // Добавляет фильтр, который указывает, что в продаже клиентом указан контрагент
                esqFiltersGroup.addItem(
                    Terrasoft.createColumnIsNotNullFilter("Account"));
                return esqFiltersGroup;
            }
        });
    });
});
```

После внесения изменений сохраните модуль.

### 3. В пользовательском пакете реализовать модуль для графика воронки

Чтобы в расчетах использовался новый модуль провайдера, создайте схему замещающей модели представления, в которой в качестве родительской схемы укажите `FunnelChartSchema` из пакета `Opportunity` (рис. 2).

Рис. 2. — Свойства замещающего модуля



#### 4. Указать новый провайдер расчетов в замещенной схеме воронки

Для этого в замещенной схеме переопределите метод формирования провайдеров расчетов воронки продаж и укажите новый класс провайдера расчетов.

Исходный код схемы замещающей модели представления:

```
define("FunnelChartSchema", ["UsrFunnelByCountDataProvider"], function() {
    return {
        entitySchemaName: "Opportunity",
        methods: {
            getProvidersCollectionConfig: function() {
                // Вызывает родительский метод, который возвращает массив провайдеров.
                var config = this.callParent();
                // Ищет провайдер данных для среза по количеству.
                var byCount = Terrasoft.findItem(config, {tag: "byNumberConversion"});
                // Заменяет на новый класс.
                byCount.item.className = "Terrasoft.UsrFunnelByCountDataProvider";
                return config;
            }
        }
    };
});
```

После сохранения схемы в воронке продаж будет использоваться новый модуль расчетов: будут отображаться только те продажи, у которых в поле [ Клиент ] указан контрагент.

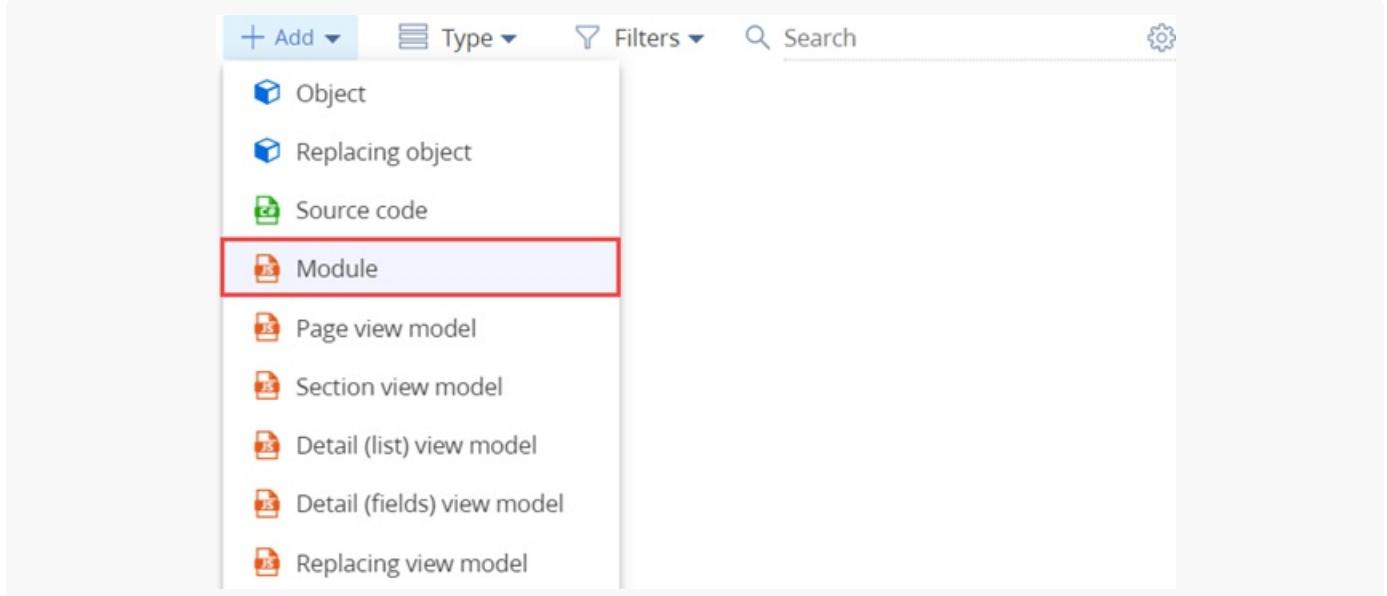
## Добавить пользовательский дашборд



**Пример.** Создать пользовательский дашборд, отображающий текущий курс валют.

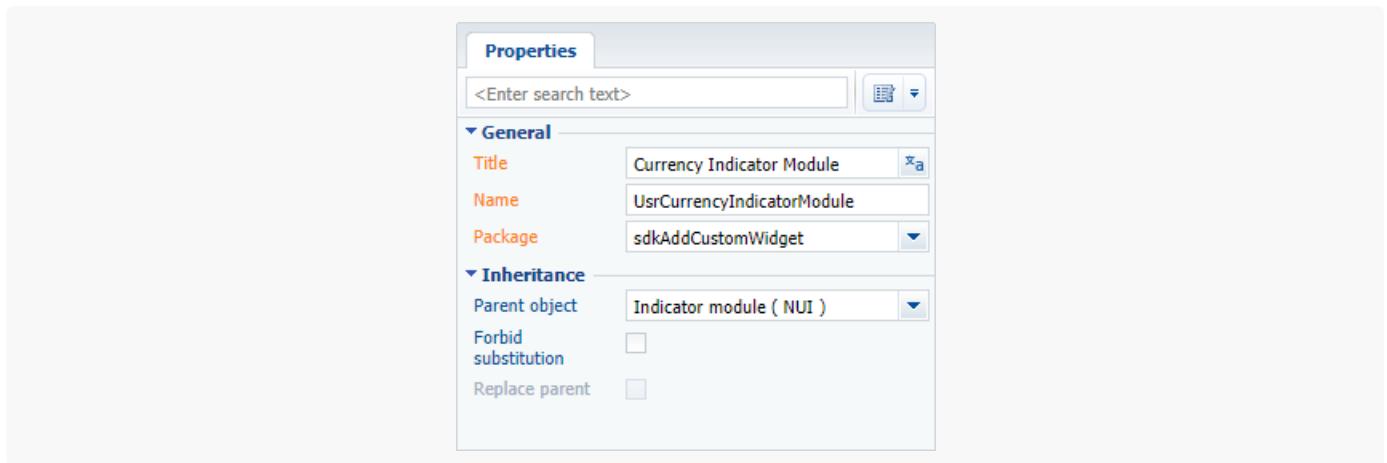
## 1. Создать модуль показателя валюты

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



### 3. Заполните **свойства схемы**.

- [ Код ] ([ Code ]) — "UsrCurrencyIndicatorModule".
- [ Заголовок ] ([ Title ]) — "Модуль показателя валюты" ("Currency Indicator Module").



### 4. Добавьте исходный код

Исходный код схемы модуля представлен ниже.

```
UsrCurrencyIndicatorModule

define("UsrCurrencyIndicatorModule", ["UsrCurrencyIndicatorModuleResources", "IndicatorModule
```

```
// Класс, генерирующий конфигурацию представления модуля показателя валюты.
Ext.define("Terrasoft.configuration.CurrencyIndicatorViewConfig", {
    extend: "Terrasoft.BaseModel",
    alternateClassName: "Terrasoft.CurrencyIndicatorViewConfig",
    // Генерирует конфигурацию представления модуля показателя валюты.
    generate: function(config) {
        var style = config.style || "";
        var fontStyle = config.fontStyle || "";
        var wrapClassName = Ext.String.format("{0}", style);
        var id = Terrasoft.Component.generateId();
        // Возвращаемый конфигурационный объект представления.
        var result = {
            "name": id,
            "itemType": Terrasoft.ViewItemType.CONTAINER,
            "classes": {wrapClassName: [wrapClassName, "indicator-module-wrapper"]},
            "styles": {
                "display": "table",
                "width": "100%",
                "height": "100%"
            },
            "items": [
                {
                    "name": id + "-wrap",
                    "itemType": Terrasoft.ViewItemType.CONTAINER,
                    "styles": {
                        "display": "table-cell",
                        "vertical-align": "middle"
                    },
                    "classes": {wrapClassName: ["indicator-wrap"]},
                    "items": [
                        // Отображение названия валюты.
                        {
                            "name": "indicator-caption" + id,
                            "itemType": Terrasoft.ViewItemType.LABEL,
                            "caption": {"bindTo": "CurrencyName"},
                            "classes": {"labelClass": ["indicator-caption"]}
                        },
                        // Отображение курса валюты.
                        {
                            "name": "indicator-value" + id,
                            "itemType": Terrasoft.ViewItemType.LABEL,
                            "caption": {
                                "bindTo": "CurrencyValue"
                            },
                            "classes": {"labelClass": ["indicator-value " + fontStyle]}
                        }
                    ]
                }
            ]
        }
    }
})
```

```

    };
    return result;
}
});

// Класс модели представления модуля показателя валюты.
Ext.define("Terrasoft.configuration.CurrencyIndicatorViewModel", {
    extend: "Terrasoft.BaseModel",
    alternateClassName: "Terrasoft.CurrencyIndicatorViewModel",
    Ext: null,
    Terrasoft: null,
    sandbox: null,
    columns: {
        // Название валюты.
        CurrencyName: {
            type: Terrasoft.core.enums.ViewModelSchemaItem.ATTRIBUTE,
            dataType: Terrasoft.DataValueType.TEXT,
            value: null
        },
        // Значение валюты.
        CurrencyValue: {
            type: Terrasoft.core.enums.ViewModelSchemaItem.ATTRIBUTE,
            dataType: Terrasoft.DataValueType.FLOAT,
            value: null
        }
    },
    onRender: Ext.emptyFn,
    // Возвращает значение валюты в зависимости от названия. Этот метод приведен в качестве
    // Для каждой конкретной задачи следует выбрать индивидуальный способ получения данны
    // например REST API, запрос к базе данных и т.п.
    getCurrencyValue: function(currencyName, callback, scope) {
        var result = 0;
        if (currencyName === "USD") {
            result = 26;
        }
        if (currencyName === "EUR") {
            result = 32.3;
        }
        if (currencyName === "RUB") {
            result = 0.45;
        }
        callback.call(scope || this, result);
    },
    // Получает и отображает данные на дашборде.
    prepareIndicator: function(callback, scope) {
        this.getCurrencyValue(this.get("CurrencyName"), function(currencyValue) {
            this.set("CurrencyValue", currencyValue);
            callback.call(scope);
        });
    }
});

```

```

        },
        // Инициализирует дашборд.
        init: function(callback, scope) {
            this.prepareIndicator(callback, scope);
        }
    });

    // Класс модуля дашборда.
Ext.define("Terrasoft.configuration.CurrencyIndicatorModule", {
    extend: "Terrasoft.IndicatorModule",
    alternateClassName: "Terrasoft.CurrencyIndicatorModule",
    // Название класса модели представления дашборда
    viewModelClassName: "Terrasoft.CurrencyIndicatorViewModel",
    // Название класса-генератора конфигурации представления.
    viewConfigClassName: "Terrasoft.CurrencyIndicatorViewConfig",
    // Подписка на сообщения сторонних модулей.
    subscribeMessages: function() {
        this.sandbox.subscribe("GenerateIndicator", this.onGenerateIndicator, this, [this
    }
}),
    return Terrasoft.CurrencyIndicatorModule;
});

```

5. Перейдите в узел [ LESS ] структуры объекта и задайте необходимые стили отображения дашборда.

### Настройка стилей отображения дашборда

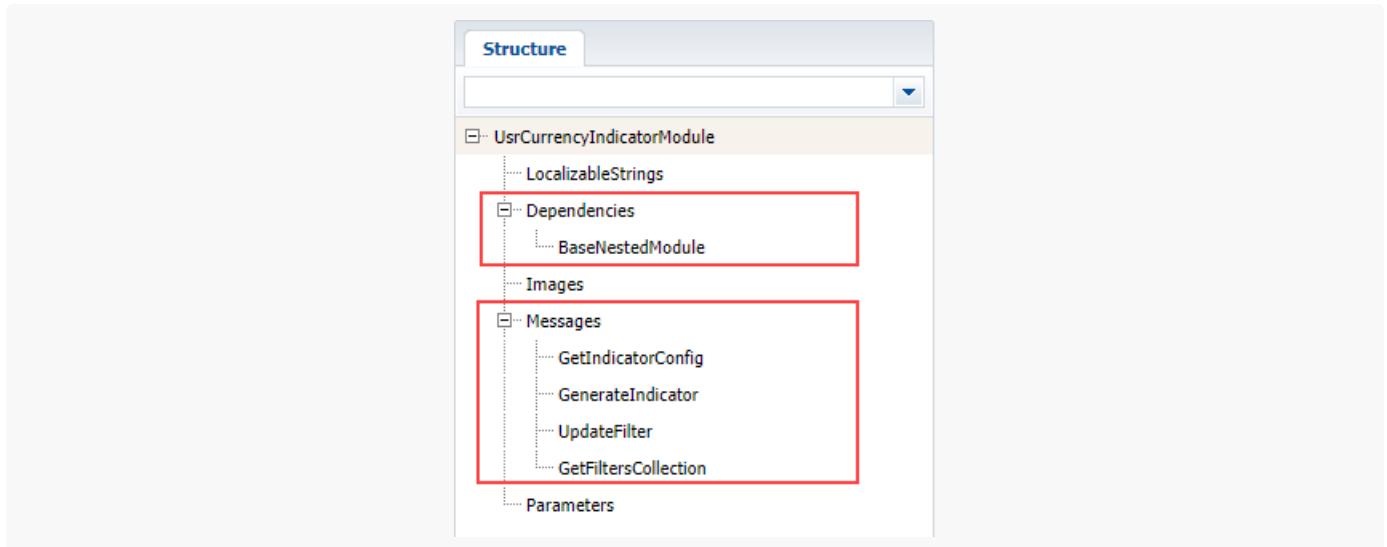
```

/* Настройка отображения текста дашборда по центру элемента.*/
.indicator-module-wrapper {
    text-align: center;
}

```

6. В созданный модуль добавьте сообщения родительского модуля:

- адресное сообщение `GetIndicatorConfig`, для которого установите направление "Публикация";
- адресное сообщение `GenerateIndicator`, для которого установите направление "Подписка".

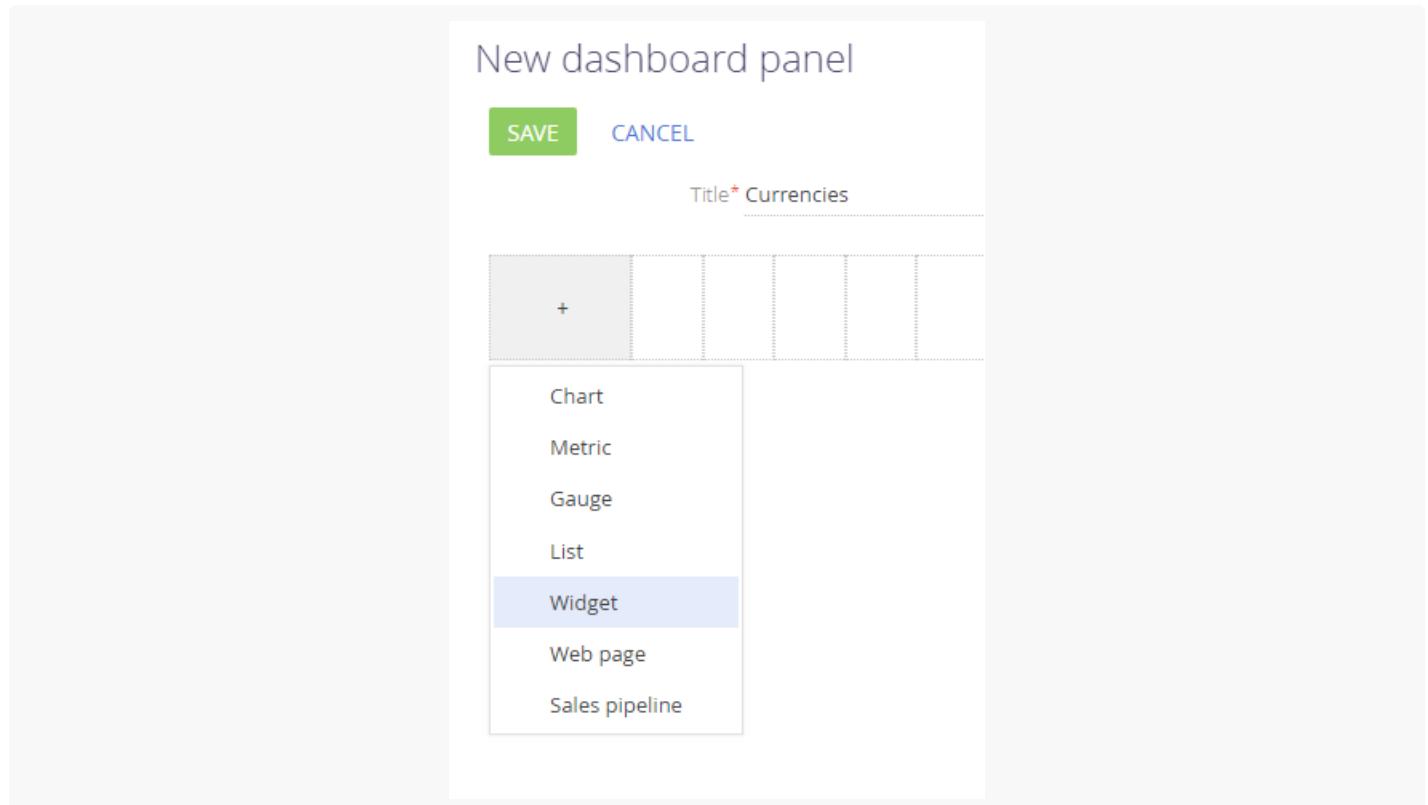


7. На панели инструментов дизайнера нажмите [ Сохранить ] ([ Save ]).

## 2. Добавить дашборд на панель итогов и указать его параметры

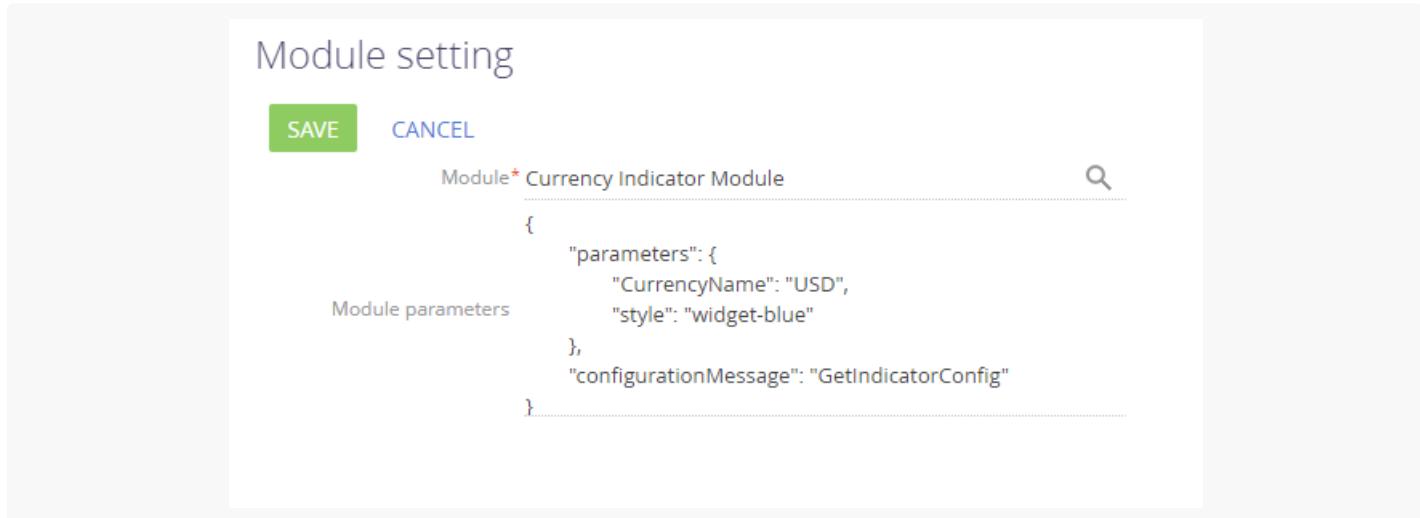
Для отображения дашборда добавьте его на панель итогов.

Рис. 3. — Добавление дашборда на панель итогов



Затем настройте параметры модуля, подключаемого к дашборду.

Рис. 4. — Настройка модуля добавляемого дашборда



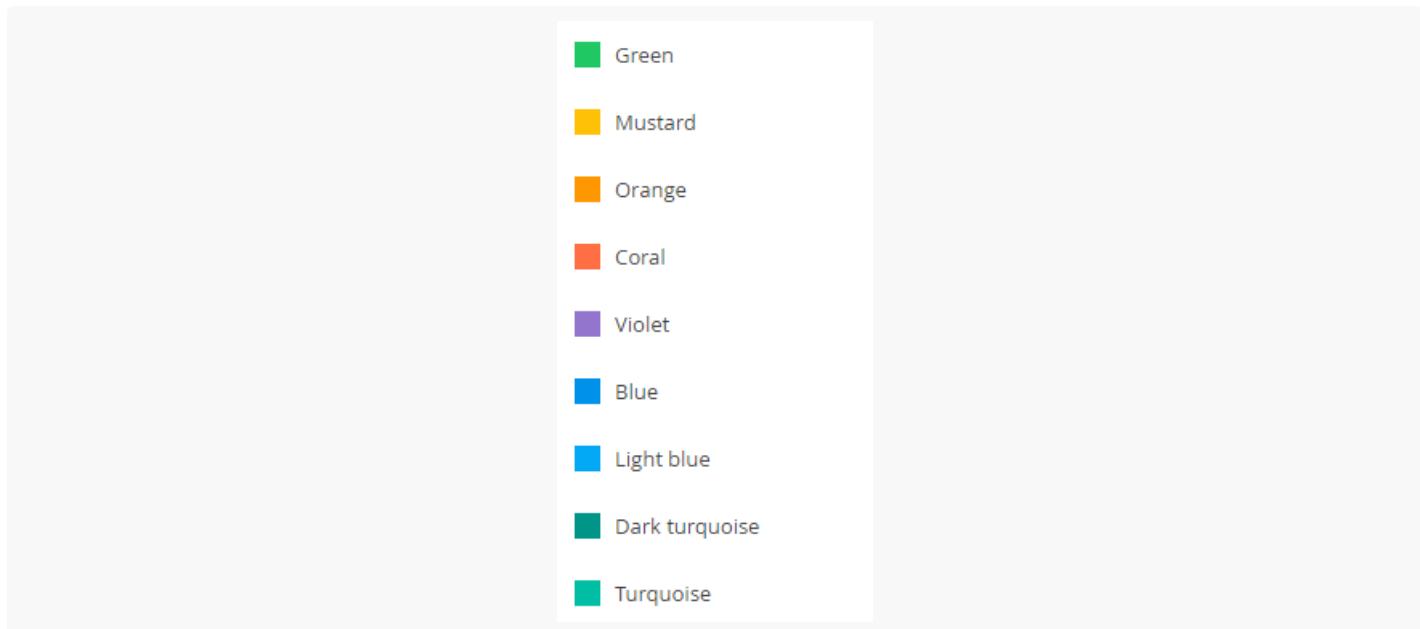
Для подключения модуля к создаваемому дашборду необходимо в поле [ Модуль ] добавить значение "Модуль показателя валюты", а в поле [ Параметры модуля ] добавить конфигурационный JSON-объект с необходимыми параметрами.

```
{
  "parameters": {
    "CurrencyName": "USD",
    "style": "widget-blue"
  },
  "configurationMessage": "GetIndicatorConfig"
}
```

Параметр `CurrencyName` устанавливает значение валюты, для которой необходимо отобразить курс, параметр `style` — стиль дашборда, а параметр `configurationMessage` — название сообщения, при помощи которого будет передан конфигурационный объект.

В параметре `style` можно указать любой цвет дашборда из существующих в Creatio.

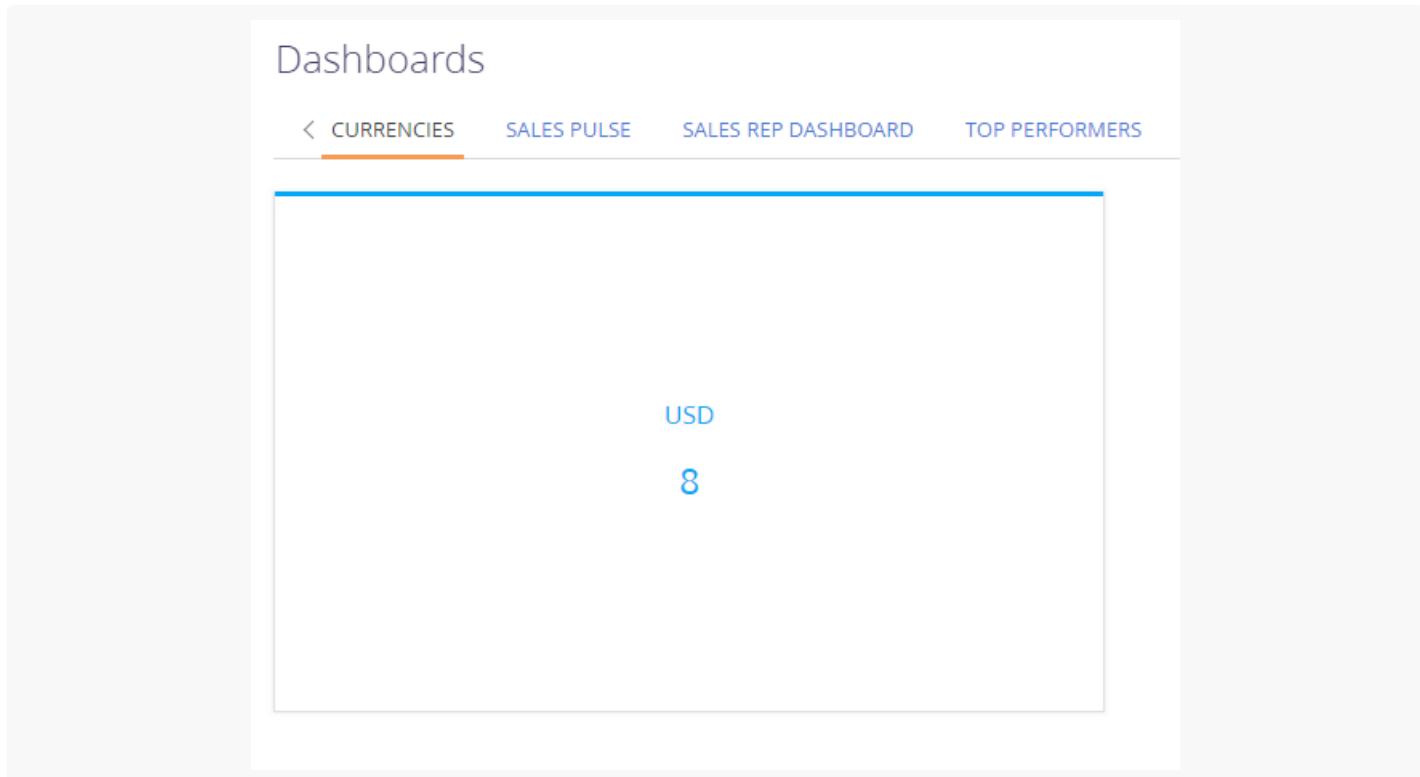
Рис. 5. — Варианты стилей дашборда



## Результат выполнения примера

После сохранения создаваемого дашборда и обновления страницы на панели итогов будет отображен пользовательский дашборд.

Рис. 6. — Дашборд курса валют



## Схема BaseWidgetDesigner



Сложный

`BaseWidgetDesigner` — базовая схема представления настройки дашбордов.

## Методы

---

`getWidgetConfig()`

Возвращает объект актуальных настроек дашборда.

`getWidgetConfigMessage()`

Возвращает название сообщения получения настроек модуля дашборда.

`getWidgetModuleName()`

Возвращает название модуля дашборда.

`getWidgetRefreshMessage()`

Возвращает название сообщения обновления дашборда.

`getWidgetModulePropertiesTranslator()`

Возвращает объект соотношения свойств модуля дашборда и модуля настройки дашборда.

## Перечисление DashboardEnums js



Сложный

`DashboardEnums` — содержит перечисление свойств, используемых в дашбордах.

`Terrasoft.DashboardEnums.WidgetType` — содержит конфигурацию дашбордов для режима просмотра (view) и режима настройки (design) итогов.

## Свойства

---

`moduleName`

Название модуля дашборда.

`configurationMessage`

Название сообщения получения настроек модуля.

---

**resultMessage**

Название сообщения для отдачи параметров настройки модуля дизайнера дашборда.

---

**stateConfig (stateObj)**

Название схемы дизайнера дашборда.