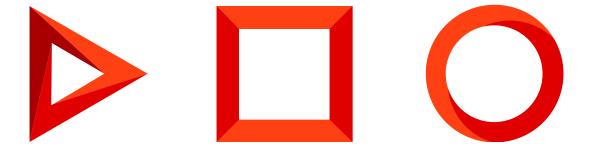


Кастомизация мобильного приложения

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

Манифест мобильного приложения	6
Свойства конфигурационного объекта для настроек синхронизации	6
Свойства конфигурационного объекта для настройки синхронизации модели	6
Свойства конфигурационного объекта фильтра модели	7
Свойство SyncOptions.ModelDataImportConfig.QueryFilter	11
Настроить меню мобильного приложения	12
Реализация примера	12
Настроить стартовую страницу и разделы меню мобильного приложения	13
Реализация примера	13
Настроить конфигурацию моделей	14
Реализация примера	15
Использовать функцию поиска подстроки для поиска данных	16
Реализация примера	16
Загрузить данные моделей при синхронизации	16
Реализация примера	16
Отобразить страницу на планшете во весь экран	17
Реализация примера	18
Добавить стандартную деталь с колонками	19
Алгоритм реализации примера	20
Добавить пользовательский дашборд в мобильное приложение	24
Алгоритм реализации примера	25
Добавить кнопку для отображения имени контакта	27
Алгоритм реализации примера	28
Свойство ModuleGroups	32
Свойство конфигурационного объекта	32
Свойство Modules	33
Свойства конфигурационного объекта	33
Свойство Icons	34
Свойства конфигурационного объекта	34
Свойства DefaultModuleImageId и DefaultModuleImageIdV2	34
Свойство Models	35
Свойства конфигурационного объекта	35
Свойство PreferedFilterFuncType	36
Функции фильтрации (Terrasoft.FilterFunctions)	36
Свойство CustomSchemas	37
Свойство SyncOptions	37

Свойства конфигурационного объекта для настроек синхронизации	38
Свойства конфигурационного объекта для настройки синхронизации модели	38
Свойства конфигурационного объекта фильтра модели	39
Свойство SyncOptions.ModelDataImportConfig.QueryFilter	43
Реестр мобильного приложения	44
Настроить реестр раздела	44
Реализация примера	44
Класс GridPage	45
Методы	45
Бизнес-правила мобильного приложения	48
Выполнить фильтрацию	48
Пример 1	48
Пример 2	49
Пример 3	49
Выделить поле по условию	50
Реализация примера	51
Сбросить отрицательные значения в 0	51
Реализация примера	52
Сгенерировать заголовок активности	52
Реализация примера	52
Свойства объекта config	53
Базовые бизнес-правила	53
Бизнес-правило Обязательность заполнения (Terrasoft.RuleTypes.Requirement)	54
Бизнес-правило Видимость (Terrasoft.RuleTypes.Visibility)	55
Бизнес-правило Доступность (Terrasoft.RuleTypes.Activation)	56
Бизнес-правило Фильтрация (Terrasoft.RuleTypes.Filtration)	57
Бизнес-правило Взаимная фильтрация (Terrasoft.RuleTypes.MutualFiltration)	58
Бизнес-правило Регулярное выражение (Terrasoft.RuleTypes.RegExp)	59
Пользовательские бизнес-правила	60
Получение данных и настроек по дашбордам	63
Примеры запросов к сервису AnalyticsService	63
Методы	63
Класс AnalyticsService	66
Методы	66
Мобильный портал	66
Настроить рабочее место пользователя мобильного портала	67
Настроить реестр обращений	68
Настроить страницу обращения	71
Настроить страницу добавления обращения	71

Манифест мобильного приложения



Описывает параметры для настройки синхронизации данных. Содержит конфигурационный объект со свойствами, представленными в таблице.

Свойства конфигурационного объекта для настроек синхронизации

ImportPageSize

Количество страниц, импортируемых в одном потоке.

PagesInImportTransaction

Количество потоков импорта.

SysSettingsImportConfig

Массив импортируемых системных настроек.

SysLookupsImportConfig

Массив импортируемых системных справочников.

ModelDataImportConfig

Массив моделей, для которых будут загружаться данные при синхронизации.

В массиве моделей мodelDataImportConfig для каждой модели можно указать дополнительные параметры синхронизации, список загружаемых колонок, а также условия фильтрации загружаемых данных модели. Если при синхронизации должна загружаться полная модель, в массиве просто указывается объект с именем модели. Если к модели должны применяться дополнительные условия при синхронизации, в массив мodelDataImportConfig добавляется конфигурационный объект со свойствами.

Свойства конфигурационного объекта для настройки синхронизации модели

Name

Название модели (см. свойство Models конфигурационного объекта манифеста).

SyncColumns

Массив колонок модели, для которых импортируются данные. Помимо явно перечисленных колонок, при синхронизации в обязательном порядке будут импортироваться системные колонки (createdOn , CreatedBy , ModifiedOn , ModifiedBy) и колонка, первичная для отображения.

SyncFilter

Фильтр, накладываемый на модель при импорте модели.

Фильтр <u>syncFilter</u>, накладываемый на модель при импорте модели представляет собой конфигурационный объект со свойствами.

Свойства конфигурационного объекта фильтра модели

type

Тип фильтра. Задается значением перечисления Terrasoft.FilterTypes . Необязательное свойство. По умолчанию Terrasoft.FilterTypes.Simple .

Возможные значения (Terrasoft.FilterTypes)

Simple	фильтр с одним условием
Group	групповой фильтр с несколькими условиями

logicalOperation

Логическая операция объединения коллекции фильтров (для фильтров с типом Terrasoft.FilterTypes.Group). Задается значением перечисления Значение по умолчанию - Terrasoft.FilterLogicalOperations.And .

BO3MOЖНЫЕ ЗНАЧЕНИЯ (Terrasoft.FilterLogicalOperations)

0r	логическая операция ИЛИ
And	логическая операция И

subfilters

Коллекция фильтров, применяемых к модели. Обязательное свойство для типа фильтра

Terrasoft.FilterTypes.Group . Фильтры между собой объединяются логической операцией, указанной в свойстве logicalOperation . Каждый фильтр представляет собой конфигурационный объект фильтра.

property

Название колонки модели, по которой выполняется фильтрация. Обязательное свойство для типа фильтра Terrasoft.FilterTypes.Simple.

valueIsMacrosType

Признак, определяющий, является ли значение для фильтрации макросом. Необязательное свойство. Может принимать значения: true, если для фильтрации используется макрос, иначе — false.

value

Значение для фильтрации колонки, указанной в свойстве property. Обязательное свойство для типа фильтра Terrasoft.FilterTypes.Simple. Может задаваться непосредственно значением для фильтрации (в том числе, может быть null) либо макросом (для этого свойство valueIsMacrosType должно иметь значение true). Макросы, которые можно использовать в качестве значения свойства, содержатся в перечислении Terrasoft.ValueMacros.

Возможные значения (Terrasoft.ValueMacros)

CurrentUserContactId	идентификатор текущего пользователя
CurrentDate	текущая дата
CurrentDateTime	текущие дата и время
CurrentDateEnd	полная дата окончания текущей даты
CurrentUserContactName	имя текущего контакта
CurrentUserContact	идентификатор и имя текущего контакта
SysSettings	значение системной настройки. Имя системной настройки передается в свойстве macrosParams
CurrentTime	текущее время
CurrentUserAccount	идентификатор и имя контрагента текущего пользователя
GenerateUId	сгенерированный идентификатор

macrosParams

Значения, которые передаются в макрос в качестве параметра. Необязательное свойство. В настоящее время используется только для макроса Terrasoft. ValueMacros. SysSettings.

isNot

Определяет, применяется к фильтру оператор отрицания. Необязательное свойство. Принимает значение true, если к фильтру применяется оператор отрицания, иначе — false.

funcType

Тип функции, которая применяется к колонке модели, заданой в свойстве property . Необязательное свойство. Может принимать значения перечисления Terrasoft.FilterFunctions . Значения аргументов для функций фильтрации задаются в свойстве funcArgs . Значение, с которым сравнивается результат функции, задается свойством value .

Возможные значения (Terrasoft.FilterFunctions)

SubStringOf	определяет, является ли строка, переданная в качестве аргумента, подстрокой колонки property
ToUpper	приводит значения колонки, заданной в property, к верхнему регистру
EndsWith	проверяет, оканчивается ли значение колонки property значением, переданным в качестве аргумента
StartsWith	проверяет, начинается ли значение колонки property значением, переданным в качестве аргумента
Year	возвращает год по значению колонки property
Month	возвращает месяц по значению колонки property
Day	возвращает день по значению колонки property
In	проверяет вхождение значения колонки property в диапазон значений, переданных в качестве аргумента функции
NotIn	проверяет невхождение значения колонки property в диапазон значений, переданных в качестве аргумента функции
Like	определяет, совпадает ли значение колонки рroperty с заданным шаблоном

funcArgs

Массив значений аргументов для функции фильтрации, заданной в свойстве funcType . Порядок значений в массиве funcArgs должен соответствовать порядку параметров функции funcType .

name

Имя фильтра или группы фильтров. Необязательное свойство.

modelName

Название модели, для которой выполняется фильтрация. Необязательное свойство. Указывается, если фильтрация выполняется по колонкам связанной модели.

assocProperty

Колонка связанной модели, по которой осуществляется связь с основной моделью. В качестве колонки для связи у основной модели выступает первичная колонка.

operation

Тип операции фильтрации. Необязательный параметр. Может принимать значения из перечисления Terrasoft.FilterOperation. По умолчанию имеет значение Terrasoft.FilterOperation.General.

Возможные значения (Terrasoft.FilterOperation)

General	стандартная фильтрация
Any	фильтрация с применением фильтра exists

compareType

Тип операции сравнения в фильтре. Необязательный параметр. Принимает значения из перечисления Terrasoft.ComparisonType. По умолчанию — Terrasoft.ComparisonType.Equal.

Возможные значения (Terrasoft.ComparisonType)

Equal	равно
LessOrEqual	меньше или равно
NotEqual	не равно
Greater	больше
GreaterOrEqual	больше или равно
Less	меньше

Свойство SyncOptions.ModelDataImportConfig.QueryFilter

Доступно в приложении, начиная с версии 7.12.1, и в мобильном приложении, начиная с версии 7.12.3.

Свойство синхронизации queryFilter позволяет настроить фильтрацию данных указанной модели при импорте с помощью <u>сервиса работы с данными DataService</u>. Ранее для фильтрации данных использовалось свойство SyncFilter, а импорт выполнялся с помощью <u>сервиса работы с данными OData</u>.

Важно. Импорт данных с помощью сервиса работы с данными DataService доступен только для платформ Android и iOS. Для платформы Windows используется OData.

Формат фильтра QueryFilter представляет собой <u>набор параметров</u> в виде JSON-объекта, передаваемых в запросе к сервису работы с данными DataService.

"subFilters": {

```
"logicalOperation": 0,
                      "filterType": 6,
                      "rootSchemaName": "ActivityParticipant",
                      "items": {
                         "ParticipantFilter":{
                            "filterType": 1,
                            "comparisonType": 3,
                            "leftExpression": {
                               "expressionType": 0,
                               "columnPath": "Participant"
                            },
                            "rightExpression": {
                               "expressionType": 1,
                               "functionType": 1,
                               "macrosType": 2
                         }
                     }
               }
            }
         }
      }
   ]
}
```

Настроить меню мобильного приложения



Пример. Настроить меню мобильного приложения, состоящего из двух групп — основной группы и группы [*Продажи*].

Реализация примера

```
CBOЙCTBO ModuleGroups

// Групы модулей мобильного приложения.

"ModuleGroups": {
    // Настройка группы основного меню.
    "main": {
```

```
// Позиция группы в главном меню.

"Position": 0
},
// Настройка группы меню [Продажи].

"sales": {
    // Позиция группы в главном меню.
    "Position": 1
}
```

Настроить стартовую страницу и разделы меню мобильного приложения



```
    Пример. Настроить разделы приложения следующим образом:
    Разделы основного меню: [ Контакты ], [ Контрагенты ].
    Стартовая страница приложения: раздел [ Контакты ].
    В блоке [ LocalizableStrings ] схемы манифеста должны быть созданы строки содержащие заголовки разделов:
```

- ContactSectionTitle CO ЗНАЧЕНИЕМ "Контакты".
- AccountSectionTitle СО ЗНАЧЕНИЕМ "Контрагенты".

Реализация примера

```
CBOЙCTBO Modules

// Модули мобильного приложения.

"Modules": {
    // Раздел "Контакт".

"Contact": {
        // Группа меню приложения, в которой размещается раздел.

"Group": "main",
        // Название модели, которая предоставляет данные раздела.

"Model": "Contact",
        // Позиция раздела в группе меню.

"Position": 0,
        // Заголовок раздела.

"Title": "ContactSectionTitle",
```

```
// Подключение пользовательского изображения к разделу.
        "Icon": {
            // Уникальный идентификатор изображения.
            "ImageId": "4c1944db-e686-4a45-8262-df0c7d080658"
        },
        // Подключение пользовательского изображения к разделу.
        "IconV2": {
            // Уникальный идентификатор изображения.
            "ImageId": "9672301c-e937-4f01-9b0a-0d17e7a2855c"
        },
        // Признак отображения в меню.
        "Hidden": false
   },
   // Раздел "Контрагент".
    "Account": {
        // Группа меню приложения, в которой размещается раздел.
        "Group": "main",
        // Название модели, которая предоставляет данные раздела.
        "Model": "Account",
        // Позиция раздела в группе меню.
        "Position": 1,
        // Заголовок раздела.
        "Title": "AccountSectionTitle",
        // Подключение пользовательского изображения к разделу.
        "Icon": {
            // Уникальный идентификатор изображения.
            "ImageId": "c046aa1a-d618-4a65-a226-d53968d9cb3d"
        },
        // Подключение пользовательского изображения к разделу.
        "IconV2": {
            // Уникальный идентификатор изображения.
            "ImageId": "876320ef-c6ac-44ff-9415-953de17225e0"
        },
        // Признак отображения в меню.
        "Hidden": false
   }
}
```

Настроить конфигурацию моделей

Сложный

Пример. Добавить в манифест конфигурацию следующих моделей:

1. контакт — указать названия схем страниц реестра, просмотра и редактирования, требуемые модели, модули расширения модели и страниц модели.

2. Адрес контакта — указать только модуль расширения модели.

Реализация примера

Свойство Models

```
// Импортируемые модели.
"Models": {
   // Модель "Контакт"
   "Contact": {
        // Схема страницы реестра.
        "Grid": "MobileContactGridPage",
        // Схема страницы просмотра.
        "Preview": "MobileContactPreviewPage",
        // Схема страницы записи.
        "Edit": "MobileContactEditPage",
        // Названия моделей, от которых зависит модель "Контакт".
        "RequiredModels": [
            "Account", "Contact", "ContactCommunication", "CommunicationType", "Department",
            "ContactAddress", "AddressType", "Country", "Region", "City", "ContactAnniversary",
            "AnniversaryType", "Activity", "SysImage", "FileType", "ActivityPriority",
            "ActivityType", "ActivityCategory", "ActivityStatus"
        1,
        // Расширения модели.
        "ModelExtensions": [
            "MobileContactModelConfig"
        ],
        // Расширения страниц модели.
        "PagesExtensions": [
            "MobileContactRecordPageSettingsDefaultWorkplace",
            "MobileContactGridPageSettingsDefaultWorkplace",
            "MobileContactActionsSettingsDefaultWorkplace",
            "MobileContactModuleConfig"
        1
   },
   // Модель "Адреса контактов".
    "ContactAddress": {
        // Страницы реестра, просмотра и редактирования сгенерированы автоматически.
        // Расширения модели.
        "ModelExtensions": [
            "MobileContactAddressModelConfig"
        1
   }
}
```

Использовать функцию поиска подстроки для поиска данных



Пример. Для поиска данных использовать функцию поиска подстроки.

Реализация примера

```
CBOЙCTBO PreferedFilterFuncType

// Для поиска данных используется функция поиска подстроки.
"PreferedFilterFuncType": "Terrasoft.FilterFunctions.SubStringOf"
```

Важно. Если в секции PreferedFilterFunctype в качестве функции фильтрации данных задается функция, отличная от Terrasoft.FilterFunctions.StartWith, то при поиске в таблицах БД индексы использоваться не будут.

Загрузить данные моделей при синхронизации



Пример. При синхронизации в мобильное приложение должны загружаться данные для таких моделей:

- 1. Активность загружаются все колонки. Выполняется фильтрация модели загружаются только те активности, у которых участником является текущий пользователь.
- 2. Тип активности загружается полная модель.

Реализация примера

```
CBOЙCTBO SyncOptions

// Настройки синхронизации.
"SyncOptions": {
```

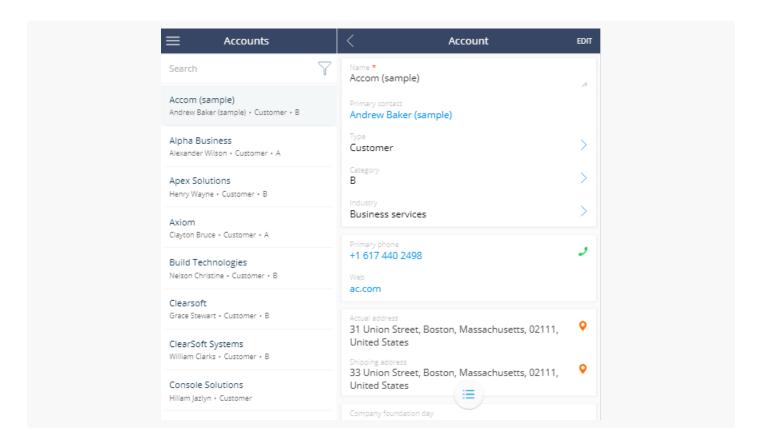
```
// Количество страниц, импортируемых в одном потоке.
    "ImportPageSize": 100,
   // Количество потоков импорта.
   "PagesInImportTransaction": 5,
   // Массив импортируемых системных настроек.
    "SysSettingsImportConfig": [
        "SchedulerDisplayTimingStart", "PrimaryCulture", "PrimaryCurrency", "MobileApplicationMc
   ],
   // Массив импортируемых системных справочников.
    "SysLookupsImportConfig": [
        "ActivityCategory", "ActivityPriority", "ActivityResult", "ActivityResultCategory", "Act
   // Массив моделей, для которых будут загружаться данные при синхронизации.
    "ModelDataImportConfig": [
        // Конфигурирование модели Activity.
        {
            "Name": "Activity",
            // Фильтр, накладываемый на модель при импорте.
            "SyncFilter": {
                // Название колонки модели, по которой выполняется фильтрация.
                "property": "Participant",
                // Название модели, для которой выполняется фильтрация.
                "modelName": "ActivityParticipant",
                // Колонка связанной модели, по которой осуществляется связь с основной моделью.
                "assocProperty": "Activity",
                // Тип операции фильтрации.
                "operation": "Terrasoft.FilterOperations.Any",
                // Для фильтрации используется макрос.
                "valueIsMacros": true,
                // Значение для фильтрации колонки — идентификатор и имя текущего контакта.
                "value": "Terrasoft.ValueMacros.CurrentUserContact"
            },
            // Массив колонок модели, для которых импортируются данные.
            "SyncColumns": [
                "Title", "StartDate", "DueDate", "Status", "Result", "DetailedResult", "Activity
        },
        // Модель ActivityType загружается полностью.
            "Name": "ActivityType",
            "SyncColumns": []
        }
   ]
}
```

Отобразить страницу на планшете во

весь экран



При просмотре страницы раздела мобильного приложения Creatio на планшете по умолчанию срабатывает режим отображения реестра раздела в левой области экрана.



Пример. Отобразить страницу на планшете во весь экран.

Реализация примера

Для отображения страницы раздела во весь экран необходимо внести изменения в манифест мобильного приложения, добавив в него свойство TabletViewMode со значением "SinglePage".

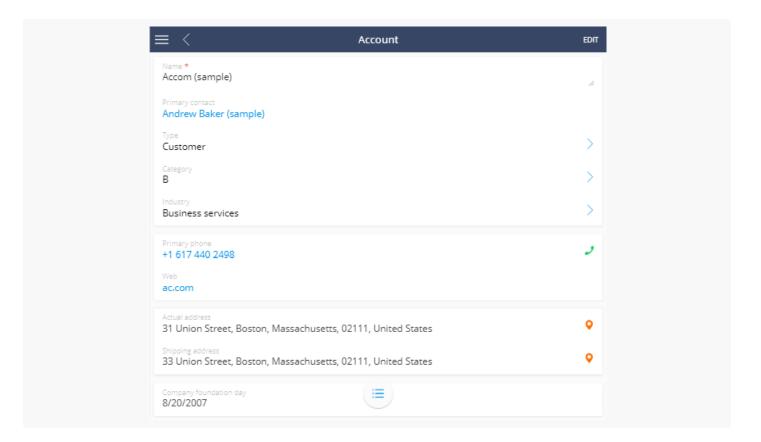
```
CBOЙCTBO TabletViewMode

{
    "TabletViewMode": "SinglePage",
    "CustomSchemas": [],
    "SyncOptions": {
        "SysSettingsImportConfig": [],
        "ModelDataImportConfig": []
},
```

```
"Modules": {},

"Models": {}
}
```

После сохранения схемы и перезагрузки мобильного приложения страница раздела на планшете будет отображаться во весь экран.



Добавить стандартную деталь с колонками



Для <u>добавления детали</u> в раздел мобильного приложения Creatio необходимо использовать мастер мобильного приложения.

Однако, если объект детали не является объектом какого-либо раздела мобильного приложения Creatio, то на странице детали вместо значений будет отображаться идентификатор связанной записи раздела. Для отображения значений необходимо выполнить конфигурирование схемы страницы детали.

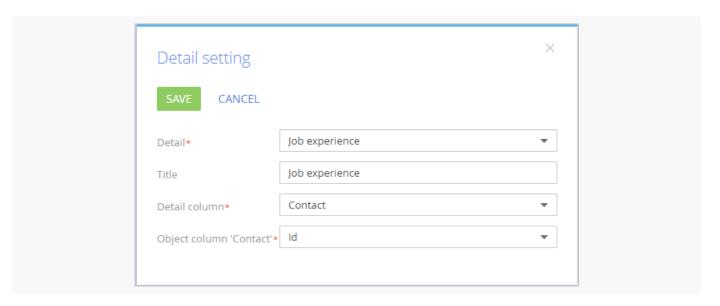
Пример. На страницу записи раздела [*Контакты*] мобильного приложения добавить деталь [*Карьера*]. В качестве основной отображать колонку [*Должность*].

Алгоритм реализации примера

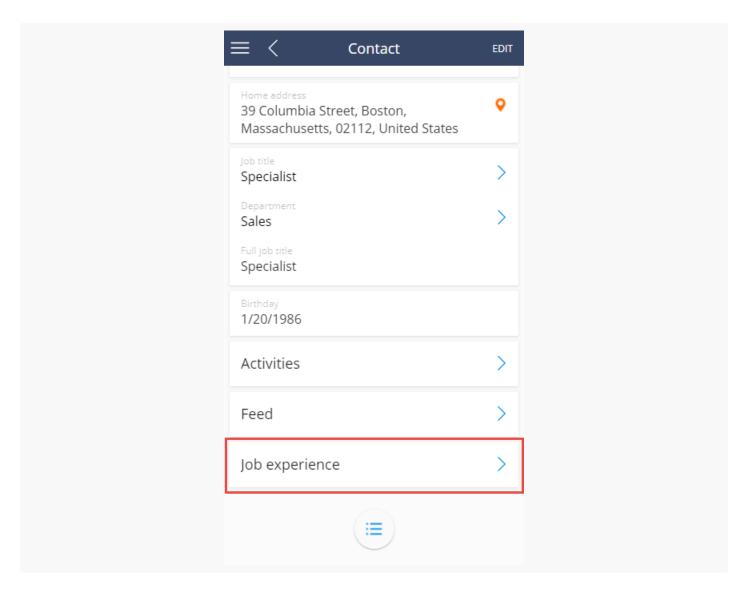
1. Добавить деталь [*Карьера*] с помощью мастера мобильного приложения

Чтобы добавить деталь на страницу записи, используйте <u>мастер мобильного приложения</u>. Для этого:

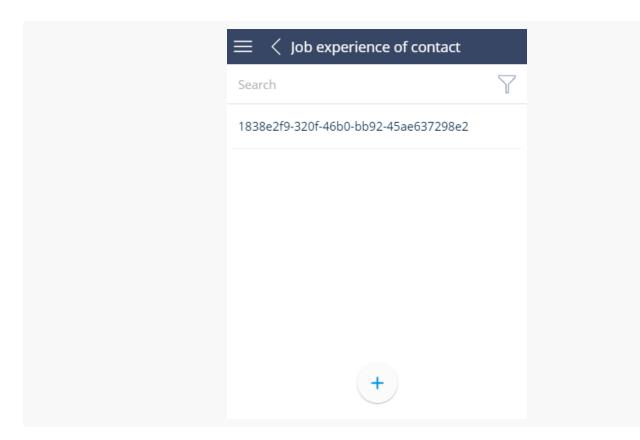
- 1. Откройте нужное рабочее место, например, [*Основное рабочее место*] ([*Main workplace*]) и нажмите кнопку [*Настроить разделы*] ([*Set up sections*]).
- 2. Выберите раздел [Контакты] и нажмите кнопку [Настроить детали] ([Details setup]).
- 3. Настройте деталь [*Карьера*] ([*Job experience*]).



После сохранения результатов настройки детали, раздела и рабочего места в мобильном приложении отобразится деталь [*Карьера*] ([*Job experience*]).



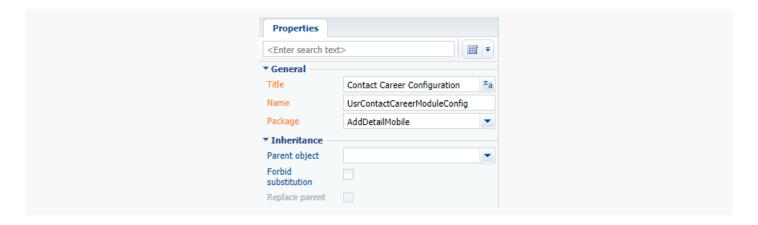
Однако, поскольку объект детали [Карьера] не является объектом раздела мобильного приложения Creatio, то на странице детали отображается значение основной колонки [Контакт] (идентификатор связанной записи контакта).



2. Создать схему модуля, в которой выполнить конфигурирование реестра детали

В разделе [*Конфигурация*] приложения Creatio в пользовательском пакете <u>создайте клиентский модуль</u> со следующими свойствами:

- [Заголовок] ([Title]) "Настройки карьеры контакта" ("Contact Career Configuration").
- [Название] ([Name]) "UsrContactCareerModuleConfig".



Добавьте в схему модуля исходный код.

UsrContactCareerModuleConfig

```
// Установка колонки [Должность] в качестве первичной.

Terrasoft.sdk.GridPage.setPrimaryColumn("ContactCareer", "JobTitle");

// Добавление колонки [Должность] в коллекцию первичных колонок.

Terrasoft.sdk.RecordPage.addColumn("ContactCareer", {
    name: "JobTitle",
    position: 1
    }, "primaryColumnSet");

// Удаление предыдущей первичной колонки [Контакт] из коллекции первичных колонок.

Terrasoft.sdk.RecordPage.removeColumn("ContactCareer", "Contact", "primaryColumnSet");
```

Здесь:

- ContactCareer название таблицы, которая соответствует детали (как правило оно совпадает с названием объекта детали).
- Job Title название колонки, которую требуется отобразить на странице.

3. Подключить схему модуля в манифесте мобильного приложения

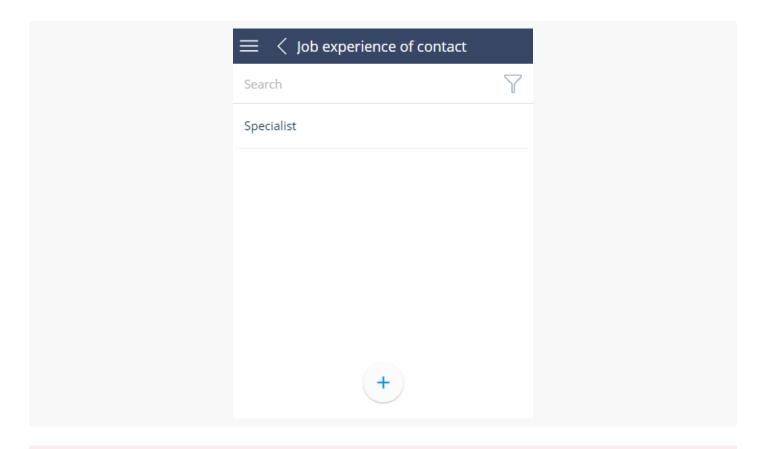
Для применения настроек реестра детали, выполненный в модуле UsrContactCareerModuleConfig , выполните следующие шаги:

- 1. Откройте в дизайнере клиентского модуля схему манифеста мобильного приложения MobileApplicationManifestDefaultWorkplace . Эта <u>схема создается</u> в пользовательском пакете мастером мобильного приложения.
- 2. Добавьте модуль UsrContactCareerModuleConfig в Секцию PagesExtensions модели ContactCareer.

```
}
```

3. Сохраните схему манифеста мобильного приложения.

В результате выполнения примера на странице детали [*Карьера*] ([*Job experience*]) будут отображаться записи по колонке [*Должность*].



Важно. Чтобы в мобильном приложении отображались сконфигурированные колонки, необходимо выполнить очистку кэша мобильного приложения. В некоторых случаях предварительно следует выполнить компиляцию приложения Creatio.

Добавить пользовательский дашборд в мобильное приложение



В приложении версии 7.10.3 (версия 7.10.5 мобильного приложения) добавлена поддержка итогов в мобильном приложении. Для получения настроек и данных итогов используется сервис <u>AnalyticsService</u>. Поддерживаются следующие дашборды — график, показатель, список и шкала.

Для добавления пользовательского типа дашборда в мобильное приложение необходимо:

1. Реализовать интерфейс настройки дашборда в приложении Creatio.

- 2. Добавить экземпляр реализованного пользовательского дашборда в приложение.
- 3. Реализовать отображение дашборда в мобильном приложении.

Важно. В данной статье описывается только реализация отображения дашборда в мобильном приложении.

Чтобы пользовательский тип дашборда отображался в мобильном приложении необходимо:

- 1. Реализовать получение данных пользовательского типа дашборда.
- 2. Добавить реализацию отображения дашборда в мобильном приложении.

Пример. На страницу итогов мобильного приложения добавить пользовательский дашборд, отображающий текущие дату и время.

Алгоритм реализации примера

1. Реализация получения данных пользовательского типа дашборда

Чтобы осуществить получение данных каждого пользовательского типа дашборда необходимо создать класс, который должен реализовать интерфейс IDashboardItemData или наследоваться от базового класса BaseDashboardItemData. Также для этого класс должен быть декорирован атрибутом DashboardItemData. Для реализации класса необходимо в пользовательский пакет добавить схему [Исходный код].

Реализовать класс получения данных для пользовательского типа дашборда customDashboardItem.

2. Реализация отображения пользовательского типа дашборда

2.1. Добавить класс отображения данных

Для этого необходимо в пользовательском пакете создать клиентский модуль (например, UsrMobileCustomDashboardItem). В созданном модуле необходимо реализовать класс, расширяющий базовый класс Terrasoft.configuration.controls.BaseDashboardItem.

```
UsrMobileCustomDashboardItem

Ext.define("Terrasoft.configuration.controls.CustomDashboardItem", {
    extend: "Terrasoft.configuration.controls.BaseDashboardItem",
    // Отображает значение, переданное через свойство customValue
    updateRawConfig: function(config) {
        this.innerHtmlElement.setHtml(config.customValue);
    }

});
```

2.2. Добавить новый тип и реализующий его класс в перечисление Terrasoft. DashboardItemClassName

Для этого в модуле, созданном на предыдущем шаге, необходимо добавить исходный код.

```
CustomDashboardItem

Terrasoft.DashboardItemClassName.CustomDashboardItem = "Terrasoft.configuration.controls.CustomDashboardItem" = "Terrasoft.configuration.customDashboardItem" = "Terrasoft.configuration.customDashboardItem" = "Terrasoft.customDashboardItem" = "Terrasoft.customDashboardI
```

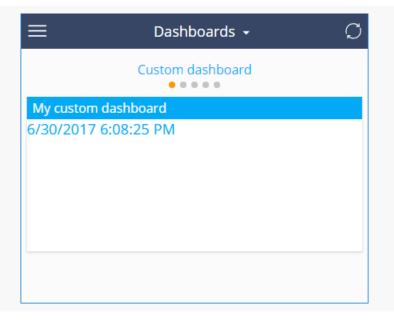
2.3. Добавить название созданной клиентской схемы в манифест мобильного приложения

В файле манифеста мобильного приложения необходимо в массив CustomSchemas добавить название созданной схемы модуля.

```
CustomSchemas

{
    "SyncOptions": {
         ...
    },
    "CustomSchemas": ["UsrMobileCustomDashboardItem"],
    "Modules": {...},
    "Models": {...}
}
```

После сохранения всех изменений дашборд будет отображаться в разделе [*Итоги*] мобильного приложения.



Важно. Чтобы дашборд отображался в мобильном приложении, он обязательно должен быть добавлен в основное приложение Creatio, с которым синхронизирована мобильная версия приложения.

Добавить кнопку для отображения имени контакта



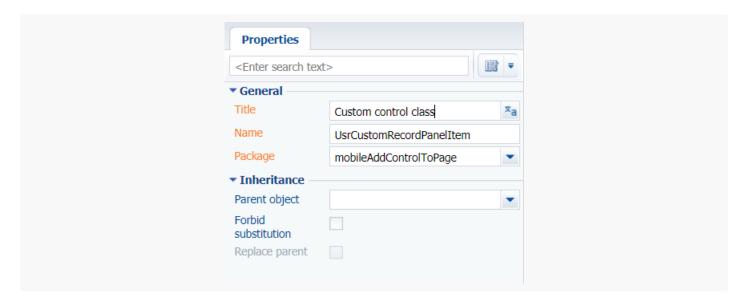
Пример. На страницу записи раздела [*Контакты*] мобильного приложения добавить кнопку, нажатие на которую будет показывать сообщение с полным именем контакта.

Алгоритм реализации примера

1. Создать пользовательский класс-наследник Terrasoft.RecordPanelItem

В разделе [*Конфигурация*] приложения Creatio в пользовательском пакете <u>создайте клиентский модуль</u> со следующими свойствами:

- [Заголовок] ([Title]) "Класс пользовательского элемента управления" ("Custom control class").
- [Название] ([Name]) "UsrCustomRecordPanelItem".



В модуль добавьте исходный код.

```
CustomRecordPanelItem
Ext.define("Terrasoft.controls.CustomRecordPanelItem", {
    extend: "Terrasoft.RecordPanelItem",
   xtype: "cftestrecordpanelitem",
   // Конфигурационный объект создаваемого элемента.
    config: {
        items: [
                xtype: "container",
                layout: "hbox",
                items: [
                    {
                        xtype: "button",
                        id: "clickMeButton",
                        text: "Full name",
                        flex: 1
                    }
```

```
]

}

// Метод инициализирует созданный элемент и добавляет метод-обработчик нажатия кнопки.
initialize: function() {
    var clickMeButton = Ext.getCmp("clickMeButton");
    clickMeButton.element.on("tap", this.onClickMeButtonClick, this);
},

// Метод-обработчик нажатия кнопки.
onClickMeButtonClick: function() {
    var record = this.getRecord();
    Terrasoft.MessageBox.showMessage(record.getPrimaryDisplayColumnValue());
}

});
```

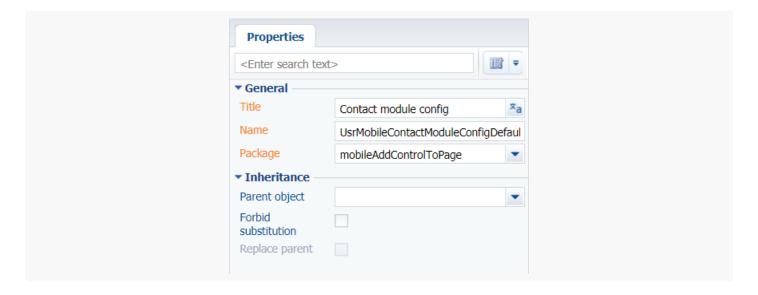
В классе описан конфигурационный объект созданного элемента управления и два метода:

- initialize() метод-обработчик события нажатия кнопки;
- onClickMeButtonClick() метод, который инициализует созданный элемент и присваивает событию нажатия на кнопку ссылку на метод-обработчик.

2. Создать схему модуля, в которой выполнить конфигурирование страницы раздела

В разделе [*Конфигурация*] приложения Creatio в пользовательском пакете <u>создайте клиентский модуль</u> со следующими свойствами:

- [Заголовок] ([Title]) "Конфигурация раздела контактов" ("Contact module config").
- [Название] ([Name]) "UsrMobileContactModuleConfigDefaultWorkplace".



Добавьте в схему модуля исходный код.

```
UsrMobileContactModuleConfigDefaultWorkplace

Terrasoft.sdk.RecordPage.addPanelItem("Contact", {
    xtype: "cftestrecordpanelitem",
    position: 1,
    componentConfig: {
    }
});
```

Здесь вызывается метода addPanelItem() класса Terrasoft.sdk.RecordPage, с помощью которого созданный элемент добавляется на страницу раздела.

- 3. Подключить схемы модулей в манифесте мобильного приложения Для применения настроек страницы раздела, выполненных в модуле UsrMobileContactModuleConfigDefaultWorkplace, выполните следующие шаги:
- 1. Откройте в дизайнере клиентского модуля схему манифеста мобильного приложения MobileApplicationManifestDefaultWorkplace. Эта схема создается в пользовательском пакете мастером мобильного приложения.
- 2. Добавьте модуль UsrCustomRecordPanelItem В СЕКЦИЮ CustomSchemas, а модуль UsrContactCareerModuleConfig В СЕКЦИЮ PagesExtensions МОДЕЛИ Contact.

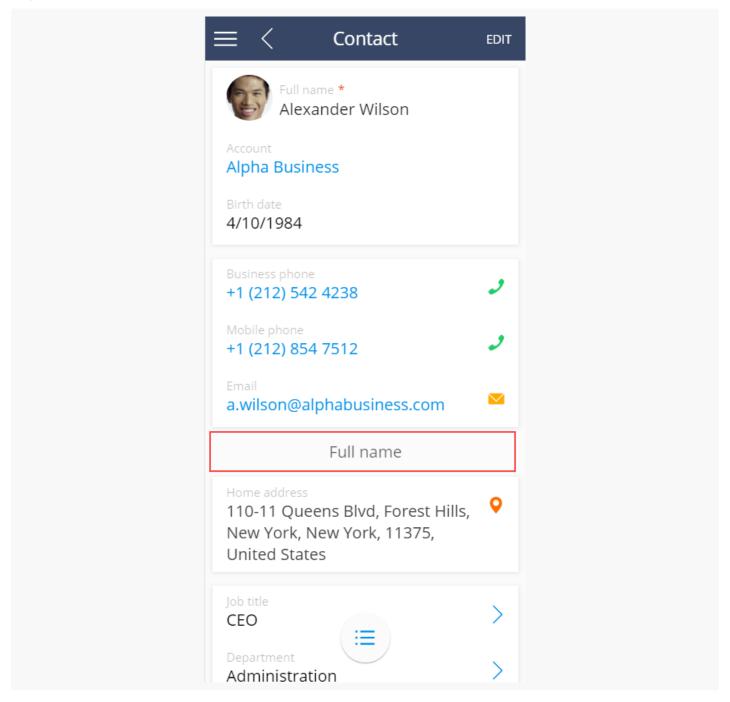
```
Добавление модулей
{
    "CustomSchemas": [
        "UsrCustomRecordPanelItem.js"
    ],
    "SyncOptions": {},
    "Modules": {},
    "Models": {
        "Contact": {
            "RequiredModels": [],
            "ModelExtensions": [],
            "PagesExtensions": [
                "UsrMobileContactModuleConfigDefaultWorkplace.js"
        }
    }
}
```

3. Сохраните схему манифеста мобильного приложения.

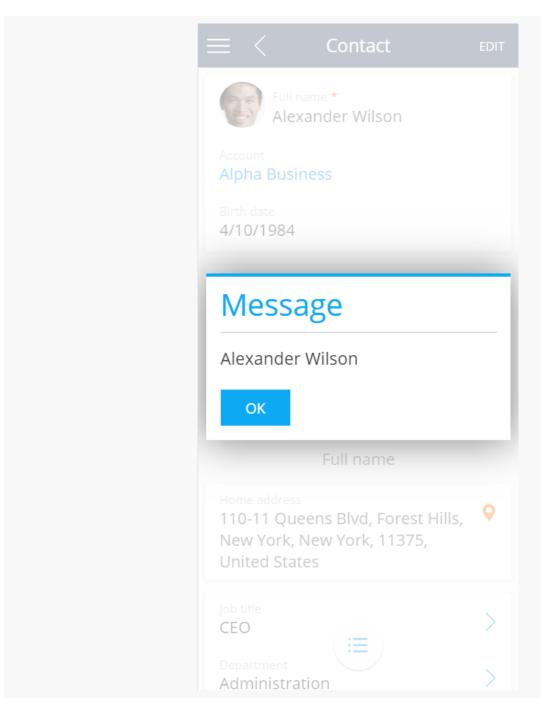
В результате выполнения примера на странице контакта появится элемент управления, при нажатии на

который отобразится сообщение с полным именем контакта.

Результат выполнения кейса. Добавление кнопки



Результат выполнения кейса. Нажатие на кнопку



Свойство ModuleGroups



Группы модулей приложения. Используется для настройки группы меню. Описывает верхнеуровневую настройку групп Position.

Свойство конфигурационного объекта

Position

Свойство Modules



• Сложный

Модули мобильного приложения. Модуль представляет собой раздел приложения. Каждый модуль в свойстве мodules конфигурационного объекта манифеста описывается конфигурационным объектом со свойствами, приведенными в таблице. Имя конфигурационного объекта раздела должно совпадать с названием модели, которая предоставляет данные раздела.

Свойства конфигурационного объекта

Group

Группа меню приложения, в которой размещается раздел. Задается строкой с названием соответствующего раздела меню из свойства ModuleGroups конфигурационного объекта манифеста.

Model

Название модели, которая предоставляет данные раздела. Задается строкой с названием одной из моделей, объявленных в свойстве Models конфигурационного объекта манифеста.

Position

Позиция раздела в группе главного меню. Задается числовым значением, начиная с 0.

Title

Заголовок раздела. Строка с названием локализованного значения заголовка раздела. Локализованное значение заголовка раздела должно быть добавлено в блок [*LocalizableStrings*] схемы манифеста.

Icon

Свойство, предназначенное для подключения пользовательского изображения к разделу в меню пользовательского интерфейса версии 1.

IconV2

Свойство, предназначенное для подключения пользовательского изображения к разделу в меню пользовательского интерфейса версии 2.

Hidden

Признак, отображается ли данный раздел в меню (true - ckpыt, false - oтображается). Необязательное свойство. По умолчанию — false.

Свойство Icons



Свойство предназначено для подключения к мобильному приложению пользовательских изображений.

Задается массивом конфигурационных объектов, каждый из которых имеет свойства, приведенные в таблице.

Свойства конфигурационного объекта

ImageListId

Идентификатор списка изображений.

ImageId

Идентификатор подключаемого изображения из списка ImageListId.

```
Подключение пользовательских изображений
```

Свойства DefaultModuleImageId и DefaultModuleImageIdV2 🖪



Свойства предназначены для установки уникальных идентификаторов изображений по умолчанию для вновь создаваемых разделов или для разделов, у которых не указаны идентификаторы изображений в свойствах Icon или IconV2 свойства мodules конфигурационного объекта манифеста.

Установка уникальных идентификаторов изображений

//Идентификатор изображения по умолчанию для пользовательского интерфейса V1. "DefaultModuleImageId": "423d3be8-de6b-4f15-a81b-ed454b6d03e3", //Идентификатор изображения по умолчанию для пользовательского интерфейса V2. "DefaultModuleImageIdV2": "1c92d522-965f-43e0-97ab-2a7b101c03d4"

Свойство Models



Содержит импортируемые модели приложения. Каждая модель в свойстве описывается конфигурационным объектом с соответствующим именем. Свойства конфигурационного объекта модели представлены в таблице.

Свойства конфигурационного объекта

Grid

Название схемы страницы реестра модели. Страница будет сгенерирована автоматически с именем Mobile[Название_модели][Тип_страницы]Раде. Не обязателен для заполнения.

Preview

Название схемы страницы просмотра элемента модели. Страница будет сгенерирована автоматически с именем Mobile[Название_модели][Тип_страницы]Раде . Не обязателен для заполнения.

Edit

Название схемы страницы элемента модели. Страница будет сгенерирована автоматически с именем Mobile[Название_модели][Тип_страницы]Раде. Не обязателен для заполнения.

RequiredModels

Названия моделей, от которых зависит данная модель. Необязательное свойство. Здесь перечисляются все модели, колонки которых добавляются в текущую модель, а также колонки, на которые у текущей модели есть внешние ключи.

ModelExtensions

Расширения модели. Необязательное свойство. Представляет собой массив названий схем, в которых реализуются дополнительные настройки для модели (например, добавление в модель бизнес-правил, событий, значений по умолчанию для полей и т.д.).

PagesExtensions

Расширения страниц модели. Необязательное свойство. Представляет собой массив названий схем, в которых реализуются дополнительные настройки для различных типов страниц модели (добавление деталей, установка заголовков и т.д.).

Свойство PreferedFilterFuncType



Свойство предназначено для явного определения операции, которая будет использоваться при поиске и фильтрации данных в реестре (в разделах, деталях, справочниках). Значение для свойства задается перечислением Terrasoft.FilterFunctions.

Функции фильтрации (Terrasoft.FilterFunctions)

SubStringOf
Определяет, является ли строка, переданная в качестве аргумента, подстрокой колонки рroperty.
ToUpper
Приводит значения колонки, заданной в ргорегту , к верхнему регистру.
EndsWith
Проверяет, оканчивается ли значение колонки ргорегту значением, переданным в качестве аргумента.
StartsWith
Проверяет, начинается ли значение колонки ргоретту значением, переданным в качестве аргумента.
Year
Возвращает год по значению колонки property.
Month
Возвращает месяц по значению колонки property.
Day

Возвращает день по значению колонки property.

Ιn

Проверяет вхождение значения колонки property в диапазон значений, переданных в качестве аргумента функции.

NotIn

Проверяет невхождение значения колонки property в диапазон значений, переданных в качестве аргумента функции.

Like

Определяет, совпадает ли значение колонки property с заданным шаблоном.

Если данное свойство явно не инициализировано в манифесте, то по умолчанию для поиска и фильтрации данных используется функция Terrasoft.FilterFunctions.StartWith, так как это обеспечивает использование соответствующих индексов в таблицах базы данных SQLite.

Свойство CustomSchemas



Свойство предназначено для подключения к мобильному приложению дополнительных схем (пользовательских схем с исходным кодом, написанным на JavaScript), расширяющих возможности приложения. Это могут быть, например, дополнительные классы, реализованные разработчиком в рамках проекта, либо утилитные классы, реализующие служебную функциональность, упрощающую работу разработчика, и т.д.

Значение свойства задается массивом с именами подключаемых пользовательских схем.

```
Подключение дополнительных пользовательских схем.

// Подключение дополнительных пользовательских схем.

"CustomSchemas": [

// Пользовательская схема регистрации действий.

"MobileActionCheckIn",

// Пользовательская схема утилит.

"CustomMobileUtilities"
]
```

Свойство SyncOptions •



Описывает параметры для настройки синхронизации данных. Содержит конфигурационный объект со свойствами, представленными в таблице.

Свойства конфигурационного объекта для настроек синхронизации

ImportPageSize	
Количество страниц, импортируемых в одном потоке.	
PagesInImportTransaction	
Количество потоков импорта.	
SysSettingsImportConfig	
Массив импортируемых системных настроек.	
SysLookupsImportConfig	
Массив импортируемых системных справочников.	
ModelDataImportConfig	

Массив моделей, для которых будут загружаться данные при синхронизации.

В массиве моделей мodelDataImportConfig для каждой модели можно указать дополнительные параметры синхронизации, список загружаемых колонок, а также условия фильтрации загружаемых данных модели. Если при синхронизации должна загружаться полная модель, в массиве просто указывается объект с именем модели. Если к модели должны применяться дополнительные условия при синхронизации, в массив мodelDataImportConfig добавляется конфигурационный объект со свойствами.

Свойства конфигурационного объекта для настройки синхронизации модели

Name

Название модели (см. свойство Models конфигурационного объекта манифеста).

SyncColumns

Массив колонок модели, для которых импортируются данные. Помимо явно перечисленных колонок,

при синхронизации в обязательном порядке будут импортироваться системные колонки (CreatedOn , CreatedBy , ModifiedOn , ModifiedBy) и колонка, первичная для отображения.

SyncFilter

Фильтр, накладываемый на модель при импорте модели.

Фильтр SyncFilter, накладываемый на модель при импорте модели представляет собой конфигурационный объект со свойствами.

Свойства конфигурационного объекта фильтра модели

type

Тип фильтра. Задается значением перечисления Terrasoft.FilterTypes . Необязательное свойство. По умолчанию Terrasoft.FilterTypes.Simple .

Возможные значения (Terrasoft.FilterTypes)

Simple	Фильтр с одним условием.
Group	Групповой фильтр с несколькими условиями.

logicalOperation

Логическая операция объединения коллекции фильтров (для фильтров с типом Terrasoft.FilterTypes.Group). Задается значением перечисления Значение по умолчанию - Terrasoft.FilterLogicalOperations.And .

Возможные значения (Terrasoft.FilterLogicalOperations)

0r	Логическая операция ИЛИ.
And	Логическая операция И.

subfilters

Коллекция фильтров, применяемых к модели. Обязательное свойство для типа фильтра Terrasoft.FilterTypes.Group. Фильтры между собой объединяются логической операцией, указанной в свойстве logicalOperation. Каждый фильтр представляет собой конфигурационный объект фильтра.

property

Название колонки модели, по которой выполняется фильтрация. Обязательное свойство для типа фильтра Terrasoft.FilterTypes.Simple .

valueIsMacrosType

Признак, определяющий, является ли значение для фильтрации макросом. Необязательное свойство. Может принимать значения: true, если для фильтрации используется макрос, иначе— false.

value

Значение для фильтрации колонки, указанной в свойстве property. Обязательное свойство для типа фильтра Terrasoft.FilterTypes.Simple. Может задаваться непосредственно значением для фильтрации (в том числе, может быть null) либо макросом (для этого свойство valueIsMacrosType должно иметь значение true). Макросы, которые можно использовать в качестве значения свойства, содержатся в перечислении Terrasoft.ValueMacros.

Возможные значения (Terrasoft.ValueMacros)

CurrentUserContactId	Идентификатор текущего пользователя.
CurrentDate	Текущая дата.
CurrentDateTime	Текущие дата и время.
CurrentDateEnd	Полная дата окончания текущей даты.
CurrentUserContactName	Имя текущего контакта.
CurrentUserContact	Идентификатор и имя текущего контакта.
SysSettings	Значение системной настройки. Имя системной настройки передается в свойстве macrosParams.
CurrentTime	Текущее время.
CurrentUserAccount	Идентификатор и имя контрагента текущего пользователя.
GenerateUId	Сгенерированный идентификатор.

macrosParams

Значения, которые передаются в макрос в качестве параметра. Необязательное свойство. В настоящее время используется только для макроса Terrasoft. ValueMacros. SysSettings.

isNot

Определяет, применяется к фильтру оператор отрицания. Необязательное свойство. Принимает значение true, если к фильтру применяется оператор отрицания, иначе — false.

funcType

Тип функции, которая применяется к колонке модели, заданой в свойстве property . Необязательное свойство. Может принимать значения перечисления Terrasoft.FilterFunctions . Значения аргументов для функций фильтрации задаются в свойстве funcArgs . Значение, с которым сравнивается результат функции, задается свойством value .

Возможные значения (Terrasoft.FilterFunctions)

SubStringOf	Определяет, является ли строка, переданная в качестве аргумента, подстрокой колонки property.
ToUpper	Приводит значения колонки, заданной в property, к верхнему регистру.
EndsWith	Проверяет, оканчивается ли значение колонки ргорегту значением, переданным в качестве аргумента.
StartsWith	Проверяет, начинается ли значение колонки property значением, переданным в качестве аргумента.
Year	Возвращает год по значению колонки property.
Month	Возвращает месяц по значению колонки property.
Day	Возвращает день по значению колонки property.
In	Проверяет вхождение значения колонки рroperty в диапазон значений, переданных в качестве аргумента функции.
NotIn	Проверяет невхождение значения колонки property в диапазон значений, переданных в качестве аргумента функции.
Like	Определяет, совпадает ли значение колонки property с заданным шаблоном.

funcArgs

Массив значений аргументов для функции фильтрации, заданной в свойстве funcType. Порядок значений в массиве funcArgs должен соответствовать порядку параметров функции funcType.

name

Имя фильтра или группы фильтров. Необязательное свойство.

modelName

Название модели, для которой выполняется фильтрация. Необязательное свойство. Указывается, если фильтрация выполняется по колонкам связанной модели.

assocProperty

Колонка связанной модели, по которой осуществляется связь с основной моделью. В качестве колонки для связи у основной модели выступает первичная колонка.

operation

Тип операции фильтрации. Необязательный параметр. Может принимать значения из перечисления Terrasoft.FilterOperation. По умолчанию имеет значение Terrasoft.FilterOperation.General.

Возможные значения (Terrasoft.FilterOperation)

General	Стандартная фильтрация.
Any	Фильтрация с применением фильтра exists.

compareType

Тип операции сравнения в фильтре. Необязательный параметр. Принимает значения из перечисления Terrasoft.ComparisonType . По умолчанию — Terrasoft.ComparisonType.Equal .

Возможные значения (Terrasoft.ComparisonType)

Equal	Равно.
LessOrEqual	Меньше или равно.
NotEqual	Не равно.
Greater	Больше.
GreaterOrEqual	Больше или равно.
Less	Меньше.

Свойство SyncOptions.ModelDataImportConfig.QueryFilter

Доступно в приложении, начиная с версии 7.12.1, и в мобильном приложении, начиная с версии 7.12.3.

Свойство синхронизации QueryFilter позволяет настроить фильтрацию данных указанной модели при импорте с помощью <u>сервиса работы с данными DataService</u>. Ранее для фильтрации данных использовалось свойство SyncFilter, а импорт выполнялся с помощью <u>сервиса работы с данными OData</u>.

Важно. Импорт данных с помощью сервиса работы с данными DataService доступен только для платформ Android и iOS. Для платформы Windows используется OData.

Формат фильтра QueryFilter представляет собой <u>набор параметров</u> в виде JSON-объекта, передаваемых в запросе к сервису работы с данными DataService.

Пример exists-фильтра

```
"SyncOptions": {
   "ModelDataImportConfig": [
      {
         "Name": "ActivityParticipant",
         "QueryFilter": {
            "logicalOperation": 0,
            "filterType": 6,
            "rootSchemaName": "ActivityParticipant",
            "items": {
               "ActivityFilter": {
                  "filterType": 5,
                  "leftExpression": {
                     "expressionType": 0,
                     "columnPath": "Activity.[ActivityParticipant:Activity].Id"
                  },
                  "subFilters": {
                     "logicalOperation": 0,
                     "filterType": 6,
                     "rootSchemaName": "ActivityParticipant",
                     "items": {
                        "ParticipantFilter":{
                            "filterType": 1,
                            "comparisonType": 3,
                            "leftExpression": {
                               "expressionType": 0,
                               "columnPath": "Participant"
                           },
                            "rightExpression": {
                               "expressionType": 1,
                               "functionType": 1,
```

```
"macrosType": 2
}
}
}
}
}
}
```

Реестр мобильного приложения



Важно. Актуально для мобильного приложения версии 7.11.1 и выше.

SDK реестра — это инструмент, позволяющий настраивать внешний вид реестра, сортировку, логику поиска и т. д. Он реализован в классе Terrasoft.sdk.GridPage.

Настроить реестр раздела



Пример. Настроить реестр раздела [*Обращения*] ([*Cases*]) таким образом, чтобы отображался заголовок с темой обращения, подзаголовок с датой регистрации и номером, а также описание обращения в виде многострочного поля.

Реализация примера

Для настройки реестра необходимо использовать приведенный ниже исходный код.

Настройка реестра раздела [Обращения] ([Cases])

```
// Настройка первичной колонки с темой обращения.

Terrasoft.sdk.GridPage.setPrimaryColumn("Case", "Subject");

// Установка подзаголовка с датой регистрации и номером обращения.

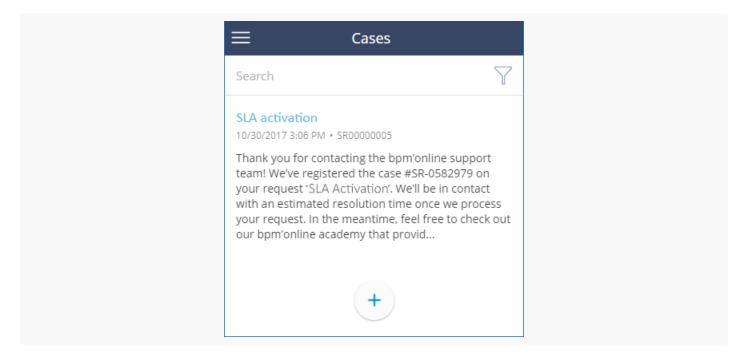
Terrasoft.sdk.GridPage.setSubtitleColumns("Case", ["RegisteredOn","Number"]);

// Добавление многострочного поля с описанием.

Terrasoft.sdk.GridPage.setGroupColumns("Case", [
```

```
name: "Symptoms",
  isMultiline: true
}]);
```

В результате реестр будет отображаться в следующем виде.



Класс GridPage



Класс позволяет настраивать внешний вид реестра, сортировку, логику поиска и т. д.

Методы

setPrimaryColumn(modelName, column)

Устанавливает первичную колонку для отображения. Настраивает отображение заголовка записи реестра.

Параметры

modelName	Название модели.
column	Название колонки.

Пример вызова

```
Terrasoft.sdk.GridPage.setPrimaryColumn("Case", "Subject");
```

setSubtitleColumns(modelName, columns)

Устанавливает колонки, которые отображаются под заголовком. Настраивает отображение подзаголовка в виде списка колонок с разделителем.

Параметры

modelName	Название модели.	
columns	Массив колонок или конфигурационных объектов колонок.	

```
Пример вызова (вариант 1)
Terrasoft.sdk.GridPage.setSubtitleColumns("Case", ["RegisteredOn","Number"]);
```

```
Пример вызова (вариант 2)

Terrasoft.sdk.GridPage.setSubtitleColumns("Case", ["RegisteredOn", {
    name: "Number",
    convertFunction: function(values){
        return values.Number;

}]);
```

setGroupColumns(modelName, columns)

Устанавливает группу с колонками, которые отображаются вертикально. Настраивает отображение группы колонок.

Параметры

modelName	Название модели.	
columns	Массив колонок или конфигурационных объектов колонок.	

```
Пример вызова (вариант 1)
Terrasoft.sdk.GridPage.setGroupColumns("Case", ["Symptoms"]);
```

```
Пример вызова (вариант 2)

Terrasoft.sdk.GridPage.setGroupColumns("Case", [{
    name: "Symptoms",
    // Отображать как многострочное поле.
    isMultiline: true,
    // Имя локализованной строки.
    label: "CaseGridSymptomsColumnLabel",
    convertFunction: function(values) {
        return values.Symptoms;
    }
}]);
```

```
setImageColumn()
Устанавливает колонку изображения.

setOrderByColumns()
Устанавливает сортировку реестра.

setSearchColumn()
Устанавливает колонку поиска.

setSearchColumns()
Устанавливает колонки поиска.

setSearchPlaceholder()
Устанавливает текст подсказки в поле поиска.
```

Устанавливает заголовок страницы реестра.

setTitle()

Бизнес-правила мобильного приложения



• Сложный

Бизнес-правила — это один из механизмов Creatio, позволяющий настраивать поведение полей на странице записи. С их помощью можно, например, настроить обязательность и видимость полей, их доступность и т. п.

Важно. Бизнес-правила работают только на страницах записи и просмотра записей.

Добавление бизнес-правила на страницу выполняется с помощью метода Terrasoft.sdk.Model.addBusinessRule(name, config), где

- name название модели, связанной со страницей записи, например, "Contact".
- config объект, определяющий свойства бизнес-правила. Перечень свойств зависит от конкретного типа бизнес-правила.

В мобильном приложении существует возможность добавлять бизнес-правило, реализующее пользовательскую логику — пользовательское бизнес-правило. Для такого бизнес-правила предусмотрен тип Terrasoft.RuleTypes.Custom.

Выполнить фильтрацию



• Сложный

Пример 1

Пример. Отфильтровать значения в колонке по условию.

На детали [*Продукты в счете*] при выборе значения из справочной колонки [*Продукт*] доступны только те продукты, у которых колонка [*Active*] содержит значение true.

Реализация примера

Пример фильтрации

```
Terrasoft.sdk.Model.addBusinessRule("InvoiceProduct", {
   ruleType: Terrasoft.RuleTypes.Filtration,
   events: [Terrasoft.BusinessRuleEvents.Load],
   triggeredByColumns: ["Product"],
   filters: Ext.create("Terrasoft.Filter", {
```

```
modelName: "Product",
    property: "Active",
    value: true
    })
});
```

Пример 2

Пример. Отфильтровать значения в колонке по значению другой колонки.

На странице записи раздела [Cчета], поле [Kонтакau] должно фильтроваться на основании значения в поле [Kонтрагенau].

Реализация примера

Пример фильтрации

```
Terrasoft.sdk.Model.addBusinessRule("Invoice", {
    ruleType: Terrasoft.RuleTypes.Filtration,
    events: [Terrasoft.BusinessRuleEvents.Load, Terrasoft.BusinessRuleEvents.ValueChanged],
    triggeredByColumns: ["Account"],
    filteredColumn: "Contact",
    filters: Ext.create("Terrasoft.Filter", {
        property: "Account"
    })
});
```

Пример 3

Пример. Добавить и удалить фильтрацию по пользовательской логике.

Реализация примера

Пример фильтрации

```
Terrasoft.sdk.Model.addBusinessRule("Activity", {
   name: "ActivityResultByAllowedResultFilterRule",
   position: 1,
   ruleType: Terrasoft.RuleTypes.Custom,
   triggeredByColumns: ["Result"],
```

```
events: [Terrasoft.BusinessRuleEvents.ValueChanged, Terrasoft.BusinessRuleEvents.Load],
    executeFn: function(record, rule, column, customData, callbackConfig) {
        var allowedResult = record.get("AllowedResult");
        var filterName = "ActivityResultByAllowedResultFilter";
        if (!Ext.isEmpty(allowedResult)) {
            var allowedResultIds = Ext.JSON.decode(allowedResult, true);
            var resultIdsAreCorrect = true;
            for (var i = 0, ln = allowedResultIds.length; i < ln; i++) {
                var item = allowedResultIds[i];
                if (!Terrasoft.util.isGuid(item)) {
                    resultIdsAreCorrect = false;
                    break;
                }
            }
            if (resultIdsAreCorrect) {
                var filter = Ext.create("Terrasoft.Filter", {
                    name: filterName,
                    property: "Id",
                    funcType: Terrasoft.FilterFunctions.In,
                    funcArgs: allowedResultIds
                });
                record.changeProperty("Result", {
                    addFilter: filter
                });
            } else {
                record.changeProperty("Result", {
                    removeFilter: filterName
                });
            }
        } else {
            record.changeProperty("Result", {
                removeFilter: filterName
            });
        Ext.callback(callbackConfig.success, callbackConfig.scope, [true]);
   }
});
```

Выделить поле по условию



Пример. Требуется выделить поле с результатом активности, если ее статус "Завершена", само поле [*Результат*] не заполнено и есть значение в колонке [*ProcessElementId*].

Реализация примера

Выделение поля по условию // Правило для страницы активности. Terrasoft.sdk.Model.addBusinessRule("Activity", { // Название бизнес-правила. name: "ActivityResultRequiredByStatusFinishedAndProcessElementId", // Тип бизнес-правила: пользовательское. ruleType: Terrasoft.RuleTypes.Custom, // Правило инициируется колонками Status и Result. triggeredByColumns: ["Status", "Result"], // Правило отработает перед сохранением данных и после изменения данных. events: [Terrasoft.BusinessRuleEvents.ValueChanged, Terrasoft.BusinessRuleEvents.Save], // Функция-обработчик. executeFn: function(record, rule, column, customData, callbackConfig) { // Признак корректности свойства и правила. var isValid = true; // Значение колонки ProcessElementId. var processElementId = record.get("ProcessElementId"); // Если значение не пустое. if (processElementId && processElementId !== Terrasoft.GUID EMPTY) { // Установка признака корректности. isValid = !(record.get("Status.Id") === Terrasoft.Configuration.ActivityStatus.Finis Ext.isEmpty(record.get("Result"))); // Изменение свойств колонки Result. record.changeProperty("Result", { // Установка признака корректности колонки. isValid: { value: isValid, message: Terrasoft.LS["Sys.RequirementRule.message"] } }); // Асинхронный возврат значений. Ext.callback(callbackConfig.success, callbackConfig.scope, [isValid]); });

Сбросить отрицательные значения в 0



Пример. Реализовать логику сбрасывания отрицательных значений в 0.

Реализация примера

C6poc отрицательных значений в 0 Terrasoft.sdk.Model.addBusinessRule("Opportunity", { name: "OpportunityAmountValidatorRule", ruleType: Terrasoft.RuleTypes.Custom, triggeredByColumns: ["Amount"], events: [Terrasoft.BusinessRuleEvents.ValueChanged, Terrasoft.BusinessRuleEvents.Save], executeFn: function(model, rule, column, customData, callbackConfig) { var revenue = model.get("Amount"); if ((revenue < 0) || Ext.isEmpty(revenue)) { model.set("Amount", 0, true); } Ext.callback(callbackConfig.success, callbackConfig.scope); } });</pre>

Сгенерировать заголовок активности



Пример. Реализовать генерацию заголовка активности для решения FieldForce.

Реализация примера

Генерация заголовка активности

Свойства объекта config



• Сложный

Базовые бизнес-правила

Базовое бизнес-правило является абстрактным классом, т.е. все бизнес-правила должны быть его наследниками.

Свойства конфигурационного объекта config могут быть использованы наследниками базового бизнесправила.

Свойства конфигурационного объекта config

ruleType

Тип правила. Значение должно входить в перечисление Terrasoft.RuleTypes.

triggeredByColumns

Массив колонок, инициирующих срабатывание бизнес-правила.

message

Текстовое сообщение, которое выводится под элементом управления, который связан с колонкой, в случае невыполнения бизнес-правила. Необходимо для правил, сообщающих пользователю предупреждающую информацию.

name

Уникальное имя бизнес-правила. Необходимо, если нужно удалить правило методами Terrasoft.sdk.

position

Позиция бизнес-правила, определяющая приоритет вызова правила в очереди текущих запускаемых правил.

events

Массив событий, определяющий время запуска бизнес-правил. Должен содержать значения, входящие в перечисление Terrasoft.BusinessRuleEvents.

Возможные значения (Terrasoft.BusinessRuleEvents)

Save	Правило отработает перед сохранением данных.
ValueChanged	Правило отработает после изменения данных (при редактировании).
Load	Правило отработает при открытии страницы записи.

Бизнес-правило [*Обязательность заполнения*] (Terrasoft.RuleTypes.Requirement) —

Определяет обязательность заполнения поля на странице записи.

Свойства конфигурационного объекта config

ruleType

Для этого правила должно содержать Значение Terrasoft.RuleTypes.Requirement.

requireType

Тип проверки. Значение должно входить в перечисление Terrasoft.RequirementTypes . Правило может проверять одну или все колонки из triggeredByColumns .

triggeredByColumns

Maccub колонок, инициирующих срабатывание бизнес-правила. Если тип проверки равен Terrasoft.RequirementTypes.Simple, то должна быть указана одна колонка в массиве.

Возможные значения (Terrasoft.RequirementTypes)

Simple	Проверка значения в одной колонке.
0ne0f	Одна из колонок, указанных в triggeredByColumns должна быть обязательно заполнена.

Пример использования

```
Terrasoft.sdk.Model.addBusinessRule("Contact", {
    ruleType: Terrasoft.RuleTypes.Requirement,
    requireType : Terrasoft.RequirementTypes.OneOf,
    events: [Terrasoft.BusinessRuleEvents.Save],
    triggeredByColumns: ["HomeNumber", "BusinessNumber"],
    columnNames: ["HomeNumber", "BusinessNumber"]
});
```

Бизнес-правило [Видимость] (Terrasoft.RuleTypes.Visibility) •

С помощью этого бизнес-правила можно скрывать и отображать поля по определенному условию.

Свойства конфигурационного объекта config

ruleType

Для этого правила должно содержать значение Terrasoft.RuleTypes.Visibility.

triggeredByColumns

Массив колонок, инициирующих срабатывание бизнес-правила.

events

Массив событий, определяющий время запуска бизнес-правил. Должен содержать значения, входящие в перечисление Terrasoft.BusinessRuleEvents.

conditionalColumns

Массив условий срабатывания бизнес-правила. Обычно это определенные значения колонок.

dependentColumnNames

Массив названий колонок, к которым применяется данное бизнес-правило.

Пример использования

```
],
  triggeredByColumns: ["Type"],
  dependentColumnNames: ["IsRx", "IsOTC"]
});
```

Поля, связанные с колонками IsRx и Isotc будут отображены, если колонка Туре будет содержать значение, определенное константой Terrasoft.Configuration.Consts.AccountTypePharmacy.

```
Terrasoft.Configuration.Consts = {
    AccountTypePharmacy: "d12dc11d-8c74-46b7-9198-5a4385428f9a"
};
```

Вместо константы можно использовать значение "d12dc11d-8c74-46b7-9198-5a4385428f9a".

Бизнес-правило [Доступность] (Terrasoft.RuleTypes.Activation) —

С помощью этого бизнес-правила можно делать поля недоступными (или наоборот — доступными) для ввода значений по определенному условию.

Свойства конфигурационного объекта config

ruleType

Для этого правила должно содержать значение Terrasoft.RuleTypes.Activation.

triggeredByColumns

Массив колонок, инициирующих срабатывание бизнес-правила.

events

Массив событий, определяющий время запуска бизнес-правил. Должен содержать значения, входящие в перечисление Terrasoft.BusinessRuleEvents.

conditionalColumns

Массив условий срабатывания бизнес-правила. Обычно это определенные значения колонок.

dependentColumnNames

Массив названий колонок, к которым применяется данное бизнес-правило.

Доступность поля, связанного с колонкой Stock, зависит от значения в колонке Ispresence.

Бизнес-правило [Φ ильтрация] (Terrasoft.RuleTypes.Filtration)

Это бизнес-правило можно использовать для фильтрации значений справочных колонок по условию, либо по значению другой колонки.

Свойства конфигурационного объекта config

Колонка, на основании которой выполняется фильтрация значений.

```
ruleType
Для этого правила должно содержать значение Terrasoft.RuleTypes.Filtration.

triggeredByColumns
Массив колонок, инициирующих срабатывание бизнес-правила.

events
Массив событий, определяющий время запуска бизнес-правил. Должен содержать значения, входящие в перечисление Terrasoft.BusinessRuleEvents.

filters
Фильтр. Свойство должно содержать экземпляр класса Terrasoft.Filter.
```

Бизнес-правило [Взаимная фильтрация] (Terrasoft.RuleTypes.MutualFiltration) —

```
• [ Страна ] — [ Область ];
```

- [Область] [Город];
- [Страна] [Город].

Свойства конфигурационного объекта config

ruleType

Для этого правила должно содержать значение Terrasoft.RuleTypes.MutualFiltration.

triggeredByColumns

Массив колонок, инициирующих срабатывание бизнес-правила.

connections

Массив объектов, конфигурирующих отношение связок.

```
Взаимная фильтрация полей [Страна], [Область] и [Город]
Terrasoft.sdk.Model.addBusinessRule("ContactAddress", {
    ruleType: Terrasoft.RuleTypes.MutualFiltration,
    triggeredByColumns: ["City", "Region", "Country"],
    connections: [
        {
            parent: "Country",
            child: "City"
        },
            parent: "Country",
            child: "Region"
        },
            parent: "Region",
            child: "City"
        }
    ]
```

```
});
```

Бизнес-правило [*Регулярное выражение*] (Terrasoft.RuleTypes.RegExp) —

Выполняет проверку соответствия значения колонки регулярному выражению.

Свойства конфигурационного объекта config

ruleType

Для этого правила должно содержать значение Terrasoft.RuleTypes.RegExp.

RegExp

Регулярное выражение, на соответствие которому проверяются все колонки из массива triggeredByColumns.

triggeredByColumns

Массив колонок, инициирующих срабатывание бизнес-правила.

Пример использования

```
Terrasoft.sdk.Model.addBusinessRule("Contact", {
   ruleType: Terrasoft.RuleTypes.RegExp,
   regExp : /^([0-9\(\)\/\+ \-]*)$/
```

```
triggeredByColumns: ["HomeNumber", "BusinessNumber"]
});
```

Пользовательские бизнес-правила

При добавлении пользовательского бизнес-правила с помощью метода

Terrasoft.sdk.Model.addBusinessRule(name, config) можно использовать свойства конфигурационного объекта config базового бизнес-правила. Также дополнительно предусмотрено свойство executeFn.

Свойства конфигурационного объекта config

ruleType

Тип правила. Для пользовательских правил должно содержать значение Terrasoft.RuleTypes.Custom.

triggeredByColumns

Массив колонок, инициирующих срабатывание бизнес-правила.

events

Maccub событий, определяющий время запуска бизнес-правил. Должен содержать значения из перечисления Terrasoft.BusinessRuleEvents. Значение по умолчанию:

Terrasoft.BusinessRuleEvents.ValueChanged.

Возможные значения (Terrasoft.BusinessRuleEvents)

Save	Правило сработает перед сохранением данных.
ValueChanged	Правило сработает после изменения данных (при редактировании).
Load	Правило сработает при открытии страницы записи.

executeFn

Функция-обработчик, содержащая пользовательскую логику выполнения бизнес-правила.

Свойства функции-обработчика executeFn

Функция-обработчик объявляется в свойстве executeFn.

Сигнатура функции-обработчика

```
executeFn: function(record, rule, checkColumnName, customData, callbackConfig, event) { }
```

Параметры

record	Запись, для которой выполняется бизнес-правило.
rule	Экземпляр текущего бизнес-правила.
checkColumnName	Название колонки, которая вызвала срабатывание бизнес-правила.
customData	Объект, разделяемый между всем правилами. Не используется. Оставлен для совместимости с предыдущими версиями.
callbackConfig	Конфигурационный объект асинхронного возврата Ext.callback.
event	Событие, по которому было запущено бизнес-правило.

При завершении работы функции необходимо обязательно вызвать или callbackConfig.success , или callbackConfig.failure .

```
      Варианты вызова

      Ext.callback(callbackConfig.success, callbackConfig.scope, [result]);

      Ext.callback(callbackConfig.failure, callbackConfig.scope, [exception]);
```

Где:

- result возвращаемое логическое значение, полученное при выполнении функции (true / false).
- exception исключение типа Terrasoft.Exception, возникшее в функции-обработчике.

Методы

В исходном коде функции-обработчика удобно использовать следующие методы модели, передаваемой в параметре record:

```
get(columnName)
```

Для получения значения колонки записи. Аргумент columnName должен содержать название колонки.

```
set(columnName, value, fireEventConfig)
```

Для установки значения колонки записи.

Параметры

columnName	Название колонки.
value	Значение, присваиваемое колонке.
fireEventConfig	Конфигурационный объект для установки свойств, передаваемых в событие изменения колонки.

changeProperty(columnName, propertyConfig)

Для изменения свойств колонки, кроме ее значения. Аргумент соlumnName должен содержать название колонки, а propertyConfig — объект, устанавливающий свойства колонки.

Свойства объекта propertyConfig

disabled	Активность колонки. Если true, то элемент управления, связанный с колонкой, будет неактивным и закрытым для работы.
readOnly	Признак "только для чтения". Если true, то элемент управления, связанный с колонкой, будет доступен только для чтения. Если false —
	для чтения и записи.
hidden	Видимость колонки. Если true, то элемент управления, связанный с колонкой, будет скрыт. Если false — отображен.
addFilter	Добавить фильтр. Если свойство указано, то в нем должен быть указан фильтр типа Terrasoft.Filter, который будет добавлен к фильтрации колонки. Свойство используется только для справочных полей.
removeFilter	Удалить фильтр. Если свойство указано, то в нем должно быть указано имя фильтра, который будет удален из фильтрации колонки. Свойство используется только для справочных полей.
isValid	Признак корректности колонки. Если свойство указано, то оно изменит признак корректности элемента управления, связанного с колонкой. Если колонка некорректна, то это может означать отмену событий по сохранению записи, а также может привести к признанию записи некорректной.

Пример изменения свойств (но не значения) колонки Owner

```
record.changeProperty("Owner", {
    disabled: false,
```

```
readOnly: false,
hidden: false,
addFilter: {
    property: "IsChief",
    value: true
},
isValid: {
    value: false,
    message: LocalizableStrings["Owner_should_be_a_chief_only"]
}
});
```

Получение данных и настроек по дашбордам



Функциональность получения настроек и данных по дашбордам реализована в сервисе AnalyticsService и в утилитном классе AnalyticsServiceUtils , пакет Platform .

Примеры запросов к сервису AnalyticsService



```
Заголовки запроса
Accept:application/json
```

Методы

GetDashboardViewConfig()

```
Запрос

POST /0/rest/AnalyticsService/GetDashboardViewConfig

{
    "id": "a71d5c04-dff7-4892-90e5-9e7cc2246915"
}
```

GetDashboardData()

```
3aπpoc

POST /0/rest/AnalyticsService/GetDashboardData

{
    "id": "a71d5c04-dff7-4892-90e5-9e7cc2246915",
    "timeZoneOffset": 120
}
```

```
]
```

GetDashboardItemData()

```
3aπpoc

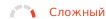
POST /0/rest/AnalyticsService/GetDashboardItemData

{
    "dashboardId": "a71d5c04-dff7-4892-90e5-9e7cc2246915",
    "itemName": "Chart4",
    "timeZoneOffset": 120
}
```

```
Ответ на запрос
  "name": "Chart4",
  "caption": "Invoice payment dynamics",
  "widgetType": "Chart",
  "chartConfig": {
    "xAxisDefaultCaption": null,
    "yAxisDefaultCaption": null,
    "seriesConfig": [
     {
        "type": "column",
        "style": "widget-green",
        "xAxis": {
          "caption": null,
          "dateTimeFormat": "Month;Year"
        },
        "yAxis": {
          "caption": "Actually paid",
          "dataValueType": 6
        },
        "schemaName": "Invoice",
        "schemaCaption": "Invoice",
        "useEmptyValue": null
     }
    ],
    "orderDirection": "asc"
 },
```

```
"style": "widget-green",
  "data": []
}
```

Класс AnalyticsService



Класс реализует функциональность получения настроек и данных по дашбордам.

Методы

Stream GetDashboardViewConfig(Guid id)

Возвращает настройки представления и виджетов на вкладке итогов по идентификатору страницы итогов.

Stream GetDashboardData(Guid id, int timeZoneOffset)

Возвращает данные по всем виджетам на вкладке итогов по идентификатору страницы итогов.

Stream GetDashboardItemData(Guid dashboardId, string itemName, int timeZoneOffset)

Возвращает данные по определенному виджету по идентификатору страницы итогов и имени виджета.

Здесь timeZoneOffset — смещение (в минутах) часового пояса относительно UTC. Данные по итогам будут получены с использованием этого часового пояса.

Мобильный портал



Основы

Функциональность мобильного портала доступна в Creatio версии 7.18.4 и выше.

Мобильный портал (мобильное приложение для портального пользователя) — мобильное рабочее место. **Назначение** мобильного портала — предоставление пользователям мобильного портала возможности создания обращений и ведения переписки со службой технической поддержки клиента.

Для мобильного портала можно настроить:

- Рабочее место пользователя мобильного портала.
- Реестр обращений.

- Страницу обращения.
- Страницу добавления обращения.

Настроить рабочее место пользователя мобильного портала

Действия по настройке рабочего места пользователя мобильного портала:

- Добавить рабочее место.
- Скрыть рабочее место.
- Удалить рабочее место.

Добавить рабочее место пользователя мобильного портала

Чтобы проверить наличие рабочего места [Portal]:

- 1. Перейдите в дизайнер системы по кнопке 🥨.
- 2. В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Мастер мобильного приложения] ([Mobile application wizard]).

Рабочее место [*Portal*] содержится в реестре раздела [*Macтер мобильного приложения*] ([*Mobile application wizard*]). По умолчанию доступно к использованию всем пользователям мобильного портала.

Если рабочее место [*Portal*] отсутствует в реестре раздела [*Macтep мобильного приложения*] ([*Mobile application wizard*]), то необходимо его добавить.

Чтобы добавить рабочее место пользователя мобильного портала:

- 1. Убедитесь, что Вы используете приложение Creatio версии 7.18.4 и выше.
- 2. Перейдите в дизайнер системы по кнопке 🧖
- 3. В блоке [*Настройка системы*] ([*System setup*]) перейдите по ссылке [*Мастер мобильного приложения*] ([*Mobile application wizard*]).
- 4. На панели инструментов раздела [*Macтep мобильного приложения*] ([*Mobile application wizard*]) нажмите на кнопку [*Добавить рабочее место*] ([*New workplace*]).
- 5. Заполните свойства рабочего места.
 - [*Название*] ([*Name*]) название рабочего места.
 - [Код]([Code]) "Portal".
- 6. На детали [*Роли*] ([*Roles*]) настройте права доступа к рабочему месту для пользователей или групп пользователей.
- 7. На панели инструментов нажмите [*Настроить разделы*] ([*Set up sections*]). По умолчанию в рабочее место пользователя мобильного портала добавлен раздел [*Обращения*] ([*Cases*]).
- 8. Сохраните настройки раздела [Мастер мобильного приложения] ([Mobile application wizard]).

В результате в приложение добавлено рабочее место пользователя мобильного портала.

Добавление рабочего места мобильного приложения подробно описано в статье Настроить рабочие места

Скрыть рабочее место пользователя мобильного портала

- 1. Перейдите в дизайнер системы по кнопке 🤽.
- 2. В блоке [*Настройка системы*] ([*System setup*]) перейдите по ссылке [*Мастер мобильного приложения*] ([*Mobile application wizard*]).
- 3. В реестре раздела откройте рабочее место [Portal].
- 4. Выполните удаление пользователей или групп пользователей рабочего места [*Portal*]. Для этого на детали [*Ponu*] ([*Roles*]) нажмите и выберите пункт [*Удалить*] ([*Delete*]).

В результате в приложении скрыто рабочее место пользователя мобильного портала.

Удалить рабочее место пользователя мобильного портала

- 1. Перейдите в дизайнер системы по кнопке 🤽.
- 2. В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Мастер мобильного приложения] ([Mobile application wizard]).
- 3. В реестре раздела выберите рабочее место [Portal] и нажмите на кнопку [Удалить] ([Delete]).

В результате в приложении удалено рабочее место пользователя мобильного портала.

Настроить реестр обращений

Действия по настройке реестра обращений для мобильного портала:

- Добавить колонку в реестр обращений.
- Скрыть заголовок колонки в реестре обращений.
- Настроить порядок сортировки обращений в реестре.

Добавить колонку в реестр обращений

- 1. Перейдите в дизайнер системы по кнопке 🤽
- 2. В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Мастер мобильного приложения] ([Mobile application wizard]).
- 3. В реестре раздела откройте рабочее место [Portal].
- 4. На панели инструментов нажмите [Настроить разделы] ([Set up sections]).
- 5. В реестре раздела выберите раздел [*Обращения*] ([*Cases*]) и нажмите на кнопку [*Настроить реестр*] ([*List setup*]).
- 6. В блоке [Подзаголовок] ([Subtitle]) или [Дополнительные колонки] ([Additional columns]) нажмите на кнопку [Добавить колонку] ([New column]) и выберите необходимую колонку.
- 7. Сохраните настройки реестра раздела [Обращения] ([Cases]).
- 8. Сохраните настройки раздела [Мастер мобильного приложения] ([Mobile application wizard]).

Добавление колонки в реестр раздела подробно описано в статье <u>Настроить реестр мобильного приложения</u>.

Скрыть заголовок колонки в реестре обращений

- 1. <u>Перейдите в раздел [Конфигурация]</u> ([Configuration]).
- 2. В пользовательском <u>пакете</u> откройте схему <u>MobileCaseGridPageSettingsPortal</u>. Если вы еще не настраивали реестр обращений через мастер мобильного приложения, то схема <u>MobileCaseGridPageSettingsPortal</u> отсутствует в пользовательском пакете.

Чтобы добавить схему MobileCaseGridPageSettingsPortal в пользовательский пакет:

- а. Перейдите в дизайнер системы по кнопке 🤽
- b. В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Мастер мобильного приложения] ([Mobile application wizard]).
- с. В реестре раздела откройте рабочее место [Portal].
- d. На панели инструментов нажмите [Настроить разделы] ([Set up sections]).
- e. В реестре раздела выберите раздел [*Обращения*] ([*Cases*]) и нажмите на кнопку [*Настроить страницу*] ([*Page setup*]).
- f. Сохраните настройки страницы раздела [Обращения] ([Cases]).
- g. Сохраните настройки раздела [Мастер мобильного приложения] ([Mobile application wizard]).
- 3. Скройте **заголовок колонки в реестре обращений**. Для этого в начало массива модификаций diff добавьте конфигурационный объект колонки, заголовок которой планируется скрыть.
 - а. В свойстве value укажите колонку. Шаблон имени колонки: \$имяколонки.
 - b. В свойстве visible (отвечает за отображение заголовка колонки) установите значение false.

Пример скрытия заголовка колонки [Status] представлен ниже.

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

Настроить порядок сортировки обращений в реестре

- 1. <u>Перейдите в раздел [Конфигурация]</u> ([Configuration]).
- 2. В пользовательском <u>пакете</u> откройте схему <u>MobileCaseGridPageSettingsPortal</u>. Если вы еще не настраивали реестр обращений через мастер мобильного приложения, то схема <u>MobileCaseGridPageSettingsPortal</u> отсутствует в пользовательском пакете.

Чтобы добавить схему MobileCaseGridPageSettingsPortal в пользовательский пакет:

- а. Перейдите в дизайнер системы по кнопке 🤽
- b. В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Мастер мобильного приложения] ([Mobile application wizard]).
- с. В реестре раздела откройте рабочее место [Portal].
- d. На панели инструментов нажмите [Настроить разделы] ([Set up sections]).
- e. В реестре раздела выберите раздел [*Обращения*] ([*Cases*]) и нажмите на кнопку [*Настроить страницу*] ([*Page setup*]).
- f. Сохраните настройки страницы раздела [Обращения] ([Cases]).
- g. Сохраните настройки раздела [Мастер мобильного приложения] ([Mobile application wizard]).
- 3. Настройте **порядок сортировки в реестре обращений**. Для этого в начало массива модификаций diff добавьте конфигурационный объект с настройками отображения реестра.
 - а. В свойстве columnPath укажите имя колонки, которую планируется использовать для сортировки.
 - b. В свойстве alias укажите алиас колонки, которую планируется использовать для сортировки.
 - с. В свойстве orderDirection укажите порядок сортировки (1 по возрастанию, 2 по убыванию).
 - d. В свойстве orderPosition укажите порядковый индекс колонки в коллекции колонок, по которой выполняется сортировка.

Пример настройки сортировки реестра обращений представлен ниже. Обращения в реестре сортируются по возрастанию значений колонки [RegisteredOn].

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

Настроить страницу обращения

Настройка страницы обращения позволяет добавить колонку на вкладку [Детали] ([Details]).

Чтобы добавить колонку на вкладку [Детали] ([Details]) страницы обращения:

- 1. Перейдите в дизайнер системы по кнопке 🤽.
- 2. В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Мастер мобильного приложения] ([Mobile application wizard]).
- 3. В реестре раздела откройте рабочее место [Portal].
- 4. На панели инструментов нажмите [Настроить разделы] ([Set up sections]).
- 5. В реестре раздела выберите раздел [*Обращения*] ([*Cases*]) и нажмите на кнопку [*Настроить страницу*] ([*Page setup*]).
- 6. В блоке [Общая информация] ([General information]) нажмите на кнопку [Добавить колонку] ([New column]) и выберите колонку [Номер] ([Number]).
- 7. Сохраните настройки страницы раздела [Обращения] ([Cases]).
- 8. Сохраните настройки раздела [Мастер мобильного приложения] ([Mobile application wizard]).

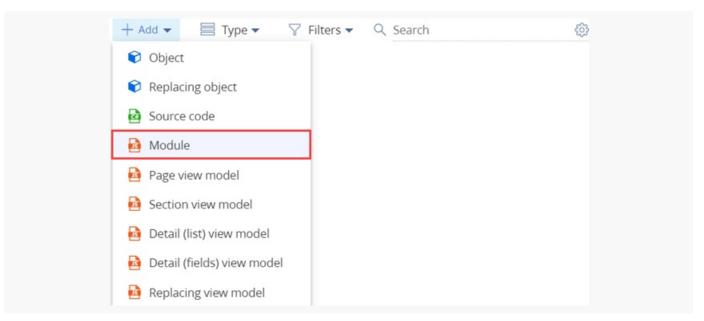
На заметку. Все колонки вкладки [*Детали*] ([*Details*]) страницы обращения доступны только для чтения.

Настроить страницу добавления обращения

Настройка страницы добавления обращения позволяет добавить колонку.

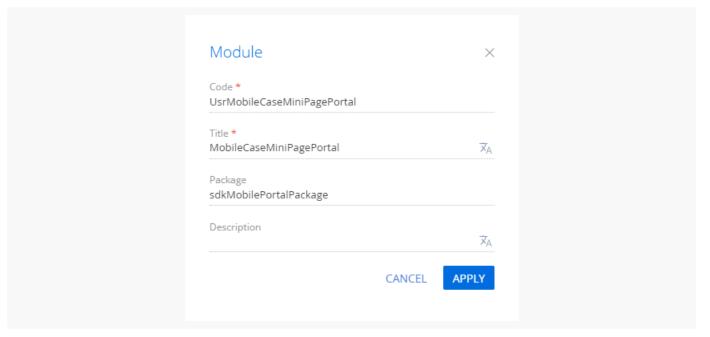
Чтобы добавить колонку на страницу добавления обращения:

- 1. <u>Перейдите в раздел [Конфигурация]</u> ([Configuration]).
- 2. Откройте схему MobileCaseMiniPagePortal Пакета CaseMobile и скопируйте ее содержимое.
- 3. Выберите пользовательский пакет, в который будет добавлена схема.
- 4. На панели инструментов реестра раздела нажмите [Добавить] -> [Модуль] ([Add] -> [Module]).



5. 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).



- 6. В пользовательский модуль добавьте скопированное содержимое схемы MobileCaseMiniPagePortal пакета CaseMobile.
- 7. В пользовательский модуль перенесите <u>локализованные строки</u> схемы MobileCaseMiniPagePortal пакета CaseMobile.
- 8. Добавьте колонку.
 - а. В свойство viewConfig добавьте элемент, который планируется использовать для редактирования

необходимой колонки. В свойстве value укажите колонку. Шаблон имени колонки: \$имяколонки . Пример добавления колонки [\$confitem] представлен ниже.

```
Пример настройки свойства viewConfig
"viewConfig": {
    "editCard": {
        "body": {
            . . .
            "items": [
                . . .
                {
                     "type": "EditField",
                     "properties": {
                         "value": "$ConfItem"
                     }
                }
            ]
        }
   }
}
```

\$ConfItem — ИМЯ КОЛОНКИ.

b. В свойство controllers добавьте описание необходимой колонки. В свойстве columnPath укажите имя колонки схемы Case объекта.

Пример добавления описания колонки [confitem] представлен ниже.

}

- 9. На панели инструментов дизайнера нажмите [Сохранить] ([Save]).
- 10.В манифесте портального рабочего места зарегистрируйте ранее созданную пользовательскую схему UsrMobileCaseMiniPagePortal.
 - a. В пользовательском <u>пакете</u> откройте схему <u>MobileApplicationManifestPortal</u>. Если вы еще не выполняли настройку приложения через мастер мобильного приложения, то схема <u>MobileApplicationManifestPortal</u> отсутствует в пользовательском пакете.

Чтобы добавить схему MobileApplicationManifestPortal в пользовательский пакет:

- а. Перейдите в дизайнер системы по кнопке 🤽.
- b. В блоке [Настройка системы] ([System setup]) перейдите по ссылке [Мастер мобильного приложения] ([Mobile application wizard]).
- с. В реестре раздела откройте рабочее место [Portal].
- d. На панели инструментов нажмите [Настроить разделы] ([Set up sections]).
- e. В реестре раздела выберите раздел [Обращения] ([Cases]) и нажмите на кнопку [Настроить страницу] ([Page setup]).
- f. Сохраните настройки страницы раздела [Обращения] ([Cases]).
- g. Сохраните настройки раздела [Мастер мобильного приложения] ([Mobile application wizard]).
- b. Зарегистрируйте **схему**.
 - a. В свойстве мodules укажите схему, которая используется для добавления записи схемы саse объекта.
 - b. В свойстве <code>models</code> укажите схему, которая используется для расширения схемы саse объекта.

Пример регистрации схемы UsrMobileCaseMiniPagePortal представлен ниже.

#