

Элементы интерфейса

Мини-карточка

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

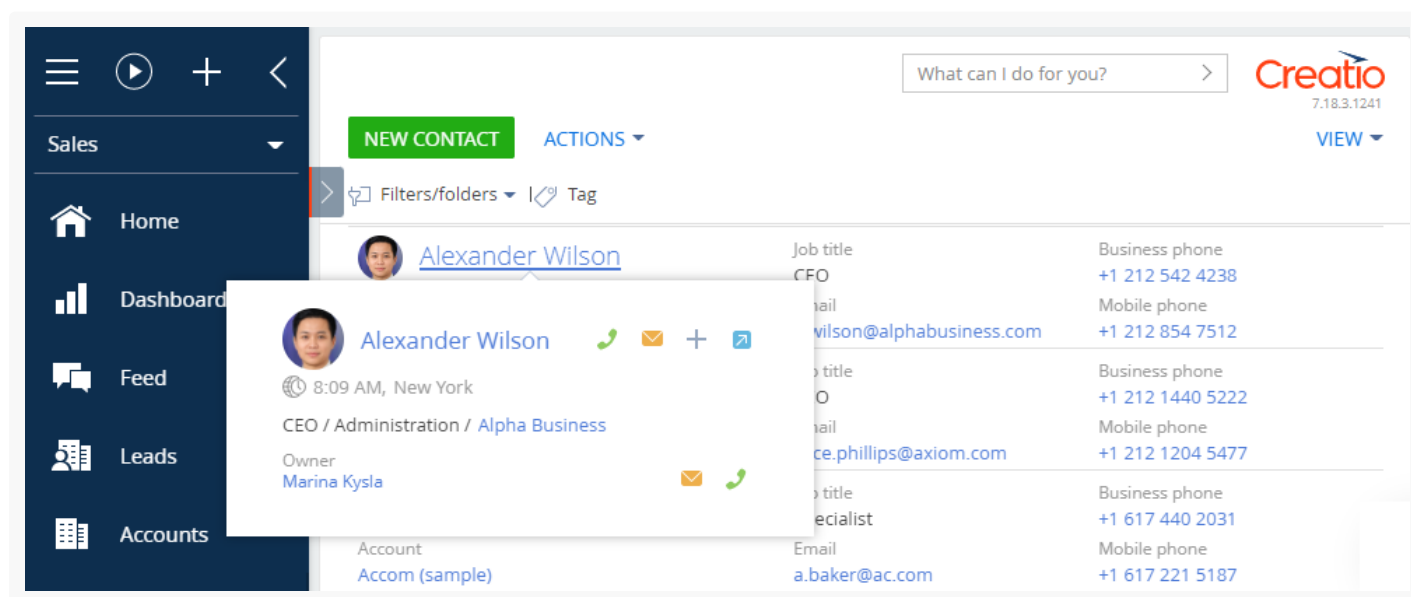
Мини-карточка	4
Схема модели представления мини-карточки	4
Операции с мини-карточками	5
Создать пользовательскую мини-карточку	6
1. Создать схему модели представления мини-карточки	6
2. Отобразить поля основного объекта	7
3. Добавить функциональную кнопку в мини-карточку	9
4. Выполнить стилизацию мини-карточки	12
5. Зарегистрировать мини-карточку в базе данных	16
6. Добавить системную настройку	17
Результат выполнения примера	18
Создать мини-карточку добавления	18
1. Создать схему модели представления мини-карточки	18
2. Отобразить поля основного объекта	19
3. Зарегистрировать мини-карточку в базе данных	21
4. Добавить системную настройку	22
Результат выполнения примера	22
Добавить мини-карточку к произвольному модулю	23
1. Создать схему модуля	24
2. Создать представление и модель представления модуля	25
3. Добавить стили модуля	27
4. Создать контейнер отображения представления	27
Результат выполнения примера	30

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



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-мини-карточек 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

1. В пользовательский [пакет](#) добавьте [схему модели представления](#) карточки.
2. В качестве родительского объекта выберите схему `BaseMiniPage`.
3. 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`.
4. Используя SQL-запрос, внесите изменения в системную таблицу `[SysModuleEdit]` базы данных.
5. Добавьте системную настройку `[HasКодРазделаMiniPageAddMode]`. 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.

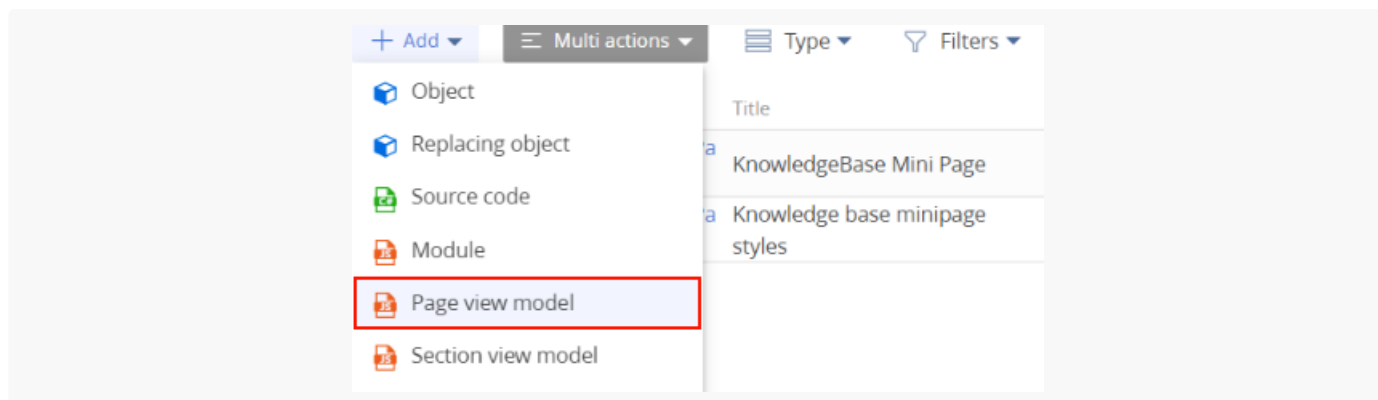
Создать пользовательскую мини-карточку



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.

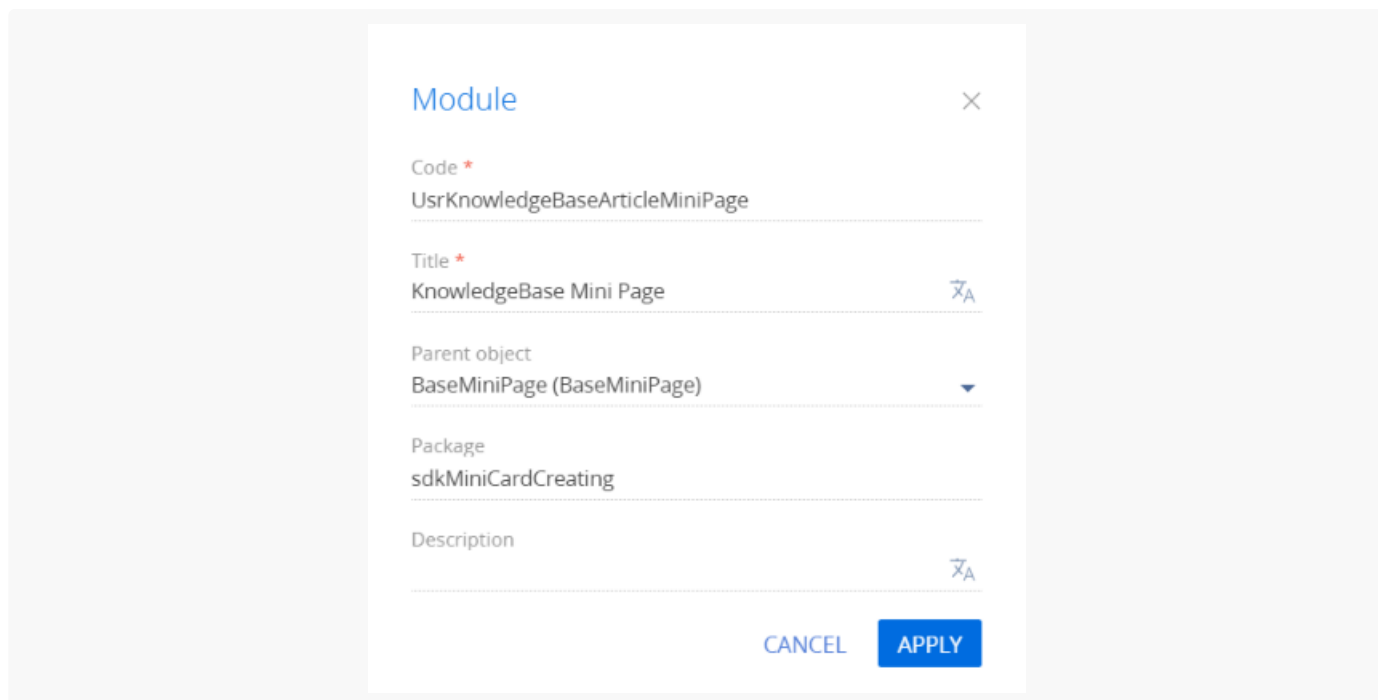
1. Создать схему модели представления мини-карточки

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 дизайнера схем измените исходный код модели представления.

Для **добавления кнопки** выбора файлов статьи базы знаний:

1. Добавьте в массив `diff` элемент `FilesButton`, который является описанием кнопки
2. Добавьте в свойство `attributes` виртуальную колонку `Article`, которая связывает основную и дополнительные записи.
3. Добавьте в свойство `attributes` трибут `MiniPageModes` — массив, который содержит коллекцию необходимых операций, выполняемых мини-карточкой.
4. Добавьте в ресурсы схемы изображение кнопки. Например, можно использовать это изображение — . Добавление изображения в ресурсы описано в статье [Добавить поле с изображением](#).
5. 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()` — иницирует загрузку файлов и их добавление в выпадающий список кнопки выбора файлов.
 - `downloadFile()` — иницирует скачивание выбранного файла.

Исходный код схемы модели представления, который добавляет функциональную кнопку, приведен ниже.

UsrKnowledgeBaseArticleMiniPage.js

```

define("UsrKnowledgeBaseArticleMiniPage",
  ["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();
    },
    /* Инициализирует 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.File);
        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);
    },
    /* Иницирует 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
            }
        }
    }
]

```

```

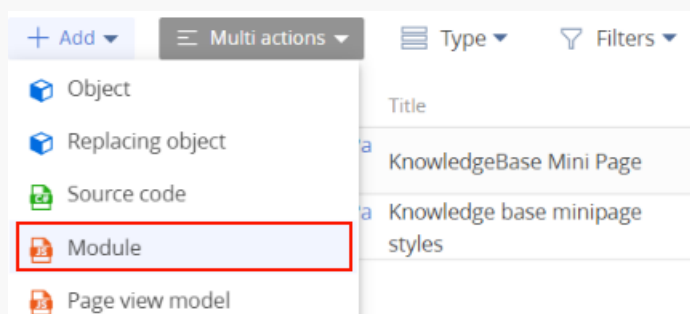
        },
        "isMiniPageModelItem": true,
        "layout": {
            "column": 0,
            "row": 1,
            "colSpan": 24
        }
    },
    {
        "operation": "insert",
        "parentName": "HeaderContainer",
        "propertyName": "items",
        "name": "FilesButton",
        "values": {
            "itemType": Terrasoft.ViewItemType.BUTTON,
            /* Настройка изображения кнопки.*/
            "imageConfig": {
                /* Изображение предварительно необходимо добавить в ресурсы мини-кар
                "bindTo": "Resources.Images.FilesImage"
            },
            /* Настройка выпадающего списка.*/
            "extendedMenu": {
                /* Название элемента выпадающего списка.*/
                "Name": "File",
                /* Название атрибута миникарточки, связывающего основную и дополните
                "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 представелния.

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



3. В дизайнере схем заполните свойства схемы:

- [Код] ([Code]) — "UsrKnowledgeBaseArticleMiniPageCss".
- [Заголовок] ([Title]) — "Стили миникарточки базы знаний" ("Knowledge base minipage styles").

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

4. В контекстном меню узла LESS укажите необходимые стили.

UsrKnowledgeBaseArticleMiniPageCss.js

```
div[data-item-marker="UsrKnowledgeBaseArticleMiniPageContainer"] > div {
width: 250px;
}
```

5. В дизайнере схем измените код схемы модели представления: добавьте загрузку этого модуля в исходном коде.

Ниже приведен полный исходный код мини-карточки:

UsrKnowledgeBaseArticleMiniPage.js

```

define("UsrKnowledgeBaseArticleMiniPage",
["terrasoft", "KnowledgeBaseFile", "ConfigurationConstants", "css!UsrKnowledgeBaseArticleMini
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.p
                var typeFilter = this.Terrasoft.createColumnFilterWithParameter(
                    this.Terrasoft.ComparisonType.EQUAL, "Type", ConfigurationConstants.F
                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

SAVE **CANCEL**

Name* HasKnowledgeBaseMiniPageAddMode Code* HasKnowledgeBaseMiniPageAddMode

Type* Boolean

Default value ☒

Cached ☒

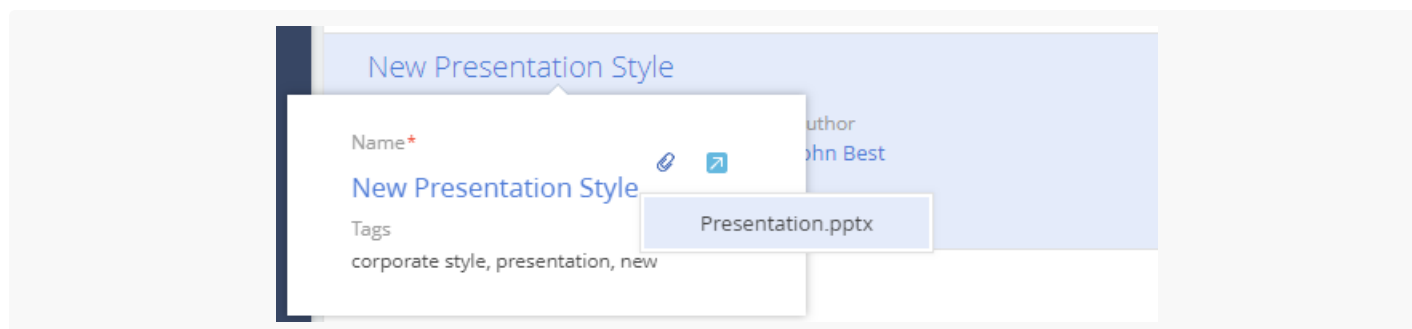
Personal ☐

Allow for portal users ☐

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.



Создать мини-карточку добавления

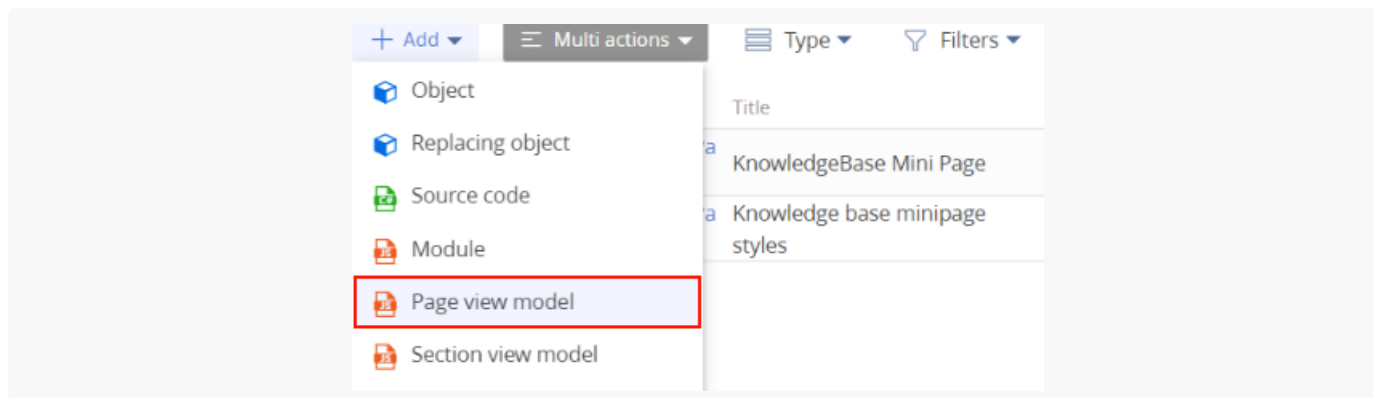
Сложный

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

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]) — "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.

HasProductMiniPageAddM... What can I do for you? > Creatio

SAVE CANCEL

Name* HasProductMiniPageAddMode Code* HasProductMiniPageAddMode

Type* Boolean Cached ☒ ⓘ

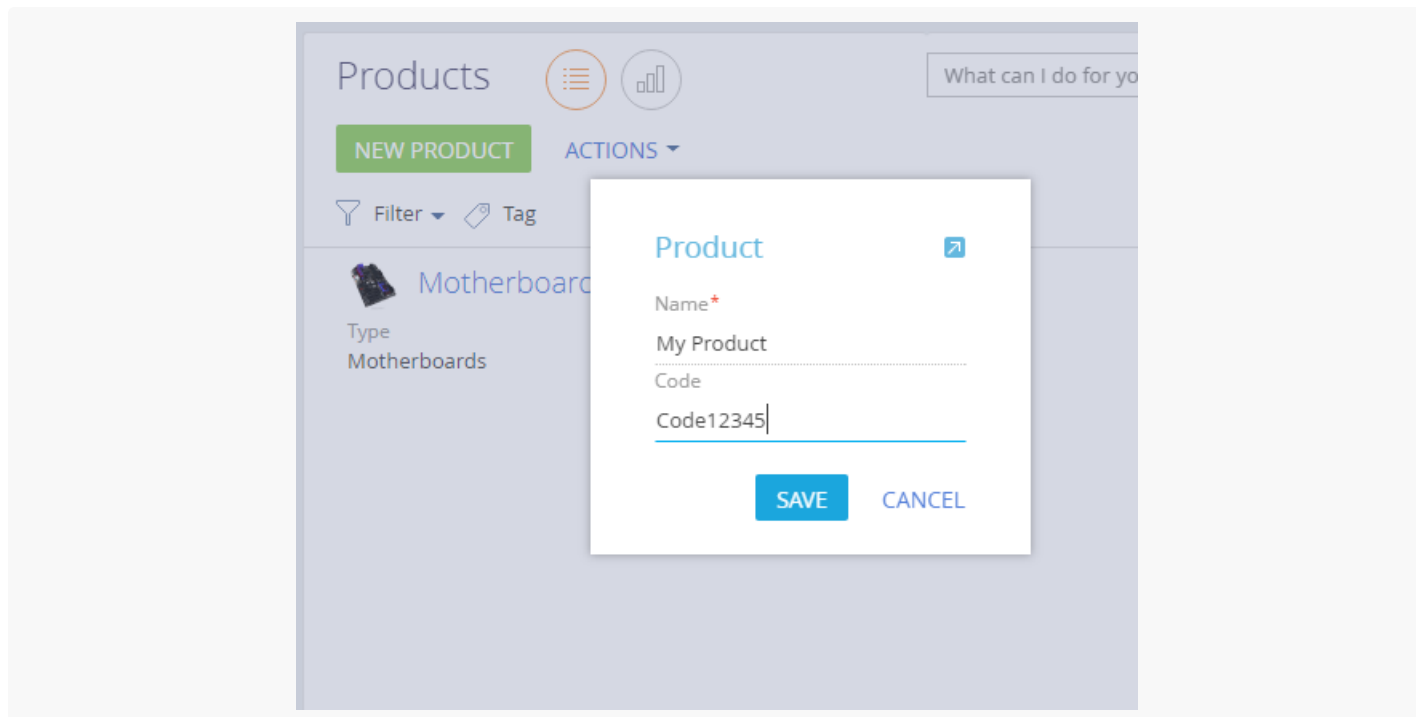
Default value ☒ Personal ☐ ⓘ

Allow for portal users ☐

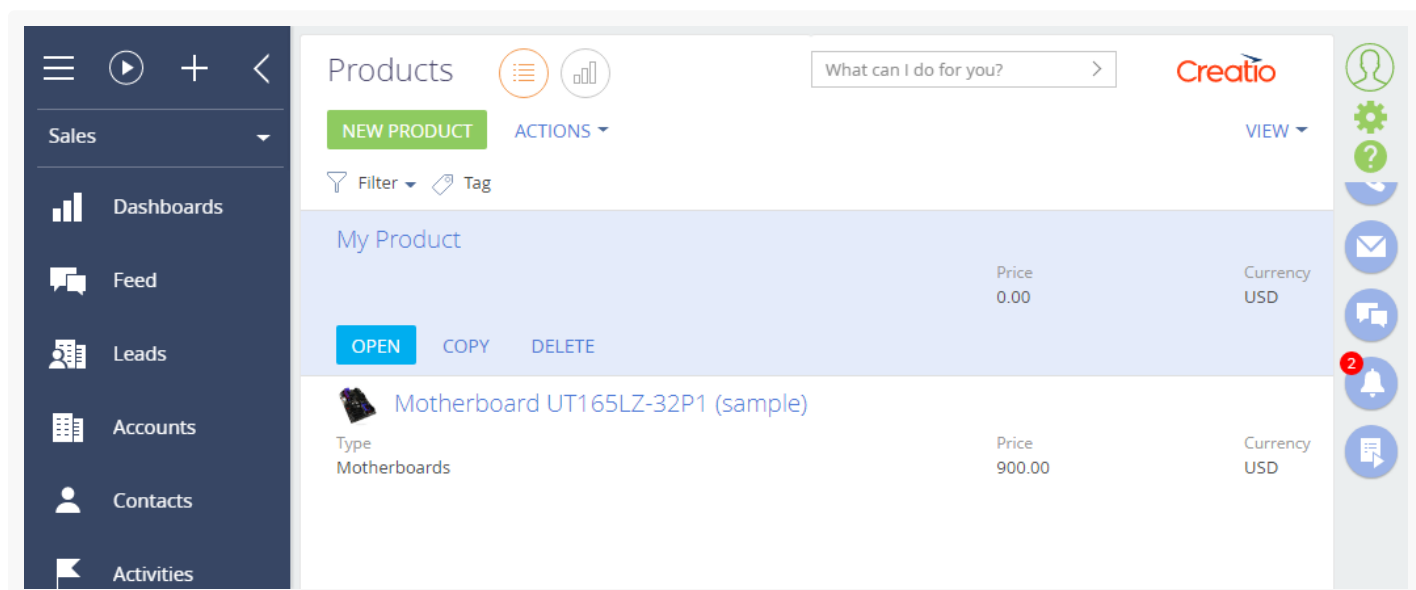
Description

Результат выполнения примера

В результате выполнения примера при добавлении нового продукта будет отображаться мини-карточка с двумя полями.



После сохранения мини-карточки соответствующая запись появится в реестре раздела.



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](#)).

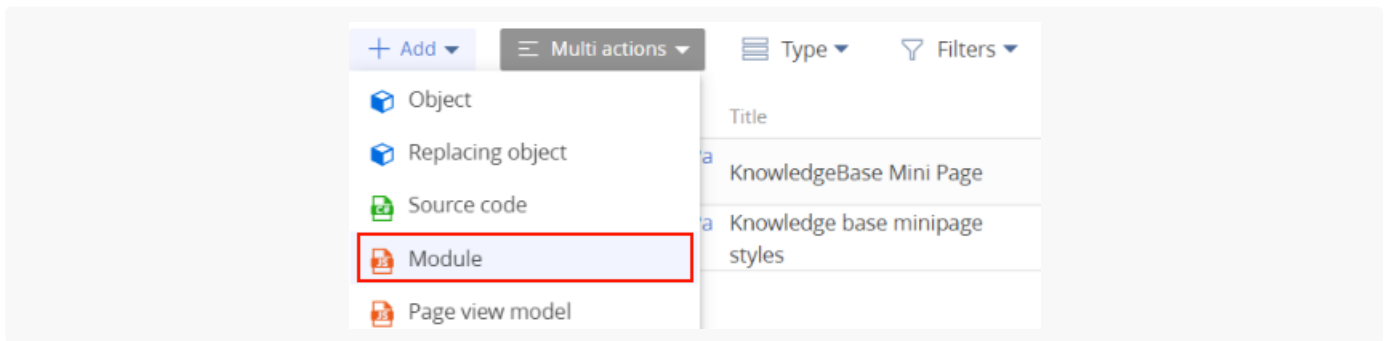
Добавить мини-карточку к произвольному модулю



Пример. Отобразить текущего пользователя в правом верхнем углу приложения возле иконки с профилем пользователя. При наведении на ссылку текущего пользователя системы открыть мини-карточку.

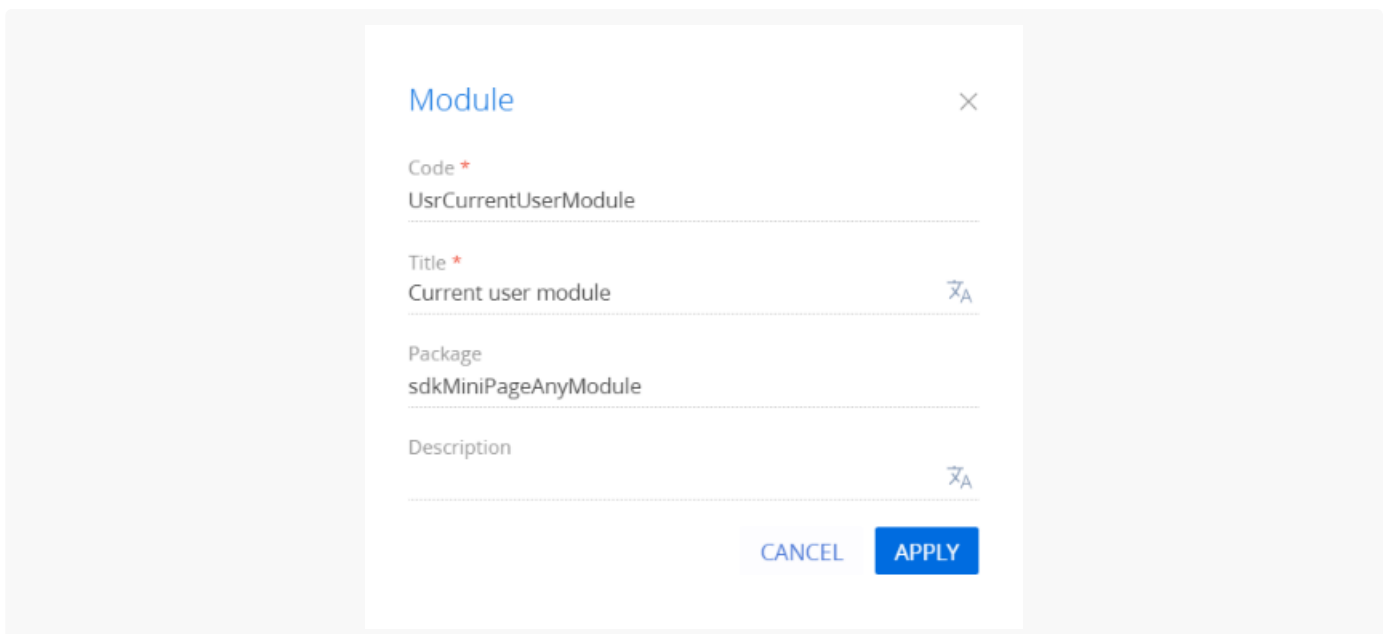
1. Создать схему модуля

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



3. В дизайнере схем заполните свойства схемы:

- [*Код*] ([*Code*]) — "UsrCurrentUserModule".
- [*Заголовок*] ([*Title*]) — "Модуль "Текущий пользователь" ("Current user module").



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.displayName,
        /* 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;
}
```

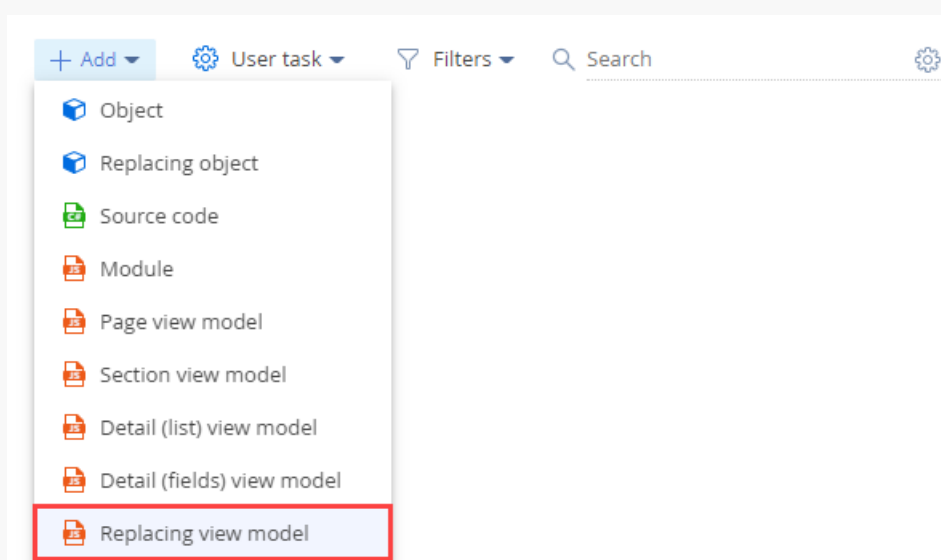
Сохраните созданный модуль.

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.

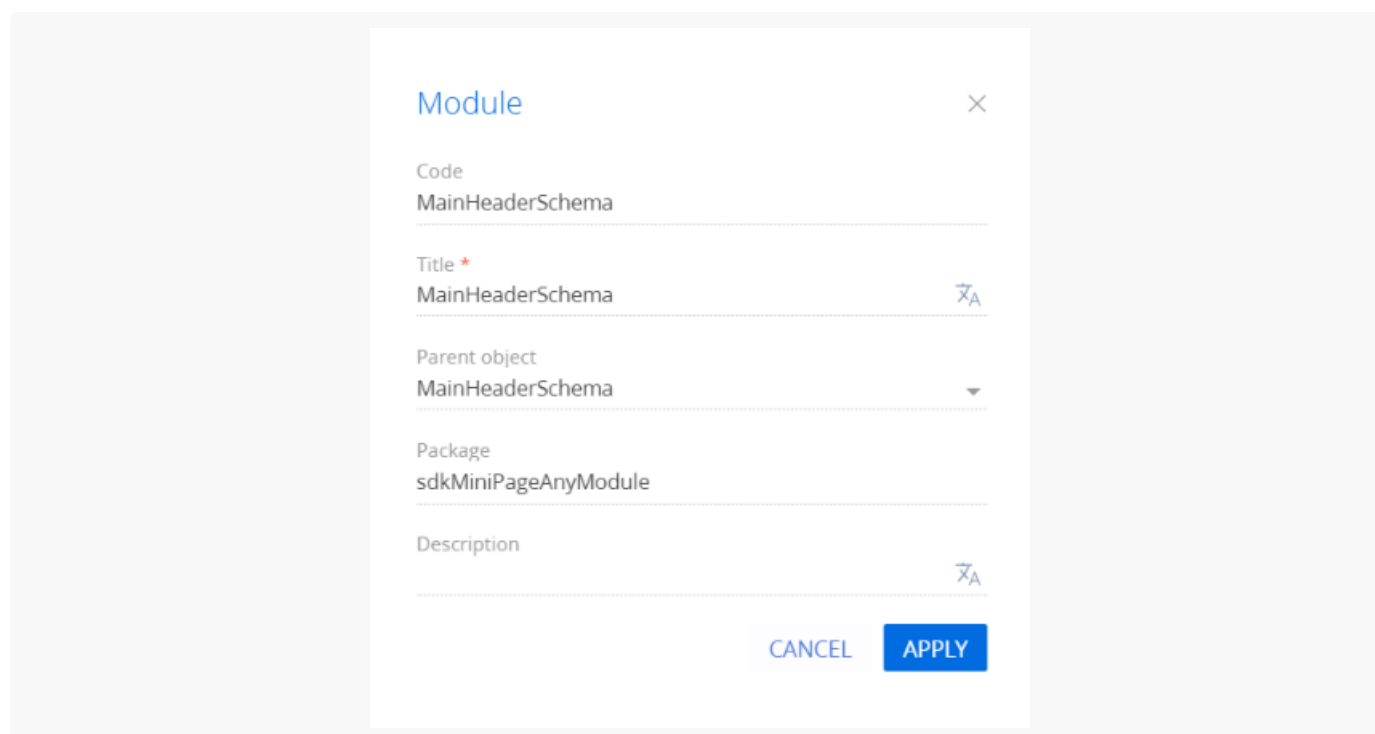
Для этого создайте схему замещающей модели представления, которая расширит функциональность схемы `MainHeaderSchema`.

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": []
                    }
                }
            ]
        }
    };
});

```

```

        }
    ]
};
});

```

5. 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.

