

ПРОГРАММА ДЛЯ ЭВМ

«Программная платформа UrbanCORE»

Руководство по установке и эксплуатации

Листов: 97

СОДЕРЖАНИЕ

1	ВВЕДЕНИЕ	4
1.1	Общие сведения и область применения	4
1.2	Термины, сокращения и определения	4
1.3	Требования к квалификации.....	4
2	ПОДГОТОВИТЕЛЬНЫЕ ОПЕРАЦИИ	5
3	СОСТАВ ДИСТРИБУТИВА	6
4	УСТАНОВКА И ЗАПУСК	7
5	СВЕДЕНИЯ ПО ЭКСПЛУАТАЦИИ	8
5.1	Системные настройки и параметры	8
5.2	Описание модулей	12
5.2.1	Общие принципы UI	12
5.2.2	Структура данных объектной модели.....	12
5.2.3	Принципы построения интеграций.....	26
5.3	Организация пользовательского интерфейса.....	29
5.4	Настройка экранных форм.....	30
5.4.1	Последовательность настройки форм	30
5.4.1.1	Общее описание.....	32
5.4.1.2	Настройка данных формы (form_data)	33
5.4.2	Настройка содержимого формы (form_content)	33
5.4.2.1	Настройка формы превью.....	38
5.4.3	Привязка формы к гриду	39
5.4.3.1	Форма по дабл-клику	39
5.4.3.2	Форма для превью	39
5.4.4	Настройки даты в форме.....	39
5.4.5	Настройка таблицы на форме.....	40
5.5	Примеры компонентов экранных форм.....	41
5.6	Примеры параметров вызова экранных форм	48
5.7	Автоматическое создание класса, представления и атрибутов.....	50
5.8	Настройка меню	51
5.8.1	Основные атрибуты s_arm.....	51
5.8.2	Основные атрибуты s_arm_relation.....	52
5.8.3	Настройка фреймов	52
5.9	Настройка таблиц.....	53
5.9.1	Последовательность настройки grid (табличного представления данных).....	53
5.10	Настройка кнопок таблиц	67
5.11	Настройка стилей оформления таблиц	72
5.12	Настройка «сложной» сортировки.....	82
5.13	Настройка отчета с параметрами	85
5.13.1	Настройка меню.....	85
5.13.2	Настройка s_core.s_form	85
5.13.3	Правила блоков в процедуре.....	86
5.14	Настройка ролей	88
5.14.1	Назначение ролей	88
5.15	Использование скриптов.....	90
5.15.1	Ролевая модель - обновить разрешения вручную для s_class_permission	90

5.15.2	CoreDev - автоматическое формирование INSERT запросов для создания скелета раздела	92
5.15.3	Автоматическое формирование скелета JSON - form_content.....	93
5.15.4	Автоматическое формирование скелета JSON - form_data.....	96

1 ВВЕДЕНИЕ

1.1 Общие сведения и область применения

Программное обеспечение «Программная платформа UrbanCORE» (далее Система) – это российское программное обеспечение, организация-разработчик: ООО «Урбантех-ИТ».

Сайт организации-разработчика: <http://urbantechgroup.ru>

Организация-правообладатель: ООО «Урбантех-ИТ».

Система предназначена для ускорения создания различных информационных систем.

Данный документ описывает порядок установки экземпляра Системы из репозитория контейнеров сервисов платформы.

1.2 Термины, сокращения и определения

Программное обеспечение (ПО) - Совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

Система - здесь: Программное обеспечение «Программная платформа UrbanCORE».

БД - База данных.

Docker - Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.

1.3 Требования к квалификации

Установка Системы должна производиться квалифицированными специалистами.

2 ПОДГОТОВИТЕЛЬНЫЕ ОПЕРАЦИИ

Система поддерживает горизонтальное масштабирование, настраиваемое вне рамок самой Системы. Для возможности горизонтального масштабирования Системы рекомендуется производить установку в кластер под управлением Kubernetes.

Все сервисы Системы могут быть установлены в рамках одного сервера, данный вариант будет описан в документе далее.

Установить Docker и Docker Compose.

Compose в актуальных версиях Docker идет в комплекте в виде плагина ("docker compose", вместо "docker-compose" при отдельном варианте установки).

3 СОСТАВ ДИСТРИБУТИВА

- conf/: директория с используемыми конфигами
- db/: директория с демонстрационными данными БД Postgres
- docker-compose.yaml: файл docker-compose для запуска всего стека ПО
- keycloak-custom-*.tar: образ контейнера с ПО
- core-backend-*.tar: образ контейнера с ПО
- core-frontend-*.tar: образ контейнера с ПО
- postgres-custom-*.tar: образ контейнера с ПО
- README_CLIENT.md: инструкция для разворачивания ПО на демо стенде

Ссылка для скачивания дистрибутива <https://s3.mvs.group/static-files/urbancore-standalone.tar.gz>

4 УСТАНОВКА И ЗАПУСК

1. Разархивировать архив и импортировать переменные

```
tar -xzvf urbancore-standalone.tar.gz  
cd urbancore-standalone
```

2. Загрузить образы в docker

```
ls *.tar | xargs -n 1 -I% docker load -i %
```

```
docker-compose -f docker-compose.yaml pull --ignore-pull-failures
```

3. Добавить в hosts-файл строки:

- Linux: /etc/hosts
- Windows: C:\Windows\System32\drivers\etc\hosts

```
127.0.0.1 urbancore-standalone.local
```

```
127.0.0.1 auth.urbancore-standalone.local
```

```
127.0.0.1 backend.urbancore-standalone.local
```

4. Запустить стек

```
docker-compose -f docker-compose.yaml up -d
```

5. Завершение настройки

Проверить логи.

```
docker-compose -f docker-compose.yaml logs -f
```

6. Подождать ~2 минуты, пока все приложения запустятся

Открыть в браузере <http://urbancore-standalone.local>

- Имя пользователя: urbancore
- Пароль: urbancore

5 СВЕДЕНИЯ ПО ЭКСПЛУАТАЦИИ

5.1 Системные настройки и параметры

Ниже представлено описание системных классов, методов, а также глобальных и локальных переменных платформы.

Таблица 1. Системные классы

Наименование	Назначение	Дополнительная информация
MainMenuTree	Главное меню для веба	Отображает структуру АРМ для текущего пользователя в вебе
MobileMainMenuTree	Главное меню для мобильных приложений	
MainMenuToolBar	Тулбар для главного меню	
MobileMenuToolBar	Тулбар для главного меню мобильного приложения	
SysClass	Системный класс	Таблица s_class
SysClassList	Список системных классов	
SysARM	Элемент АРМ	
SysView	Системное представление	
SysViewList	Список системных представлений	
SysViewToolBar	Тулбар для системного представления	
SysAttribute	Атрибуты класса	Описание класса - на фронте запрос при первом обращении к классу, затем кешируется на фронте.
SysViewAttribute	Атрибуты системного представления	Описание представления - на фронте запрос при первом обращении к классу, затем кешируется на фронте.
SysDataTypeOperationList	Список системных операций для	

	указанного типа данных	
SysForm	Системная форма	
SysFormToolBar	Тулбар для системной формы	
MainLayersMap	Слои карты	
SysParamsDiff	Перечень системных параметров по времени изменения	
SysIntegration	Интеграция	
SysDataType	Типы данных	
SysClassSearchTree	Дерево поиска для класса	
SysClassJoinTree	Дерево соединений классов	
AccountPrefList	Настройки текущего пользователя	Параметры: ID_CLASS - ИД текущего класса, ID_CALL_POINT - ИД точки вызова.
AccountPrefInfo	Указанная настройка	Параметры: ID_OBJ - ИД искомой настройки.
SysAccountID	ИД текущего пользователя	Определяется по токену
SysAccountRoleList	Список ролей УЗ	Список уникальных ролей (item_type = 1) в списке полномочий пользователя. Используется при синхронизации ролей с KeyCloack

Таблица 2. Системные методы

№	Наименование	Назначение	Дополнительная информация
1	buildTree	Построение дерева по данным, возвращаемым запросом	Параметры: className - системное наименование класса, parentIdFieldName - наименование поля, содержащего ИД корневого элемента, idFieldName - наименование поля, содержащего ИД текущего узла (значение ИД может быть пустым в случае конечного элемента), queryParams : [{ name : "наименование параметра", value : "значение

			параметра" }}] - дополнительные параметры для класса
2	docSave	Запись файла с дополнительной метаинформацией	<p>Параметры:</p> <p>[DocSaveQueryInput]! - перечень дополнительных полей с метаинформацией (id: "ИД записи в таблице", table: "Наименование таблицы", field: "Наименование поля", value: "Значение поля", dataType: "Тип данных"), file - содержимое файла</p>

Таблица 3. Системные глобальные переменные

№	Наименование	Тип	Назначение	Обработка переменной
1	{USER_LOGIN\$}	varchar	Логин пользователя	Да
2	{ID_OBJ\$}		ИД основной таблицы	
4	{ID_CLASS\$}	int4	ИД класса	
5	{ID_OBJECT\$}		ИД объекта	
6	{ID_ACCOUNT\$}	varchar	ИД пользователя	Да
7	{TIMESTAMP_START\$}	timestamp	Дата и время начала процесса	
8	{TIMESTAMP_END\$}	timestamp	Дата и время окончания процесса	
9	{TIME_START\$}	time	Время начала процесса	
10	{TIME_END\$}	time	Время окончания процесса	
11	{DATE_START\$}	date	Дата начала процесса	
12	{DATE_END\$}	date	Дата окончания процесса	

Таблица 4. Системные локальные переменные

№	Наименование	Тип данных	Назначение	Дополнительная информация
1	__currentDate__	date	Текущая дата клиента	обработка на уровне передачи параметра на сервер. Используется в setValues

				Чтобы отобразить строку __currentDate__ как есть в форме, необходимо использовать \$\$__currentDate__\$\$. То же касается и других системных локальных переменных
2	__currentTime__	time	Текущее время клиента	обработка на уровне передачи параметра на сервер. Используется в setValues
3	__currentDateTime__	timestamp	Текущая дата и время клиента	обработка на уровне передачи параметра на сервер. Используется в setValues
4	__currentIdArm__	int4	ИД текущего пункта меню	обработка на уровне передачи параметра на сервер. Используется в setValues
5	7. __currentIdAccount__	int4	ИД текущего пользователя	обработка на уровне передачи параметра на сервер
6	8. __currentIdSubTree__	int4	ИД текущего поддерева	обработка на уровне передачи параметра на сервер. Используется в setValues
7	9. __currentLayerComponent__	vvarchar	Наименование типа элемента текущего слоя	обработка на уровне передачи параметра на сервер.
8	10. __currentLayerElement__	int4	ИД текущего элемента слоя	обработка на уровне передачи параметра на сервер.
9	11. __workspace__	TObject	Рабочая область приложения	

5.2 Описание модулей

5.2.1 Общие принципы UI

1. При входе в систему из БД/через бек запрашиваются настройки, отражающие последнее состояние UI, с которым работал пользователь (в каком состоянии завершалась сессия - в таком состоянии начинается работа: открытый пункт меню, примененные фильтры, установленные размеры объектов, установленные настройки объектов);
2. Настройки меню, таблиц, форм запрашиваются однократно для каждого объекта по мере необходимости за сеанс работы и сохраняются в кеше клиента. (все настройки сразу не вычитываются);
3. При каждом изменении параметров UI объекта производится запись настройки, повторное вычитывание при этом из БД не осуществляется;
4. При самом первом отображении объекта UI применяются общесистемные настройки для данного класса объектов;
5. Если пользователь не изменяет параметры отображения, то запись настроек в БД не производится.

5.2.2 Структура данных объектной модели

- `s_core.s_component_class` - перечень компонентов;
- `s_core.s_component_struct` - свойства компонентов;
- `s_core.s_component_group` - группировка компонентов в панели инструментов конструктора форм;
- `SysComponentPropList` - класс для получения структуры компонента;
 - Параметры;

```
/* AND scs.id_component = {Sid_component$}::int4*/  
/* AND scs.id_component = (SELECT id FROM s_core.s_component_class WHERE cmp_class_name =  
                           {Scomponent_name$}::text) */
```

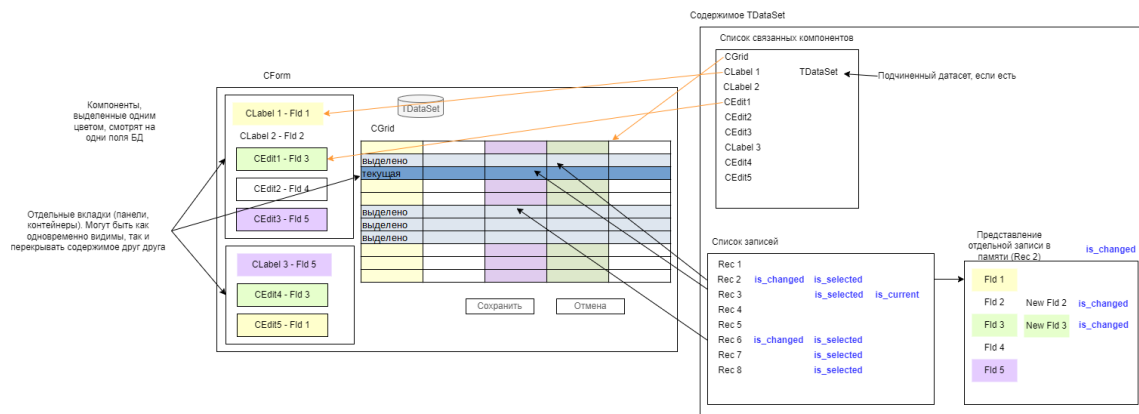
Наименования свойств в объектной модели не привязаны к реализации. Данная объектная модель будет применяться для всех типов приложений (web, десктоп, мобильные).

Если при описании компонента не указаны значения каких-либо свойств или **null**, то:

1. если установлен признак `is_parent_inherit`, то значение брать из родительского компонента - **в текущий момент**
2. если родительский компонент отсутствует или `is_parent_inherit = false`, то значение получать из темы, выбранной пользователем в качестве текущей - **в перспективе**.
3. Если пользователь не выбрал для себя тему, то брать общесистемную тему, указанную в настройках (`s_core.s_params:main_theme`) - **в перспективе**.

4. Если параметр не указан или отсутствует, то необходимо получить описание темы из `s_core.s_theme:theme_name='default'` - в перспективе.
5. Если тема **default** отсутствует, то устанавливать только те значения, которые пришли в описании компонента как есть (в настоящее время формируется описание стилей и тем) - в текущий момент.

Для всех компонентов, поддерживающих операцию выделения и копирования в буфер, необходимо реализовать в новых формах (CLabel, CEdit, CComboBox, CDatePicker...).



Общая схема взаимодействия элементов формы:

SysArmCategory	CForm	Вызов формы из таблицы основной рабочей области: <pre>{ "params": [{ "name": "\$SysArmCategory::CForm.dsSysArmCategory::TDataSet.params.first_param.value\$", "value": "\$self::CForm.SysArmCategory::CGrid.data.selected[1].id.value\$" }] }</pre>
dsSysArmCategory	TDataSet	dbClass = SysArmCategory dbTable = s_core.s_arm_category params = [{ name = "first_param", value = "", type = "int4" }]
SysArmCategory	SQL	SELECT ac.id , ac.category_name , ac.category_description , sac.account_caption AS creator , sae.account_caption AS editor FROM s_core.s_arm_category ac LEFT JOIN s_core.s_account sac ON ac.id_creator = sac.id LEFT JOIN s_core.s_account sae ON ac.id_editor = sae.id WHERE 1=1 /* AND ac.id = {first_param}::int4 */
edId	CEdit	dataSet = dsSysArmCategory field = id
edCategoryName	CEdit	dataSet = dsSysArmCategory field = categoryName
edCategoryDescription	CEdit	dataSet = dsSysArmCategory field = categoryDescription
btSave	CButton	action = execMethod component = dsSysArmCategory::TDataSet methodName = saveCurrentRow
btCancel	CButton	action = execMethod component = dsSysArmCategory::TDataSet methodName = cancelEditRow action = execMethod component = SysArmCategory::CForm methodName = close

Открытие формы для редактирования:

Исходное состояние формы

SysArmCategory	CForm	
dsSysArmCategory	TDataSet	
edId	CEdit	123
edCategoryName	CEdit	INV
edCategoryDescription	CEdit	Роли, относящиеся к проекту Инвентори
btSave	CButton	
btCancel	CButton	

Состав dsSysArmCategory

Служебная информация				Данные из класса БД		
idRow	isChanged	isCurrent	isSelected	id	categoryName	categoryDescription
1	false	true	true	123	INV	Роли, относящиеся к проекту Инвентори
				1	2	3

Пользователь вносит изменения в поля:

Редактирование данных

SysArmCategory	CForm	
dsSysArmCategory	TDataSet	
edId	CEdit	1234
edCategoryName	CEdit	INV
edCategoryDescription	CEdit	Роли, относящиеся к проекту Инвентори
btSave	CButton	
btCancel	CButton	

Состав dsSysArmCategory после редактирования

Служебная информация				Данные из класса БД		
idRow	isChanged	isCurrent	isSelected	id	categoryName	categoryDescription
1	true	true	true	1234	INV	Роли, относящиеся к проекту Инвентори
				1	2	3

При редактировании создается список измененных полей, по которым в итоге формируется список параметров для отправки в FormSave. oldValue - с учетом развития, возможность отмены изменений для каждого отдельного поля.

Список изменений isEdited

idRow	attr_num	oldValue
1	1	123

При нажатии на кнопку "Сохранить" выполнение метода компонента dsSysArmCategory::TDataSet с названием methodName = saveCurrentRow (=отправка FormSave):

FormSave

```
mutation formSave($query: [FormSaveQueryInput]!, $callFunction: [FormSaveCallFunctionInput]) {
  formSave(query: $query, callFunction: $callFunction) {
    id
    table
    key
    __typename
  }
}
```

```
{
  "query": [
    {
      "table": "s_core.s_arm_category", - dbTable
      "field": "id", - dbClass.attributes.attrDbName
      "dataType": "int4", - dbClass.attributes.dataTypeName
    }
  ]
}
```

"value": "1234", - новое значение поля

"id": 123 - пока отправляем значение первого поля, наименование которого указано в TDataSet.primaryKeys[]. Далее метод FormSave будет доработан

```
}
}
]
```

Результат сохранения:

После получения положительного ответа от бека.

После сохранения данных

SysArmCategory	CForm	
dsSysArmCategory	TDataSet	
edId	CEdit	1234
edCategoryName	CEdit	INV
edCategoryDescription	CEdit	Роли, относящиеся к проекту Инвентори
btSave	CButton	
btCancel	CButton	

Состав dsSysArmCategory после сохранения

Служебная информация				Данные из класса БД		
idRow	isChanged	isCurrent	isSelected	id	categoryName	categoryDescription
1	false	true	true	1234	INV	Роли, относящиеся к проекту Инвентори

Список изменений isEdited

idRow	attr_num	oldValue
-------	----------	----------

Изменение нескольких записей одновременно:

При навигации по записям таблицы автоматически меняются значения полей ввода информации

CLabel1: field = num	108
CEdit1: field = address	а/д М-8 «Холмогоры», 56км+614м, в Москву
CEdit2: field = lat	56.184357
CEdit3: field = lon	38.032165

CLabel1: field = num	108
CEdit4: field = lat	56.184357
CEdit5: field = lon	38.032165

Номер	Адрес	Широта	Долгота	Комплексы	Код зоны
110	а/д М-8 «Холмогоры», 65км+260м, из Москвы	56.23311	38.136528	AS5000208	433
1003	а/д «Москва-Егорьевск-Тума-Касимов», 56км+095м, н.п. Соболево	55.51974	38.70462	AS5000312	733
11	а/д М-10 «Россия», 63км+340м, н.п. Солнечногорск	56.180729	36.988799	AS5000456	353
1124	а/д "МБК-подъезд к санаторию "Русь", 1км+800м, н.п. Руза, из Волоколамска / в Волоколамск	55.711739	36.177192	AS5000571	370
108	а/д М-8 «Холмогоры», 56км+614м, в Москву	56.184357	38.032165	AS5000591	431
1116	а/д М-7 «Волга», 87км+285м, н.п. Малая Дубна, из Москвы	55.873779	38.952681	AS5000634	521
107	а/д М-8 «Холмогоры», 56км+585м, из Москвы	56.18406	38.03226	AS5000639	430
1130	ул. 1-я Советская, д.78, н.п. Дорохово, к ул. Комсомольской / от ул. Комсомольской	55.558268	36.378593	AS5000653	303
1142	а/д "Можайск - М-1"Беларусь", 1км+610м, н.п. Ямская.	55.482199	36.022903	AS5000685	625

<< < > >> Изменить Сохранить

Переключение текущей записи кнопками

idRow	isChanged	isCurrent	isSelected	num	address	lat	lon	name	road_km
1	true	false	false	110	а/д М-8 «Холмогоры», 65км+260м, из Москвы	56.23311	38.136528	AS5000208	433
2	false	false	false	1003	а/д «Москва-Егорьевск-Тума-Касимов», 56км+095м, н.п. Соболево,	55.51974	38.70462	AS5000312	733
3	true	false	false	11	а/д М-10 «Россия», 63км+340м, н.п. Солнечногорск,	56.180729	36.988799	AS5000456	353
4	true	false	false	1124	а/д "МБК-подъезд к санаторию "Русь", 1км+800м, н.п. Руза, из Волоколамска / в Волоколамск	55.711739	36.177192	AS5000571	370
5	false	true	true	108	а/д М-8 «Холмогоры», 56км+614м, в Москву	56.184357	38.032165	AS5000591	431
6	false	false	false	1116	а/д М-7 «Волга», 87км+285м, н.п. Малая Дубна, из Москвы	55.873779	38.952681	AS5000634	521
7	true	false	false	107	а/д М-8 «Холмогоры», 56км+585м, из Москвы	56.18406	38.03226	AS5000639	430
8	false	false	false	1130	ул. 1-я Советская, д.78, н.п. Дорохово, к ул. Комсомольской / от ул. Комсомольской	55.558268	36.378593	AS5000653	303
9	false	false	false	1142	а/д "Можайск - М-1"Беларусь", 1км+610м, н.п. Ямская,	55.482199	36.022903	AS5000685	625

idRow	attr_num	oldValue
1	2	а/д М-8 «Холмогоры», 65км+260м, из Москвы
3	3	56.180729
3	4	36.988799
4	2	а/д "МБК-подъезд к санаторию "Русь", 1км+800м, н.п. Руза, из Волоколамска / в Волоколамск
4	3	55.711739
4	4	36.177192
4	5	AS5000571
7	1	107
7	2	а/д М-8 «Холмогоры», 56км+585м, из Москвы

Сохранение нескольких TDataSet:

```

btSave          CButton      action = execMethod
                                   component = dsSysArmCategory::TDataSet
                                   methodName = saveCurrentRow
                                   action = execMethod
                                   component = dsSysArm::TDataSet
                                   methodName = saveAllChanged

```

Связанные TDataSet-ы:

```

DataSet1       TDataSet      dbClass = SysArmCategory
                                   dbView = SysArmCategory
DataSet2       TDataSet      dbClass = SysArm
                                   dbView = SysArm
                                   masterDS = $self::CForm.DataSet1::TDataSet$
                                   keyMapping = [
                                     masterField = "id",
                                     detailField = "id_arm_category",
                                     type = "int"
                                   ]
                                   params = [
                                     name = "id_arm_category",
                                     value = "$self::CForm.DataSet1::TDataSet.data.CurrentRow.id.value$",
                                     type = "int"
                                   ]

```

id	category_name	category_description	
1	AM	Роли для аналитического модуля	
2	INV	Роли для Инвентори	
id	id_category	arm_name	arm_description
292	1	ambddHeathOld	Очаги
204	1	ambddDtp	ДТП
252	2	drtMap	Карта
107	2	map_road_lock_list_check_rg	Проверка РГ
106	2	map_road_lock_list_consideration_rg	Рассмотрение РГ
277	2	drtCustomOkrugLayer9	Дороги.Поселение Рязановское
661	2	role_test1	role-test1
197	1	ambddRoot	АМБДД
203	2	mobile_contragents	Организации MPM dev
201	1	ambddHeath	Очаги
537	2	new	новая

Формирование данных для сохранения для подчиненного TDataSet:

```

mutation formSave($query: [FormSaveQueryInput]!, $callFunction: [FormSaveCallFunctionInput]) {
  formSave(query: $query, callFunction: $callFunction) {
    id
    table
    key
    __typename
  }
}

{
  "query": [
    { --данные, указанные в форме пользователем
      "table": "s_core.s_arm", - dbTable
      "field": "arm_name", - dbClass.attributes.attrDbName
      "dataType": "varchar", - dbClass.attributes.dataTypeName
      "value": "ambddHeathOld", - новое значение поля, указанное пользователем
      "id": null - при создании новой записи.
    },
  ],
}

```

```

{ --для каждого поля keyMapping, по которым связываются TDataSet
"table": "s_core.s_arm", - dbTable
"field": "id_arm_category", - dbClass.attributes.attrDbName (keyMapping.detailField)
"dataType": "int", - dbClass.attributes.dataTypeName
"value": "masterDS::TDataSet.data.CurrentRow.id.value$", - значение поля keyMapping.masterField
"id":
}
]
}
    
```

Взаимодействие с БД:

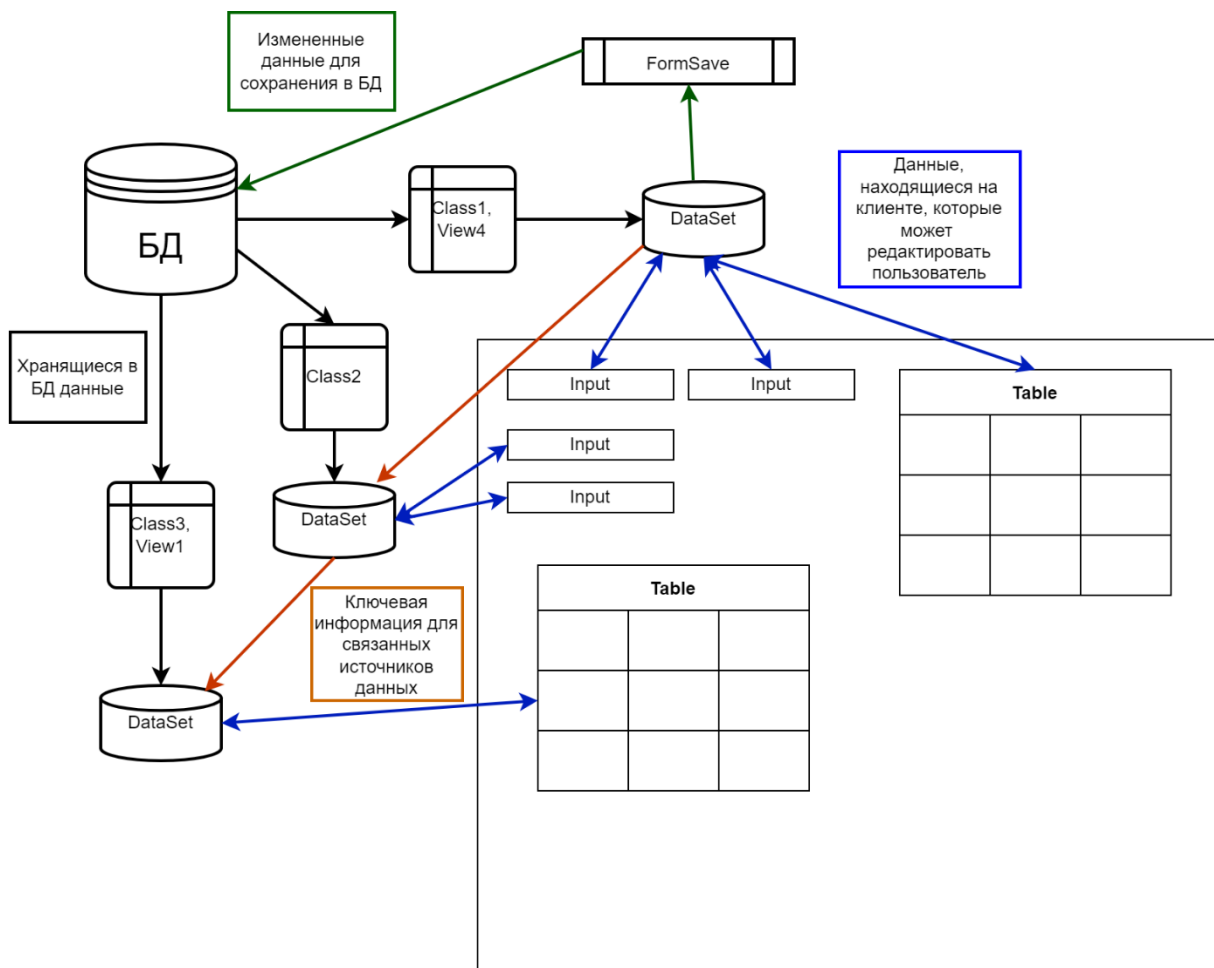


Таблица 5. Перечень компонентов и их свойств

Наименование класса	Наименование свойства	UI наименование	Тип	Краткое описание
CDBPicture Deck		Набор изображений ДБ		Отображение первого изображения из набора с возможностью просмотра и выбора остальных изображений

CDBTree		Дерево ДБ		Компонент для отображения иерархической структуры данных контроль ввода циклических ссылок - родитель!=потомок, потомок !=родитель верхнего уровня
	objectName	Наименование экземпляра компонента	varchar	
	dataSet	наименование источника данных		
	parentField	поле для указания владельца		
	childField	поле для указания потомка		
	keyField	поле для указания ID		saveForm = "id": "primaryKeyField"
	keyChildField	наименование поля для указания ID текущего узла		используется в паре с keyParentField, не равно keyField в ряде случаев (дерево меню)
	keyParentField	наименование поля для указания ID родительского узла		saveForm = " значение из атрибута keyParentField " : " значение keyChildField для того узла, в который перетаскивается элемент ", " значение из атрибута keyChildField " : " значение keyChildField для того узла, который перетаскивается "
	paramName	наименование параметра в запросе		{SID_OBJECT\$}
	paramNullValue	указание значения для null значений		COALESCE(id_parent_item, 0) - 0 в данном случае значение параметра вместо null. т.к. нельзя писать id_parent_item = null
	ordField	поле для указания порядка расположения		itemOrd - системное название атрибута, в запросе может отличаться от названия в БД
	saveOrdTable	таблица для сохранения порядка		saveForm = "table": "s_core.s_arm" - наименование в БД

	saveOrdField	поле в БД для сохранения порядка		saveForm = "field": "item_ord" - наименование в БД (только целое, "dataType": "int")
	interval			расстояние от иконок до текста
	parentIcon	по умолчанию изображение для родительских узлов		
	childIcon	по умолчанию изображение для подчиненных узлов		
	unknownIcon	по умолчанию для неопределенных узлов при асинхронной загрузке		
	+displayFieldList	перечень полей для вывода в строке		Элементы располагаются также как элементы в этом массиве "displayFieldList": [{ "name": "idItem", "isIcon": false, "colorDefinition": "red" }, { "name": "idParentItemIcon", "isIcon": true }]
	-name	наименование поля		
	-isIcon	признак изображения		
	-colorDefinition	определение цвета		
	fieldSeparator	разделитель текстовых полей		
	showLine	показывать направляющие		

	depthLevel	уровень отображения		<p>0 - все открыто 1 - первый уровень отобразить 2 - отобразить до 2-го уровня (2-ой уровень схлопнут) доставлять фильтр</p> <pre>[{ "dataTypeName": "int", "dbInfo": "lvl", "operationType": "<=", "value1": "-depthLevel-" }]</pre>
	dragDropMode	режим перетаскивания в дереве		<p>all - перетаскивание любого элемента на любой уровень currentLevel - перемещение только внутри текущего узла leaf - todo parent - todo</p>
	isAsyncLoad	загрузка данных только при раскрытии узла		false (при данном значении depthLevel не учитывать)
	checkStrictly	режим выбора нескольких элементов		<p>true - выделять только указанный элемент, по умолчанию false - выделять указанный элемент вместе с подчиненными элементами. При снятии у родителя - у потомков остаются. При isAsyncLoad: true, раскрыть все подчиненные узлы и выбрать все подчиненные элементы</p>
	searchMode	режим поиска		
	+background			
	-color			
	-image			
	+SelectedStyle			
	+CheckedStyle			
	+ToolBar			для контекстно зависимых операций?
	isMultiselect			
TDataSet		Источник данных		Наименование класса для получения/сохранения данных
	ObjectName	Название датасета	varchar	

	ClassName	Системное наименование класса	varchar ?	
	MasterDataSet	Источник данных - родитель	{ }	
	KeyField			
	MasterKey			
	DetailKey			
CContainer		Базовый контейнер		
	ObjectName			
	Caption			
	Hint			
	TPosition			
	TStyle			
	Resizable			
	Parent			
	enable			
	fragmentation			

Общие свойства:

1. Высота, Ширина;
2. (X, Y);
3. Стилль;
4. Hint;
5. Привязка, якоря;
6. Контейнер-владелец;
7. Источник данных (s_class + s_attribute) для контролов, работающих с данными;

Таблица 6. Перечень элементов управления (control)

Элемент управления	Тип	Свойства	Дополнительно
Форма		<p>Вызов по кнопке:</p> <p>"formName": "_имя_формы_", - наименование вызываемой формы</p> <p>"closeFormAfterSave": false, - закрывать форму после нажатия кнопки "Сохранить"</p> <p>"formType": "drawer", - тип формы modal/drawer</p> <p>"operation": "create", - тип операции create/edit</p>	
Простое однострочное поле	simple_field	<p>dataset: s_class</p> <p>data_field: s_attribute</p>	
Многострочный текст	multiline_field	<p>dataset: s_class</p> <p>data_field: s_attribute</p>	
Дата	data_field	<p>dataset: s_class</p> <p>data_field: s_attribute</p>	
Дата/время	datetime_field	<p>dataset: s_class</p> <p>data_field: s_attribute</p>	
Время	time_field	<p>dataset: s_class</p> <p>data_field: s_attribute</p>	
Выпадающий список	dropdown_field	<p>dataset: s_class</p> <p>data_field: s_attribute</p> <p>values:[key1:text1, key2:text2]</p>	<p>заданные значения для отображения и ключи для БД</p> <p>0- системный, 1- пользовательский</p> <p>true - Да, false - нет, null - Не установлено</p>

Чекбокс	checkbox_field	dataset: s_class data_field: s_attribute	
Ссылка на карточку	object_ref_field	dataset: s_class data_field: s_attribute	
Автокомплит	autocomplete_field	list_dataset: s_class1, key_list: s_attribute1, display_field:[s_attribute1, s_attribute2], visible_row_count: 20	из справочника с динамический выбор по подстроке
Связанные автокомплиты	autocomplete_relation_field		
Интервал дат	date_interval_field		
Интервал времени	time_interval_field		
Автокомплит с ограничением	ext_autocomplete_field		заданные значения, доп. условия фильтрации
Радиокнопка	radio_field		
Текст	label		

Изображение	Picture	<p>Таблица</p> <p>Поле</p> <p>Высота</p> <p>Ширина</p> <p>Левый верхний угол</p> <p>Правый нижний угол</p> <pre>"stagePhoto": { ---"ord": 3, "componentType": { "componentName": "Picture" }, "type": "picture", "table": "public.ref_stage", "field": "stage_photo", "dataType": "???", "label": "Фото"</pre>	
ссылка на файл ссылка url	ref_field		
Аккордеон	accordion		
Закладки	Tabs		
Панель	panel		
Кнопка	button		
Кнопка со списком	drop_down_button		
Многострочный текст с расширением по заполнению	autoresize_text_field		

Список с мультивыбором	multiselect_autocomplete_field		
Автокомплит с вызовом справочника			
Элемент с вызовом карты/ синхронизация с картой		<p>Отображение карты - в модальном окне при клике на иконку</p> <p>возможность ручного ввода/ только с карты</p> <p>начальная область для отображения</p> <p>Текстовая подсказка водяными знаками</p> <p>Отметка обязательности</p> <p>Индикация изменения значения</p> <p>Вид карты</p> <p>Поля (x, y, point)</p>	
Область карты с перечнем объектов	RLMap		
Плитка	PictureList	<p>Таблица</p> <p>Поле для изображения (хранилище, путь, имя)</p> <p>Ширина изображения</p> <p>Высота изображения (для сохранения пропорции исходного фото можно указывать значение "auto")</p> <p>Массив иконок (отображаются поверх изображения)</p> <p>Перечень фильтров</p> <p>Сортировка</p>	

5.2.3 Принципы построения интеграций

Таблица 7. Типы адаптеров (s_core.s_integration.integration_type).

ИД	Наименование	Описание	Дополнительно
1	s3	Адаптер для хранения файлов во внешних хранилищах	
2	string	Адаптер для хранения внешних ссылок	Использовалось в пробках для сохранения внешних ссылок как файлов.
3	API	Адаптер для взаимодействия с API внешних систем	Адаптер для взаимодействия с API внешних систем
4	email	Адаптер для взаимодействия с почтовыми ящиками	Взаимодействие с почтой
5	DBLink	Адаптер для взаимодействия с внешними БД	Источники данных/транспорт п.3 Параметры адаптера: Внешние (remote) БД
6	БД	Адаптер для хранения файлов в БД	
7	Telegram	Адаптер для взаимодействия с Telegram	

Таблица 8. Взаимодействие со всеми внешними системами в таблице s_core.s_integration:

№	Наименование	Тип	Обязательно	По умолчанию	Комментарий
1	id	int4	да	sequence	ИД записи
2	integration_name	varchar	да		Системное наименование интеграции
3	integration_caption	varchar	да		UI наименование интеграции
4	integration_description	varchar			Описание интеграции

5	dt_create	timestamp	да	now()	Дата создания записи
6	dt_edit	timestamp			Дата последнего изменения записи
7	dt_delete	timestamp			Дата логического удаления записи
8	dt_start	timestamp			Дата начала действия
9	dt_expire	timestamp			Дата окончания действия
10	id_creator	int4			ИД пользователя, создавшего запись
11	id_editor	int4			ИД пользователя, изменившего запись последним
12	integration_params	json			Параметры внешней системы
13	integration_type	int2			Тип адаптера

Организация вызовов из UI:

1. Описать s_integration;
2. Описать s_core.s_exchange_param;
3. Описать логику компонента.

Пример вызова сервиса по кнопке:

```
{
  "multiMode": false,
  "btnShowCondition": "{id}!=null",
  "btnBlockingMode": "disabled",
  "idIntegration": 11, - параметры интеграции
  "useCurrentAccount": false,
  "idExchangeParam": 13, - протокол импорта/экспорта
  "params": [
    {
      "name": "packageVersionId",
      "value": "7"
    },
    {
      "name": "complexGroupId",
```

```
    "value": "id.value"
  },
  {
    "name": "force",
    "value": false
  }
],
"confirmedParams": [
  {
    "name": "packageVersionId",
    "value": "7"
  },
  {
    "name": "complexGroupId",
    "value": "id.value"
  },
  {
    "name": "force",
    "value": true
  }
],
"notification": {
  "success": "Пакет обновлений успешно назначен!",
  "error": "{message} {debugMessage}",
  "confirm": "{message}"
}
}
```

5.3 Организация пользовательского интерфейса

Рабочие области пользовательского интерфейса:

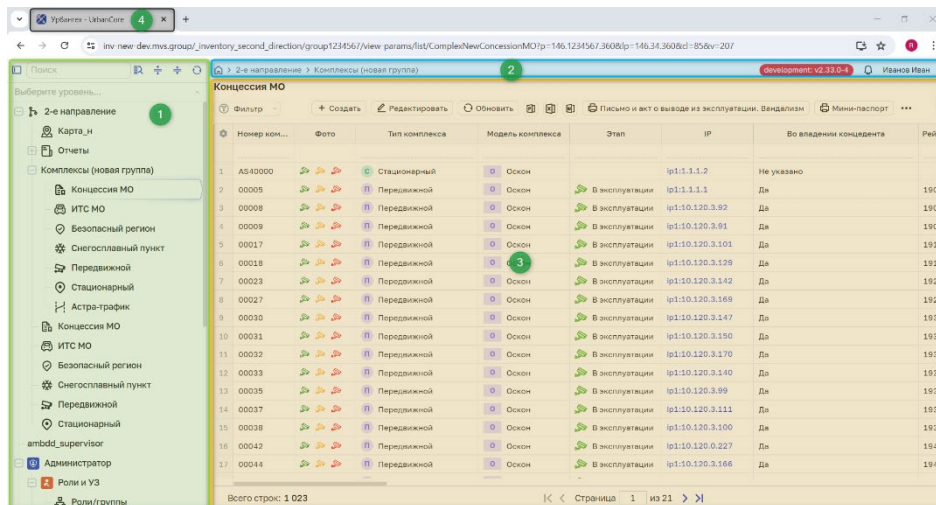



Таблица 9. Описание элементов пользовательского интерфейса.

№	Элемент/регион	Описание
1	панель меню	<p>внешний вид зависит от текущих общих настроек системы и персональных настроек пользователя</p> <p>1 - дерево</p> <p>2 - дерево с выпадающим списком</p> <p>3 - плашки</p> <p>4 - панель элементами одного уровня сворачивается в панель быстрого доступа. Видимость кнопок на панели быстрого доступа регулируется настройками пользователя</p>
2	заголовок	информация о текущих характеристиках режима работы
3	рабочая зона пункта меню	<p>1 - таблица</p> <p>2 - отчет</p> <p>3 - карта</p> <p>4 - дашборд</p> <p>5 - календарь/расписание</p> <p>6 - фрейм</p> <p>7 - произвольная форма</p>

4	заголовок браузера	выводится информация из системных параметров s_core.s_params:app_caption
5	статусная строка	отображается информация о версии приложения, + дополнительная информация (какая? уведомления?) Пример статусной строки 

5.4 Настройка экранных форм

Программное -

<https://confluence.mvs.group/pages/viewpage.action?pageId=292716708>

5.4.1 Последовательность настройки форм

Создается запрос SQL извлечения данных для формы из БД:

1. Если для таблицы (грида) нужно использовать форму редактирования или превью - один из столбцов грида должен называться **id** и содержать идентификатор записи (может быть невидимым).
2. Если основной класс для формы редактирования или превью запрос - обязательно добавляем условие [Идентификатор записи] = {\$ID_OBJ\$}. Это позволяет выбрать *единственную* запись для отображения в форме (в отличии от grid).

Пример:

```
SELECT
    c.id,
    c.name,
    c.line_address,
    c.contragent_id
FROM ambdd.complex c
WHERE
    c.id = {$ID_OBJ$}
```

3. Комментарии в запросе предпочтительно делать в формате: /* ... */

Создается запись для класса формы **s_core.s_class**

Задаются атрибуты класса, см. таблицу ниже. Здесь и далее не показаны колонки, которые заполняются автоматически (их можно игнорировать при заполнении).

Название	Тип данных	Комментарий	Пример заполнения для класса
----------	------------	-------------	------------------------------

cls_name	varchar	Системное наименование класса	<i>AmbddHearthEdit</i>
cls_caption	varchar	UI наименование класса	<i>Редактирование очага</i>
cls_description	varchar	Описание класса	<i>Редактирование очага</i>
id_cls_type	int2	Тип класса (1 - таблица, 2 - вью в БД, 3 - запрос) TODO - справочник типов классов	3
cls_schema	varchar	Наименование схемы размещения таблицы/представления	<i>s_core</i>
cls_query	text	Текст запроса/ наименование таблицы/представления	<p>Пример запроса</p> <pre> WITH event_type_names AS (SELECT eh.hearth_id , string_agg(DISTINCT e.id::varchar, ', ') as event_id_names , string_agg(DISTINCT to_char(e.date_finish_plan, 'DD Mon.YYYY'), ', ') as event_dates FROM ambdd.event_hearth eh JOIN ambdd.event e ON e.id = eh.event_id AND e.user_name = 'SKPI' /* JOIN ambdd.event_type et ON e.type_id = et.id*/ GROUP BY eh.hearth_id) select h.id , h.uid /*, h.is_verified*/ , h.lost_count , h.stricken_count /*, h.date_created*/ , event_id_names from ambdd.hearth h left outer join ambdd.event_hearth eh on h.id = eh.hearth_id left join event_type_names on event_type_names.hearth_id = h.id where h.id = {\$ID_OBJ\$} </pre>
id_cls_kind	int2	Вид класса (1- системный, не предназначен для корректировки через интерфейс администратора, 2 - пользовательский, с возможностью настройки через интерфейс администратора) TODO - справочник видов классов	2

cls_db_name	varchar	Наименование таблицы в БД или alias ведущей таблицы в запросе	<i>h</i>
cls_export_ord	int4	Зависимости при экспорте/импорте	<i>[NULL]</i>
id_parent_cls	int4	ИД родительского класса	
id_default_view	int4	ИД системного представления по умолчанию	<i>[NULL]</i>
cls_search_options	json	Настройки поиска для класса по умолчанию	<i>[NULL]</i>
cls_log_enable	bool	Признак логирования класса. true - логирование включено, false или null - логирование отключено	
stat_log	bool	Признак сбора статистики по использованию класса. true - собирать статистику, false - исключить из сбора	

Атрибуты формы `s_core.s_attribute`

В отличие от `grid`, заполнять таблицу атрибутов для формы не нужно.

Редактируется запись класса `grid s_core.s_view`:

В таблице `s_core.s_view` для ранее созданного класса `grid` корректируются три поля:

1. в поле `id_dbl_click_operation` - указывается "2";
2. `dbl_click_params` (см. ниже)
3. `preview_params` (см. ниже).

5.4.1.1 Общее описание

Настройки внешнего вида форм и их наполнения данными хранятся в таблице `s_core.s_form`.

Основные поля таблицы:

Поле	Назначение
id_form_type	Тип формы (1-edit_form, 2-preview, 3-report)
form_name	Наименование формы

form_caption	UI наименование формы
form_description	Описание формы
widget_mode_enable	Доступность в режиме виджета. true - доступно, false или null - недоступно
id_form_kind	Вид формы (1-web, 2-mobile)
form_content	JSON. Содержимое формы. Здесь указываем настройки отображения полей, мэппинг полей на таблицы БД при сохранении и т.д.
form_data	JSON. Данные формы. Здесь задаём имя класса и набор полей, которые будут выведены на форму

3. Все поля, перечисленные в **form_data**, должны быть указаны так же и в **form_content** (т. е. набор полей в обоих json должен совпадать).

5.4.1.2 Настройка данных формы (form_data)

В данном JSON нужно перечислить необходимые поля из класса, на основании которого форма заполняется данными.

Пример:

```
{
  "AmbddComplexEdit": [
    {
      "id": {
        "attr_name": "id"
      }
    }, {
      "name": {
        "attr_name": "name"
      }
    }, {
      "lineAddress": {
        "attr_name": "line_address"
      }
    }, {
      "contragentId": {
        "attr_name": "contragent_id"
      }
    }
  ]
}
```

JSON содержит массив, название массива соответствует названию класса (**AmbddComplexEdit**).

Каждый элемент массива содержит ключ, который соответствует имени поля на форме (**id**, **name**, **lineAddress**, **contragentId**) и объект, определяющий соответствующее поле в запросе или таблице ("**attr_name**": "**id**", "**attr_name**": "**line_address**" и т.д.)

5.4.2 Настройка содержимого формы (form_content)

Для настройки содержимого формы используем имена полей из **form_data**.

```
{
  "id": {
```

```
"ord": 0,
"componentType": {
  "componentName": "Label",
  "disabled": true
},
"type": "number",
"hide": false,
"table": "ambdd.complex",
"field": "id",
"dataType": "int4",
"label": "ИД комплекса",
"action": "formSave",
"key": "$cid$"
},
"name": {
  "ord": 1,
  "componentType": {
    "componentName": "Input"
  },
  "label": "Номер комплекса",
  "type": "string",
  "hide": false,
  "table": "ambdd.complex",
  "field": "name",
  "dataType": "text",
  "idFieldName": "id"
},
"lineAddress": {
  "ord": 2,
  "componentType": {
    "componentName": "Input"
  },
  "label": "Адрес рубежа",
  "type": "string",
  "hide": false,
  "table": "ambdd.complex",
  "field": "line_address",
  "dataType": "text",
  "idFieldName": "id"
},
"contragentId": {
  "ord": 3,
  "componentType": {
    "componentName": "CustomSelect",
    "refDataSet": "AmbddContragent",
    "refKeyField": "id",
    "refFieldList": [
      "name"
    ]
  },
  "type": "reference",
  "table": "ambdd.complex",
  "field": "contragent_id",
  "dataType": "int4",
  "label": "Экспл. орг.",
  "idFieldName": "id"
},
"btnSave": {
  "ord": 100,
  "componentType": {
    "componentName": "Button",
    "action": "saveObjectAction"
  }
},
```

```












    "label": "Сохранить"
  },
  "btnCancel": {
    "ord": 110,
    "componentType": {
      "componentName": "Button",
      "action": "cancelObjectAction"
    },
    "label": "Отмена"
  }
}

```

Для каждого поля необходимо задать набор свойств.

1. **ord** - порядок расположения элемента на форме. В формах редактирования элементы расположены в один столбец, в формах превью по умолчанию в два.
2. **hide** - скрыть поле
3. Объект "**componentType**"

1. **componentName**- тип компонента на форме. Возможные значения:

1. **Label** - метка (нередатируемое поле)
2. **Input** - поле ввода, допускаются любые символы
3. **TextArea** - поле ввода текста, допускаются любые символы (в том числе и в Unicode, например:           ). Количество строк не ограничено.

Функционал применим не только к **TextArea**, в **Label** тоже работает

```

"clsQuery": {
  "ord": 99,
  "type": "text",
  "componentType": {
    "componentName": "TextArea",
    "disabled": true,
    "style": {
      "height": "300px"
    }
  },
  "label": "Запрос",
  "style": {
    "gridColumn": "span 2"
  }
}

```

4. **InputNumber** - поле ввода, допускаются только цифры
5. **Checkbox** - булево поле, представляет собой check box
6. **CustomSelect** - выпадающий список. Имеет расширенные настройки:
 1. **refDataSet** - название класса, на основании которого будет заполнен данные список

2. **refKeyField** - атрибут класса, который будет сохранен в заданное поле таблицы (как правило это поле с ID)
 3. **refFieldList** - атрибут класса, содержащий значение для вывода в списке. Массив.
7. **Button** - кнопка. Расширенные настройки:
1. **action**. Возможные значения:
 1. **saveObjectAction** - сохранение объекта
 2. **cancelObjectAction** - отмена
 3. **printAction** - печать по шаблону
 2. **actionParams**. Пока применяется только для действия "printAction". Содержит массив: [{"name": "templateName", "value": "decommissioningReportReconstruction"}], в котором задаём имя шаблона для печати.
8. **Upload** - компонент для загрузки файла, id хранилища (справочник s_core.s_integration) передается в id_integration_storage, форма с этим компонентом должна вызываться с обязательными параметрами, прописанными в s_core.s_object_operation.operation_params, пример:

```
{
  "formName": "dceRegionDocStorageCreate",
  "closeFormAfterSave": true,
  "uploadDoc": true,
  "keyField": [
    {
      "value": "$fileSize$",
      "table": "dce.region_docs",
      "field": "doc_size",
      "dataType": "int4"
    },
    {
      "value": "$filePath$",
      "table": "dce.region_docs",
      "field": "doc_path",
      "dataType": "varchar"
    },
    {
      "value": "$fileName$",
      "table": "dce.region_docs",
      "field": "doc_name",
      "dataType": "varchar"
    }
  ]
}
```

переменные \$fileSize\$, \$filePath\$, \$fileName\$ заполняются бэком при сохранении файла

пример настройки формы:

```
{
  "id": {
    "ord": 1,
    "componentType": {
```

```
        "componentName": "Label"
    },
    "type": "number",
    "table": "dce.region_docs",
    "field": "id",
    "dataType": "int4",
    "label": "ИД записи"
},
"docUpload": {
    "ord": 2,
    "componentType": {"componentName": "Upload"},
    "type": "upload",
    "label": "Документ"
},
"idIntegrationStorage": {
    "ord": 10,
    "componentType": {
        "componentName": "CustomSelect",
        "disabled": true,
        "refDataSet": "SysIntegration",
        "refKeyField": "id",
        "refFieldList": [
            "integration_caption"
        ]
    },
    "type": "reference",
    "table": "dce.region_docs",
    "field": "id_integration_storage",
    "dataType": "int2",
    "initialValue": 2,
    "label": "Хранилище документов"
},
"btnSave": {
    "ord": 100,
    "componentType": {
        "componentName": "Button",
        "action": "saveObjectAction"
    },
    "label": "Сохранить"
},
"btnCancel": {
    "ord": 101,
    "componentType": {
        "componentName": "Button",
        "action": "cancelObjectAction"
    },
    "label": "Отмена"
}
}
```

ограничение: upload-форма работает только в тулбарах таблиц, находящихся в форме.

2. **disabled** - сделать поле неактивным (нет возможности отредактировать)
4. **type** - тип данных на форме. Возможные значения:
 1. number
 2. string
 3. boolean
 4. reference - указывается для компонента "CustomSelect", ссылка на другую таблицу

5. text

5. **table** - таблица, в которую будет записано значение с формы (обязательно указываем схему!)
6. **field** - поле таблицы, в которую будет записано значение с формы
7. **dataType** - тип данных в БД
8. **label** - название поля
9. **action** - данное свойство задаём только для тех полей, который являются ключевыми (по ним идентифицируется запись в таблице, которую необходимо изменить). Значение: **formSave**
10. **key** - идентификатор ключевого поля. Этот ключ нужно указывать в том случае, если данные на форме сохраняются в несколько разных таблиц. Для каждой таблицы должен указывать разный **key**
11. **idFieldName** - указываем имя поля, содержащего идентификатор необходимой таблицы

Свойства формы, которые используются **только** в сложных формах (где присутствует grid, или поля формы расположены в несколько колонок), в простых формах они **не нужны**.

5.4.2.1 Настройка формы превью

Настройка аналогична форме редактирования, но т.к. в данной форме информация приведена только для просмотра, нет необходимости задавать полный набор свойств (в какую таблицу записывать и т.д.)

На форме превью не должно быть кнопок (например, btnSave, btnCancel). Т. е. при копировании формы редактирования в форму превью надо обязательно удалить кнопки, иначе превью показываться не будет.

Для добавления кнопок на форму редактирования необходимо использовать toolbar.

Пример:

```
{
  "id": {
    "ord": 0,
    "componentType": {
      "componentName": "Label"
    },
    "label": "ИД комплекса"
  },
  "name": {
    "ord": 1,
    "componentType": {
      "componentName": "Label"
    },
    "label": "Номер комплекса"
  },
  "lineAddress": {
    "ord": 2,
    "componentType": {
      "componentName": "Label"
    }
  }
}
```

```
    },  
    "label": "Адрес рубежа"  
  }  
}
```

5.4.3 Привязка формы к гриду

Для привязки формы редактирования или формы превью к таблице используются поля таблицы `s_core.s_view`:

5.4.3.1 Форма по дабл-клику

Записать JSON в поле `s_core.s_view.dbl_click_params`

Пример:

```
{  
  "formName": "ambddComplexEdit",  
  "operation": "edit",  
  "formType": "modal",  
  "closeFormAfterSave": true  
}
```

Комментарии

- Если не указывать `"formType": "modal"` - форма будет выдвигаться справа (поведение по умолчанию)
- `"closeFormAfterSave": true` - закрытие формы по нажатию "Сохранить"

5.4.3.2 Форма для превью

Записать JSON в поле `s_core.s_view.preview_params`

Пример:

```
{"formName": "ambddComplexPreview"}
```

5.4.4 Настройки даты в форме

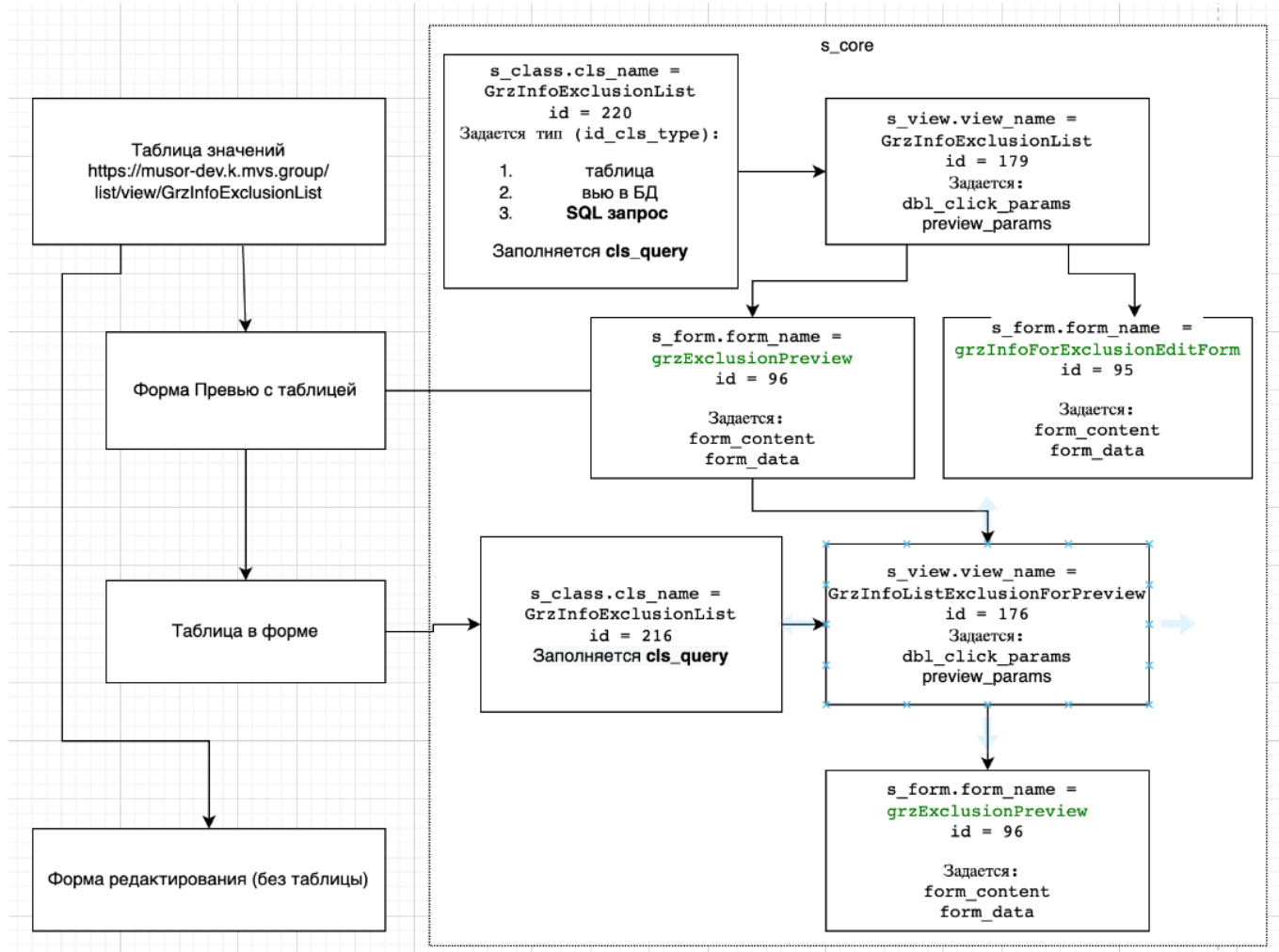
Пример настройки поля даты в форме:

Пример:

```
"purchaseDatetime": {  
  "ord": 12,  
  "componentType": {  
    "componentName": "DatePicker"  
  },  
  "type": "string",  
  "table": "public.complex",  
  "field": "purchase_datetime",  
  "dataType": "date",  
  "label": "Дата покупки",  
  "idFieldName": "complexId"  
},
```

5.4.5 Настройка таблицы на форме

Схема взаимодействия классов, и основные настройки в них:

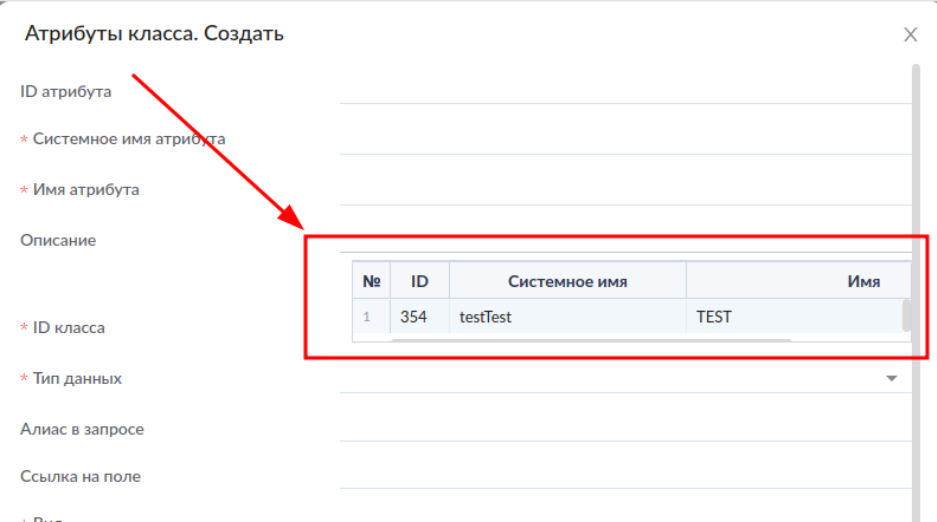


5.5 Примеры компонентов экранных форм

Описание компонентов, их параметров и примеров реализации в настройках форм. Структура компонентов описана в рамках параметра ComponentType.

SimpleGrid

Описание	Позволяет отображать таблицу для того или иного представления
Структура	<pre data-bbox="359 571 901 1008">"componentType": { "componentName": "SimpleGrid", "viewName": "VIEW_NAME", "keyField": [{ "dataType": "DATA_TYPE_NAME", "paramName": "objectId", "value": "VALUE" }], }, "type": "gridSelect",</pre> <p data-bbox="327 1041 383 1075">где:</p> <ul data-bbox="375 1120 1484 1747" style="list-style-type: none"> • VIEW_NAME - имя представления, которое необходимо выводить в таблицу. Требования: <ul style="list-style-type: none"> ○ представление должно иметь фильтр с ID_ОБЪЕКТ (поле s_view.view_rule), который будет принимать значение id_object от компонента. Например: <code>sv.id=\${ID_ОБЪЕКТ\$}</code> • DATA_TYPE_NAME - тип значения переменной id_object которое будет передаваться в правило представления. Требования: <ul style="list-style-type: none"> ○ тип значения должен соответствовать полю, с которым оно участвует в выражение фильтра представления (поле s_view.view_rule). Например, если выражение <code>sv.id=\${ID_ОБЪЕКТ\$}</code> и sv.id это int4, то и dataType должен принимать int4 • VALUE - значение из параметров представления, может использоваться как: <ul style="list-style-type: none"> ○ Значение из другого параметра формы - ИМЯ_ПАРАМЕТРА.value, например: <code>className.value</code> или <code>id.value</code> ○ Значение ID редактируемого элемента: <ul style="list-style-type: none"> ▪ для главного грида - objectId ▪ для грида, внутри формы главного грида - objectId.objectId

<p>Пример</p>	<pre>"class": { "ord": 50, "required": true, "hide": false, "componentType": { "componentName": "SimpleGrid", "viewName": "SysAdminClassObject", "keyField": [{ "dataType": "int4", "paramName": "objectId", "value": "objectId.objectId" }], "height": 70 }, "type": "gridSelect", "table": "s_core.s_attribute", "field": "id_class", "dataType": "int4", "label": "ID класса", "idFieldName": "id" },</pre>								
<p>Скриншот</p>	 <p>Атрибуты класса. Создать</p> <p>ID атрибута</p> <p>* Системное имя атрибута</p> <p>* Имя атрибута</p> <p>Описание</p> <table border="1"> <thead> <tr> <th>№</th> <th>ID</th> <th>Системное имя</th> <th>Имя</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>354</td> <td>testTest</td> <td>TEST</td> </tr> </tbody> </table> <p>* ID класса</p> <p>* Тип данных</p> <p>Алиас в запросе</p> <p>Ссылка на поле</p> <p>...</p>	№	ID	Системное имя	Имя	1	354	testTest	TEST
№	ID	Системное имя	Имя						
1	354	testTest	TEST						

CustomSelect

Описание	Позволяет отображать таблицу для того или иного представления
Структура	<pre> "componentType": { "componentName": "CustomSelect", "refDataSet": "CLASS_NAME", "refKeyField": "REF_KEY", "refFieldList": ["GET_FIELD1", "GET_FIELD2", ...] }, "type": "reference", </pre> <p>где:</p> <ul style="list-style-type: none"> • CLASS_NAME - имя класса, из которого необходимо получить список значений. Требования: <ul style="list-style-type: none"> ○ в классе должны быть описаны атрибуты, которые будут использоваться в компоненте • REF_KEY - название ключевого поля в таблице из CLASS_NAME, по которому будет выбран список значений. Значения для ключевого поля берется из полученного значения для атрибута формы. Требования: <ul style="list-style-type: none"> ○ тип значения поля REF_KEY должен соответствовать типу значения для поля, связанного с атрибутом формы, для которого используется компонент • GET_FIELD1, GET_FIELD2 - список полей в таблице из CLASS_NAME для отображения в виде значений. Если полей несколько - значения выводятся через запятую, в порядке индекса списка. Требования: <ul style="list-style-type: none"> ○ поля, используемые в списке, должны быть описаны атрибутами класса CLASS_NAME

Пример	<pre> "dataType": { "ord": 55, "required": true, "componentType": { "componentName": "CustomSelect", "refDataSet": "SysDataType", "refKeyField": "id", "refFieldList": ["data_type_caption", "data_type_name"] }, "type": "reference", "table": "s_core.s_attribute", "field": "id_data_type", "dataType": "int2", "label": "Тип данных", "idFieldName": "id" }, </pre>
Скриншот	

Label

Сам компонент используется в основном для форм превью. Ниже приведен список вариантов использования Label для различных типов полей

Timestamp

Описание	Форматировать в строку с нужным форматом
Структура	<pre> "componentType": { "componentName": "Label", "format": "DATE_FORMAT" }, "type": "date", </pre> <ul style="list-style-type: none"> • DATE_FORMAT - формат даты, где: <ul style="list-style-type: none"> ○ DD - день (01-31) ○ MM - месяц (01-12) ○ YYYYYY - год полный

	<ul style="list-style-type: none"> ○ HH - часы (00-23) ○ mm - минуты (00-59) ○ ss - секунды (00-59) 								
Пример	<pre>"dtCreate": { "ord": 14, "type": "date", "componentType": { "componentName": "Label", "format": "DD.MM.YYYY HH:mm:ss" }, "label": "Дата создания" },</pre>								
Скриншот	<p>The screenshot shows a table with the following data:</p> <table border="1"> <thead> <tr> <th>Создано</th> <th>Дата создания</th> </tr> </thead> <tbody> <tr> <td></td> <td>11.09.2022 15:58:24</td> </tr> <tr> <th>Изменено</th> <th>Дата изменения</th> </tr> <tr> <td></td> <td>12.10.2022 13:28:57</td> </tr> </tbody> </table>	Создано	Дата создания		11.09.2022 15:58:24	Изменено	Дата изменения		12.10.2022 13:28:57
Создано	Дата создания								
	11.09.2022 15:58:24								
Изменено	Дата изменения								
	12.10.2022 13:28:57								

Text

Описание	Отображение текста в виде текстового поля. Таким же образом можно отображать поля JSON, предварительно конвертируемые в запросе классе в тип text
Структура	<pre>"componentType": { "componentName": "TextArea", "disabled": true, "style": { "height": "300px" } }, "type": "text",</pre>

Пример

```
"clsQuery": {
  "ord": 99,
  "type": "text",
  "componentType": {
    "componentName": "TextArea",
    "disabled": true,
    "style": {
      "height": "300px"
    }
  },
  "label": "Запрос",
  "style": {
    "gridColumn": "span 2"
  }
},
},
```

Скриншот

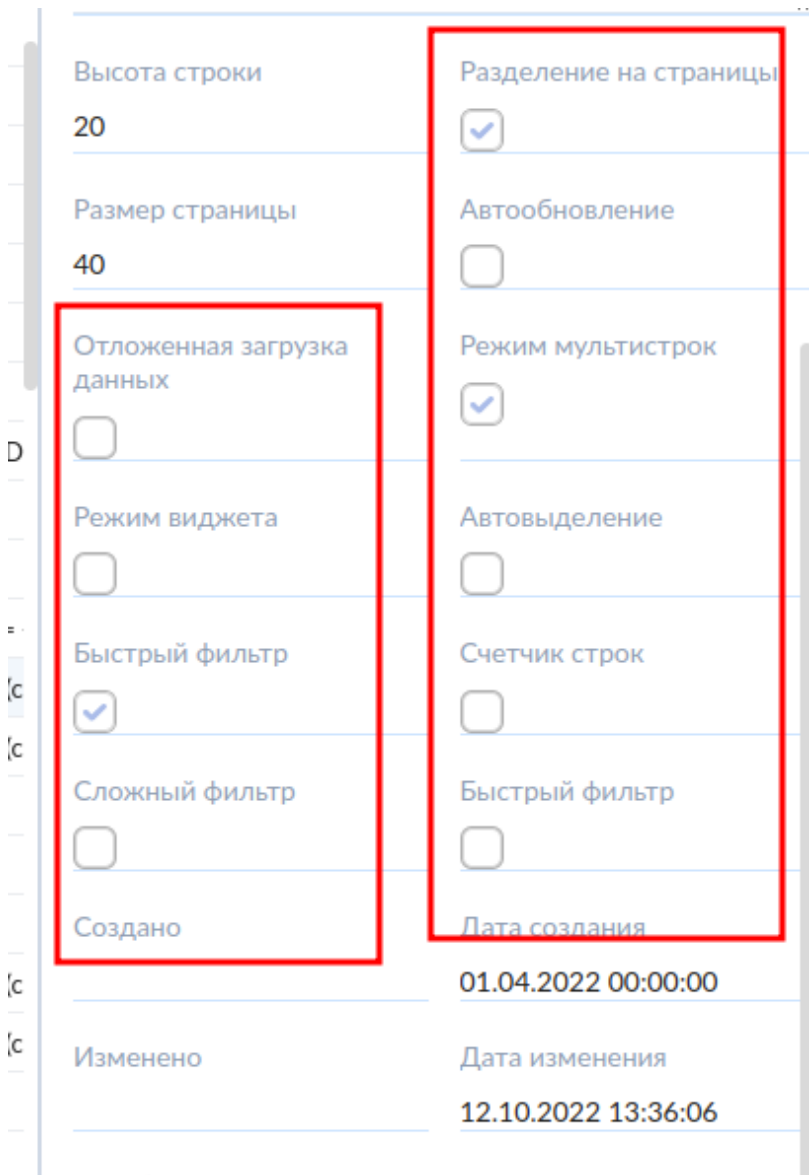
Атрибуты класса

№	ID ...	Системное имя ат...	Имя атрибута
7	3824	complexModelName	Модель комплекса
8	3825	stageName	Этап
9	3826	begin	Дата поверки
10	3827	end	Срок окончания по
11	3828	docnum	№ поверки

Запрос

```
SELECT
  mh.sid
  , c."name" complex_name
  , rct.name complex_type_name
  , rcm."name" complex_model_name
  , rls."name" stage_name
  , mh."begin"
  , mh."end"
  , mh."number" docnum
  , mh.insert_dt
  , mh.author
  , mh.muster_sert_state
FROM public.muster_history mh
```

Bool

Описание	Отображение в виде CheckBox
Структура	<pre>"componentType": { "componentName": "Checkbox", "disabled": true }, "type": "boolean",</pre>
Пример	<pre>"clsLogEnable": { "ord": 7, "type": "boolean", "componentType": { "componentName": "Checkbox", "disabled": true }, "label": "Логирование" },</pre>
Скриншот	 <p>The screenshot shows a settings page with various options. Two red boxes highlight specific checkbox settings:</p> <ul style="list-style-type: none"> Left box: Includes "Отложенная загрузка данных" (unchecked), "Режим виджета" (unchecked), "Быстрый фильтр" (checked), and "Сложный фильтр" (unchecked). Right box: Includes "Разделение на страницы" (checked), "Автообновление" (unchecked), "Режим мультистрок" (checked), "Автовыделение" (unchecked), "Счетчик строк" (unchecked), and "Быстрый фильтр" (unchecked). <p>Other visible settings include "Высота строки" (20), "Размер страницы" (40), "Создано" (01.04.2022 00:00:00), and "Изменено" (12.10.2022 13:36:06).</p>

5.6 Примеры параметров вызова экранных форм

Параметры вызова той или иной можно задать в разных таблицах, в зависимости от способа перехода на форму:

- **s_view.dbl_click_params** - двойной щелчок мыши по строке грида;
- **s_view.preview_params** - щелчок мыши по любой строке грида;
- **s_object_operation.operation_params** - нажатие по кнопке в гриде или в форме.

Общее описание параметров:

Название параметра	Описание параметра	Значение	Описание значения	Пример значений
formName	Ссылка на вызываемую форму	ИМЯ_ФОРМЫ	Имя формы form_name из таблицы s_form	"formName": "objectNomenclatureStageChange"
closeFormAfterSave	Действия с формой после выполнения	true	Закрывать	"closeFormAfterSave": true
		false	Не закрывать	
formType	Тип формы	modal	Форма откроется в модальном окне	"formType": "modal"
		drawer	Форма получит анимацию появления справа	
operation	Тип операции формы, по умолчанию берется тип из описания операции	create	режим создания новой записи	"operation": "create"
		edit	режим изменения выделенной записи	

keyField	Автоматическое заполнение полей в момент выполнения. Такие поля описывать в самой форме не нужно	Список справочников в формате json			<pre> "keyField": [{ "dataType": "int4", "field": "id_object", "paramName": "objectId", "table": "public.object_stag e_history", "value": "object.value" }, { "dataType": "int4", "field": "id_object_category ", "paramName": "objectId", "table": "public.object_stag e_history", "value": 3 }] </pre>
		Параметры справочника:			
		Название	Значение	Описание	
		dataType	<i>тип данных из таблицы s_data_type</i>	тип данных, в котором необходимо записать значение	
		field	<i>строка</i>	имя поля в таблице БД, в которое необходимо записать значение	
		paramName	objectId	???	
table	<i>строка</i>	имя таблицы БД, в которой содержится поле			
value	<ul style="list-style-type: none"> • фиксированное значение • значение из другого параметра формы: ИМЯ_АТТРИБУТА.value • системное значение для преобразования: <ul style="list-style-type: none"> ○ __currentIdAccount__ ○ __currentDate__ ○ __currentTime__ ○ __currentDateTime__ 	значение для передаваемого параметра, можно использовать значение других атрибутов формы, либо фиксированное значение			

			<ul style="list-style-type: none"> ○ <code>__currentIdArm_</code> ○ <code>__currentIdSubTree__</code> 	
formWidth	Задаёт ширину формы	значение в формате CSS	https://www.w3schools.com/cssref/pr_dim_width.php	"formWidth": "97%"
formHeight	Задаёт высоту формы	значение в формате CSS	https://www.w3schools.com/cssref/pr_dim_height.php	"formHeight": "100vh"

5.7 Автоматическое создание класса, представления и атрибутов

Для таблицы, физически существующей в БД, можно автоматически создать класс, представление и их атрибуты с помощью процедуры БД "s_core.create_class_w_attributes".

Процедура извлекает из системных таблиц БД информацию о самой таблице и её полях (в том числе типы данных и комментарии), на основании этой информации создаёт новый класс со всей необходимой структурой.

Вызов процедуры:

Пример вызова

```
call s_core.create_class_w_attributes
(
'public',      --tableSchema
'ref_voltage', --tableName
'RefVoltage',  --newSClassName
false,        --isMarkDeletedExistClass
true          --isCreateView
)
```

Входные параметры:

1. tableSchema - схема таблицы;
2. tableName – таблица;
3. newSClassName - название нового класса;
4. isMarkDeletedExistClass - если в справочнике s_class уже существует класс с таким именем, пометить его удалённым;
5. isCreateView - создать связанное с классом представление.

Создание класса на основании запроса:

Если класс нужно создать на основании запроса:

1. Создаём на основании запроса таблицу (крайне желательно в своей схеме):

Пример

```
SELECT
    r.id
    , r.name
INTO gkovalenko.tmp_dpl_reasons
FROM ambdd.reasons r
```

2. Выполняем для созданной таблицы процедуру "s_core.create_class_w_attributes".

3. После этого таблицу удаляем.

Комментариев в таком случае не будет, однако названия полей и их типы будут заполнены корректно.

5.8 Настройка меню

Для настройки меню используются 2 таблицы:

1. s_core.s_arm для настройки непосредственно пунктов;
2. s_core.s_arm_relation для настройки иерархии пунктов меню.

5.8.1 Основные атрибуты s_arm

Название поля	Комментарий	Возможные значения
item_name	Уникальное название пункта меню	
item_caption	Видимое название пункта	
item_description	Описание	
item_params	Параметры вызова пункта меню	<p>Основные параметры:</p> <ol style="list-style-type: none"> 1. NULL - группирующий элемент меню (часто выступает в качестве роли). 2. /map - используется для карты Инвентори 3. /list - список (grid) 4. /report - вызов формы отчета 5. /iframe/"Название формы в сторонней системе" - используется для отображения форм сторонних систем в интерфейсе платформы. Пример будет ниже

item_type	Тип элемента	<ol style="list-style-type: none"> 1. 1-роль/корневой элемент 2. 2 - группировка 3. 3-вызываемый элемент 4. 4- стартовая страница 5. 5-слой карты
id_class	ID класса. Заполнение этого поля актуально для пунктов меню с типами /list и /report	
id_object	ID объекта. Аналогично прошлому атрибуту, актуально для /list (содержит id_view) и /report (содержит id_report)	
item_icon	Название иконки. Список иконок здесь	
quick_panel_visible	Отображать на панель быстрого доступа	
mobile_arm_sync	Отображать или нет пункт меню в мобильном приложении (если true - отображать)	

5.8.2 Основные атрибуты s_arm_relation

Название поля	Комментарий
id_item	ID пункта меню
id_parent_item	ID родительского пункта меню
item_ord	Порядок расположения пункта меню в рамках родительского пункта

5.8.3 Настройка фреймов

Используется для отображения форм сторонних систем в интерфейсе платформы.

При настройке пункта меню, вызывающего фрейм, необходимо в столбце item_para,s таблицы s_arm указать:

- */iframe/"Название формы в сторонней системе"*

5.9 Настройка таблиц

5.9.1 Последовательность настройки grid (табличного представления данных)

Для настройки табличного представления данных на платформе UrbanCore настраиваются следующие классы:

- `s_core.s_class`;
- `s_core.s_attribute`;
- `s_core.s_view`;
- `s_core.s_view_attribute`.

При создании представления данных в виде таблицы (grid), может быть использованы следующие источники (поле `s_core.s_class.id_cls_type` принимает одно из трех значений):

1 - таблица в БД,

2 - вью в БД,

3 - SQL запрос.

Далее мы рассмотрим вариант 3 - создание табличного представления на базе SQL запроса к БД

1. Создаем запрос SQL извлечения данных для формы из БД

1. Для корректной работы фильтров в конце запроса добавляем условие "1 = 1".

Пример:

```
SELECT
    c.id,
    c.name,
    c.line_address,
    c.contragent_id
FROM ambdd.complex c
WHERE
    1 = 1
```

2. Комментарии в запросе предпочтительно делать в формате: `/* ... */`, т.к. комментарии в стиле: `-- комментарий` не всегда корректно обрабатываются, особенно если расположены в конце запроса.

2. Создаем новый класс `s_core.s_class`

Назначение класса - выборка данных из БД. Задаются свойства класса, см. таблицу ниже. Здесь и далее не показаны колонки, которые заполняются автоматически (их можно игнорировать при заполнении).

Название	Тип данных	Комментарий	Пример заполнения для класса
cls_name	varchar	Системное наименование класса	<i>AmbddLine</i>
cls_caption	varchar	UI наименование класса	<i>Рубежи</i>
cls_description	varchar	Описание класса	<i>Рубежи</i>
id_cls_type	int2	Тип класса (1 - таблица, 2 - вью в БД, 3 - запрос) TODO - справочник типов классов	3
cls_schema	varchar	Наименование схемы размещения таблицы/представления	<i>ambdd</i>

cls_query	text	Текст запроса/ наименование таблицы/представления	<pre>-- Рубежи WITH line_complex_name AS (SELECT lc.line_id, string_agg(c.name, ', ') as complexes FROM ambdd.line_complex lc LEFT JOIN ambdd.complex c on lc.complex_id = c.id GROUP BY lc.line_id) select distinct l.id , l.name as line_name , l.address as address , l.latitude as latitude , l.longitude as longitude , r."name" as road_name , l.piketage as piketage , ct.title as city_name , m.municipality_name as municipality_name /*, case when lcn.line_id is null then 'Нет' else 'Да' end as complex_is*/ , case when lcn.line_id is null then 0 else 1 end as complex_is , lcn.complexes as complexes -- здесь выводится весь список соединенных с рубежом комплексов , lt.name as type_name , l.additional_info as additional_info</pre>
-----------	------	---	--

```
, l.ref_gibdd_name as
ref_gibdd_name

, l.concessor_name as
concessor_name

/*, case

when (l.powered_by_akb = 0 )
then 'Нет'

when (l.powered_by_akb = 1 )
then 'Да'

when (l.powered_by_akb =-1)
then 'Не указано'

end as powered_by_akb_name*/

, l.powered_by_akb as
powered_by_akb

/*, case

when (l.energy_con_reserved
= 0 ) then 'Нет'

when (l.energy_con_reserved
= 1 ) then 'Да'

when (l.energy_con_reserved
=-1) then 'Не указано'

end as
energy_con_reserved_name*/

, l.energy_con_reserved as
energy_con_reserved

, l.source_of_point_name as
source_of_point_name

, l.source_of_point_date as
source_of_point_date

, l.source_of_point_num as
source_of_point_num

from ambdd.line l

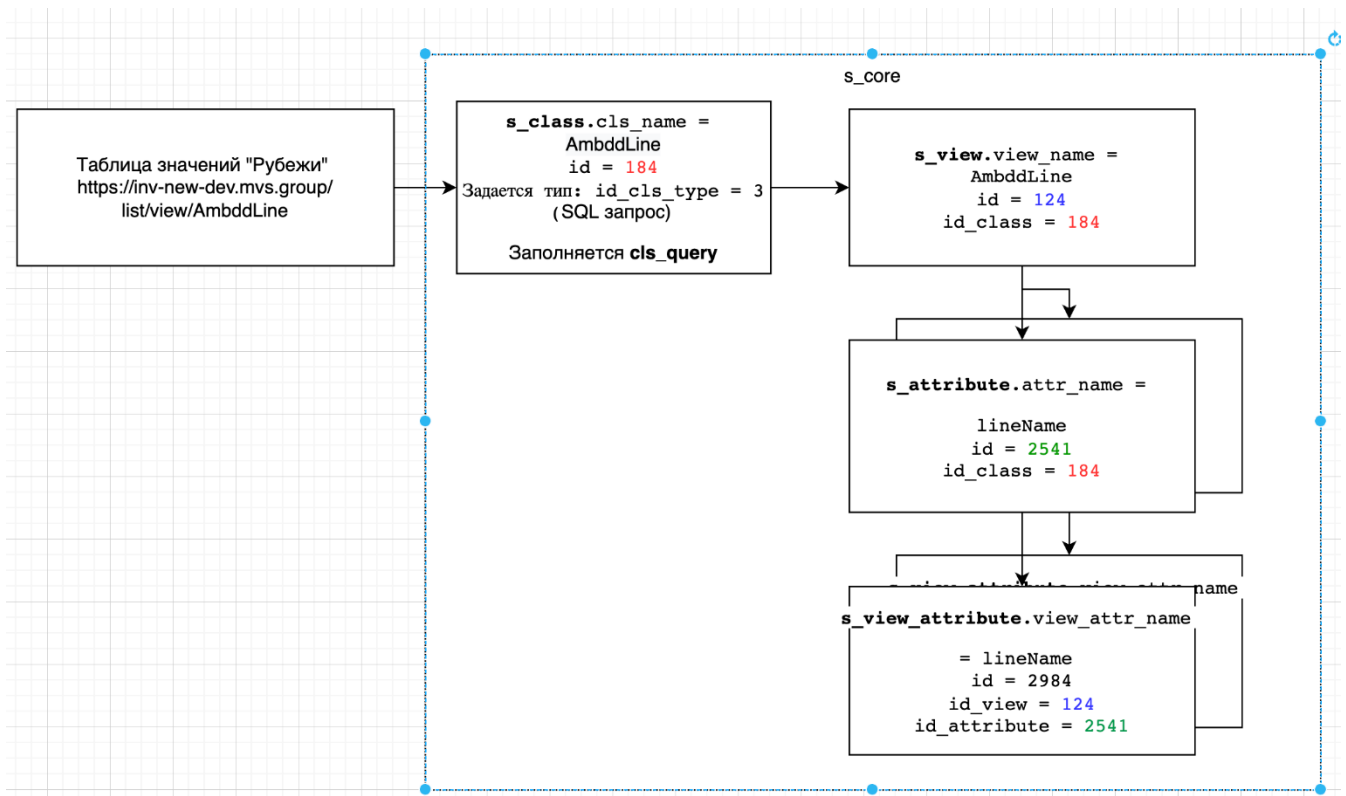
left outer join ambdd.road r
on l.road_id = r.id

left outer join ambdd.city
ct on l.city_id = ct.id

LEFT JOIN line_complex_name
lcn ON lcn.line_id = l.id

left outer join
ambdd.line_municipality lm
on l.id = lm.line_id
```

			<pre> left outer join ambdd.municipality m on lm.municipality_id = m.id left outer join ambdd.line_type lt on lt.id = l.line_type_id where l=1 </pre>
id_cls_kind	int2	Вид класса (1- системный, не предназначен для корректировки через интерфейс администратора, 2 - пользовательский, с возможностью настройки через интерфейс администратора) TODO - справочник видов классов	2
cls_db_name	varchar	Наименование или alias в БД	l
cls_export_ord	int4	Зависимости при экспорте/импорте	[NULL]
id_parent_cls	int4	ИД родительского класса	[NULL]
id_default_view	int4	ИД системного представления по умолчанию	[NULL]
cls_search_options	json	Настройки поиска для класса по умолчанию	[NULL]
cls_log_enable	bool	Признак логирования класса. true - логирование включено, false или null - логирование отключено	
stat_log	bool	Признак сбора статистики по использованию класса. true - собирать статистику, false - исключить из сбора	



3. Создаём новое представление `s_core.s_view`

Назначение представления - общая настройка грида для вывода данных (в том числе привязка к пункту меню). Задаются свойства представления, см. таблицу ниже. Представлений может быть несколько для одного и того же класса.

Название	Тип данных	Комментарий	Пример заполнения для класса
<code>id_class</code>	<code>int4</code>	ИД класса	184
<code>view_name</code>	<code>varchar</code>	Наименование системного представления	AmbddLine
<code>view_caption</code>	<code>varchar</code>	UI наименование системного представления	Рубежи
<code>view_description</code>	<code>varchar</code>	Описание системного представления	Рубежи
<code>paging_enable</code>	<code>bool</code>	Режим пагинации. true - режим пагинации включен	true
<code>page_size</code>	<code>int4</code>	Размер страницы при включенном режиме пагинации	50

lazy_load_data	bool	Отложенная загрузка данных (до момента выбора пользователем фильтра или выбора действия)	false
view_rule	varchar	Дополнительные условия фильтрации для представления	
multi_line_enable	bool	Режим работы с несколькими записями. true - разрешить выделение нескольких строк, false - запретить выделение нескольких строк	
vidget_mode_enable	bool	Доступность в режиме виджета. true - доступно, false или null - недоступно	
auto_refresh	bool	Автообновление страницы при выполнении операций. true - автоматически обновлять страницу при удалении, изменении состояния данных (редактировании), false - не обновлять, выполнять обновление по запросу пользователя (по кнопке обновить или при переходе на другую страницу)	
id_dbl_click_operation	int4	ИД операции, выполняемой при двойном клике на таблице	
dbl_click_params	json	Параметры операции, выполняемой по двойному клику на таблице	
preview_params	json	Параметры панели предварительного просмотра	{"formName": "ambddLineEdit", "operation": "edit", "formType": "modal", "closeFormAfterSave": true}
auto_select	bool	Автоматически выбирать первую запись	{"formName": "ambddLinePreview"}

row_height	int2	Высота строки по умолчанию	
floating_filter	bool	Видимость строки быстрого фильтра	true
view_type_params	json	Параметры типа представления	
auto_row_count	bool	Отображение счетчика строк. true - автоматически при каждом запросе в БД, false - по клику пользователя	

4. Атрибуты класса s_core.s_attribute

Назначение атрибутов класса - настройка правил обработки конкретных полей запроса. Заполняются записи в таблице атрибутов, для каждого значащего поля запроса (или таблицы).

Название	Тип данных	Комментарий	Пример заполнения для класса
id_class	int4	ИД класса	184
attr_name	varchar	Системное имя атрибута	lineName
attr_caption	varchar	UI наименование атрибута	Номер рубежа
attr_description	varchar	Описание атрибута	Номер рубежа
id_data_type	int2	ИД типа данных	6
attr_size	int2	Размер поля	
attr_precision	int2	Точность поля	
attr_rule	json	Правила ввода	
attr_log_enable	bool	Признак логирования атрибута	
attr_read_only	bool	Признак "Только для чтения"	
attr_p_key_ord	bool	Порядок вхождения в первичный ключ	
id_attr_kind	int2	Вид атрибута (1-системный, 2-пользовательский)	2
attr_sortable	bool	Возможность сортировки	true
seach_enable	bool	Доступность поиска по этому полю	
attr_db_name	varchar	Алиас атрибута в запросе/таблице	line_name

attr_full_db_name	varchar	Название поля в БД с указанием таблицы (алиаса) для расширенного поиска. Поле обязательно должно быть заполнено, иначе не будет работать сортировка по колонке (появится ошибка ERR во всех строках грида)	l.name
attr_quick_filter	bool	Доступность поля для поиска в быстром фильтре QUICK_FILTER	
attr_search_str	varchar	Информация для поиска по атрибуту???	
attr_search_component	json	Описание компонента, который будет использоваться для построения сложного запроса по этому полю	

5. Атрибуты представления s_core.s_view_attribute

Атрибуты представления предназначены для настройки отображения данных.

Название	Тип данных	Комментарий	Пример заполнения для класса
id_view	int4	ИД системного представления	
id_attribute	int4	ИД атрибута	
view_attr_name	varchar	Системное наименование атрибута представления	
view_attr_caption	varchar	UI наименование атрибута представления	
view_attr_description	varchar	Описание атрибута представления	
view_attr_visible	bool	Видимость при отображении. true - видимый, false или null - невидимый	
view_attr_width	int2	Ширина колонки при отображении в таблице	
id_view_attr_format	varchar	Формат отображения данных. Формат даты надо писать НЕ сюда, а в поле view_attr_style	
filter_enable	bool	Доступен поиск по столбцу при отображении таблицы	

view_attr_editor	text	Ссылка на редактор. Если не пусто, то поле доступно для редактирования через таблицу.	
view_attr_filter_options	json	Настройки для фильтра в таблице. Указание, из какого справочника брать данные, какие поля отображать, какие ключи использовать. Просто перечисление списка значений с ключами. Ограничение значений справочника.	
view_attr_ord	int4	Порядковый номер при отображении в таблице	
view_attr_sort_ord	int2	Порядковый номер при сортировке	
view_attr_sort_mode	int2	Режим сортировки (1- ASC NULL FIRST, 2- ASC NULL LAST, 3- DESC NULL FIRST, 4- NULL LAST)	
horizontal_alignment	int2	Горизонтальное выравнивание в ячейке при отображении (1-left, 2-center,3-right)	
wrap_text	bool	Переносить по строкам. true - переносить, false или null- не переносить	
view_attr_style	json	Стиль отображения ячейки, например, формат отображения даты/timestamp. Чтобы указанный формат работал, обязательно должно быть заполнено поле grid_column_filter .	{"dateFormat":"DD.MM.YYYY"}
vertical_alignment	int2	Вертикальное выравнивание при отображении (1-top, 2-center, 3-bottom)	

grid_column_filter	varchar	Тип фильтра для столбца при отображении в таблице. Поле должно быть заполнено, чтобы применялся формат, указанный в view_attr_style	agDateColumnFilter
view_attr_case_sensitive	bool	Регистрозависимость при поиске (true- учитывать регистр, false - не учитывать)	
fix_column	bool	Зафиксированный столбец, порядок определяется по view_attr_ord	
auto_height	bool	Автоподбор высоты по содержимому	
icon_name	varchar	Наименование иконки	

6. Частные настройки

Настройка именованного списка. В которых состав значений не расширяем и соответствует цифровым индексам.

Откорректированный код SQL, ранее выводивший значения непосредственно в столбцах, но не позволяющих фильтровать :

```
-- В SQL
-- БЫЛО
case
when (l.powered_by_akb = 0 ) then 'Нет'
when (l.powered_by_akb = 1 ) then 'Да'
when (l.powered_by_akb =-1 ) then 'Не указано'
end as powered_by_akb_name
-- стало
l.powered_by_akb as powered_by_akb
```

Настройка отображения и справочника

```
s_core.s_attribute.attr_full_db_name = l.powered_by_ak
s_core.s_attribute.id_data_type = 3
s_core.s_attribute.seach_enable = true
```

```
id_data_type = 2 (2x битное целое)
s_view_attribute.grid_column_filter = agSetColumnFilter
s_view_attribute.view_attr_style = {
  "mapping": {
    "replaceValue": true,
    "filter": {
      "-1": {"caption": "Не указано"},

```

```

    "0": {"caption": "Нет"},
    "1": {"caption": "Да"}
  },
  "cell": {
    "-1": {"caption": "Не указано"},
    "0": {"caption": "Нет"},
    "1": {"caption": "Да"}
  }
}
}
}

```

7. Настройки для вывода значений из справочника, и фильтрации по ним

В `s_core.s_class` - создается класс для таблицы справочника, или может использоваться ранее созданный класс:

Например, для типа рубежа используется класс `RefComplexType`, для него определяется источник в виде таблицы `ref_complex_type` и атрибуты (см. ниже).

В запросе указывается связь между рубежом и типом:

```
left outer join public.ref_complex_type rct on l.line_type_sid = rct.sid
```

А для идентификации типа конкретного рубежа в атрибуте, возвращается:

```
rct.id
```

По нему заводится атрибут `s_core.s_attribute`

А для идентификации типа конкретного рубежа в атрибуте, возвращается:

```

attr_name           =                               typeLineId
attr_db_name        =                               type_line_id
attr_full_db_name   =                               rct.id
attr_search_component = {"filterType": "agNumberColumnFilter"}

```

По нему заводится атрибут `s_core.s_view_attribute` :

```

view_attr_name = typeLineId
attr_db_name = line_type_id
attr_full_db_name = rct.id
view_attr_style =
{
  "refMapping": {
    "replaceValue": true,
    "refName": "RefComplexType",

```

```
"keyField": "id",
"filter": {
  "fieldList": ["name"],
  "fieldSeparator": ", ",
  "iconField": "iconName",
  "iconWidth": 20,
  "iconPosition": "left"
},
"cell": {
  "fieldList": ["name"],
  "fieldSeparator": ", ",
  "iconField": "iconName",
  "iconWidth": 20,
  "iconPosition": "left"
}
}
```

Где:

RefComplexType - тип класса справочника,

id - атрибут RefComplexType по которому осуществляется выборка элемента из справочника,

name - атрибут RefComplexType по которому осуществляется отображение элемента из справочника,

iconName - атрибут RefComplexType по которому отображается иконка в ячейке элемента.

8. Настройки кнопки выгрузки в Excel

В таблице `s_core.s_operation` - заданы различные операции, в числе которых с `id = 4` есть операция выгрузки данных в Excel (`exportData`).

Для того чтобы эта операция была доступна для настраиваемого грида (например, класс 184, вью 123), необходимо добавить запись в таблицу `s_core.s_object_operation`

у которой `id_operation = 4`, `id_object = 123` идентификатор `view` - для которой подключается операция, `id_class = 11` (системное представление `View`).

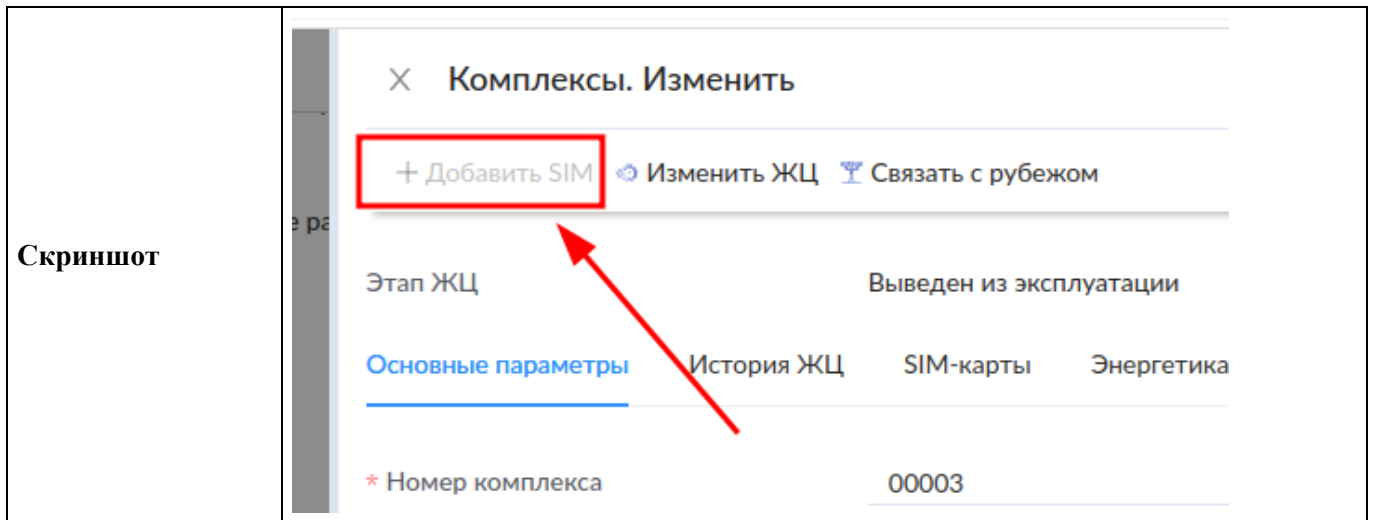
5.10 Настройка кнопок таблиц

Описание возможностей действий для кнопок в toolbar.

Описание оформления предназначено для столбца `s_core.s_object_operation.operation_params`.

1. Блокировка кнопки

Описание	Позволяет выполнить проверку на доступность кнопки и описать тип ее отключения
Структура	<pre>{ "btnShowCondition": CONDITION, "btnBlockingMode": BLOCK_TYPE }</pre> <p>где:</p> <ul style="list-style-type: none"> • CONDITION - Условие в виде строки, разрешающее отобразить кнопку. В условии используются наименование атрибутов класса (<code>s_attributes.attr_name</code>), который используется для формы или представления где отображается кнопка. Сами атрибуты необходимо использовать в фигурных скобках • BLOCK_TYPE - Тип блокировки кнопки, если условия отображения не соблюдаются, может принимать значения: <ul style="list-style-type: none"> ○ <i>invisible</i> ○ <i>disabled</i>
Пример <code>operation_params</code>	<pre>{ "btnShowCondition": "{objectNomenclature}!=null", "btnBlockingMode": "disabled", "formName": "objectComplexSimLink", "closeFormAfterSave": true, "formType": "modal", "formHeight": "auto", "keyField": [{ "dataType": "int4", "field": "id_object", "paramName": "objectId", "table": "public.object_sim_history", "value": "id.value" }, { "dataType": "int4", "field": "id_object_category", "paramName": "objectId", "table": "public.object_sim_history", "value": "objectCategory.value" }] }</pre>



2. Вызов функций

Описание	Позволяет по нажатию кнопки вызывать несколько функций в определенном порядке
----------	---

Структура	<pre>{ "requestQueue": { "beforeAllRows": [FUNCTION_1_PARAM, FUNCTION_2_PARAM, ...], "everyRow": [FUNCTION_1_PARAM, FUNCTION_2_PARAM, ...], "afterAllRows": [FUNCTION_1_PARAM, FUNCTION_2_PARAM, ...] } }</pre> <p>где:</p> <ul style="list-style-type: none">○ FUNCTION_N_PARAM - описание параметров вызываемой функции в виде словаря:○ {○ "mutation": "MUTATION",○ "params": [○ {○ "PARAM": "VALUE",○ ...○ },○ ...○]○ } <p>где:</p> <ul style="list-style-type: none">▪ MUTATION - структура вызываемой функции, список функций можно посмотреть здесь: https://inv-backend.k.mvs.group/altair (Раздел Docs)▪ PARAM - параметр функции. Наименования параметров зависят от той или иной функции▪ VALUE - значение параметра. Можно использовать как статичные значения, так и значения из атрибутов класса, привязанного к объекту кнопки (форма или представление) <p>Последовательность выполнения функций:</p> <ol style="list-style-type: none">1) beforeAllRows2) everyRow3) afterAllRows
-----------	---

**Пример
operation_params**

```

{
  "btnShowCondition": "{complexModel}=== 'Скат' ||
{complexModel}=== 'Скат-С' || {complexModel}=== 'Скат СР'
||{complexModel}=== 'Скат ПСР' || {complexModel}=== 'RNcam'",
  "btnBlockingMode": "invisible",
  "requestQueue": {
    "beforeAllRows": [
      {
        "mutation": "mutation formSave($params:
[FormSaveQueryInput]!, $callFunction: [FormSaveCallFunctionInput])
{\n formSave(query: $params, callFunction: $callFunction) {\n
id\n table\n key\n __typename\n }\n\n",
        "params": [
          {
            "field": "id_complex",
            "dataType": "int4",
            "table": "public.object_erd_use_history",
            "value": "id.value"
          },
          {
            "field": "operation_name",
            "dataType": "varchar",
            "table": "public.object_erd_use_history",
            "value": "erdResetComplex"
          }
        ]
      },
      {
        "mutation": "mutation erd ($params:
[NamedAndTypedParamInput]!) {erd {customCommand(params:
$params)}}",
        "params": [
          {
            "name": "ip",
            "value": "ip.value",
            "dataType": "varchar"
          },
          {
            "name": "protocol",
            "value": "udp",
            "dataType": "varchar"
          },
          {
            "name": "port",
            "value": "10161",
            "dataType": "int"
          },
          {
            "name": "community",
            "value": "public",
            "dataType": "varchar"
          },
          {
            "name": "oid",
            "value": "1.3.6.1.4.1.40418.2.6.2.2.1.3.1.3",
            "dataType": "varchar"
          },
          {
            "name": "command",
            "value": "2",
            "dataType": "int"
          }
        ]
      }
    ]
  }
}

```


5.11 Настройка стилей оформления таблиц

Описание возможностей оформления столбцов на примерах с различными типами данных.

Описание оформления предназначено для столбца `s_core.s_view.view_attr_style`.

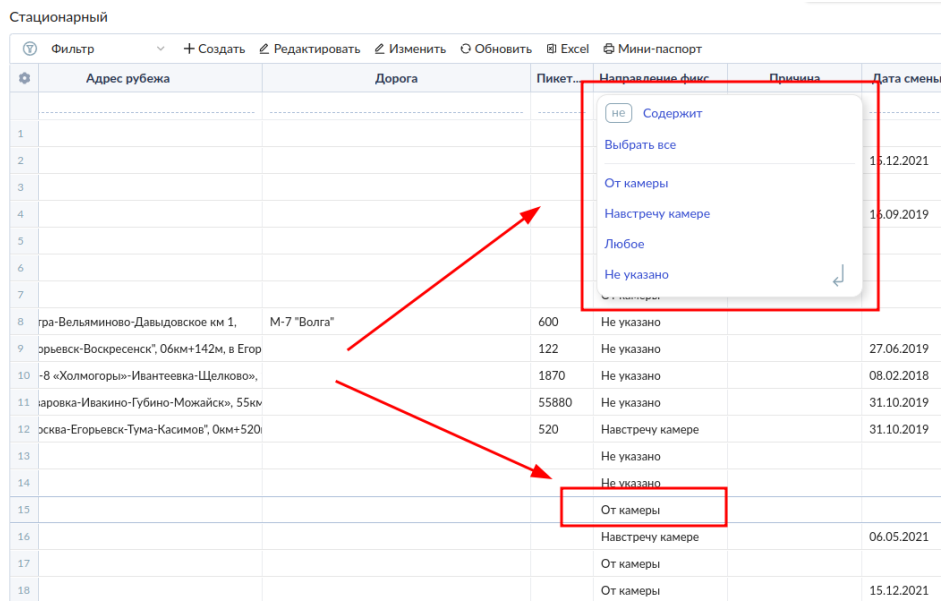
1. Гиперссылка

Описание	Позволяет создавать гиперссылки
Требования к типу данных атрибута класса	json или text
Структура	<pre>{ "cellRender": { "LINK_OBJECT_NAME": { "componentType": { "componentName": "LinkList" } } } }</pre> <p>где:</p> <ul style="list-style-type: none"> • LINK_OBJECT_NAME - Название ключа в словаре json,
Пример значения поля, полученного из класса	<pre>{"complexLink" : [{"href": "/list/view/ObjectComplex/form-view/objectComplexEdit/3888", "text": "AT0775"}]}</pre>
Пример view_attr_style	<pre>{ "cellRender": { "complexLink": { "ord": 3, "componentType": { "componentName": "LinkList" } } } }</pre>

Скриншот	Рубежи (новое)		
	Поиск <input type="text"/> + Создать ✎ Изменить ↻ Обновить ☒		
	№	Номер рубежа	Комплекс
	1	1.28	
	2	2.23	
	3	7708441	АТ0775
	4	7700606	АТ0358
	5		
	6	--04267	
	7		

2. Простой маппинг

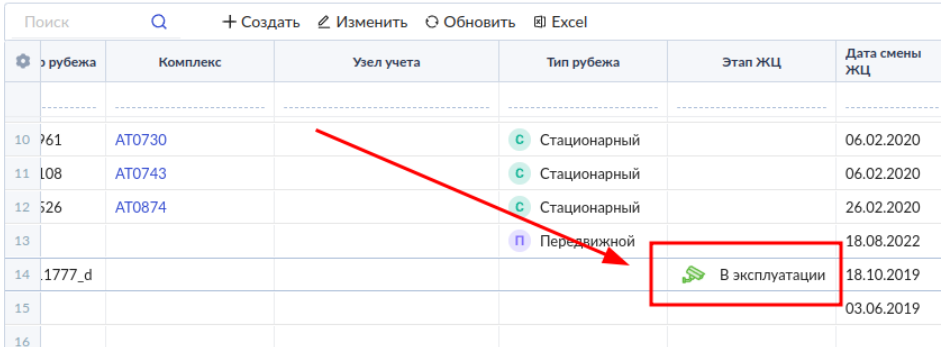
Описание	Переназначать значения по собственному маппингу
Требования к типу данных атрибута класса	Любое
Структура	<pre> { "mapping": { "replaceValue": true, "filter": { "VALUE_1": {"caption": "CAPTION_FILTER_1"}, "VALUE_2": {"caption": "CAPTION_FILTER_2"}, ... }, "cell": { "VALUE_1": {"caption": "CAPTION_CELL_1"}, "VALUE_2": {"caption": "CAPTION_CELL_2"}, ... } } } </pre> <p>где:</p> <ul style="list-style-type: none"> • VALUE_X - Значение столбца, полученное из класса • CAPTION_CELL_X - Новое значение столбца, отображаемое в столбце грида • CAPTION_FILTER_X - Новое значение столбца, отображаемое в фильтре столбца

Пример значения поля, полученного из класса	0
Пример view_attr_style	<pre>{ "mapping": { "replaceValue": true, "filter": { "null": {"caption": "Не указано"}, "0": {"caption": "От камеры"}, "1": {"caption": "Навстречу камере"}, "2": {"caption": "Любое"} } }, "cell": { "null": {"caption": "Не указано"}, "0": {"caption": "От камеры"}, "1": {"caption": "Навстречу камере"}, "2": {"caption": "Любое"} } }</pre>
Скриншот	 <p>Скриншот интерфейса приложения, отображающего таблицу данных. Таблица имеет следующие столбцы: Адрес рубежа, Дорога, Пикет, Направление фикс, Приёмка, Дата смены. В столбце 'Направление фикс' открыто выпадающее меню с вариантами: 'Не указано', 'От камеры', 'Навстречу камере', 'Любое', 'Не указано'. Красные стрелки указывают на соответствие значений в таблице и в меню.</p>

3. Значение справочника

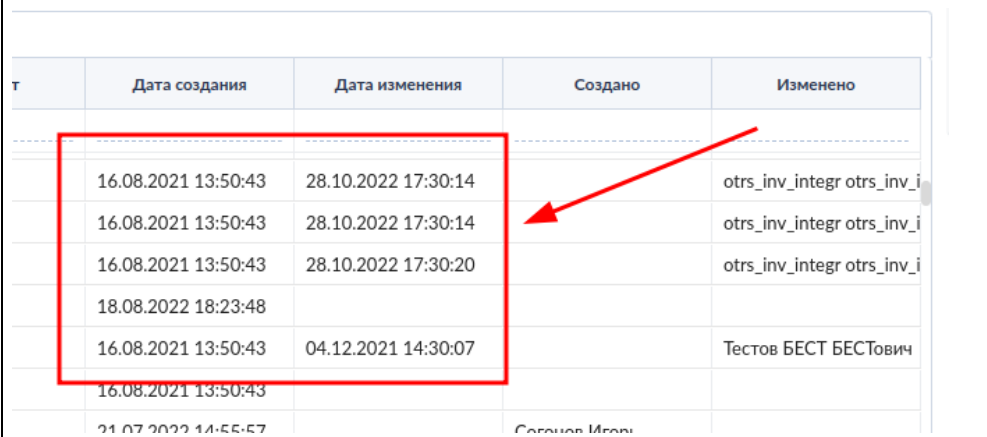
Описание	Выводить значения справочника по ключу
Требования к типу данных атрибута класса	<ul style="list-style-type: none"> • int4 • int2 • numeric • varchar

<p>Структура</p>	<pre> { "refMapping": { "replaceValue": true, "refName": "REF_NAME", "keyField": "KEY_FIELD", "cell": { "fieldList": ["NAME_FIELD"], "fieldSeparator": "SEPARATOR", "iconField": "ICON_FIELD", "iconWidth": ICON_WIDTH, "iconPosition": "ICON_POSITION" }, "filter": { "fieldList": ["NAME_FIELD"], "fieldSeparator": "SEPARATOR", "iconField": "ICON_FIELD", "iconWidth": ICON_WIDTH, "iconPosition": "ICON_POSITION" } } } </pre> <p>где:</p> <ul style="list-style-type: none"> • REF_NAME - Название класса-справочника • KEY_FIELD - ключевое поле справочника, по которому необходимо смаппить значение из столбца • NAME_FIELD - отображаемые значение из справочника, в виде списка атрибутов справочника. В большинстве случаев используется только один элемент списка атрибутов • SEPARATOR - разделитель между значениями атрибутов справочника в виде подстроки, работает только если в fieldList описано больше 1-го атрибута • ICON_FIELD - поле справочника, содержащее название иконки, указывается для добавления к атрибуту из справочника иконки • ICON_WIDTH - размер иконки в px , работает при наличии параметра iconField • ICON_POSITION - позиция иконки (left, right), работает при наличии параметра iconField
<p>Пример значения поля, полученного из класса</p>	<p>3</p>

Пример view_attr_style	<pre>{ "refMapping": { "replaceValue": true, "refName": "RefStageOperational", "keyField": "id", "cell": { "fieldList": ["name"], "fieldSeparator": ", ", "iconField": "iconName", "iconWidth": 30, "iconPosition": "left" }, "filter": { "fieldList": ["name"], "fieldSeparator": ", ", "iconField": "iconName", "iconWidth": 30, "iconPosition": "left" } } }</pre>																																																
Скриншот	<p>Рубежи (новое)</p>  <table border="1"> <thead> <tr> <th>рубежа</th> <th>Комплекс</th> <th>Узел учета</th> <th>Тип рубежа</th> <th>Этап ЖЦ</th> <th>Дата смены ЖЦ</th> </tr> </thead> <tbody> <tr> <td>10 761</td> <td>AT0730</td> <td></td> <td>Стационарный</td> <td></td> <td>06.02.2020</td> </tr> <tr> <td>11 108</td> <td>AT0743</td> <td></td> <td>Стационарный</td> <td></td> <td>06.02.2020</td> </tr> <tr> <td>12 526</td> <td>AT0874</td> <td></td> <td>Стационарный</td> <td></td> <td>26.02.2020</td> </tr> <tr> <td>13</td> <td></td> <td></td> <td>Передвижной</td> <td></td> <td>18.08.2022</td> </tr> <tr> <td>14 1777_d</td> <td></td> <td></td> <td></td> <td>В эксплуатации</td> <td>18.10.2019</td> </tr> <tr> <td>15</td> <td></td> <td></td> <td></td> <td></td> <td>03.06.2019</td> </tr> <tr> <td>16</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	рубежа	Комплекс	Узел учета	Тип рубежа	Этап ЖЦ	Дата смены ЖЦ	10 761	AT0730		Стационарный		06.02.2020	11 108	AT0743		Стационарный		06.02.2020	12 526	AT0874		Стационарный		26.02.2020	13			Передвижной		18.08.2022	14 1777_d				В эксплуатации	18.10.2019	15					03.06.2019	16					
рубежа	Комплекс	Узел учета	Тип рубежа	Этап ЖЦ	Дата смены ЖЦ																																												
10 761	AT0730		Стационарный		06.02.2020																																												
11 108	AT0743		Стационарный		06.02.2020																																												
12 526	AT0874		Стационарный		26.02.2020																																												
13			Передвижной		18.08.2022																																												
14 1777_d				В эксплуатации	18.10.2019																																												
15					03.06.2019																																												
16																																																	

4. Дата

Описание	Рендер формата даты
Требования к типу данных атрибута класса	<ul style="list-style-type: none"> • timestamp • timestamptz • date

<p>Структура</p>	<pre>{ "dateFormat": "DATE_FORMAT" }</pre> <p>где:</p> <ul style="list-style-type: none"> • DATE_FORMAT - формат даты, где: <ul style="list-style-type: none"> ○ DD - день (01-31) ○ MM - месяц (01-12) ○ YYYYYY - год полный ○ HH - часы (00-23) ○ mm - минуты (00-59) ○ ss - секунды (00-59) 																																								
<p>Пример значения поля, полученного из класса</p>	<p>2020-02-06 17:07:00.694</p>																																								
<p>Пример view_attr_style</p>	<pre>{"dateFormat": "DD.MM.YYYY HH:mm:ss" }</pre>																																								
<p>Скриншот</p>	 <p>The screenshot shows a table with the following columns: ID, Дата создания, Дата изменения, Создано, and Изменено. The data rows are as follows:</p> <table border="1"> <thead> <tr> <th>ID</th> <th>Дата создания</th> <th>Дата изменения</th> <th>Создано</th> <th>Изменено</th> </tr> </thead> <tbody> <tr> <td></td> <td>16.08.2021 13:50:43</td> <td>28.10.2022 17:30:14</td> <td></td> <td>otrs_inv_integr otrs_inv_i</td> </tr> <tr> <td></td> <td>16.08.2021 13:50:43</td> <td>28.10.2022 17:30:14</td> <td></td> <td>otrs_inv_integr otrs_inv_i</td> </tr> <tr> <td></td> <td>16.08.2021 13:50:43</td> <td>28.10.2022 17:30:20</td> <td></td> <td>otrs_inv_integr otrs_inv_i</td> </tr> <tr> <td></td> <td>18.08.2022 18:23:48</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>16.08.2021 13:50:43</td> <td>04.12.2021 14:30:07</td> <td></td> <td>Тестов БЕСТ БЕСТович</td> </tr> <tr> <td></td> <td>16.08.2021 13:50:43</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td>21.07.2022 14:55:57</td> <td></td> <td>Сергей Иван</td> <td></td> </tr> </tbody> </table>	ID	Дата создания	Дата изменения	Создано	Изменено		16.08.2021 13:50:43	28.10.2022 17:30:14		otrs_inv_integr otrs_inv_i		16.08.2021 13:50:43	28.10.2022 17:30:14		otrs_inv_integr otrs_inv_i		16.08.2021 13:50:43	28.10.2022 17:30:20		otrs_inv_integr otrs_inv_i		18.08.2022 18:23:48					16.08.2021 13:50:43	04.12.2021 14:30:07		Тестов БЕСТ БЕСТович		16.08.2021 13:50:43					21.07.2022 14:55:57		Сергей Иван	
ID	Дата создания	Дата изменения	Создано	Изменено																																					
	16.08.2021 13:50:43	28.10.2022 17:30:14		otrs_inv_integr otrs_inv_i																																					
	16.08.2021 13:50:43	28.10.2022 17:30:14		otrs_inv_integr otrs_inv_i																																					
	16.08.2021 13:50:43	28.10.2022 17:30:20		otrs_inv_integr otrs_inv_i																																					
	18.08.2022 18:23:48																																								
	16.08.2021 13:50:43	04.12.2021 14:30:07		Тестов БЕСТ БЕСТович																																					
	16.08.2021 13:50:43																																								
	21.07.2022 14:55:57		Сергей Иван																																						

5. Просмотр изображений

<p>Описание</p>	<p>Рендер формата даты</p>
<p>Требования к типу данных атрибута класса</p>	<ul style="list-style-type: none"> • timestamp • timestamptz • date

<p>Структура</p>	<pre> { "cellRender":{ "LINK_OBJECT_NAME": { "ord": ICON_ORDER, "type": "button", "componentType": { "componentName": "ViewFile", "iconStyle": {"color": "ICON_COLOR"}, "operation": { "operationIcon": "ICON_NAME", "operationDescription": "DESCRIPTION", "operationParams": { "method": "facadePhoto" } } } } } } </pre> <p>где:</p> <ul style="list-style-type: none"> • LINK_OBJECT_NAME - имя ключа из справочника json, полученного из атрибута • ICON_ORDER - порядок иконки, работает если LINK_OBJECT_NAME указано несколько • ICON_COLOR - цвет иконки, формат CSS color • ICON_NAME - название иконки • DESCRIPTION - описание
<p>Пример значения поля, полученного из класса</p>	<pre> { "greenIcon" : { "data" : {"complexName" : "AS5000237", "photoType" : "TRANSIT"}, "formTitle" : "AS5000237. Фото проезда"}, "orangeIcon" : { "data" : {"complexName" : "AS5000237", "photoType" : "TRANSIT_OVERVIEW_VIOLATION_TAG"}, "formTitle" : "AS5000237. Обзорное фото" }, "redIcon" : { "data" : {"complexName" : "AS5000237", "photoType" : "TRANSIT_LAST_VIOLATION"}, "formTitle" : "AS5000237. Фото последнего нарушения" } } </pre>

Пример view_attr_style

```
{
  "cellRender":{
    "greenIcon": {
      "ord": 1,
      "type": "button",
      "componentType": {
        "componentName": "ViewFile",
        "iconStyle": {"color": "green"},
        "operation": {
          "operationIcon": "camera",
          "operationDescription": "Последнее фото
комплекса",
          "operationParams": {
            "method": "facadePhoto"
          }
        }
      }
    },
    "orangeIcon": {
      "ord": 2,
      "type": "button",
      "componentType": {
        "componentName": "ViewFile",
        "iconStyle": {"color": "orange"},
        "operation": {
          "operationIcon": "camera",
          "operationDescription": "Фото признака нарушения
на комплексе",
          "operationParams": {
            "method": "facadePhoto"
          }
        }
      }
    },
    "redIcon": {
      "ord": 3,
      "type": "button",
      "componentType": {
        "componentName": "ViewFile",
        "iconStyle": {"color": "red"},
        "operation": {
          "operationIcon": "camera",
          "operationDescription": "Фото последнего нарушения
на комплексе",
          "operationParams": {
            "method": "facadePhoto"
          }
        }
      }
    }
  }
}
```

Скриншот

Комплексы

Фильтр + Создать Редактировать Изменить Обновить Excel

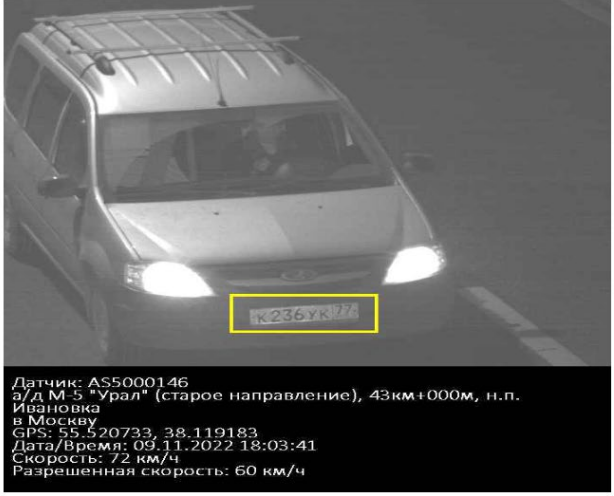
№	Номер комплекса	Фото	Тип комплекса	M
1	AS5000146		С Стационарный	A
2	AS5000610-AS5000265		С Стационарный	АС
3	AS5000020		С Стационарный	A
4	AS5000479		С Стационарный	A
5	AS5000101		С Стационарный	A
6	AS5000110		С Стационарный	A
7	AS5000086		С Стационарный	A
8	AS5000064		С Стационарный	A
9	13182		Б Безопасный регион	Б
10	15504		Б Безопасный регион	Б
11	31163		Б Безопасный регион	Б

Комплексы

Фильтр + Создать Редактировать Изменить Обновить Excel Мини-паспорт

№	Номер комплекса	Фото	Тип комплекса	Модель комплекса	Этап
1	AS5000146				
2	AS5000610-AS5000265				
3	AS5000020				
4	AS5000479				
5	AS5000101				
6	AS5000110				
7	AS5000086				
8	AS5000064				
9	13182				
10	15504				
11	31163				
12	18383				
13	13313				
14	15542				
15	17769				
16	17773				
17	AS5000085-AS5000904				
18	AS5000027				
19	AS5000337				
20	F250		С Стационарный	Ф Форсаж	Э Эксплуатация
21	F212		С Стационарный	Ф Форсаж	Э Эксплуатация

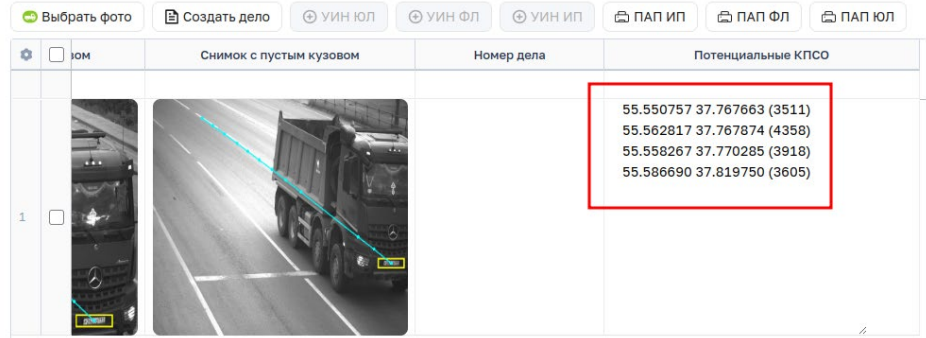
AS5000146. Фото проезда



Датчик: AS5000146
 а/д М-5 "Урал" (старое направление), 43км+000м, н.п. Ивановка в Москву
 GPS: 55.520733, 38.119183
 Дата/Время: 09.11.2022 18:03:41
 Скорость: 72 км/ч
 Разрешенная скорость: 60 км/ч

6. Мультистрочный текст

Описание	Текст, с наличием символов перехода на новую строку
Требования к типу данных атрибута класса	<ul style="list-style-type: none"> text

<p>Структура</p>	<pre> { "cellRender": { "TMP_ATTR_NAME": { "componentType": { "componentName": "TextArea", "style": { "height": "HEIGHT_BLOCK", "color": "#000" }, "disabled": true } } } } </pre> <p>где:</p> <ul style="list-style-type: none"> • TMP_ATTR_NAME - временное наименование атрибута из сформированного json, которое приходит как значение поля атрибута (пример значения ниже) • HEIGHT_BLOCK - высота области текста. Если текста будет больше заданной высоты - появится полоса прокрутки, поэтому желательно заранее знать максимальную высоту строки грида, где будет отображаться значение
<p>Пример значения поля, полученного из класса</p>	<pre> {"info" : "55.550757 37.767663 (3511)\n55.562817 37.767874 (4358)\n55.558267 37.770285 (3918)\n55.586690 37.819750 (3605)"} </pre>
<p>Пример view_attr_style</p>	<pre> { "cellRender": { "info": { "componentType": { "componentName": "TextArea", "style": { "height": "250px", "color": "#000" }, "disabled": true } } } } </pre>
<p>Скриншот</p>	 <p>The screenshot shows a web application interface with a table. The table has columns: 'Снимок с пустым кузовом', 'Номер дела', and 'Потенциальные КПСО'. A red box highlights the 'Потенциальные КПСО' column, which contains the following text:</p> <pre> 55.550757 37.767663 (3511) 55.562817 37.767874 (4358) 55.558267 37.770285 (3918) 55.586690 37.819750 (3605) </pre>

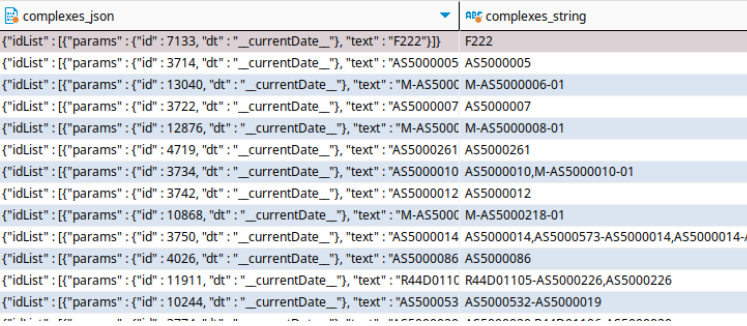
5.12 Настройка «сложной» сортировки

Сортировка по столбцу CoreLinkList

Для того чтобы работала сортировка по столбцу **CoreLinkList** необходимо правильно настроить:

- описание атрибута в классе;
- описание db_name и full_db_name для атрибута;
- описание стиля отображения колонки в view_attribute.

Рассмотрим конкретный пример – форма "Редактирование ДТП", грид "Верификация привязок: Рубежи" - сортировка по колонке "Комплексы"

Краткое описание настройки	Пример настройки	Примечания к настройке
<p>Класс AmDtpLine Link</p>	<pre>WITH complexes AS (SELECT vc.line_id ,json_build_object('idList', json_agg(json_build_object('params', json_build_object('id' , vc.id, 'dt' , '__currentDate__'), 'text', vc.name)) AS complexes_json -- Комплексы, в формате json ,string_agg(vc .name, ',') AS complexes_string -- Комплексы, в формате строки</pre>	<p>Важно чтобы в конечной выборке у нас было два типа колонок по одной логической:</p> <ul style="list-style-type: none"> • complexes_json - комплексы в формате json, для дальнейшего рендера в виде ссылок в таблице • complexes_string - комплексы в формате строки, для выполнения самой сортировки <p>Также важно отметить, что столбец "complexes_string" выводить в конечном запросе НЕОБЯЗАТЕЛЬНО, главное, чтобы он был доступен</p> <p>Выглядят оба столбца так:</p>  <p>Требования к json: массив (даже если элемент один) из объектов, у которых должны быть следующие ключи:</p> <ul style="list-style-type: none"> • params <ul style="list-style-type: none"> ○ id - ИД, который нужно использовать в датасете формы, которая будет открываться по ЛКМ на объекте.

```

FROM
ambdd.v_complex vc
GROUP BY
vc.line_id
)
SELECT
    dl.id
    ,dl.dtp_id
    ,dl.line_id
    ,dl.is_deleted
    ,true as
status
    ,json_build_ob
ject(
        'idList',
json_build_array(
        json_build_obj
ect(
            'params',
json_build_object('id'
, dl.line_id, 'dt',
'__currentDate__'),
            'text', l.name
        )
        ) as line_name
    ,c.complexes_j
son AS complex_name
    ,json_build_ob
ject(
        'lineCoords',
json_build_array(
        json_build_obj
ect(
            'href',
(SELECT param_value
FROM s_core.s_params
WHERE param_name =
'ambdd_iframe_prefix')
|| '/map/lines/' ||
l."name" ,
            'text',
round(l.lat::numeric,
6)::TEXT || ', ' ||
round(l.lon::numeric,
6)::TEXT
        )
    )

```

Описание датасета и формы
выполняется во вью-атрибуте

- o text - наименование объекта, которое
будет отображаться в виде ссылки

	<pre>) AS line_coords , l.road_km , l.address FROM ambdd.dtp_line dl LEFT JOIN public.line l ON l.sid = dl.line_id LEFT JOIN complexes c ON c.line_id = dl.line_id WHERE dl.is_deleted IS NOT TRUE /* AND dtp_id = {\$id_dtp\$} */ </pre>	
атрибут complexName	<p>Указываем для атрибута следующие параметры:</p> <ul style="list-style-type: none"> • attr_name - complexName • attr_caption - Комплекс • id_data_type - 6 • attr_sortable - true • attr_db_name - complex_name • attr_full_db_name - c.complexes_string • attr_search_component - {"filterType": "agTextColumnFilter"} 	<p>Здесь описываем атрибуты:</p> <ul style="list-style-type: none"> • attr_db_name - нужно указать именно АЛИАС поля complexes_json в запросе класса • attr_full_db_name - а тут мы указываем то самое строковое поле c.complexes_string, которое доступно, но не выведено в список столбцов в запросе • id_data_type - указываем также json, т.к. этот тип дальше уже используется вьюхой

вью-атрибут complexName	Указываем для атрибута следующие параметры: <ul style="list-style-type: none"> • view_attr_name - complexName • view_attr_caption - Комплекс • view_attr_style: <ul style="list-style-type: none"> • { • "cellRender": <ul style="list-style-type: none"> • { • "idList": { • "ord": 1, • "componentType": { • "componentName": "CoreLinkList", • "formName": "fmComplexCard", • "dataSet": "dsComplex", • "mode": "modal" • } • } • } 	Для вью атрибута описываем стиль отображения, то есть как нужно выполнить рендер того json, который приходит в атрибуте: <ul style="list-style-type: none"> • cellRender - наименование типа рендера, всегда такое • idList - наименование ключа в json, который мы формируем • componentName - CoreLinkList, всегда такое, означает переход по ссылкам внутри платформы • formName - наименование формы, которую нужно открыть. форма должна быть описана в form_struct • dataSet - наименование главного датасета в этой форме, в который будет передаваться id из каждого объекта json атрибута • mode - тип формы, модальная или выезжающая, стандартное свойство
----------------------------	---	---

5.13 Настройка отчета с параметрами

5.13.1 Настройка меню

Настройка вызова отчета из меню s_core.s_arm

- **item_params** = /report;
- **id_class** = 43 - системный класс;
- **id_object** = <номер класса формы>.

5.13.2 Настройка s_core.s_form

Пример настройки интервала дат и кнопки формирования отчета:

Заполняется поле s_core.s_form.form_content

Пример

```
{
  "dtStart": {
    "ord": 1,
```

```

    "type": "string",
    "dataType": "datetime",
    "label": "Начало периода",
    "componentType": {
      "componentName": "DatePicker"
    }
  },
  "dtEnd": {
    "ord": 2,
    "type": "string",
    "dataType": "datetime",
    "label": "Окончание периода",
    "componentType": {
      "componentName": "DatePicker"
    }
  },
  "btnBuildReport": {
    "ord": 3,
    "componentType": {
      "componentName": "Button",
      "action": "callDbProc",
      "procName": "public.build_report_loses_GUPS"
    },
    "label": "Сформировать"
  }
}

```

"procName": "public.build_report_loses_GUPS" - определяет формирующую отчет процедуру.

5.13.3 Правила блоков в процедуре

Пример состава процедуры:

```

create function build_report_loses_gups(in_params text DEFAULT '{}'::text) returns
json
  strict
  security definer
  language plpgsql
as
$$
DECLARE
  v_dt_start timestamp DEFAULT in_params::json ->> 'dt_start'; -- Блок параметров

begin
  IF v_dt_end IS NULL THEN
    v_dt_end = now();
  END IF;

  IF v_dt_start IS NULL THEN
    v_dt_start = v_dt_end - '14 days'::interval;
  END IF;

  -- состав панели инструментов
  SELECT jsonb_agg(json_build_object(
    'operationOrd', soo.operation_ord,
    'operationName', so.operation_name,
    'operationCaption', so.operation_caption,
    'operationDescription', so.operation_description,

```

```
'operationMethod', so.operation_method,
'operationIcon', so.operation_icon,
'operationParams', soo.operation_params,
'id', soo.id)
FROM public.s_object_operation soo JOIN public.s_operation so ON soo.id_operation
= so.id WHERE id_class = 43 AND id_object = 5 INTO v_toolbar; -- TODO id_object =
72

-- перечень столбцов
v_grid_option = jsonb_build_array(
    json_build_object('viewAttrOrd', 1, 'viewAttrName', 'model',
'attrDbName', 'model', 'viewAttrCaption', 'Модель', 'dataTypeName', 'string',
'viewAttrVisible', 'true', 'viewAttrWidth', '220'),
    json_build_object('viewAttrOrd', 2, 'viewAttrName', 'complex',
'attrDbName', 'complex', 'viewAttrCaption', 'Комплекс', 'dataTypeName', 'string',
'viewAttrVisible', 'true', 'viewAttrWidth', '120'),
    json_build_object('viewAttrOrd', 3, 'viewAttrName', 'stopreason',
'attrDbName', 'downtimereason', 'viewAttrCaption', 'Причина простоя',
'dataTypeName', 'string', 'viewAttrVisible', 'true', 'viewAttrWidth', '220'),
    json_build_object('viewAttrOrd', 4, 'viewAttrName',
'date_beg_raspisaniya', 'attrDbName', 'date_beg_raspisaniya', 'viewAttrCaption',
'Начало неработоспособности', 'dataTypeName', 'datetime', 'viewAttrVisible',
'true', 'viewAttrWidth', '220'),
    json_build_object('viewAttrOrd', 5, 'viewAttrName',
'date_end_raspisaniya', 'attrDbName', 'date_end_raspisaniya', 'viewAttrCaption',
'Планируемая дата выставления на рубеж', 'dataTypeName', 'datetime',
'viewAttrVisible', 'true', 'viewAttrWidth', '220'),
    json_build_object('viewAttrOrd', 6, 'viewAttrName', 'info',
'attrDbName', 'info', 'viewAttrCaption', 'Примечание', 'dataTypeName', 'string',
'viewAttrVisible', 'true', 'viewAttrWidth', '220'),
    json_build_object('viewAttrOrd', 7, 'viewAttrName', 'disab_reason',
'attrDbName', 'disab_reason', 'viewAttrCaption', 'Тип проблемы', 'dataTypeName',
'string', 'viewAttrVisible', 'true', 'viewAttrWidth', '220')
);

-- текст запроса (выборка потерь по причине СГЭ)
v_sql = '...';

v_sql = FORMAT(v_sql, v_dt_start::date::text, v_dt_end::date::text,
'DD.MM.YYYY HH24:MI:SS');

EXECUTE v_sql INTO v_data;

RETURN json_build_object(
    'dt_start', v_dt_start,
    'dt_end', v_dt_end,
    'caption', 'Заголовок отчета',
    'toolbar', v_toolbar,
    'viewTypeParams', json_build_object(
        'SimpleGrid', json_build_object(
            'attributes', v_grid_option)),
    'data', v_data,
    'sql', v_sql);

EXCEPTION
WHEN OTHERS THEN
RETURN json_build_object(
    'SQLSTATE', SQLSTATE,
    'SQLERRM', SQLERRM,
    'sql', v_sql);

END
```

\$\$;

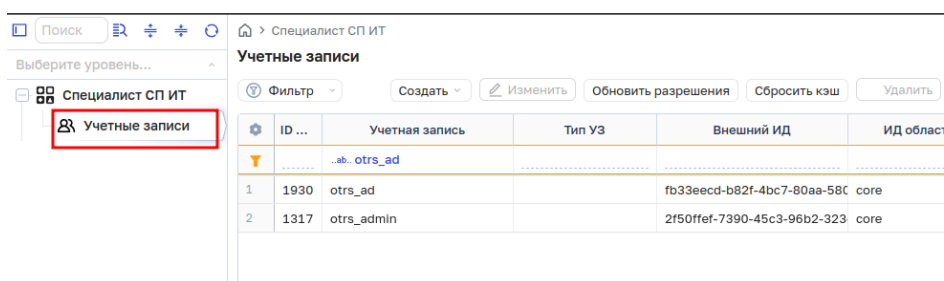
```
alter function build_report_mobcompxinoperability(text) owner to <автор>;
```

5.14 Настройка ролей

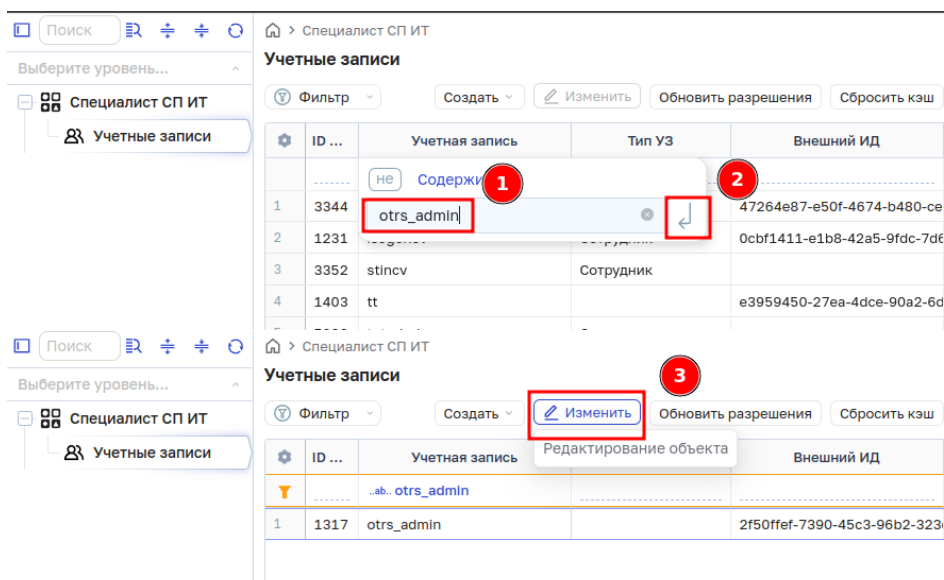
5.14.1 Назначение ролей

Для авторизации пользователя необходимо:

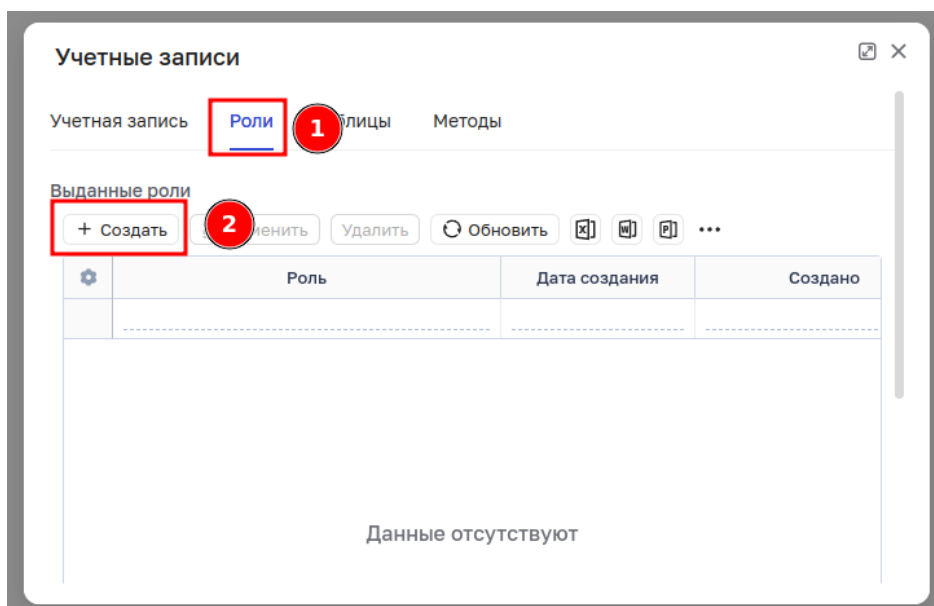
1. Перейти в раздел управления ролями: **Специалист СП ИТ → Учетные записи:**



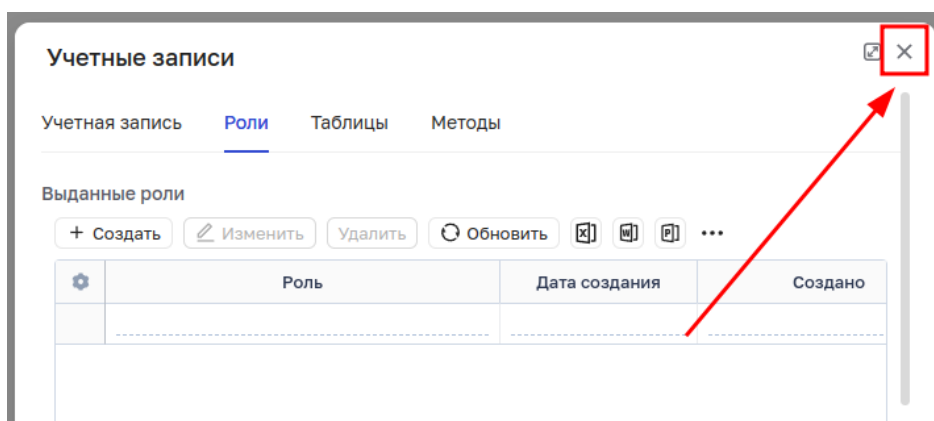
2. Ввести в поле **Учетная запись** логин нового пользователя (1), затем ЛКМ на кнопке **Применить** (2) и ЛКМ на кнопке **Изменить** (3):



3. Перейти на вкладку **Роли** (1) и добавить роли согласно таблице "Матрица ролей "подразделение".xlsx" с помощью кнопки **Создать** (2):

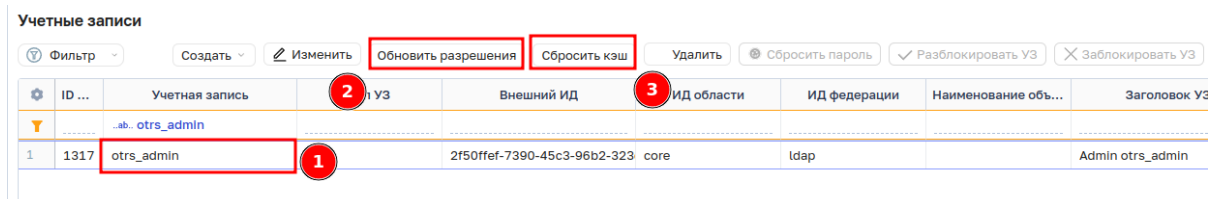


4. После завершения добавления ролей закрыть окно Учетные записи:



5. ЛКМ на новой учетной записи (1), после ЛКМ на кнопку **Обновить разрешения** (2) и в заключении ЛКМ на кнопку **Сбросить кэш** (3):

6.



5.15 Использование скриптов

Реализованы скрипты для автоматической настройки функционала:

Заголовок	Описание	Язык
Ролевая модель - обновить разрешения вручную для s_class_permission	Ручное обновление разрешений для s_class_permission	SQL
CoreDev - автоматическое формирование INSERT запросов для создания скелета раздела	Автоматическое формирование скелета раздела: представление, атрибуты класса, атрибуты представления, формы и кнопки на гриде	python
form_content - Автоматическое формирование скелета JSON	Автоматическое формирование скелета JSON для контента формы	python
form_data - Автоматическое формирование скелета JSON	Автоматическое формирование скелета JSON для данных формы	python

5.15.1 Ролевая модель - обновить разрешения вручную для s_class_permission

Описание	Ручное обновление разрешений для s_class_permission
Язык	SQL

1. Подготовить список таблиц, для которых нужно выдать разрешение.
2. Убедиться что нужные таблицы активированы в соответствующем разделе.
3. Перейти в настройки нужной роли и во вкладке **Таблицы** добавить классы **нужных** таблиц

Класс	UI наименование	Вставка	Изменение	Удаление
ObjectWashsystemCounter	ObjectWashsystemCounter	Да	Да	Да
LineStageHistory	LineStageHistory	Да	Да	Да
SourceOfPoint	SourceOfPoint	Да	Да	Да
LineOrderForm	LineOrderForm	Да	Да	Да
Complex	Комплексы	Да	Да	Да
Line	Рубежи	Да	Да	Да
ComplexStageHistory	История жизненного цикла	Да	Да	Да
ComplexMintransActHistory	Перечень актов минтранса для комплекса	Да	Да	Да

4. Добавить нужному пользователю настроенную роль
5. Выполнить скрипт обновления разрешений на классы (скрипт ниже)

где:

- 1318 - id аккаунта из таблицы s_core.s_account, в данном случае otrs_inv_integr

```

INSERT INTO s_core.s_class_permission (id_class, permission_option, class_db_name,
is_logical_delete, dt_create)
WITH res AS (
    SELECT
        sc.id id_class
        , sc.cls_schema || '.' || sc.cls_query class_db_name
        , s_core.get_class_permission(sc.id) permission_option
        , sc.is_logical_delete
    FROM s_core.s_class sc
    JOIN (
        SELECT DISTINCT id_class
        FROM s_core.s_account_class_permission
        WHERE dt_delete IS NULL
    ) acp ON sc.id = acp.id_class
    LEFT JOIN s_core.s_class_permission scp ON acp.id_class = scp.id_class
    WHERE sc.id_cls_type = 1
    AND sc.dt_delete IS NULL
    AND (
        scp.id_class IS NULL
        OR s_core.get_class_permission(sc.id)::text !=
COALESCE(scp.permission_option::text, '')
        OR COALESCE(sc.is_logical_delete, TRUE) !=
COALESCE(scp.is_logical_delete, TRUE)
        OR (sc.cls_schema || '.' || sc.cls_query)::text !=
COALESCE(scp.class_db_name, '')
    )
    AND (
        sc.id IN (
            SELECT id_class
            FROM s_core.s_account_class_permission
            WHERE id_class_accessor = (SELECT id FROM
s_core.s_class WHERE cls_name = 'SysAccount')
            AND id_object_accessor = 1318
        )
        OR
        sc.id IN (
            SELECT scp.id_class
            FROM s_core.s_account_class_permission scp
            LEFT JOIN s_core.s_permission sp ON
                sp.id_class_permission = (SELECT id FROM
s_core.s_class WHERE cls_name = 'SysArm')
            AND sp.id_object_permission =
scp.id_object_accessor
            WHERE scp.id_class_accessor = (SELECT id FROM
s_core.s_class WHERE cls_name = 'SysArm')
            AND sp.id_account = 1318
            GROUP BY scp.id_class
        )
    )
)
SELECT id_class, permission_option, class_db_name, is_logical_delete, now() FROM
res
ON CONFLICT (id_class) DO UPDATE SET permission_option =
EXCLUDED.permission_option, class_db_name = EXCLUDED.class_db_name,
is_logical_delete = EXCLUDED.is_logical_delete, dt_edit = now();

```

5.15.2 CoreDev - автоматическое формирование INSERT запросов для создания скелета раздела

Описание	Автоматическое формирование скелета раздела: представление, атрибуты класса, атрибуты представления, формы и кнопки на гриде
Язык	python

Использование модуля

Разместите скрипт для подключения к модулю в каталоге с файлами модуля из репозитория, с содержимым:

```
cd ~/coredev
touch run.py
chmod 755 run.py
mv auth.py_example auth.py
```

Добавить в файле auth.py учетные данные для работы с БД devmvs_dash:

```
auth = {
    'name': 'devmvs_dash',
    'host': 'postgres-dev.k.mvs.group',
    'port': '5432',
    'user': 'isogonov',
    'pass': '*****'
}
```

Запуск:

```
~/coredev/run.py --class CLS_NAME ИЗ ТАБЛИЦЫ S_CLASS
```

где параметры запуска:

Параметр	Описание
--class	<p>Имя класса в таблице s_class</p> <p>Требования к формату SQL:</p> <ul style="list-style-type: none"> каждое поле должно содержать алиас таблицы и самое название столбца если используется алиас поля, то алиас указывается через пробел или функцию AS положение запятой значения не имеет, но мы оформляем запятую перед полем у каждой таблицы в FROM и JOIN должно быть обязательно указано СХЕМА и НАЗВАНИЕ САМОЙ ТАБЛИЦЫ

Пример:

```

SELECT
    oc.id
    ,oc.name
    ,oc.db_table
    ,oc.dt_create
    ,oc.dt_edit
    ,sa1.account_caption AS account_creator
    ,sa2.account_caption AS account_editor
FROM public.object_category oc
LEFT JOIN s_core.s_account sa1 ON sa1.id = oc.id_creator
LEFT JOIN s_core.s_account sa2 ON sa2.id = oc.id_editor
WHERE 1=1

```

5.15.3 Автоматическое формирование скелета JSON - form_content

Описание	Автоматическое формирование скелета JSON для контента формы
Язык	python

```

#!/home/isogonov/venv/bin/python
import sys
import re
import json
import psycopg2
import psycopg2.extras
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('-table', '--table', action='store', required=True,
help="Таблица")
parser.add_argument('-view', '--view', action='store', required=True,
help="view_name")
args = parser.parse_args()

db = psycopg2.connect(dbname='devmvs_dash', user='isogonov', password='***',
host='postgres-dev.k.mvs.group', port=5432)
cursor = db.cursor(cursor_factory=psycopg2.extras.DictCursor)

q = '''SELECT sva.view_attr_name , sa.attr_db_name , sva.view_attr_caption ,
sdt.data_type_name , sva.view_attr_ord, sva.view_attr_visible
FROM s_view_attribute sva
LEFT JOIN s_attribute sa ON sa.id = sva.id_attribute
LEFT JOIN s_view sv ON sv.id = sva.id_view
LEFT JOIN s_data_type sdt ON sdt.id = sa.id_data_type
WHERE sv.view_name = '%(view_name)s'
ORDER BY sva.view_attr_ord ASC
''' % {'view_name': args.view}
cursor.execute(q)
res = cursor.fetchall()
db.close()

TPL = {
    'num': ''
    "%(view_attr_name)s": {
        "ord": %(view_attr_ord)s,
        "componentType": {

```

```
        "componentName": "InputNumber"
    },
    "type": "number",
    "table": "%(table)s",
    "field": "%(attr_db_name)s",
    "dataType": "%(data_type_name)s",
    "label": "%(view_attr_caption)s",
    "idFieldName": "id"
}, '',
    'str': ''
"%(view_attr_name)s": {
    "ord": %(view_attr_ord)s,
    "componentType": {
        "componentName": "Input"
    },
    "type": "string",
    "table": "%(table)s",
    "field": "%(attr_db_name)s",
    "dataType": "%(data_type_name)s",
    "label": "%(view_attr_caption)s",
    "idFieldName": "id"
}, '',
    'text': ''
"%(view_attr_name)s": {
    "ord": %(view_attr_ord)s,
    "componentType": {
        "componentName": "TextArea",
        "style": {
            "height": "50px"
        }
    },
    "type": "text",
    "table": "%(table)s",
    "field": "%(attr_db_name)s",
    "dataType": "%(data_type_name)s",
    "label": "%(view_attr_caption)s",
    "idFieldName": "id"
}, '',
    'checkbox': ''
"%(view_attr_name)s": {
    "ord": %(view_attr_ord)s,
    "componentType": {
        "componentName": "Checkbox"
    },
    "type": "boolean",
    "table": "%(table)s",
    "field": "%(attr_db_name)s",
    "dataType": "%(data_type_name)s",
    "label": "%(view_attr_caption)s",
    "idFieldName": "id"
}, '',
    'date': ''
"%(view_attr_name)s": {
    "ord": %(view_attr_ord)s,
    "componentType": {
        "componentName": "DatePicker"
    },
    "type": "string",
    "table": "%(table)s",
    "field": "%(attr_db_name)s",
    "dataType": "%(data_type_name)s",
    "label": "%(view_attr_caption)s",
    "idFieldName": "id"
```

```

},'',
    'id': ''
"% (view_attr_name)s": {
    "ord": %(view_attr_ord)s,
    "hide": false,
    "componentType": {
        "componentName": "Input",
        "disabled": true
    },
    "type": "number",
    "table": "% (table)s",
    "field": "% (attr_db_name)s",
    "dataType": "% (data_type_name)s",
    "label": "% (view_attr_caption)s"
},'',
    'hidden': ''
"% (view_attr_name)s": {
    "ord": %(view_attr_ord)s,
    "hide": false,
    "componentType": {
        "componentName": "Input",
        "disabled": true
    },
    "type": "number",
    "table": "% (table)s",
    "field": "% (attr_db_name)s",
    "dataType": "% (data_type_name)s",
    "label": "% (view_attr_caption)s",
    "idFieldName": "id"
},'',
}

js = ''

for r in res:
    param = dict(r)
    param['table'] = args.table
    if r['view_attr_name'] == 'id':
        tpl = 'id'
    elif r['view_attr_visible'] == False:
        tpl = 'hidden'
    elif r['data_type_name'] in ['int4', 'int2', 'int8', 'float4', 'float8',
'numeric', 'int8']:
        tpl = 'num'
    elif r['data_type_name'] in ['bool']:
        tpl = 'checkbox'
    elif r['data_type_name'] in ['text']:
        tpl = 'text'
    elif r['data_type_name'] in ['timestamp', 'timestampz', 'date']:
        tpl = 'date'
    else:
        tpl = 'str'

    js += TPL[tpl] % param

js = ''{% (js)s
"btnSave": {
    "ord": 99,
    "componentType": {
        "componentName": "Button",
        "action": "saveObjectAction"
    },
    "label": "Сохранить"
}

```

```

},
"btnCancel": {
  "ord": 100,
  "componentType": {
    "componentName": "Button",
    "action": "cancelObjectAction"
  },
  "label": "Отмена"
}
}''' % {'js': js}

print(js)

```

5.15.4 Автоматическое формирование скелета JSON - form_data

Описание	Автоматическое формирование скелета JSON для данных формы
Язык	python

```

#!/home/isogonov/venv/bin/python
import sys
import re
import json
import psycopg2
import psycopg2.extras
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('-class', '--class_name', action='store', required=True,
help="Класс - cls_name")
parser.add_argument('-view', '--view', action='store', required=True, help="Вью -
view_name")
args = parser.parse_args()

db = psycopg2.connect(dbname='devmvs_dash', user='isogonov', password='***',
host='postgres-dev.k.mvs.group', port=5432)
cursor = db.cursor(cursor_factory=psycopg2.extras.DictCursor)

q = '''SELECT sva.view_attr_name , sa.attr_name
FROM s_view_attribute sva
LEFT JOIN s_attribute sa ON sa.id = sva.id_attribute
LEFT JOIN s_view sv ON sv.id = sva.id_view
LEFT JOIN s_class sc ON sc.id = sv.id_class
WHERE sv.view_name = '%(view_name)s'
AND sc.cls_name = '%(cls_name)s'
ORDER BY sva.view_attr_ord ASC
''' % {
    'view_name': args.view,
    'cls_name': args.class_name
}
cursor.execute(q)
res = cursor.fetchall()
db.close()

js = '''{
  "%(class_name)s": ['' % {'class_name': args.class_name}

l = len(res)
for r in res:

```

```
        params = dict(r)
        if l == 1:
            js += '''
{
    "%(view_attr_name)s": {
        "attr_name": "%(attr_name)s"
    }
}''' % params
        else:
            js += '''
{
    "%(view_attr_name)s": {
        "attr_name": "%(attr_name)s"
    }
},''' % params
        l -= 1

js += '''
]
}'''

print(js)
```