Акционерное общество «ИК Интеграция»

ПРОГРАММА ДЛЯ ЭВМ: «КОМПЛЕКС УСКОРЕННОЙ РАЗРАБОТКИ СИСТЕМ (КУРС)»

Руководство программиста КУРС

ЛИСТОВ 113

Москва, 2022

ОГЛАВЛЕНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Определения, обозначения и сокращения, используемые в настоящем документе, приведены в таблице 1.

Таблица 1. Определения, обозначения и сокращения

Обозначения/ сокращения	**Описание (определения)**
CSS	Формальный язык описания внешнего вида документа (веб-страницы)
DOCX	Расширение имени файла, используемое для файлов, представляющих текст, с разметкой или без
HTML	Стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере
JSON	Текстовый формат обмена данными
Microsoft Excel, MS Excel	Программа для работы с электронными таблицами
Microsoft Word, MS Word	Текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов
SQL	Structured Query Language — язык структурированных запросов
XML	eXtensible Markup Language — расширяемый язык разметки
БД	База данных
КУРС, Программа	Комплекс ускоренной разработки систем
Прикладная информационная система	Разрабатываемые информационные системы с использованием компонентов КУРС
СУБД	Система управления базами данных

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1. Назначение Программы

Комплекс ускоренной разработки систем (далее – Программа, КУРС) предназначен для сокращения затрат на разработку прикладных информационных систем за счет использования готовых программных компонентов для разработки веб-интерфейсов, отчетов и других компонентов веб-приложений; сокращения затрат на сопровождение и администрирование веб-приложений за счет предоставления удобных готовых модулей для решения типовых задач администрирования.

1.2. Состав и функции Программы

Программа имеет следующие функциональные модули:

- Модуль «Сущности». Модуль предназначен для описания в специально разработанном XML формате всех аспектов визуального представления и поведения объекта. А также модуль предназначен для автоматического формирования списков (табличных представлений) и форм для просмотра/корректировки информации о выбранном объекте (записи БД) на базе создаваемых XML описаний (метаданных).
- Модуль «Отчетность». Модуль предназначен для формирования отчетов на базе создаваемых XML описаний (метаданных) и на основании SQL выборок из различных БД с большим выбором инструментариев для настойки визуальных шаблонов отчетов, поддержкой отчетов с параметрами, задаваемыми пользователем, а также с поддержкой внешнего программного управления процессом формирования отчетов.
- Модуль «Управление доступом». Модуль предназначен для управления учетными записями пользователей, разграничения прав доступа в соответствии с ролевой моделью на базе создаваемых XML описаний (метаданных), авторизации и аутентификации пользователей (с поддержкой доменной аутентификации Windows), а также для протоколирования действий пользователей в прикладной информационной системе.

1.3. Условия применения

Для функционирования Программы необходимо обеспечить следующие минимальные требований к рабочей станции программиста:

- Процессор – 4 ядра Intel Core i-5 3 ГГц или аналогичный по производительности процессор.
- Архитектура процессора – x64 (64 бита).

- Оперативная память – 8 ГБ.
- Жесткий диск – 20 ГБ.
- Сетевая плата – Ethernet 10 Мбит.

Программа обеспечивает совместимость со следующими компонентами программного обеспечения:

- Операционная система – Microsoft Windows, Linux
- СУБД - Microsoft SQL Server, PostgreSQL, MySql.
- Среда разработки прикладной системы – Microsoft .NET.
- Инструментарий разработки – Microsoft Visual Studio.
- Язык разработки – C#.
- Браузер - Mozilla Firefox (версии 100 и выше), Google Chrome (версии 101 и выше), Яндекс.Браузер (версии 22 и выше)

2. ОПИСАНИЕ МЕТАДАННЫХ МОДУЛЕЙ ПРОГРАММЫ

2.1. Описание метаданных модуля «Сущности»

2.1.1. Общее описание

Метаданные сущностей хранятся в файлах формата XML в заранее выбранной папке внутри папки приложения. Относительный путь (относительно корневой папки приложения – папки, на которую настроен IIS) к папке, содержащей метаданные, задается в конфигурационном файле приложения в секции KURSConfiguration в разделе MetadataLocations с помощью ключа entities.

Пример,

```
<KURSConfiguration>
    <MetadataLocations entities="~/_Metadata" accessControl="~/_SystemMetadata"
reports="~/_Reports" />
```

Каждая сущность должна быть описана в отдельном файле метаданных.

2.1.2. Структура метаданных сущности

2.1.2.1. Entity – Параметры сущности

Комплексный тип. Формирует список из полей.

Дочерние элементы:

- References (0..1) – настройки связанных сущностей
- Data (1..1) – настройки взаимодействия с БД
- Filter (0..1) – глобальные настройки фильтра
- List (0..1) – глобальные настройки списка
- Form (0..1) – глобальные настройки формы
- Fields (1..1) – настройки атрибутов сущности

Атрибуты представлены в таблице 2.

Таблица 2. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
Id	String	Уникальный ключ сущности (желательно совпадение с названием, содержащим его, файла XML)	Обязательный
allowFormAdd	Boolean	Разрешать добавление новых объектов (записей) через форму ввода	Необязательный
allowFormEdit	Boolean	Разрешать редактирование объекта (записи) через форму ввода	Необязательный
allowFormView	Boolean	Разрешать просмотр объекта (записи) через форму просмотра	Необязательный

allowFormDelete	Boolean	Разрешать удаление объекта (записи) в форме редактирования	Необязательный
allowListSelector	Boolean	Разрешать множественное добавление объектов (записей) с использованием связи многие-ко-многим	Необязательный
allowListAdd	Boolean	Разрешать добавление новых объектов через новую строку в списке	Необязательный
allowListEdit	Boolean	Разрешать редактирование данных списка (редактируемый список)	Необязательный
allowListDelete	Boolean	Разрешать удаление объектов (записей) в списке	Необязательный

Пример

```
<Entity id="LIC_APPLICATIONS_List" allowFormAdd="true" allowFormEdit="true"
allowFormView="true" allowFormDelete="false" allowListDelete="false"
allowListSelector="false">
```

2.1.2.2. Entity/References – Связи с сущностями

Элемент комплексного типа. Задает ссылки на связанные с текущей сущностью сущности.

Дочерних элементов нет.

Атрибуты представлены в таблице 3.

Таблица 3. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
addEntityId	String	Сущность для формы добавления объекта (записи)	Необязательный
editEntityId	String	Сущность для формы редактирования объекта (записи)	Необязательный
viewEntityId	String	Сущность для формы просмотра объекта (записи)	Необязательный
selectorEntityId	String	Сущность для связи многие-ко-многим. Указывается сущность, при выборе записей которой должны сформироваться записи в основной сущности	Необязательный
selectorFieldName	String	Поле для связки с сущностью, указанной в selectorEntityId.	Необязательный
exportEntityId	String	Сущность для экспорта данных в Word и Excel	Необязательный

Пример,

```
<References addEntityId="LIC_APPLICATIONS_Tab1" editEntityId="LIC_APPLICATIONS_Tab1"
viewEntityId="LIC_APPLICATIONS_Tab1" exportEntityId="LIC_APPLICATIONS_List" />
```

2.1.2.3. Entity/Data – Взаимодействие с БД

Элемент комплексного типа, отвечает за взаимодействие с БД.

Дочерние элементы:

- WhereSection (0..1) – настройки фильтрации данных
- SortExpression (0..1) – настройка сортировки данных
- AddMode (0..1) – настройка режима видимости полей формы в режиме создания
- EditMode (0..1) – настройка режима видимости полей формы в режиме редактирования
- ViewMode (0..1) – настройка режима видимости полей формы в режиме просмотра
- Initialize (0..1) – настройка значений по умолчанию
- SelectCount (0..1) –переопределения запроса на подсчет количества записей в списке
- SelectMany (0..1) – переопределение запроса на получение записей списка
- SelectSingle (0..1) – переопределение запроса на получение записи для формы
- Insert (0..1) – переопределение запроса на вставку записи
- Update (0..1) – переопределение запроса на обновление записи
- Delete (0..1) – переопределение запроса на удаление записи
- BeforeInsert (0..N) – настройка действий перед вставкой
- BeforeUpdate (0..N) – настройка действий перед обновлением
- BeforeDelete (0..N) – настройка действий перед удалением
- AfterInsert (0..N) – настройка действий после вставки
- AfterUpdate (0..N) – настройка действий после обновления
- AfterDelete (0..N) – настройка действий после удаления

Атрибуты представлены в таблице 4.

Таблица 4. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
tableName	String	Имя таблицы в подключаемой базе	Обязательный
connectionStringName	String	Имя подключения к базе, указанное в файле web.config	Необязательный
newIdMode	NewIdModesEnum	Способ генерации первичного ключа	Необязательный
selectorDuplicateCheck	Boolean	Использовать ли проверку дубликатов в случае использования selector. По умолчанию проверка включена	Необязательный

Пример

```
<Data tableName="LIC_APPLICATIONS" connectionStringName="AKNDPP_REGIONAL"
newIdMode="ServerGuid">
```

2.1.2.4. Entity/Data/WhereSection – Фильтрация данных

Элемент текстового типа (string). Содержит WHERE-секцию SQL-выражения, которая содержит условие фильтрации данных.

Пример,

```
<WhereSection>
    <![CDATA[
LIC_APPLICANTS.DELETED = 0
        ]]>
</WhereSection>
```

2.1.2.5. Entity/Data/SortSection – Сортировка данных

Элемент текстового типа (string). Содержит ORDER-BY-секцию SQL-выражения, задающую сортировку загружаемых данных (по возрастанию или по убыванию)

Пример,

```
<SortSection>LIC_APPLICANTS.ID</SortSection>
```

2.1.2.6. Entity/Data/SelectMany – Переопределение запроса на получение записей списка

Элемент текстового типа (string). Содержит SQL-выражение (SELECT-запрос) на выборку данных, для отображения в списке.

2.1.2.7. Entity/Data/SelectSingle – Переопределение запроса на получение записи формы

Элемент текстового типа (string). Содержит SQL-выражение (SELECT-запрос) на выборку данных, для отображения в списке.

2.1.2.8. Entity/Data/Insert – Переопределение запроса на вставку записи

Элемент текстового типа (string). Содержит SQL-выражения, задающие значения по умолчанию для полей на форме в режиме создания.

Пример,

```
<Insert>
    <![CDATA[
INSERT INTO [dbo].[FORM_4_1B_ELEMENTS]
([ID]
,[FORM_4_1A_FK]
,[NPA_NAME_TYPES_FK]
,[TOTAL]
,[NUMBER]
,[ACC_DATE]
```

```
,[HEAD_DATE]
,[PUB_DATE]
,[NPA_NAME]
,[STRING_NUMBER]
,[DELETED]
,[PARENT_NPA_NAME_TYPES_FK])
SELECT
    NEWID()
    ,@FORM_4_1A_FK
    ,null
    ,null
    ,@NUMBER
    ,@ACC_DATE
    ,@HEAD_DATE
    ,@PUB_DATE
    ,@NPA_NAME
    ,@STRING_NUMBER
    ,0
    ,(SELECT ID FROM FORM_NPA_TYPES WHERE NAME = N'НПА органа, исполнительной власти
субъекта Российской Федерации, осуществляющего переданные полномочия Российской
Федерации в сфере образования')
        ]]>
</Insert>
```

2.1.2.9. Entity/Data/Update – Переопределение запроса на обновление записи

Элемент текстового типа (string). Содержит SQL-выражения, задающие значения по умолчанию для полей на форме в режиме создания.

Пример

```
<Update>
    <![CDATA[
UPDATE [dbo].[RPT_SECTIONS] SET
[RPT_FORM_FK] = @RPT_FORM_FK,
[NAME] = @NAME,
[CODE] = null
WHERE [ID] = @ID ;
    ]]>
</Update>
```

2.1.2.10. Entity/Data/Delete – Переопределение запроса на удаление записи

Элемент текстового типа (string). Содержит SQL-выражения, задающие значения по умолчанию для полей на форме в режиме создания.

Пример

```
<Delete>
    <![CDATA[
UPDATE FORM_4_5V SET DELETED = 1 WHERE ID = @CurrentObjectId;
        ]]>
</Delete>
```

2.1.2.11. Entity/Data/BeforeInsert – Действие перед вставкой записи

Содержит SQL-выражение, выполняемое до создания записи.

Атрибуты представлены в таблице 5.

Таблица 5. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
mode	EventOperationModesEnum	Режим выполнения действия. По умолчанию ForEach (для каждой обрабатываемой записи)	Необязательный
connectionStringName	String	Имя подключения к базе, указанное в файле web.config	Необязательный

Пример,

```
<BeforeInsert connectionStringName="AKNDPP_FEDERAL">
    <![CDATA[
INSERT INTO [dbo].[RPT_INDEXES]
(
[ID],
[RPT_SECTION_FK],
[STRING_NUMBER],
[NAME],
[UNIT],
[OKEI_CODE]
)
VALUES
(
@ID,
@RPT_SECTION_FK,
@STRING_NUMBER,
@NAME,
@UNIT,
@OKEI_CODE
) ;
    ]]>
</BeforeInsert>
```

2.1.2.12.Entity/Data/BeforeUpdate – Действие перед обновлением записи

Содержит SQL-выражение, выполняемое до редактирования записи.

Атрибуты представлены в таблице 6

Таблица 6. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
mode	EventOperationModesEnum	Режим выполнения действия. По умолчанию ForEach (для каждой обрабатываемой записи)	Необязательный
connectionStringName	String	Имя подключения к базе, указанное	Необязательный

		в файле web.config	

Пример,

```
<BeforeUpdate connectionStringName="AKNDPP_FEDERAL">
    <![CDATA[
UPDATE [dbo].[RPT_INDEXES] SET
[RPT_SECTION_FK] = @RPT_SECTION_FK,
[STRING_NUMBER] = @STRING_NUMBER,
[NAME] = @NAME,
[UNIT] = @UNIT,
[OKEI_CODE] = @OKEI_CODE
WHERE [ID] = @ID ;
    ]]>
</BeforeUpdate>
```

2.1.2.13.Entity/Data/BeforeDelete – Действие перед удалением записи

Содержит SQL-выражение, выполняемое до удаления записи.

Атрибуты представлены в таблице 7

Таблица 7. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
mode	EventOperationModesEnum	Режим выполнения действия. По умолчанию ForEach (для каждой обрабатываемой записи)	Необязательный
connectionStringName	String	Имя подключения к базе, указанное в файле web.config	Необязательный

Пример

```
<BeforeDelete connectionStringName="AKNDPP_FEDERAL">
    <![CDATA[
DELETE FROM [dbo].[RPT_INDEXES] WHERE [ID] = @CurrentObjectId ;
    ]]>
</BeforeDelete>
```

2.1.2.14.Entity/Data/AfterInsert – Действие после вставки записи

Содержит SQL-выражение, выполняемое после создания записи.

Атрибуты представлены в таблице 8

Таблица 8. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
mode	EventOperationModesEnum	Режим выполнения действия. По умолчанию ForEach (для каждой обрабатываемой записи)	Необязательный
connectionStringName	String	Имя подключения к базе, указанное в файле web.config	Необязательный

Пример

```
<AfterInsert connectionStringName="AKNDPP_FEDERAL">
    <![CDATA[
INSERT INTO [dbo].[RPT_INDEXES]
(
[ID],
[RPT_SECTION_FK],
[STRING_NUMBER],
[NAME],
[UNIT],
[OKEI_CODE]
)
VALUES
(
@ID,
@RPT_SECTION_FK,
@STRING_NUMBER,
@NAME,
@UNIT,
@OKEI_CODE
) ;
    ]]>
</AfterInsert>
```

2.1.2.15.Entity/Data/AfterUpdate – Действие после обновления записи

Содержит SQL-выражение, выполняемое после редактирования записи.

Атрибуты представлены в таблице 9

Таблица 9. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
mode	EventOperationModesEnum	Режим выполнения действия. По умолчанию ForEach (для каждой обрабатываемой записи)	Необязательный

connectionStringName	String	Имя подключения к базе, указанное в файле web.config	Необязательный

Пример,

```
<AfterUpdate connectionStringName="AKNDPP_FEDERAL">
    <![CDATA[
UPDATE [dbo].[RPT_INDEXES] SET
[RPT_SECTION_FK] = @RPT_SECTION_FK,
[STRING_NUMBER] = @STRING_NUMBER,
[NAME] = @NAME,
[UNIT] = @UNIT,
[OKEI_CODE] = @OKEI_CODE
WHERE [ID] = @ID ;
    ]]>
</AfterUpdate>
```

2.1.2.16.Entity/Data/AfterDelete – Действие после удаления записи

Содержит SQL-выражение, выполняемое после удаления записи.

Атрибуты представлены в таблице 10

Таблица 10. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
mode	EventOperationModesEnum	Режим выполнения действия. По умолчанию ForEach (для каждой обрабатываемой записи)	Необязательный
connectionStringName	String	Имя подключения к базе, указанное в файле web.config	Необязательный

Пример,

```
<AfterDelete connectionStringName="AKNDPP_FEDERAL">
    <![CDATA[
DELETE FROM [dbo].[RPT_INDEXES] WHERE [ID] = @CurrentObjectId ;
    ]]>
</AfterDelete>
```

2.1.2.17.Entity/Data/SqlActions – SQL- действия для вызова KURS.runSqlAction

Описывает перечень SQL-действий с помощью дочернего элемента SqlAction.

Дочерние элементы:

- SqlAction (1..N) – SQL-действие

Пример,

```
<SqlActions>
    <SqlAction id="autoFill" connectionStringName="AKNDPP_REGIONAL">
        <![CDATA[
EXECUTE [sp_ReportFormAutoFill]  @ID, 'FORM1' ;
EXECUTE [sp_ReportSectionComputations] '8c56f2c5-4225-4f61-b428-24e9e78ddabe', @ID ;
EXECUTE [sp_ReportSectionComputations] 'a1d6e641-bd18-4f7b-a4d4-8508d6f97dd5', @ID ;
EXECUTE [sp_ReportSectionComputations] '510d370b-b620-47d6-9317-b0b3d17affe5', @ID ;
EXECUTE [sp_ReportSectionComputations] '74e2c99c-5332-460b-ba27-c621a3f96276', @ID ;
]]>
    </SqlAction>
</SqlActions>
```

2.1.2.18.Entity/Data/SqlActions/SqlAction – SQL- действие

Элемент строкового типа. Содержит SQL-выражения.

Атрибуты представлены в таблице 11

Таблица 11. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Идентификатор действия	Обязательный
connectionStringName	String	Имя подключения к базе, указанное в файле web.config	Необязательный

Пример

```
<SqlAction id="autoFill" connectionStringName="AKNDPP_REGIONAL">
    <![CDATA[
EXECUTE [sp_ReportFormAutoFill]  @ID, 'FORM1' ;
EXECUTE [sp_ReportSectionComputations] '8c56f2c5-4225-4f61-b428-24e9e78ddabe', @ID ;
EXECUTE [sp_ReportSectionComputations] 'a1d6e641-bd18-4f7b-a4d4-8508d6f97dd5', @ID ;
EXECUTE [sp_ReportSectionComputations] '510d370b-b620-47d6-9317-b0b3d17affe5', @ID ;
EXECUTE [sp_ReportSectionComputations] '74e2c99c-5332-460b-ba27-c621a3f96276', @ID ;
]]>
</SqlAction>
```

2.1.2.19.Entity/Filter – Параметры фильтра

Задает параметры отображения списка на странице. Раздел содержит глобальные настройки фильтра.

Дочерние элементы:

- Initialize (0..1) – Инициализация полей фильтра по умолчанию
- FieldGroups (0..1) – группы полей фильтра

Атрибуты представлены в таблице 12

Таблица 12. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
displayMode	PanelDisplayModesEnum	Режим отображения блока фильтра. По умолчанию используются глобальные настройки отображения	Необязательный

cssPrefix	string	Имя css-класса для блока фильтра в целом	Необязательный

Пример

```
<Filter displayMode="Global" cssPrefix="cssPrefix1">
```

2.1.2.20.Entity/Filter/Initialize – Инициализация полей фильтра по умолчанию

Элемент текстового типа (string). Содержит SQL-выражения, задающие значения по умолчанию для полей на фильтре.

Пример

```
<Filter displayMode="Inline">
    <Initialize>
        SELECT N'STAT_VALUES.END_DATE', CONVERT(nvarchar(50), getdate(), 104) ;
        SELECT N'STAT_VALUES.REGIONS_FK', (SELECT TOP 1 r.ID FROM REGIONS r WHERE
r.DELETED = 0 ORDER BY r.NAME) ;
    </Initialize>
</Filter>
```

2.1.2.21.Entity/Filter/FieldGroups – Группы полей фильтра

Описывает группы полей фильтрации с помощью дочернего элемента FieldGroup. Группы полей (отображаются как блоки с заголовком).

Дочерние элементы:

– FieldGroup (1..N) – группа полей

Пример

```
<FieldGroups>
    <FieldGroup id="Gen" caption="Общие сведения"/>
    <FieldGroup id="PPE" caption="Сведения о ППЭ"/>
    <FieldGroup id="Contact" caption="Контакты специалистов в ППЭ"/>
    <FieldGroup id="Prepare" caption="Подготовка к экзамену"/>
    <FieldGroup id="Proved" caption="Проведение экзамена"/>
    <FieldGroup id="Peredacha" caption="Передача файлов актов и журналов"/>
</FieldGroups>
```

2.1.2.22.Entity/Filter/FieldGroups/FieldGroup – Группа полей

Пустой элемент. Создает группу фильтров, которая формирует список из фильтров, и может сворачиваться при отображении.

Дочерние элементы:

– FieldGroup (0..N) – вложенная группа полей

Атрибуты представлены в таблице 13

Таблица 13. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Название группы	Обязательный
caption	string	Заголовок группы	Обязательный
tooltip	string	Всплывающая подсказка	Необязательный

collapsed	Boolean	Состояние свернутости группы по умолчанию (true – свернута, false – раскрыта)	Необязательный

Пример

```
<FieldGroup id="Gen" caption="Общие сведения"/>
```

2.1.2.23.Entity/List – Параметры списка

Задает параметры отображения списка на странице. Раздел содержит глобальные настройки списка.

Дочерние элементы:

- JsSection (0..1) – JS-секция интерфейса списка
- ControlPanel (0..1) – настройка панели управления списка
- PagerPanel (0..1) – настройка панели пейджера
- ColumnGroups (0..1) – настройка групп столбцов списка
- StyleConditions (0..1) – настройка условной стилизации списка

Атрибуты представлены в таблице 14

Таблица 14. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
pageSize	unsignedInt	Кол-во отображаемых строк на странице (по умолчанию – 20 строк)	Необязательный
headerText	String	Заголовок списка (по умолчанию заголовок отсутствует)	Необязательный
cacheType		Включает режим кеширование по справочнику ListCacheTypesEnum	Необязательный
cacheDurationInSeconds	unsignedInt	Время хранения кеша в секундах	Необязательный
cacheKey	String	Ключ данных в кеш-таблице	Необязательный
cacheCountOnly	Boolean	Кешировать только количество записей списка	Необязательный
cssPrefix	String	Задает настройки css для всего списка. Настройки задаются с помощью указания тега настроек из файла css. по умолчанию применяются настройки без тега.	Необязательный
formatSchema	String	Схема форматирования, которая задается в файле конфигурации	Необязательный
showTableHeader	Boolean	Показывать заголовок таблицы. По умолчанию показывается	Необязательный

allowSort	Boolean	Разрешать пользовательскую сортировку списка. По умолчанию сортировка разрешена	Необязательный
allowResize	Boolean	Разрешать пользователю изменение ширины столбцов списка. По умолчанию разрешена	Необязательный
fixTableHeader	Boolean	Фиксировать заголовок таблицы при скроллинге. По умолчанию не фиксируется	Необязательный
fixFirstColumns	unsignedInt	Фиксировать первые N-столбцов при скроллинге. По умолчанию не фиксируется	Необязательный
fixLastColumns	unsignedInt	Фиксировать последние N-столбцов при скроллинге. По умолчанию не фиксируется	Необязательный

Пример

```
<List
    pageSize="20" headerText="Список организаций"
    cacheType="FilteredData" cacheDurationInSeconds="600" cacheKey="SCHOOLS"
    cacheCountOnly="false" cacheByUser="false"
    cssPrefix="cssPrefix1" formatSchema="formatSchema1"
    showTableHeader="true" allowSort="true" fixTableHeader="false">
```

2.1.2.24.Entity/List/JsSection – JS-секция списка

Элемент текстового типа (string). Содержит JS-выражения для интерфейса списка.

Пример

```
<JsSection>
    <![CDATA[
function afterGridRefresh(listid)
{
    if(listid == 'LIC_APPLICANTS_List_IP' || listid == 'LIC_APPLICANTS_List_OO')
    {
        HideProccessButtons();
    }
}
     ]]>
</JsSection>
```

2.1.2.25.Entity/List/Initialize – Инициализация полей новой записи списка

Элемент текстового типа (string). Содержит SQL-выражения, задающие значения по умолчанию для новой строки в списке.

Пример

```
<List pageSize="99999" headerText="Разрезы">
    <Initialize>
        <![CDATA[
```

```
DECLARE @NEW_NUMBER int = isnull((SELECT MAX(i.SYS_CODE) FROM RPT_DIMENSIONS i WHERE
i.RPT_SECTION_FK = @CurrentObjectId),0) + 1;
SELECT N'RPT_DIMENSIONS.SYS_CODE', @NEW_NUMBER ;
SELECT N'RPT_DIMENSIONS.SYS_NAME', 'DIM' + CAST(@NEW_NUMBER AS nvarchar(50)) ;
SELECT N'RPT_DIMENSIONS.ORDER_NUMBER', @NEW_NUMBER ;
        ]]>
    </Initialize>
</List>
```

2.1.2.26.Entity/List/ControlPanel – Панель управления списка

Элемент комплексного типа. Задает отображение панели управления списком.

Дочерние элементы:

– Button (0..N) – Дополнительная кнопка в панели управления

Атрибуты представлены в таблице 15

Таблица 15. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
position	PanelPositionsEnum	Расположение панели управления	Необязательный
allowListSettings	Boolean	Показывать кнопку настройки списка	Необязательный
allowExportToExcel	Boolean	Показывать кнопку экспорта в Excel	Необязательный
allowExportToWord	Boolean	Показывать кнопку экспорта в Word	Необязательный

Пример

```
<List pageSize="20" headerText="" >
    <ControlPanel allowExportToExcel="true" allowExportToWord="true">
        <Button caption="Подписать все">
            <JsOnClick>
                <![CDATA[
KURS.openPopupWindow('../SignTransmission/SignTransfer.aspx?TransmissionsId=LICENSES_AL
L&EntityId=LICENSES&RecordId=') ;
            ]]>
            </JsOnClick>
        </Button>
    </ControlPanel>
</List>
```

2.1.2.27.Entity/List/ControlPanel/Button – Кнопка в панели управления списка

Элемент комплексного типа. Задает отображение панели управления списком.

Дочерние элементы:

– JsOnClick (0..1) – Выполнение JS-выражения по нажатию на кнопку

Выполняемые действия в рамках одной кнопки могут комбинироваться в любом порядке.

Атрибуты представлены в таблице 16

Таблица 16. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
caption	String	Название кнопки	Обязательный
cssPrefix	String	Имя css-класса для стилизации кнопки	Необязательный

Пример

```
<Button caption="Подписать все">
    <JsOnClick>
        <![CDATA[
KURS.openPopupWindow('../SignTransmission/SignTransfer.aspx?TransmissionsId=LICENSES_ALL&EntityId=LICENSES&RecordId=') ;
    ]]>
    </JsOnClick>
</Button>
```

2.1.2.28.Entity/List/ControlPanel/Button/JsOnClick – Выполнение JS-выражения по нажатию на кнопку

Элемент строкового типа. Содержит JS-выражения.

Пример

```
<JsOnClick>
    <![CDATA[
KURS.openPopupWindow('../SignTransmission/SignTransfer.aspx?TransmissionsId=LICENSES_ALL&EntityId=LICENSES&RecordId=') ;
]]>
</JsOnClick>
```

2.1.2.29.Entity/List/PagerPanel – Панель пейджера списка

Элемент комплексного типа. Задает вид отображения информации об открытом списке. С помощью атрибутов. Дочерних элементов нет.

Атрибуты представлены в таблице 17

Таблица 17. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
position	PanelPositionsEnum	Расположение панели пейджера	Необязательный
previousButtonText	String	Задает наименование кнопки перехода на предыдущую страницу	Необязательный
nextButtonText	String	Задает наименование кнопки перехода на следующую страницу	Необязательный
firstButtonText	String	Задает наименование кнопки перехода на первую страницу	Необязательный

lastButtonText	String	Задает наименование кнопки перехода на последнюю страницу	Необязательный
firstLastButtonTextMode	FirstLastButtonTextModesEnum	Режим отображения кнопок перехода на первую/последнюю страницу	Необязательный
buttonsOrder	PagerButtonsOrdersEnum	Порядок кнопок перехода по страницам	Необязательный
showPrevPreviousAndNextButtons	Boolean	Признак отображать кнопки перехода на предыдущую/следующую страницу (по умолчанию true)	Необязательный
showInfo	Boolean	Признак показывать информацию об открытых записях и их общем кол-ве (по умолчанию true)	Необязательный
infoTextFormat	String	Формат отображения текста об открытых записях и их общем кол-ве (используются спец-слова: "{from}" (порядковый номер первой отображаемой записи), "{to}" (порядковый номер последней отображаемой записи), "{total}" (всего записей))	Необязательный
pageButtonsCount	unsignedInt	Кол-во кнопок перехода на страницы	Необязательный

Пример

```
<PagerPanel position="None" />
```

2.1.2.30.Entity/List/ColumnGroups – Группировка столбцов списка

Описывает группы столбцов списка с помощью дочернего элемента ColumnGroup.

Дочерние элементы:

– ColumnGroup (1..N) – группа столбцов

Атрибуты представлены в таблице 18.

Таблица 18. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
fitHeight	Boolean	Выравнивать содержимое ячеек заголовка таблицы по высоте	Необязательный

Пример

```
<ColumnGroups fitHeight="true">
```

```
    <ColumnGroup id="f1" caption="Наименование показателя" order="300" />
    <ColumnGroup id="f2" caption="№ строки" order="400" />
    <ColumnGroup id="f3" caption="Значение" order="500" />
</ColumnGroups>
```

2.1.2.31.Entity/List/ColumnGroups/ColumnGroup – Группа столбцов

Элемент описывает группу столбцов.

Дочерние элементы:

– ColumnGroup (0..N) – вложенная группа полей

Атрибуты представлены в таблице 19.

Таблица 19. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Название группы	Обязательный
caption	string	Заголовок группы	Обязательный
order	Integer	Порядок следования группы столбцов	Необязательный
fitRows	Integer	Количество строк, которое необходимо объединить для этой ячейки группы	Необязательный
cssPrefix	String	Имя css-класса для стилизации группы	Необязательный

Пример

```
<ColumnGroup id="f1" caption="Наименование показателя" order="300" />
```

2.1.2.32.Entity/List/SummaryRow – Итоговая строка списка

Настройка итоговой строки списка.

Дочерние элементы:

– Field (1..N) – поле итоговой строки списка

Атрибуты представлены в таблице 20

Таблица 20. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
position	SummaryRowPositionsEnum	Режим отображения итоговой строки. По умолчанию итоговая строка отображается после списка	Необязательный

Пример

```
<SummaryRow position="Bottom">
    <Field positionField="NAME" mode="Single"><![CDATA[N'Итого:']]></Field>
    <Field positionField="F1S4P0IND1DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND2DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND3DIM1" mode="Sum" />
```

```
    <Field positionField="F1S4P0IND4DIM1" mode="Sum" />
</SummaryRow>
```

2.1.2.33.Entity/List/SummaryRow/Field – Поле итоговой строки списка

Настройка поля в итоговой строке списка.

Атрибуты представлены в таблице 21.

Таблица 21. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
positionField	String	Поле для отображения итогового значения	Обязательный
mode	SummaryFieldModesEnum	Режим агрегации итогового значения	Обязательный
fitColumns	Integer	Количество столбцов, которое необходимо объединить для этой ячейки	Необязательный
Вложенный текст	Вложенный текст	SQL-выражение, используемое для формирования итогового значения. Если не указано, то по умолчанию используется значения поля, указанного в positionField	Необязательный

Пример

```
<Field positionField="NAME" mode="Single"><![CDATA[N'Итого:']]></Field>
```

2.1.2.34.Entity/List/RowGroups – Группировка строк списка

Настройка группировки строк списка.

Дочерние элементы:

- Field (1..N) – поле строки группы
- SummaryRow (0..1) – итоговая строка группы
- RowGroups (0..1) – вложенная группировка строк (группировка 2-го, N-го уровней)

Атрибуты представлены в таблице 22

Таблица 22. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
groupByField	string	Поле, по которому необходимо выполнить группировку строк	Обязательный
orderByField	string	Поле, по которому необходимо выполнить сортировку групп	Обязательный
collapsed	Boolean	Признак свернутости групп по умолчанию	Необязательный

Пример

```
<RowGroups groupByField="FEDERAL_DISTRICTS_ID" orderByField="FEDERAL_DISTRICTS_NAME"
collapsed="true">
    <Field positionField="NAME"
mode="Single"><![CDATA[FEDERAL_DISTRICTS.NAME]]></Field>
    <Field positionField="F1S4P0IND1DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND2DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND3DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND4DIM1" mode="Sum" />
</RowGroups>
```

2.1.2.35.Entity/List/RowGroups/Field – Поле в строке группы списка

Настройка поля в группе строк.

Атрибуты представлены в таблице 23

Таблица 23. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
positionField	String	Поле для отображения итогового значения	Обязательный
mode	SummaryFieldModesEnum	Режим агрегации итогового значения	Обязательный
fitColumns	Integer	Количество столбцов, которое необходимо объединить для этой ячейки	Необязательный
Вложенный текст	Вложенный текст	SQL-выражение, используемое для формирования итогового значения. Если не указано, то по умолчанию используется значения поля, указанного в positionField	Необязательный

Пример

```
<Field positionField="NAME" mode="Single"><![CDATA[N'Итого:']]></Field>
```

2.1.2.36.Entity/List/RowGroups/SummaryRow – Итоговая строка группы

Настройка итоговой строки группы строк.

Дочерние элементы:

– Field (1..N) – поле итоговой строки списка

Атрибуты представлены в таблице 24

Таблица 24. Атрибуты

Имя атрибута	Тип атрибута	Комментарий	Встречаемость

position	SummaryRowPositionsEnum	Режим отображения итоговой строки. По умолчанию итоговая строка отображается после списка	Необязательный

Пример

```
<SummaryRow position="Bottom">
    <Field positionField="NAME" mode="Single"><![CDATA[N'Итого:']]></Field>
    <Field positionField="F1S4P0IND1DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND2DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND3DIM1" mode="Sum" />
    <Field positionField="F1S4P0IND4DIM1" mode="Sum" />
</SummaryRow>
```

2.1.2.37.Entity/List/RowGroups/SummaryRow/Field – Поле итоговой строки группы

Настройка поля в итоговой строке группы.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
positionField	String	Поле для отображения итогового значения	Обязательный
mode	SummaryFieldModesEnum	Режим агрегации итогового значения	Обязательный
fitColumns	Integer	Количество столбцов, которое необходимо объединить для этой ячейки	Необязательный
Вложенный текст	Вложенный текст	SQL-выражение, используемое для формирования итогового значения. Если не указано, то по умолчанию используется значения поля, указанного в positionField	Необязательный

Пример

```
<Field positionField="NAME" mode="Single"><![CDATA[N'Итого:']]></Field>
```

2.1.2.38.Entity/List/StyleConditions – Условная стилизация строк списка

Элемент комплексного типа. Описывает условия стилизации строк списка с помощью дочернего элемента StyleCondition.

Дочерние элементы:

– StyleCondition (1..N) – условие стилизации

Пример

```
<StyleConditions>
    <StyleCondition cssPrefix="grey">
```

```
        <![CDATA[
LIC_APPLICATION_STATUSES_FK_LIC_APPLICATION_STATUSES.PROCESS_CODE in
('FINISHED','RETURNED','CANCEL','ERROR')
]]>
    </StyleCondition>
</StyleConditions>
```

2.1.2.39.Entity/List/StyleConditions/StyleCondition – Условие стилизации

Элемент комплексного типа. Содержит имя css-класса и SQL-условие, при котором css-класс должен применяться.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
cssPrefix	String	Название css-класса, который нужно применить в случае выполнения условия	Обязательный
otherwiseCssPrefix	String	Название css-класса, который нужно применить в случае невыполнения условия	Необязательный

Пример

```
<StyleCondition cssPrefix="grey">
    <![CDATA[
LIC_APPLICATION_STATUSES_FK_LIC_APPLICATION_STATUSES.PROCESS_CODE in
('FINISHED','RETURNED','CANCEL','ERROR')
]]>
</StyleCondition>
```

2.1.2.40.Entity/List/Validations – Валидация редактируемого списка

Элемент комплексного типа. Задает параметры валидации редактируемого списка с помощью дочерних элементов.

Дочерние элементы:

- JsValidation (0..N) – валидация через JS-выражение
- SqlCreateValidation (0..N) – валидация создания записи через SQL-выражение
- SqlUpdateValidation (0..N) – валидация обновление записи через SQL-выражение
- SqlDeleteValidation (0..N) – валидация удаления записи через SQL-выражение

2.1.2.41.Entity/List/Validations/JsValidation – Валидация через JS-выражение

Задает выражение для валидации полей с помощью JS-выражения содержащегося в теле элемента.

2.1.2.42.Entity/List/Validations/SqlCreateValidation – Валидация создания записи через SQL-выражение

Задает выражение для валидации полей с помощью SQL-выражения содержащегося в теле элемента.

2.1.2.43.Entity/List/Validations/SqlUpdateValidation – Валидация обновление записи через SQL-выражение

Задает выражение для валидации полей с помощью SQL-выражения содержащегося в теле элемента.

2.1.2.44.Entity/List/Validations/SqlDeleteValidation – Валидация удаления записи через SQL-выражение

Задает выражение для валидации полей с помощью SQL-выражения содержащегося в теле элемента.

2.1.2.45.Entity/Form – Параметры формы

Задает параметры отображения в форме. Раздел содержит настройки формы.

Дочерние элементы:

- JsSection (0..1) – JS-секция интерфейса списка
- TabsControl (0..1) – настройка вкладок формы
- FieldGroups (0..1) – группы полей фильтра
- Validations (0..1) – Настройка ФЛК при сохранении формы

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
addHeaderText	String	Заголовок формы создания	Необязательный
editHeaderText	String	Заголовок формы редактирования	Необязательный
viewHeaderText	String	Заголовок формы просмотра	Необязательный
cssPrefix	String	Задает настройки css для всей формы. Настройки задаются с помощью указания тега настроек из файла css. по умолчанию применяются настройки без тега.	Необязательный
formatSchema	String	Схема форматирования, которая задается в файле конфигурации	Необязательный

Пример

```
<Form viewHeaderText="Просмотр лицензий" addHeaderText="Создание лицензий"
editHeaderText="Редактирование лицензий">
    <TabsControl>
        <Tab caption="Общие сведения" entityId="LICENSES_Tab1_SIGNED" />
        <Tab caption="Приложения" entityId="LICENSES_Tab2_SIGNED" />
        <Tab caption="Решения по результатам контроля (надзора)"
entityId="LICENSES_Tab3_SIGNED" />
    </TabsControl>
```

```
</Form>
```

2.1.2.46.Entity/Form/JsSection – JS-секция формы

Элемент текстового типа (string). Содержит JS-выражения для интерфейса формы.

Пример

```
<JsSection>
    <![CDATA[
function HideProccessButtons()
{
    var ShowAddButton = GetFormFieldValue('LIC_APPLICATIONS_ShowAddButton');
    var ShowSelectButton = GetFormFieldValue('LIC_APPLICATIONS_ShowSelectButton');
    if(!ShowAddButton)
    {
        $('div[subgrid-field="LIC_APPLICATIONS_IDsub3"] .kurs-action-add').hide();
        $('div[subgrid-field="LIC_APPLICATIONS_IDsub4"] .kurs-action-add').hide();
    }
    if(!ShowSelectButton)
    {
        $('div[subgrid-field="LIC_APPLICATIONS_IDsub3"] .kurs-action-grid-
select').hide();
        $('div[subgrid-field="LIC_APPLICATIONS_IDsub4"] .kurs-action-grid-
select').hide();
    }
}

$(document).ready(function ()
{
    HideFormField('LIC_APPLICATIONS_ShowAddButton');
    HideFormField('LIC_APPLICATIONS_ShowSelectButton');
    HideProccessButtons();
});

function afterGridRefresh(listid)
{
    if(listid == 'LIC_APPLICANTS_List_IP' || listid == 'LIC_APPLICANTS_List_OO')
    {
        HideProccessButtons();
    }
}
     ]]>
</JsSection>
```

2.1.2.47.Entity/Data/Initialize – Значения по умолчанию при создании

Элемент текстового типа (string). Содержит SQL-выражения, задающие значения по умолчанию для полей на форме в режиме создания.

Пример

```
<Initialize>
    <![CDATA[
SELECT N'LIC_APPLICANTS.CAPTION1', N'Заявление на лицензирование образовательной
деятельности №' + a.REG_NUMBER + N' от ' + isnull(CONVERT(nvarchar(50),a.REG_DATE,
104),'') FROM dbo.LIC_APPLICATIONS a WHERE ID = @CurrentObjectId ;
        ]]>
</Initialize>
```

2.1.2.48.Entity/Data/AddMode – Режим отображения полей при создании

Элемент текстового типа (string). Содержит SQL-выражения, задающие правила отображения полей на форме в режиме создания.

Пример

```
<AddMode>
    <![CDATA[
IF (@END_DATE is NULL)
SELECT N'LICENSES.END_DATE', N'NotVisible' ;
    ]]>
</AddMode>
```

2.1.2.49.Entity/Data/EditMode – Режим отображения полей при редактировании

Элемент текстового типа (string). Содержит SQL-выражения, задающие правила отображения полей на форме в режиме редактирования.

Пример

```
<EditMode>
    <![CDATA[
IF (@END_DATE is not NULL)
SELECT N'LICENSES.END_DATE', N'Control' ;
    ]]>
</EditMode>
```

2.1.2.50.Entity/Data/ViewMode – Режим отображения полей при просмотре

Элемент текстового типа (string). Содержит SQL-выражения, задающие правила отображения полей на форме в режиме просмотра.

Пример

```
<ViewMode>
    <![CDATA[
IF (@END_DATE is NULL)
SELECT N'LICENSES.END_DATE', N'ReadOnlyText' ;
    ]]>
</ViewMode>
```

2.1.2.51.Entity/Form/FieldGroups – Группы полей фильтра

Описывает группы полей фильтрации с помощью дочернего элемента FieldGroup. Группы полей (отображаются как блоки с заголовком).

Дочерние элементы:

- FieldGroup (1..N) – группа полей

Пример

```
<FieldGroups>
    <FieldGroup id="Gen" caption="Общие сведения"/>
    <FieldGroup id="PPE" caption="Сведения о ППЭ"/>
```

```
    <FieldGroup id="Contact" caption="Контакты специалистов в ППЭ"/>
    <FieldGroup id="Prepare" caption="Подготовка к экзамену"/>
    <FieldGroup id="Proved" caption="Проведение экзамена"/>
    <FieldGroup id="Peredacha" caption="Передача файлов актов и журналов"/>
</FieldGroups>
```

2.1.2.52.Entity/Form/FieldGroups/FieldGroup – Группа полей

Пустой элемент. Создает группу фильтров, которая формирует список из фильтров, и может сворачиваться при отображении.

Дочерние элементы:

- FieldGroup (0..N) – вложенная группа полей

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Название группы	Обязательный
caption	string	Заголовок группы	Обязательный
collapsed	Boolean	Состояние свернутости группы по умолчанию (true – свернута, false – раскрыта)	Необязательный

Пример

```
<FieldGroup id="Gen" caption="Общие сведения"/>
```

2.1.2.53.Entity/Form/Validations – Валидация формы

Простой элемент комплексного типа. Задает параметры валидации формы с помощью дочерних элементов.

Дочерние элементы:

- JsValidation
- SqlValidation

Пример

```
<Validations>
    <JsValidation>
        <![CDATA[
if(!(window.location.href.indexOf("View_A.aspx") > -1) &&
!(window.location.href.indexOf("Edit_A.aspx") > -1))
{
    var type_fk = GetFormFieldValue('LIC_APPLICATIONS_APPLICANT_TYPES_FK') ;
    var proc_fk = GetFormFieldValue('LIC_APPLICATIONS_LIC_PROCEDURES_FK') ;
    if(NullOrEmpty(type_fk))
    {
        alert('Необходимо заполнить поле - Тип заявителя') ;
        return false;
    }
    else if(type_fk == 'ecbb204b-4c28-4783-8bb2-d3aa57f41047' && proc_fk != 'f49fab05-
f321-91bf-b415-98eff0ac1954' &&
        NullOrEmpty(GetFormFieldValue('LIC_APPLICATIONS_SCHOOLS_FK')))
```

```
    {
        alert('Необходимо заполнить поле - Заявитель. Юр. Лицо') ;
        return false;
    }
    else if(type_fk == 'ac8fc3ad-fbe2-4f26-9b74-3ff818371a89' && proc_fk != 'f49fab05-
f321-91bf-b415-98eff0ac1954' &&
        NullOrEmpty(GetFormFieldValue('LIC_APPLICATIONS_INDIVIDUALS_FK')))
    {
        alert('Необходимо заполнить поле - Заявитель. ИП') ;
        return false;
    }
}
        ]]>
    </JsValidation>
</Validations>
```

2.1.2.54.Entity/Form/Validations/JsValidation – Валидация через JS-выражение

Задает выражение для валидации полей с помощью JS-выражения содержащегося в теле элемента.

Пример

```
<JsValidation>
        <![CDATA[
if(!(window.location.href.indexOf("View_A.aspx") > -1) &&
!(window.location.href.indexOf("Edit_A.aspx") > -1))
{
    var type_fk = GetFormFieldValue('LIC_APPLICATIONS_APPLICANT_TYPES_FK') ;
    var proc_fk = GetFormFieldValue('LIC_APPLICATIONS_LIC_PROCEDURES_FK') ;
    if(NullOrEmpty(type_fk))
    {
        alert('Необходимо заполнить поле - Тип заявителя') ;
        return false;
    }
    else if(type_fk == 'ecbb204b-4c28-4783-8bb2-d3aa57f41047' && proc_fk != 'f49fab05-
f321-91bf-b415-98eff0ac1954' &&
        NullOrEmpty(GetFormFieldValue('LIC_APPLICATIONS_SCHOOLS_FK')))
    {
        alert('Необходимо заполнить поле - Заявитель. Юр. Лицо') ;
        return false;
    }
    else if(type_fk == 'ac8fc3ad-fbe2-4f26-9b74-3ff818371a89' && proc_fk != 'f49fab05-
f321-91bf-b415-98eff0ac1954' &&
        NullOrEmpty(GetFormFieldValue('LIC_APPLICATIONS_INDIVIDUALS_FK')))
    {
        alert('Необходимо заполнить поле - Заявитель. ИП') ;
        return false;
    }
}
        ]]>
</JsValidation>
```

2.1.2.55.Entity/Form/Validations/SqlValidation – Валидация через SQL-выражение

Задает выражение для валидации полей с помощью SQL-выражения содержащегося в теле элемента.

Пример

```
<SqlValidation>
    <![CDATA[
SELECT
    N' - «Срок действия» (' + convert(varchar(10),s.END_DATE,104) +
    ') должен быть больше чем «Дата распорядительного документа» (' +
    convert(varchar(10),s.ORDER_DOCUMENT_SIGN_DATE,104) +
    ') в решении по лицензированию'
FROM
(
    select top (1) l.END_DATE, d.ORDER_DOCUMENT_SIGN_DATE
    from licenses l
    left join LIC_DECISIONS d
    on l.ID = d.LICENSES_FK
    where l.ID = @ID
) s
WHERE s.END_DATE < s.ORDER_DOCUMENT_SIGN_DATE
    ]]>
</SqlValidation>
```

2.1.2.56.Entity/Fields – Атрибуты сущности

Элемент комплексного типа. Содержит описание атрибутов сущности и связей атрибутов с полями таблиц в базе данных.

Дочерние элементы:

- Field (1..N) – Атрибут сущности
- Relation (0..N) – Отношение (связь с другой таблицей)

Пример

```
<Fields>
    <Field fieldName="Id" fieldType="Guid" isPrimaryKey="true" />
    <Field fieldName="Login" fieldType="String" isNameField="true">
        <Filter filterMode="Substring" caption="Логин:" groupId="filter_common"
input="Text"  order="1" />
        <List caption="Логин" order="1" />
        <Form caption="Логин:" input="Text" order="1" required="true" />
    </Field>
    <Field fieldName="PasswordWithSaltHash" fieldType="String">
        <Form caption="Пароль:" input="Password" order="2" />
    </Field>
    <Field fieldName="Enabled" fieldType="Boolean">
        <Filter filterMode="Equals" caption="Активен:" input="Text"  order="2" />
        <List caption="Активен" order="2" />
        <Form caption="Активен:" input="Boolean" order="3" />
    </Field>
</Fields>
```

2.1.2.57.Entity/Fields/Field – Атрибут сущности

Элемент комплексного типа. Описывает связь с полем в таблице и визуализацию атрибута в интерфейсе.

Дочерние элементы:

- Data (0..1) – Описание взаимодействия с БД в рамках атрибута\поля.
- Filter (0..1) – Визуализация поля фильтра в рамках атрибута\поля

- List (0..1) – Визуализация столбца списка в рамках атрибута\поля
- Form (0..1) – Визуализация поля форма в рамках атрибута\поля

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
fieldName	String	Название атрибута\поля в таблице	Обязательный
fieldType	FieldDataTypesEnum	Тип данных	Обязательный
isPrimaryKey	Boolean	Является первичным ключом	Необязательный
isNameField	Boolean	Является отображаемым полем (для подстановки)	Необязательный
isCodeField	Boolean	Является кодовым полем (для внутренних связей)	Необязательный
stringLength	UnsignedInt	Длина строки для типа String	Необязательный

Пример

```
<Field fieldName="FULL_NAME" fieldType="String" isNameField="true">
    <Filter filterMode="Substring" caption="Полное наименование" input="Text"
size="NormalLong" lineBreakAfterInput="true" order="60" />
    <List caption="Полное наименование" hidden="false" order="60" />
    <Form addMode="ReadOnlyText" editMode="ReadOnlyText" viewMode="ReadOnlyText"
caption="Полное наименование" input="Text" size="NormalLong" textRows="4"
lineBreakAfterInput="true" required="true" order="30" groupId="Gen_Inf"/>
</Field>
```

2.1.2.58.Entity/Fields/Field/Data – Взаимодействие с БД

Элемент комплексного типа. В этом разделе описывается взаимодействие с базой данных в рамках атрибута\поля: описание вычисляемого поля, дополнительное взаимодействие с БД (например, использование справочника).

Дочерние элементы:

- Dictionary (0..1) – использование справочника
- Subset (0..1) – использование связи многие-ко-многим
- SelectFragment (0..1) – SQL-выражение для получения вычисляемого поля
- WhereFragment – SQL-выражение для фильтрации по атрибуту\полю
- SortFragment (0..1) – SQL-выражение для сортировки по атрибуту\полю

Пример использования справочника

```
<Data>
    <Dictionary tableName="SCHOOL_PROPERTY_FORMS" keyFieldName="ID"
displayFieldName="NAME" orderFieldName="NAME" cacheDurationInSeconds="3600"
cacheByUser="false" >
        <Select>
            <![CDATA[
select ID ,NAME  from SCHOOL_PROPERTY_FORMS  where NOT_TRUE = 0 order by NAME
    ]]>
        </Select>
    </Dictionary>
</Data>
```

Пример вычисляемого поля

```
<Data>
    <SelectFragment>
        <![CDATA[
(
SELECT TOP 5
o.CODE + N'; '
FROM LIC_APPLICATION_OKVED lo
INNER JOIN OKVED o on o.id=lo.OKVED_FK
WHERE lo.LIC_APPLICATIONS_FK=LIC_APPLICATIONS.ID
ORDER BY o.CODE
FOR XML PATH('')
)
]]>
    </SelectFragment>
    <WhereFragment>
        <![CDATA[
(SELECT TOP 1 1 FROM LIC_APPLICATION_OKVED lo WHERE
lo.LIC_APPLICATIONS_FK=LIC_APPLICATIONS.ID AND lo.OKVED_FK =
@LIC_APPLICATIONS_OKVED_FK) = 1
    ]]>
    </WhereFragment>
</Data>
```

2.1.2.59.Entity/Fields/Field/Data/Dictionary - Справочник

Элемент комплексного типа. Раздел описывает справочник, который необходимо использовать при работе с атрибутом.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
tableName	String	Название используемой в качестве справочника таблицы в БД	Обязательный
keyFieldName	String	Название ключевого поля	Обязательный
displayFieldName	String	Название отображаемого поля	Обязательный
codeFieldName	String	Название кодового поля (для внутренних связей)	Необязательный
orderFieldName	String	Название поля сортировки	Необязательный
alias	String	Псевдоним выборки данных справочника	Необязательный
cacheDurationInSeconds	UnsignedInt	Длительность хранения кеша. 0 – кеш не используется	Необязательный
cacheByUser	Boolean	Требуется кешировать данные в рамках пользователя	Необязательный

Пример

```
<Dictionary tableName="SCHOOL_TYPES" keyFieldName="ID" displayFieldName="NAME"
orderFieldName="NAME" cacheDurationInSeconds="3600" cacheByUser="false" >
    <Select>
```

```
        <![CDATA[
select ID ,NAME  from SCHOOL_TYPES  where NOT_TRUE = 0 order by NAME
        ]]>
    </Select>
</Dictionary>
```

2.1.2.60.Entity/Fields/Field/Data/Dictionary/Select – SQL-выражение для получения справочника

Пустой элемент типа String. В этом разделе задается SQL-выражение для получения выборки данных, используемой в качестве справочника.

Пример

```
<Select>
   <![CDATA[
select ID ,NAME  from SCHOOL_TYPES  where NOT_TRUE = 0 order by NAME
   ]]>
</Select>
```

2.1.2.61.Entity/Fields/Field/Data/Subset – Подмножество значений

Элемент комплексного типа. Описывает связь многие ко многим, используется, когда необходимо при заполнении формы ввода указывать несколько значений в одном поле ввода.

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
tableName	String	Название используемой в качестве связи многие-ко-многим	Обязательный
keyFieldName	String	Название ключевого поля из таблицы связи	Обязательный
baseFieldName	String	Название ссылочного поля из таблицы связки для связи с главной сущностью	Обязательный
dictionaryFieldName	String	Название ссылочного поля из таблицы связки для связи со справочника (Dictionary)	Обязательный
cacheDurationInSeconds	UnsignedInt	Длительность хранения кеша. 0 – кеш не используется	Необязательный
cacheByUser	Boolean	Требуется кешировать данные в рамках пользователя	Необязательный
newIdMode	NewIdModesEnum	Способ генерации первичного ключа	Необязательный

Пример

```
<Field fieldName="VIRTUAL_REASONS" fieldType="Complex">
    <Data>
```

```
        <Dictionary tableName="LIC_REASONS" keyFieldName="ID" displayFieldName="NAME"
orderFieldName="NAME" />
        <Subset tableName="LIC_APPLICATION_REASONS" keyFieldName="ID"
baseFieldName="LIC_APPLICATIONS_FK" dictionaryFieldName="LIC_REASONS_FK" />
    </Data>
    <Form addMode="NotVisible" editMode="Default" viewMode="Default" caption="Причины
обращения" input="MultiDropDown" size="Long" lineBreakAfterInput="true" />
</Field>
```

2.1.2.62.Entity/Fields/Field/Data/SelectFragment – Вычисляемое поле

Пустой элемент текстового типа. Содержит SQL-выражение для вычисляемого поля.

Пример

```
<Field fieldName="Owner" fieldType="String">
    <Data>
        <SelectFragment>
            <![CDATA[
isnull(SCHOOLS_FK_SCHOOLS.FULL_NAME,'') + isnull(INDIVIDUALS_FK_INDIVIDUALS.IP_NAME,'')
            ]]>
        </SelectFragment>
    </Data>
    <List caption="Обладатель лицензии" hidden="false" order="10" />
</Field>
```

2.1.2.63.Entity/Fields/Field/Data/WhereFragment – Фильтрация по вычисляемому полю

Пустой элемент текстового типа. Содержит SQL -выражение для фильтрации (если требуется нестандартная фильтрация или используется вычисляемое поле).

В раздел передаются параметры, указанные в блоке фильтров. Параметры формируются по следующему правилу: @ + <название таблицы> + _ + <название атрибута>

Пример

```
<WhereFragment>
    <![CDATA[MonitoringBaseViewTable.ExamDate = @MonitoringBaseViewTable_ExamDate]]>
</WhereFragment>
```

2.1.2.64.Entity/Fields/Field/Data/SortFragment – Сортировка по вычисляемому полю

Пустой элемент текстового типа. Содержит SQL -выражение для сортировки (если требуется нестандартная сортировка или используется вычисляемое поле).

Пример

```
<SortFragment>
   <![CDATA[
isnull(SCHOOLS_FK_SCHOOLS.FULL_NAME,'') + isnull(INDIVIDUALS_FK_INDIVIDUALS.IP_NAME,'')
   ]]>
</SortFragment>
```

2.1.2.65.Entity/Fields/Field/Filter – Описание поля фильтра

Элемент комплексного типа. Описывает поле фильтра по заданному атрибуту.

Дочерние элементы:

- Picker (0..1) – Выбор из списка
- Buttons (0..1) – Кнопки

Атрибуты:

Имя атрибута	Тип Атрибута	Комментарий	Встречаемость
filterMode	FilterModesEnum	Режим фильтрации	Обязательный
caption	String	Заголовок фильтра (по умолчанию заголовок отсутствует)	Обязательный
input	FilterInputTypesEnum	Способ ввода данных (по умолчанию – None)	Обязательный
defaultValue	String	Значение по умолчанию. Для множественного выбора значения указываются через разделитель - запятая	Необязательный
order	unsignedInt	Порядковый номер поля ввода в фильтре (по умолчанию в порядке их объявления)	Необязательный
lineBreakAfterInput	Boolean	Включает ввод с новой строки следующего поля (по умолчанию – false, тогда следующее поля ввода появится на той же строке что и текущее)	Необязательный
size	InputSizesEnum	Ширина поля на форме (по умолчанию – Normal)	Необязательный
groupId	String	имя группы, к которой присоединяется поле ввода (по умолчанию группа отсутствует)	Необязательный
cssPrefix	String	Имя css-класса для стилизации	Необязательный
textRows	unsignedInt	Задает количество строк в текстовом поле ввода (по умолчанию – 1)	Необязательный

Пример

```
<Filter filterMode="Equals" caption="Статус" input="MultiDropDown" size="NormalLong"
lineBreakAfterInput="true" order="320" defaultValue="334B1ACC-6CD5-92C9-02E9-
118EA8D2339D,B5A6A4D6-FD3E-3F64-9A8F-6488F90FD481" />
```

2.1.2.66.Entity/Fields/Field/Filter/Picker – Выбор из списка

Элемент комплексного типа. Выбор значения или значений из списка.

Дочерние элементы:

- Dependency (0..N) – Зависимость полей

Атрибуты:

Имя атрибута	Тип Атрибута	Комментарий	Встречаемость
entityId	String	Наименование сущности, используемой для выбора записи	Обязательный
hasLink	Boolean	Отображать ссылку для перехода на выбранную запись	Необязательный

Пример

```
<Filter filterMode="Equals" caption="Вид организационно-правовой формы"
input="MultiPicker" size="NormalLong" lineBreakAfterInput="true" order="80">
    <Picker entityId="SCHOOL_PROPERTY_KINDS" />
</Filter>
```

2.1.2.67.Entity/Fields/Field/Filter/Picker/Dependency – Зависимость полей

Элемент комплексного типа. Позволяет при выборе значения из списка заполнять другое поле фильтра или формы на основе выбранной записи.

Атрибуты:

Имя атрибута	Тип Атрибута	Комментарий	Встречаемость
pickerFieldName	String	Наименование поля из списка (пикера)	Обязательный
formFieldName	String	Наименование поля на форме (главная сущность)	Обязательный

Пример

```
<Picker entityId="EDU_PROGRAMS_FOR_ACCR" hasLink="true">
    <Dependency pickerFieldName="CODE" formFieldName="CALC_EDU_PROGRAMS_CODE"/>
    <Dependency pickerFieldName="NAME" formFieldName="CALC_EDU_PROGRAMS_NAME"/>
</Picker>
```

2.1.2.68.Entity/Fields/Field/Filter/Buttons – Кнопки фильтров

Элемент комплексного типа. Выбор значения или значений из списка.

Дочерние элементы:

- Button (1..N) – Кнопка фильтра

2.1.2.69.Entity/Fields/Field/Filter/Buttons/Button/JsAction – JS-действие при нажатии кнопки

Элемент строкового типа. Содержит JS-выражения.

Пример

```
<JsAction>
    <![CDATA[
if ($('span.hidden').text()) {
alert( $('span.hidden').text().substring(1) );
return false;
}
    ]]>
</JsAction>
```

2.1.2.70.Entity/Fields/Field/List – Описание столбца списка

Элемент комплексного типа. Задает настройки отображения столбца в списке.

Дочерние элементы:

- StyleConditions (0..1) - настройка условной стилизации столбца\ячеек

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
hidden	Boolean	Поле скрыто по умолчанию	Необязательный
caption	String	Заголовок столбца в списке	Обязательный
order	unsignedInt	Порядковый номер (по умолчанию отображается в их порядке объявления в XML-файле)	Необязательный
groupId	unsignedInt	Наименование группы, в которую требуется включить столбец	Необязательный
cssPrefix	String	Указывает имя css-класса для данного поля, для применения необходимых настроек шрифта	Необязательный
formatSchema	String	Схема форматирования, которая задается в файле конфигурации	Необязательный
input	ListCellInputTypesEnum	Способ ввода данных (по умолчанию – None)	Необязательный

defaultValue	String	Значение по умолчанию	Необязательный
required	Boolean	Обязательный для ввода	Необязательный
inputMask	String	Маска ввода	Необязательный
maskValidationEnabled	Boolean	Использовать валидацию по маске ввода	Необязательный
size	ListCellInputTypesEnum	Размер (ширина) столбца	Необязательный
fixColumn	Boolean	Фиксировать столбец списка	Необязательный

2.1.2.71.Entity/Fields/Field/List/StyleConditions – Условии стилизация столбца

Элемент комплексного типа. Описывает группы полей фильтрации с помощью дочернего элемента FieldGroup. Группы полей (отображаются как блоки с заголовком).

Дочерние элементы:

- StyleCondition (1..N) – условие стилизации

Пример

```
<StyleConditions>
    <StyleCondition cssPrefix="grey">
        <![CDATA[
LIC_APPLICATION_STATUSES_FK_LIC_APPLICATION_STATUSES.PROCESS_CODE in
('FINISHED','RETURNED','CANCEL','ERROR')
]]>
    </StyleCondition>
</StyleConditions>
```

2.1.2.72.Entity/Fields/Field/List/StyleConditions/StyleCondition – Условие стилизации столбца

Элемент комплексного типа. Содержит имя css-класса и SQL-условие, при котором css-класс должен применяться.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
cssPrefix	String	Название css-класса	Обязательный

Пример

```
<StyleCondition cssPrefix="grey">
    <![CDATA[
LIC_APPLICATION_STATUSES_FK_LIC_APPLICATION_STATUSES.PROCESS_CODE in
('FINISHED','RETURNED','CANCEL','ERROR')
]]>
</StyleCondition>
```

2.1.2.73.Entity/Fields/Field/Form – Описание поля формы

Задает настройки формы редактирования.

- Picker (0..1) – выбор из списка
- SubGrid (0..1) – подсписок
- ViewReport – визуализация отчета
- File (0..1) – управление файлом
- Buttons (0..1) – кнопки на форме
- JsOnChange – JS-выражение при изменении значения поля

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
addMode	FormFieldModesEnum	Режим создания	Необязательный
editMode	FormFieldModesEnum	Режим редактирования	Необязательный
viewMode	FormFieldModesEnum	Режим просмотра	Необязательный
caption	String	Задает заголовок	Необязательный
input	FormInputTypesEnum	Способ ввода данных	Необязательный
defaultValue	String	Значение по умолчанию	Необязательный
order	unsignedInt	Порядковый номер поля ввода (по умолчанию поля появляются в порядке их объявления)	Необязательный
lineBreakAfterInput	Boolean	Включает ввод с новой строки следующего поля (по умолчанию – false, тогда следующее поля ввода появится на той же строке что и текущее)	Необязательный
size	InputSizesEnum	Ширина поля на форме (по умолчанию – Normal)	Необязательный
required	Boolean	Поле обязательно для заполнения на форме (По умолчанию – false)	Необязательный
groupId	String	Id группы/подгруппы полей ввода, к которой прикрепляется данное поле ввода (по умолчанию – нет группы)	Необязательный
cssPrefix	String	Настраивает css для отдельного поля формы	Необязательный
formatSchema	String	Задает схему форматирования для отдельного поля формы	Необязательный
textRows	unsignedInt	Задает количество строк в текстовом поле ввода (по умолчанию – 1)	Необязательный
inputMask	String	Маска ввода	Необязательный
maskValidationEnabled	Boolean	Включить валидацию по маске ввода	Необязательный
hideIfEmpty	Boolean	Скрывать поле, если пустое	Необязательный

2.1.2.74.Entity/Fields/Field/Form/Picker – Выбор из списка

Элемент комплексного типа. Выбор значения или значений из списка.

Дочерние элементы:

- Dependency (0..N) – Зависимость полей

Атрибуты:

Имя атрибута	Тип Атрибута	Комментарий	Встречаемость
entityId	String	Наименование сущности, используемой для выбора записи	Обязательный
hasLink	Boolean	Отображать ссылку для перехода на выбранную запись	Необязательный

Пример

```
<Filter filterMode="Equals" caption="Вид организационно-правовой формы"
input="MultiPicker" size="NormalLong" lineBreakAfterInput="true" order="80">
    <Picker entityId="SCHOOL_PROPERTY_KINDS" />
</Filter>
```

2.1.2.75.Entity/Fields/Field/Form/Picker/Dependency – Зависимость полей

Элемент комплексного типа. Позволяет при выборе значения из списка заполнять другое поле фильтра или формы на основе выбранной записи.

Атрибуты:

Имя атрибута	Тип Атрибута	Комментарий	Встречаемость
pickerFieldName	String	Наименование поля из списка (пикера)	Обязательный
formFieldName	String	Наименование поля на форме (главная сущность)	Обязательный

Пример

```
<Picker entityId="EDU_PROGRAMS_FOR_ACCR" hasLink="true">
    <Dependency pickerFieldName="CODE" formFieldName="CALC_EDU_PROGRAMS_CODE"/>
    <Dependency pickerFieldName="NAME" formFieldName="CALC_EDU_PROGRAMS_NAME"/>
</Picker>
```

2.1.2.76.Entity/Fields/Field/Form/SubGrid

Пустой элемент. С помощью атрибутов задаёт параметры подсписка, связанного с текущей формой по полю данной формы как 1-N. Подсписок может находиться на форме в любой очередности с другими полями, а также крепиться к группам полей.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
fieldName	String	Наименование поля главной сущности для связи с подсписком	Обязательный
entityId	String	Наименование сущности, используемой в качестве подсписка	Обязательный
filterFieldName	String	Наименование поля, по которому список связан с полем формы	Обязательный

Пример

```
<Field fieldName="IDsub2" fieldType="Complex">
    <Form addMode="Default" editMode="Default" viewMode="Default" input="SubGrid"
caption="Раздел 1" lineBreakAfterInput="true" order="45" groupId="BRANCH">
        <SubGrid fieldName="ID" entityId="SCHOOLS_FOR_BRANCH"
filterFieldName="PARENT_SCHOOL_FK"   />
    </Form>
</Field>
```

2.1.2.77.Entity/Fields/Field/Form/ViewReport

Пустой элемент. С помощью атрибутов задаёт параметры отображения отчета на форме.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
reportCode	String	Наименование отчета, используемого для отображения на форме	Обязательный
showGenerateButton	Boolean	Отображать кнопку генерации отчета	Обязательный
generateButtonText	String	Наименование кнопки генерации отчета	Обязательный

2.1.2.78.Entity/Fields/Field/Form/File

Пустой элемент. В этом разделе задаются параметры для управления файлом.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
fileContentField	String	Наименование поля, содержащего информацию о файле (сам файл, либо реквизит доступа к файлу)	Обязательный
fileNameField	String	задает тип файлового поля	Обязательный

Пример

```
<Field fieldName="SELF_EXAM_REPORT_FILE" fieldType="Complex">
    <Form addMode="Default" editMode="Default" viewMode="Default" caption="Сведения о
реализации основных образовательных программ" input="File" size="NormalLong"
textRows="1" lineBreakAfterInput="true" required="false" order="140" groupId="Dop">
```

```
        <File fileNameField="SELF_EXAM_REPORT_FILE_NAME"
fileContentField="SELF_EXAM_REPORT_FILE_PATH"/>
    </Form>
</Field>
```

2.1.2.79.Entity/Fields/Field/Form/Buttons

Пустой элемент. В этом разделе задаются описание кнопок на форме.

Пример

```
<Field fieldName="ButtonsControl1" fieldType="Complex">
    <Form caption="" input="Buttons" order="82" addMode="NotVisible"
editMode="NotVisible" viewMode="Default">
        <Buttons>
            <Button caption="Отправить на подпись">
                <SqlAction id="CERTIFICATES_Tab1"
connectionStringName="AKNDPP_REGIONAL">
                    <![CDATA[
UPDATE dbo.CERTIFICATES SET SIGN_STATUSES_FK = (SELECT ID FROM dbo.SIGN_STATUSES WHERE
CODE = 'READY')
WHERE ID = @CurrentObjectId
                    ]]>
                </SqlAction>
                <JsAction>
                    <![CDATA[
alert( 'Свидетельство отправлено на подпись руководителю!' ) ;
                    ]]>
                </JsAction>
            </Button>
            <Button caption="Печатная форма свидетельства">
                <JsAction>
                    <![CDATA[
self.location.href =
'../Reporting/Report.ashx?code=REPORT_ACCREDITATION&format=WORD&p3=RecordId=' +
getUrlVar()['RecordId'] ;
                    ]]>
                </JsAction>
            </Button>
        </Buttons>
    </Form>
</Field>
```

2.1.2.80.Entity/Fields/Field/Form/Buttons/Button – Кнопка в панели управления списка

Элемент комплексного типа. Задает отображение панели управления списком.

Дочерние элементы:

- JsAction (0..N) – Выполнение JS-выражения
- SaveAction (0..1) – Выполнение сохранения данных
- SqlAction (0..N) – Выполнение SQL-выражения
- UrlAction (0..1) – Выполнение перехода по ссылке
- RefreshAction (0..1) – Выполнение обновления данных
- CustomAction (0..N) – Доп. действие (определяется в C#)

Выполняемые действия в рамках одной кнопки могут комбинироваться в любом порядке.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
caption	String	Название кнопки	Обязательный
cssPrefix	String	Имя css-класса для стилизации кнопки	Необязательный

Пример

```
<Button caption="Отправить на подпись">
     <SqlAction id="CERTIFICATES_Tab1" connectionStringName="AKNDPP_REGIONAL">
         <![CDATA[
UPDATE dbo.CERTIFICATES SET SIGN_STATUSES_FK = (SELECT ID FROM dbo.SIGN_STATUSES WHERE
CODE = 'READY')
WHERE ID = @CurrentObjectId
         ]]>
      </SqlAction>
      <JsAction>
         <![CDATA[
alert( 'Свидетельство отправлено на подпись руководителю!' ) ;
         ]]>
      </JsAction>
</Button>
```

2.1.2.81.Entity/Fields/Field/Form/Buttons/Button/JsOnClick – JS-действие при нажатии кнопки

Элемент строкового типа. Содержит JS-выражения.

Пример

```
<JsAction>
    <![CDATA[
if ($('span.hidden').text()) {
alert( $('span.hidden').text().substring(1) );
return false;
}
    ]]>
</JsAction>
```

2.1.2.82.Entity/Fields/Field/Form/Buttons/Button/UrlAction – Переход по ссылке при нажатии кнопки

Элемент простого типа без вложений.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
url	String	Название кнопки	Обязательный

В значение параметра возможно добавление переменных – переменная должна быть заключена в фигурные скобки. В составе переменной указывается название таблицы и название поля.

Пример

```
<UrlAction
url="../Generic/Edit_A.aspx?RecordId={LIC_APPLICATIONS.ID}&amp;EntityId=LIC_APPLICATION
S_Tab1"/>
```

2.1.2.83.Entity/Fields/Field/Form/Buttons/Button/RefreshAction – Обновление данных при нажатии кнопки

Элемент комплексного типа. Задает признак необходимости обновления данных.

Пример

```
<RefreshAction />
```

2.1.2.84.Entity/Fields/Field/Form/Buttons/Button/CustomAction – Дополнительное действие при нажатии кнопки

Элемент простого типа без вложений. Задает отображение панели управления списком.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Идентификатор действия	Обязательный

2.1.2.85.Entity/Fields/Field/Form/JsOnChange

Элемент строкового типа. Содержит JS-выражение, выполняемое при изменении значения поля.

Скрипт выполняется не только после изменения значения в поле, но и сразу после загрузки страницы (для отработки варианта, когда условное значение уже сохранено в БД и требуется выполнить ряд действий, связанных с ним).

Внутрь скрипта всегда передается 3 значения:

- text - текст, отображаемый пользователю.
- value - значение элемента управления (для текстовых полей, чисел и дат совпадает с text)
- id - идентификатор текущего поля ввода в html-коде страницы

Пример

```
<Form caption="Тип заявителя" input="DropDown" groupId="Owner" required="true"
addMode="NotVisible" viewMode="Default" editMode="ReadOnlyText">
    <JsOnChange>
        <![CDATA[
var ptype = GetFormFieldValue('CERTIFICATES_TYPE')
if(ptype == 'UL')
{
ShowFormField('CERTIFICATES_SCHOOLS_FK');
HideFormField('CERTIFICATES_INDIVIDUALS_FK');
}
else if(ptype == 'IP')
{
```

```
ShowFormField('CERTIFICATES_INDIVIDUALS_FK');
HideFormField('CERTIFICATES_SCHOOLS_FK');
}
else
{
HideFormField('CERTIFICATES_SCHOOLS_FK');
HideFormField('CERTIFICATES_INDIVIDUALS_FK');
}
        ]]>
    </JsOnChange>
</Form>
```

2.1.2.86.Entity/Fields/Relation

Задает связь между полем связи таблицы и таблицей, на которую ссылается поле связи. Если элемент объявлен внутри элемента Fields, то поле связи является полем основной таблицы списка (таблицы, указанной в ~/Entity/DataSettings). Если элемент объявлен внутри другого элемента Relation, то полем связи является полем таблицы, указанной в родительском элементе Relation.

Дочерние элементы:

- JoinCondition (0..1) – дополнительное условие связки
- Field (1..N) – атрибут\поле из связанной таблице
- Relation (0..N) – вложенное отношение

И содержит три обязательных атрибута:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
fieldName	String	имя поля связи	Обязательный
relatedTableName	String	имя присоединяемой к полю таблицы	Обязательный
relatedTableField	String	Указывает поле на которое ссылается присоединяемая таблица, если связь таблиц обратная (присоединяем главную таблицу)	Необязательный
relatedTableAlias	String	Задание альтернативного имени связанной таблицы внутри списка, для возможности использования нескольких таблиц с одним именем в списке.	Необязательный
mustExist	Boolean	Тип соединения (inner join – true, left join – false)	Обязательный

Пример

```
<Relation fieldName="SCHOOLS_FK" relatedTableName="SCHOOLS" relatedTableField="ID"
relatedTableAlias="SCHOOLS_FK_SCHOOLS" mustExist="false">
    <Field fieldName="ID" fieldType="Guid" isPrimaryKey="true"/>
    <Field fieldName="INN" fieldType="String">
        <Filter filterMode="Substring" caption="ИНН ОО" input="Text" size="Normal"
lineBreakAfterInput="true" order="500" />
    </Field>
</Relation>
```

2.1.2.87.Entity/Fields/Relation/JoinCondition

Задает дополнительное условие на присоединение таблицы посредством явного SQL-выражения.

Пример

```
<Relation fieldName="SCHOOLS_FK" relatedTableName="SCHOOLS" relatedTableField="ID"
relatedTableAlias="SCHOOLS_FK_SCHOOLS" mustExist="false">
    <JoinCondition>
        <![CDATA[
AND UpdateMonitoring.system_id = version_id_UpdateVersionLog.systemId AND
UpdateMonitoring.start_datetime < version_id_UpdateVersionLog.datetime AND
UpdateMonitoring.finish_datetime > version_id_UpdateVersionLog.datetime
        ]]>
    </JoinCondition>
    <Field fieldName="ID" fieldType="Guid" isPrimaryKey="true"/>
    <Field fieldName="INN" fieldType="String">
        <Filter filterMode="Substring" caption="ИНН ОО" input="Text" size="Normal"
lineBreakAfterInput="true" order="500" />
    </Field>
</Relation>
```

2.1.3. Справочные элементы

2.1.3.1. NewIdModesEnum – режимы создания первичного ключа

Элемент типа string, не содержит дочерних элементов и атрибутов. Может принимать одно из следующих значений:

- None – не требуется
- ServerGuid - значение генерируется сервером
- Database - значение генерируется БД
- UserInput - значение задается вручную

2.1.3.2. FormFieldModesEnum – режимы видимости полей на форме просмотра\корректировки данных

Элемент типа string, не содержит дочерних элементов и атрибутов. Принимает одно из следующих значений:

- Default – по умолчанию (определяется настройками ролевой модели и также режимом отображения карточки в целом)
- Control – отображается как элемент управления (например, текст. поле ввода или календарь)
- ReadOnlyControl – отображается как элемент управления только на чтение
- ReadOnlyText – отображается как текст
- NotVisible – не отображается (скрыто)

2.1.3.3. FieldDataTypesEnum – типы данных атрибутов сущности

Элемент типа string, не содержит дочерних элементов и атрибутов. Задает один из следующих типов форматов поля:

- String – текстовый
- Integer – целочисленный
- Float – вещественный
- Decimal - вещественный
- DateTime – формат даты
- Boolean – логический формат
- Guid – уникальный идентификатор
- Complex - сложный тип атрибута (используется для элементов: подсписок, работа с файлом, множественный выбор из списка, кнопки и другие элементы управления)

2.1.3.4. FilterInputTypesEnum – типы элементов управления в блоке фильтра

Элемент типа string, не содержит дочерних элементов и атрибутов. Задает один из следующих типов форматов, вводимых данных в фильтре:

- Text – текстовый ввод
- Boolean –выпадающий список из трех вариантов: да, нет, любой
- StrictBoolean – элемент управления «CheckBox»
- RadioBoolean – элемент управления «радиокнопки», доступны варианты: да, нет, все
- Date – выбор даты (без времени)
- DateTime – выбор даты и времени
- DatesRange – диапазон нескольких дат
- DateTimesRange – диапазон нескольких дат и времени
- DropDown – выпадающий список
- MultiDropDown – выпадающий список с множественным выбором значений
- AutoCompleteDropDown – список вариантов по поисковой строке
- LazyDropDown – выпадающий список отложенной инициализации (инициализируется в момент обращения к нему)
- RelatedDropDowns – связанный выпадающий список
- Number – ввод числа
- NumbersRange – диапазон нескольких чисел
- Picker – выбор значения из всплывающего окна со списком
- MultiPicker – множественный выбор значения из всплывающего окна со списком
- Buttons – кнопки управления
- Custom – прикладной элемент управления (реализуется в прикладной системе по правилам платформы)

2.1.3.5. FilterModesEnum – режимы фильтрации данных

Простой элемент текстового типа, задает режим фильтрации. Принимает одно из следующих значений:

- Equals – значение равное заданному
- Substring – часть строки
- FullText – полный текст
- Between – значение, попадающее в заданный диапазон
- In - значение равное одному из множества заданных значений
- EqualsIgnoreCase – значения равное заданному без учета регистра.

- Custom – фильтрация по sql-условию в веденному в xml списка.
- CustomIn – фильтрация по sql-условию в веденному в xml списка равное одному из множества заданных значений.
- Default – по умолчанию (в зависимости от типа атрибута и элемента управления)

2.1.3.6. ListCellInputTypesEnum - типы элементов управления в списке

Простой элемент текстового типа, задает тип форма для вводимых данных. Принимает одно из следующих значений:

- None – ячейка не является редактируемой (по умолчанию)
- Text – текстовый ввод
- Boolean – элемент управления «CheckBox»
- Date – выбор даты (без времени)
- DateTime – выбор даты и времени
- DropDown – выпадающий список
- AutoCompleteDropDown – список вариантов по поисковой строке
- LazyDropDown – выпадающий список отложенной инициализации (инициализируется в момент обращения к нему)

2.1.3.7. FormInputTypesEnum - типы элементов управления в блоке формы просмотра\корректировки данных

Простой элемент текстового типа, задает тип форма для вводимых данных. Принимает одно из следующих значений:

- Text – текстовый ввод
- Boolean – элемент управления «CheckBox»
- StrictBoolean –
- Date – выбор даты (без времени)
- DateTime – выбор даты и времени
- DropDown – выпадающий список
- MultiDropDown – выпадающий список с множественным выбором значений
- AutoCompleteDropDown – список вариантов по поисковой строке
- LazyDropDown – выпадающий список отложенной инициализации (инициализируется в момент обращения к нему)
- RelatedDropDowns – связанный выпадающий список
- LongText –ввод большого текста
- Password – элемент ввода защищенных символов (ввод экранируется символом *)
- Picker – выбор значения из всплывающего окна со списком
- MultiPicker – множественный выбор значения из всплывающего окна со списком
- SubGrid – элемент управления «подсписок»
- ViewReport – элемент управления «отчет»
- File – элемент управления файлом (загрузка, выгрузка, удаление)
- Buttons – кнопки управления
- Custom – прикладной элемент управления (реализуется в прикладной системе по правилам платформы)

2.1.3.8. InputSizesEnum – размеры элементов управления

Простой элемент текстового типа. Задает длину поля ввода на форме. Принимает одно из трех значений:

- Long – длинный
- NormalLong – средне-длинный
- Normal – нормальный
- ShortNormal – средне-короткий
- Short – короткий

2.1.3.9. FirstLastButtonTextModesEnum – режимы отображения кнопок навигации списка

Задает вид отображения кнопок перехода на первую и последнюю страницы списка. Принимает одно из двух значений:

- ExplicitText – отображение в виде текста, например, Первая, Последняя
- PageNumber – отображение в виде номера страниц, например, 1, 20, где 20 – номер последней страницы списка.

2.1.3.10. PagerButtonsOrdersEnum – последовательности отображения кнопок навигации списка

Задает порядок следования кнопок перехода между предыдущей, следующей и первой и последней страницами. Принимает одно из двух значений:

- FirstPrevPagesNextLast – кнопки перехода на предыдущую/следующую страницы расположены между кнопками перехода на первую/последнюю страницы
- PrevFirstPagesLastNext – кнопки перехода на первую/последнюю страницы расположены между кнопками перехода на предыдущую/следующую страницы

2.1.3.11. PanelPositionsEnum – режимы отображения панелей списка

Задает тип отображения информации о записях в списке (кол-во страниц, кол-во записей на странице и т.д.). Принимает одно из 4-х значений:

- Top – информация отображается только над списком
- Bottom – информация отображается только под списком
- Both – информация отображается под и над списком
- None – информация не отображается

2.1.3.12. ListCacheTypesEnum – режимы кеширования данных списка

Время жизни кэшируемого объекта, в секундах

- None – объект не помещается в кэш
- FilterData – кэширование фильтрованных данных
- AllData – кэширование всех данных

2.1.4. Таблицы соответствий типов данных и типов элементов управления

Тип данных	Тип элемента управления в фильтре	Тип элемента управления в форме	Дополнительные требования к описанию
FieldDataTypesEnum	**FilterInputTypesEnum**	**FormInputTypesEnum**	
Integer	Text	Text	
Integer	DropDown	DropDown	Требуется описание Dictionary
Integer	MultiDropDown	-	Требуется описание Dictionary
Integer	Picker	Picker	Требуется описание Picker
Integer	MultiPicker	-	Требуется описание Picker
String	Text	Text	
DateTime	Date	Date	
DateTime	DatesRange	-	
DateTime	DateTime	DateTime	
DateTime	DateTimesRange	-	
Boolean	Boolean	Boolean	
Boolean	StrictBoolean	-	
Boolean	RadioBoolean	-	
Guid	Text	Text	
Guid	DropDown	DropDown	Требуется описание Dictionary
Guid	MultiDropDown	-	Требуется описание Dictionary
Guid	Picker	Picker	Требуется описание Picker
Guid	MultiPicker	-	Требуется описание Picker
Complex	-	SubGrid	Требуется описание SubGrid
Complex	-	ViewReport	Требуется описание ViewReport
Complex	-	File	Требуется описание File
Complex	Buttons	Buttons	Требуется описание Buttons
Complex	-	MultiDropDown	Требуется описание Dictionary и Subset
Complex	-	MultiPicker	Требуется описание Dictionary и Subset

2.1.5. Платформенные SQL-параметры

В тексты SQL-выражений можно использовать стандартные параметры:

- @CurrentUserId - текущий пользователь
- @CurrentRecordId – идентификатор объекта, отображаемого в текущей карточке (данный параметр передается, только если список работает в контексте родительской карточки)

- @OwnerRecordId – идентификатор родительского объекта, параметр передается в дочернюю сущность
- @OwnerEntityId – код сущности родительского объекта, параметр передается в дочернюю сущность
- Параметры, формируемые по следующему правилу: @ + <название атрибута>

2.1.6. Платформенные JS-функции

Платформенные JS-функции можно использовать в качестве готовых методов при описании алгоритмов на JavaScript.

В метаданных сущности определены следующие типы событий:

- JsSection – загрузка списка или формы. В секции могут определяться прикладные функции и вызовы функций
- JsOnChange – изменение состояния поля. Событие срабатывает при установке значения в поле формы или фильтра
- JsOnClick – нажатие кнопки. Событие срабатывает при нажатии прикладной кнопки списка, фильтра или формы
- JsValidation – валидация сохраняемых данных на стороне клиента. Событие срабатываем при попытке сохранения данных в списке или форме

Далее приведены описания JS-функций, которые можно использовать в тексте JS-выражений.

2.1.6.1. KURS.getFormClientId – получение уникального признака экземпляра формы

Старое название метода – GetFormClientIdOrDefault

Для миграции достаточно переименовать

Описание:

Вспомогательная функция обеспечивает получение идентификатора формы на странице. Обеспечивает получение идентификатора (уникального признака экземпляра) формы по вложенному в нее элементу (кнопке, полю и т.п.) или по идентификатору сущности - в случае, если форм на странице несколько, например, при использовании popup-форм.

Определение функции:

KURS.getFormClientId (element, entityId)

Описание входных параметров:

- **element** - jQuery-объект, соответствующий элементу формы (чаще всего - кнопки, поля). В случае, если параметр не передан - будет возвращен идентификатор самой первой формы на странице. Параметр определяется системой. В большинстве случаев по событию нажатия кнопки в качестве значения параметра передается $sender. **НЕОБЯЗАТЕЛЬНЫЙ**
- **entityId** –получение идентификатора (уникального признака) формы по идентификатору сущности. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки – идентификатор (уникальный признак экземпляра) формы на странице

Пример

```
var formClientId = KURS.getFormClientId();
var formClientId = KURS.getFormClientId($sender);
```

2.1.6.2. KURS.getCellRecordId– получение идентификатора строки списка

Описание:

Вспомогательная функция обеспечивает получение идентификатора формы на странице. Обеспечивает получение идентификатора (уникального признака экземпляра) формы по вложенному в нее элементу (кнопке, полю и т.п.) или по идентификатору сущности - в случае, если форм на странице несколько, например, при использовании popup-форм.

Определение функции:

KURS.getCellRecordId ($cellElement)

Описание входных параметров:

- **$cellElement** - . **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки – идентификатор (уникальный признак экземпляра) формы на странице

Пример

```
var formClientId = KURS.getFormClientId();
var formClientId = KURS.getFormClientId($sender);
```

2.1.6.3. KURS.getFormRecordId– получение идентификатора записи формы

Описание:

Вспомогательная функция обеспечивает получение идентификатора формы на странице. Обеспечивает получение идентификатора (уникального признака экземпляра) формы по вложенному в нее элементу (кнопке, полю и т.п.) или по идентификатору сущности - в случае, если форм на странице несколько, например, при использовании popup-форм.

Определение функции:

KURS.getFormRecordId (formClientId)

Описание входных параметров:

- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки – идентификатор записи формы на странице

Пример

```
var formClientId = KURS.getFormClientId();
var formClientId = KURS.getFormClientId($sender);
```

2.1.6.4. KURS.getListClientId – получение идентификатора списка

Описание:

Вспомогательная функция обеспечивает получение идентификатора списка на странице. Обеспечивает получение идентификатора (уникального признака экземпляра) списка по вложенному в нее элементу (кнопке, полю и т.п.) или по идентификатору сущности - в случае, если списков на странице несколько, например, при использовании popup-форм.

Определение функции:

KURS.getListClientId (element, entityId)

Описание входных параметров:

- **element** - jQuery-объект, соответствующий элементу формы (чаще всего - кнопки, поля). В случае, если параметр не передан - будет возвращен идентификатор самой первой формы на странице. Параметр определяется системой. В большинстве случаев по событию нажатия кнопки в качестве значения параметра передается $sender. **НЕОБЯЗАТЕЛЬНЫЙ**
- **entityId** –получение идентификатора (уникального признака) списка по идентификатору сущности. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки – идентификатор (уникальный признак экземпляра) формы на странице

Пример

```
var listClientId = KURS.getListClientId ();
var listClientId = KURS.getListClientId ($sender);
```

2.1.6.5. KURS.getForm – получение jQuery-объекта формы

Описание:

Функция обеспечивает получение объекта формы на странице по вложенному в нее элементу (кнопке, полю и т.п.) или по идентификатору сущности - в случае, если форм на странице несколько, например, при использовании popup-форм.

Определение функции:

KURS.getForm (element, entityId)

Описание входных параметров:

- **element** - jQuery-объект, соответствующий элементу формы (чаще всего - кнопки, поля). В случае, если параметр не передан - будет возвращен идентификатор самой первой формы

на странице. Параметр определяется системой. В большинстве случаев по событию нажатия кнопки в качестве значения параметра передается $sender. **НЕОБЯЗАТЕЛЬНЫЙ**

- **entityId** –получение идентификатора (уникального признака) списка по идентификатору сущности. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- jQuery-объект формы

Пример

```
var form = KURS.getForm ();
var form = KURS.getForm ($sender);
```

2.1.6.6. KURS.getFormMode – получение режима отображения формы

Описание:

Функция обеспечивает получение объекта формы на странице по вложенному в нее элементу (кнопке, полю и т.п.) или по идентификатору сущности - в случае, если форм на странице несколько, например, при использовании popup-форм.

Определение функции:

KURS.getFormMode (element, entityId)

Описание входных параметров:

- **element** - jQuery-объект, соответствующий элементу формы (чаще всего - кнопки, поля). В случае, если параметр не передан - будет возвращен идентификатор самой первой формы на странице. Параметр определяется системой. В большинстве случаев по событию нажатия кнопки в качестве значения параметра передается $sender. **НЕОБЯЗАТЕЛЬНЫЙ**
- **entityId** –получение идентификатора (уникального признака) списка по идентификатору сущности. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- jQuery-объект формы

Пример

```
var form = KURS.getForm ();
var form = KURS.getForm ($sender);
```

2.1.6.7. KURS.getList – получение jQuery-объекта списка

Описание:

Функция обеспечивает получение объекта списка на странице по вложенному в нее элементу (кнопке, полю и т.п.) или по идентификатору сущности - в случае, если списков на странице несколько, например, при использовании popup-форм.

Определение функции:

KURS.getList (element, entityId)

Описание входных параметров:

- **element** - jQuery-объект, соответствующий элементу формы (чаще всего - кнопки, поля). В случае, если параметр не передан - будет возвращен идентификатор самой первой формы на странице. Параметр определяется системой. В большинстве случаев по событию нажатия кнопки в качестве значения параметра передается $sender. **НЕОБЯЗАТЕЛЬНЫЙ**
- **entityId** –получение идентификатора (уникального признака) списка по идентификатору сущности. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- jQuery-объект списка

Пример

```
var form = KURS.getForm ();
var form = KURS.getForm ($sender);
```

2.1.6.8. KURS.getListCellValue - получение значения поля списка

Описание:

Функция предназначена для получения значения с поля формы и последующей обработки полученного значения. Может быть использована при наступлении некоторого события, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.getListCellValue (recordId, fieldName, listId)

Описание входных параметров:

- **recordId** – идентификатор записи
- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- listId - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки, для ссылочных полей возвращается идентификатор

Пример

```
var type_fk = KURS.getFormFieldValue ('LIC_APPLICATIONS_APPLICANT_TYPES_FK') ;
```

2.1.6.9. KURS.getListAttrValue - получение значения служебного параметра списка

Описание:

Функция предназначена для получения значения с поля формы и последующей обработки полученного значения. Может быть использована при наступлении некоторого события,

например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.getListAttrValue (recordId, fieldName, listId)

Описание входных параметров:

- **recordId** – идентификатор записи
- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- listId - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки, для ссылочных полей возвращается идентификатор

Пример

```
var type_fk = KURS.getFormFieldValue ('LIC_APPLICATIONS_APPLICANT_TYPES_FK') ;
```

2.1.6.10.KURS.setListCellValue - установка значения поля списка

Описание:

Функция предназначена для установки значения в поле формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.setListCellValue (recordId, fieldName, value, formClientId)

Описание входных параметров:

- **recordId** – идентификатор записи
- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **value** – значение в виде строки, для ссылочных полей передается идентификатор. **ОБЯЗАТЕЛЬНЫЙ**
- listId - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.setFormFieldValue ('LIC_APPLICATIONS_CREATE_NEW_APPLICANT', 'false') ;
KURS.setFormFieldValue ('LIC_APPLICATIONS_REG_DATE', '') ;
KURS.setFormFieldValue ('FOUNDERS_COUNTRIES_FK', '643eeeee-eeee-eeee-eeee-eeeeeeeeeeee') ;
```

2.1.6.11.KURS.getFormFieldValue - получение значения поля формы

Описание:

Функция предназначена для получения значения с поля формы и последующей обработки полученного значения. Может быть использована при наступлении некоторого события, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.getFormFieldValue (fieldName, formClientId)

Описание входных параметров:

- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки, для ссылочных полей возвращается идентификатор

Пример

```
var type_fk = KURS.getFormFieldValue ('LIC_APPLICATIONS_APPLICANT_TYPES_FK') ;
```

2.1.6.12.KURS.setFormFieldValue - установка значения поля формы

Описание:

Функция предназначена для установки значения в поле формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.setFormFieldValue (fieldName, value, formClientId)

Описание входных параметров:

- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **value** – значение в виде строки, для ссылочных полей передается идентификатор. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.setFormFieldValue ('LIC_APPLICATIONS_CREATE_NEW_APPLICANT', 'false') ;
KURS.setFormFieldValue ('LIC_APPLICATIONS_REG_DATE', '') ;
KURS.setFormFieldValue ('FOUNDERS_COUNTRIES_FK', '643eeeee-eeee-eeee-eeee-eeeeeeeeeeee') ;
```

2.1.6.13.KURS.getFilterFieldValue - получение значения поля фильтра

Описание:

Функция предназначена для получения значения с поля фильтра и последующей обработки полученного значения. Может быть использована при наступлении некоторого события, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке списка (JsSection).

Определение функции:

KURS.getFilterFieldValue (fieldName, listClientId)

Описание входных параметров:

- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **listClientId** - идентификатор списка на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки, для ссылочных полей возвращается идентификатор

Пример

```
var result = KURS.getFilterFieldValue ('LIC_APPLICATIONS_REG_DATE');
```

2.1.6.14.KURS.setFilterFieldValue - установка значения поля фильтра

Описание:

Функция предназначена для установки значения в поле фильтра по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке списка (JsSection).

В случае если параметр listClientId не передан - для его получения будет автоматически вызван метод KURS.getListClientId

Определение функции:

KURS.setFilterFieldValue (fieldName, value, listClientId)

Описание входных параметров:

- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **value** – значение в виде строки, для ссылочных полей передается идентификатор. **ОБЯЗАТЕЛЬНЫЙ**
- **listClientId** - идентификатор списка на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.setFilterFieldValue ('LIC_APPLICATIONS_REG_DATE', '01.01.2017') ;
```

2.1.6.15.KURS.applyExternalFilter – применение произвольного фильтра для списка\подсписка

Описание:

Функция обновления списка по некоторому событию.

Определение функции:

KURS.applyExternalFilter (entityId, filterString, listClientId)

Описание входных параметров:

- **entityId** - идентификатор метаданных. **ОБЯЗАТЕЛЬНЫЙ**
- **filterString** - строка фильтра, аналогичная url-строке. **ОБЯЗАТЕЛЬНЫЙ**
- **listClientId** - идентификатор списка на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.applyExternalFilter ('SearchOO_GRID',
'SortDir=Ascending&SortCol=vwm_SCHOOLS.Name&vwm_SCHOOLS.NAME_OR_INN_FILTER=9909437740&vwm_SCHOOLS.REGIONFK=5&vwm_SCHOOLS.AKND_CHECK_PURPOSE_FILTER=3|4&ID_vwm_SCHOOLS_ADDITIONAL.HAS_NOTFILLED_PRESCRIPTIONS=false&vwm_SCHOOLS.AKND_COUNT_DAYS_FROM_FILTER_tovalue=101&vwm_SCHOOLS.PARENTFK=0C44556423C738225AE7BEB1D59FF5B0')
```

2.1.6.16.KURS.hideFormField - скрытие поля формы

Описание:

Функция предназначена для скрытия поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.hideFormField (fieldName, formClientId)

Описание входных параметров:

- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.hideFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
KURS.hideFormField ('Users_CERTIFIED_EXPERTS_FK');
```

2.1.6.17.KURS.showFormField – снятие скрытия с поля формы

Описание:

Функция предназначена для снятия скрытия с поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.showFormField (fieldName, formClientId)

Описание входных параметров:

- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.showFormField ('Users_CERTIFIED_EXPERT_ORGANIZATIONS_FK');
KURS.showFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
```

2.1.6.18.KURS.toggleFormField – переключение видимости поля

Описание:

Функция предназначена для переключения видимости поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Если поле скрыто (через HideFormField), то при выполнении функции ToggleFormField поле отобразится, иначе при выполнении функции ToggleFormField поле скроется.

Определение функции:

KURS.toggleFormField (fieldName, formClientId)

Описание входных параметров:

- **fieldName** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.toggleFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
```

2.1.6.19.KURS.hideFormGroup - скрытие поля формы

Описание:

Функция предназначена для скрытия поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.hideFormGroup (groupId, formClientId)

Описание входных параметров:

- **groupId** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.hideFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
KURS.hideFormField ('Users_CERTIFIED_EXPERTS_FK');
```

2.1.6.20.KURS.showFormGroup - скрытие поля формы

Описание:

Функция предназначена для скрытия поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.showFormGroup (groupId, formClientId)

Описание входных параметров:

- **groupId** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.hideFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
KURS.hideFormField ('Users_CERTIFIED_EXPERTS_FK');
```

2.1.6.21.KURS.toggleFormGroup - скрытие поля формы

Описание:

Функция предназначена для скрытия поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.toggleFormGroup (groupId, formClientId)

Описание входных параметров:

- **groupId** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.hideFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
KURS.hideFormField ('Users_CERTIFIED_EXPERTS_FK');
```

2.1.6.22.KURS.hideFormGroupContent - скрытие поля формы

Описание:

Функция предназначена для скрытия поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.hideFormGroupContent (groupId, formClientId)

Описание входных параметров:

- **groupId** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.hideFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
KURS.hideFormField ('Users_CERTIFIED_EXPERTS_FK');
```

2.1.6.23.KURS.showFormGroupContent - скрытие поля формы

Описание:

Функция предназначена для скрытия поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.showFormGroupContent (groupId, formClientId)

Описание входных параметров:

- **groupId** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.hideFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
KURS.hideFormField ('Users_CERTIFIED_EXPERTS_FK');
```

2.1.6.24.KURS.toggleFormGroupContent - скрытие поля формы

Описание:

Функция предназначена для скрытия поля формы по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.toggleFormGroupContent (groupId, formClientId)

Описание входных параметров:

- **groupId** – имя поля, необходимо передать название таблицы и название поля, разделенные подчеркиванием, в кавычках, например, 'Users_RoleId'. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.hideFormField ('FOUNDERS_MUNICIPAL_ORGANS_FK');
KURS.hideFormField ('Users_CERTIFIED_EXPERTS_FK');
```

2.1.6.25.KURS.getEntityRecord - получение строки данных по сущности

Описание:

Функция предназначена для получения строки данных по сущности из базы данных и последующей обработки полученного значения. Может быть использована при наступлении некоторого события, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.getEntityRecord (entityId, recordId, async, onAsyncGet)

Описание входных параметров:

- **entityId** – код сущности. **ОБЯЗАТЕЛЬНЫЙ**
- **recordId** – идентификатор записи. **ОБЯЗАТЕЛЬНЫЙ**
- **async** - признак асинхронной обработки, по умолчанию обработка СИНХРОННАЯ и функция возвращает данные сама. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onAsyncGet** – функция, вызываемая при получении успешного ответа с сервера, только в АСИНХРОННОМ режиме. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра (в асинхронном режиме – параметр result):

- Значение в виде JSON с данными по указанной записи

Пример

```
//Пример синхронного вызова:
var result = KURS.getEntityRecord('MY_ENTITY', 'myRecordId');
alert('Получен результат!');
```

```
//Пример асинхронного вызова:
KURS.getEntityRecord ('MY_ENTITY', 'myRecordId', true, function (result) {alert ('Получен результат!'); });
```

2.1.6.26.KURS.getRecordFieldValue - получение значения поля из строки данных по сущности

Описание:

Функция предназначена для получения значения поля из строки данных по сущности. Используется совместно с функциями KURS.getEntityRecord и KURS.getReferenceRecord (если возвращаемый результат не picker).

Определение функции:

KURS.getRecordFieldValue (record, fieldName)

Описание входных параметров:

- **record** – строка данных в виде JSON, полученная в результате выполнения функции KURS.getEntityRecord. **ОБЯЗАТЕЛЬНЫЙ**
- **fieldName** – наименование поля, по которому нужно получить значение. **ОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- Значение в виде строки, для ссылочных полей возвращается идентификатор

Пример

```
KURS.getEntityRecord ('MY_ENTITY', 'myRecordId', true, function (result
{
   var ApplStatus = KURS.getRecordFieldValue (result,
'LIC_APPLICATIONS_LIC_APPLICATION_STATUSES_FK');
   var ApplType = KURS.getRecordFieldValue (result,
'LIC_APPLICATIONS_LIC_APPLICATION_TYPE_FK');
});
```

2.1.6.27.KURS.getReferenceRecord - получение данных по связанной с полем записи из БД

Описание:

Функция предназначена для получение связанной записи по полю сущности формы из базы данных

Определение функции:

KURS.getReferenceRecord (entityId, fieldName, fieldValue, currentRecordId, ownerRecordId, async, onAsyncGet)

Описание входных параметров:

Описание входных параметров:

- **entityId** – код сущности. **ОБЯЗАТЕЛЬНЫЙ**
- **fieldName** – ссылочное поле сущности, по которому нужно получить связанную запись. **ОБЯЗАТЕЛЬНЫЙ**
- **fieldValue** – значение ссылочного поля сущности, по которому нужно получить связанную запись. **ОБЯЗАТЕЛЬНЫЙ**
- **currentRecordId** – идентификатор текущей записи (например, открытой записи формы). **НЕОБЯЗАТЕЛЬНЫЙ**
- **ownerRecordId** – идентификатор родительской записи (например, при создании в подсписке доступен идентификатор родительской записи). **НЕОБЯЗАТЕЛЬНЫЙ**
- **async** - признак асинхронной обработки, по умолчанию обработка СИНХРОННАЯ и функция возвращает данные сама. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onAsyncGet** – функция, вызываемая при получении успешного ответа с сервера, только в АСИНХРОННОМ режиме. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра (в асинхронном режиме – параметр result):

- Значение в виде JSON с данными по указанной записи. В случае picker возвращает объект с полями Id, Name, Dependencies

Пример

```
var result = KURS.getReferenceRecord ('SCHOOLS', 'REGIONS_FK', '77');
```

2.1.6.28.KURS.refreshForm – обновление формы

Описание:

Функция предназначена для обновления формы по некоторому событию, например, по нажатию кнопки (JsOnClick). Функция выполняет обновление текущей формы, данный метод НЕ сохраняет данные формы. В случае если параметр formClientId не передан - для его получения будет автоматически вызван метод KURS.getFormClientId. В случае если formMode не передан - режим формы останется без изменений.

Определение функции:

KURS.refreshForm (formClientId, formMode)

Описание входных параметров:

- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**
- **formMode** - режим формы, принимает одно из значений: 'Edit' или 'View' (для режимов редактирования или просмотра соответственно). **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.refreshForm ();
```

2.1.6.29.KURS.refreshList – обновление данных списка

Описание:

Функция предназначена для обновления списка по некоторому событию, например, по нажатию кнопки (JsOnClick). Функция выполняет обновление текущего списка, данный метод НЕ сохраняет данные списка. В случае если параметр listClientId не передан - для его получения будет автоматически вызван метод KURS.getListClientId.

Определение функции:

KURS.refreshList (listClientId)

Описание входных параметров:

- **listClientId** - идентификатор списка на странице. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.refreshList ();
```

2.1.6.30.KURS.refreshAllForms – обновление всех форм на странице

Описание:

Функция предназначена для обновления всех форм на странице по некоторому событию, например, по нажатию кнопки (JsOnClick). Функция выполняет только обновление форм, данный метод НЕ сохраняет данные форм.

Определение функции:

KURS.refreshAllForms ()

Пример

```
KURS.refreshAllForms ();
```

2.1.6.31.KURS.refreshAllLists – обновление всех списков на странице

Описание:

Функция предназначена для обновления всех списков на странице по некоторому событию, например, по нажатию кнопки (JsOnClick). Функция выполняет только обновление списков, данный метод НЕ сохраняет данные списков.

Определение функции:

KURS.refreshAllLists ()

Пример

```
KURS.refreshAllLists ();
```

2.1.6.32.KURS.saveForm – сохранение формы

Описание:

Функция предназначена для сохранения формы по некоторому событию, например, по нажатию кнопки (JsOnClick). Функция выполняет сохранение и обновление текущей формы. В случае если параметр formClientId не передан - для его получения будет автоматически вызван метод KURS.getFormClientId. В случае если formMode не передан - режим формы останется без изменений.

Определение функции:

KURS.saveForm (formClientId, formMode, onSaveSuccess)

Описание входных параметров:

- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**
- **formMode** - режим формы, принимает одно из значений: 'Edit' или 'View' (для режимов редактирования или просмотра соответственно). **НЕОБЯЗАТЕЛЬНЫЙ**
- onSaveSuccess – функция выполняемая после успешного сохранения формы

Пример

KURS.saveForm ();

```
KURS.saveForm(null, null, function()
{
    KURS.showDialog('Подтверждение действия', 'Вы уверены, что хотите перестроить
форму?', '', 'Да', 'Нет', function()
    {
        var recordId = UTILS.getUrlParameter('RecordId');
        KURS.runCustomAction('BuildPrimaryForm', 'PrimaryForms_Tab1', recordId, null,
true, {}, function(result)
        {
           KURS.showMessage('Сообщение', 'Перестроение формы выполнено!',
result.response);
        });
    })
});
```

2.1.6.33.KURS.saveList – сохранение списка

Описание:

Функция предназначена для сохранения формы по некоторому событию, например, по нажатию кнопки (JsOnClick). Функция выполняет сохранение и обновление текущей формы. В случае если параметр formClientId не передан - для его получения будет автоматически вызван метод KURS.getFormClientId. В случае если formMode не передан - режим формы останется без изменений.

Определение функции:

KURS.saveList (listClientId, entityId)

Описание входных параметров:

- **listClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**
- **entityId** - режим формы, принимает одно из значений: 'Edit' или 'View' (для режимов редактирования или просмотра соответственно). **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.saveForm ();

KURS.saveForm(null, null, function()
{
    KURS.showDialog('Подтверждение действия', 'Вы уверены, что хотите перестроить
форму?', '', 'Да', 'Нет', function()
    {
        var recordId = UTILS.getUrlParameter('RecordId');
        KURS.runCustomAction('BuildPrimaryForm', 'PrimaryForms_Tab1', recordId, null,
true, {}, function(result)
        {
           KURS.showMessage('Сообщение', 'Перестроение формы выполнено!',
result.response);
        });
    })
});
```

2.1.6.34.KURS.navigateToTab – переход на вкладку без сохранения формы

Описание:

Функция предназначена для перехода на определенную вкладку формы по некоторому событию, например, по нажатию кнопки (JsOnClick). Функция НЕ выполняет сохранение текущей формы. В случае если параметр formClientId не передан - для его получения будет автоматически вызван метод KURS.getFormClientId. В случае если tabFormMode не передан - новая вкладка загрузится в том же режиме, что и текущая.

Определение функции:

KURS.navigateToTab (tabEntityId, formClientId, tabFormMode)

Описание входных параметров:

- **tabEntityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**
- **tabFormMode** - режим формы, принимает одно из значений: 'Edit' или 'View' (для режимов редактирования или просмотра соответственно). **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.navigateToTab ('LIC_APPLICATIONS_Tab2');
```

2.1.6.35.KURS.saveAndNavigateToTab – переход на вкладку с сохранением формы

Описание:

Функция предназначена для перехода на определенную вкладку формы по некоторому событию, например, по нажатию кнопки (JsOnClick). Перед переходом на выбранную вкладку функция выполняет сохранение текущей формы. В случае если параметр formClientId не передан - для его получения будет автоматически вызван метод KURS.getFormClientId. В случае если tabFormMode не передан - новая вкладка загрузится в том же режиме, что и текущая.

Важно! Следует учитывать, что сохранение формы всегда происходит в асинхронном режиме, поэтому последовательный вызов KURS.saveForm и KURS.navigateToTab будет некорректным (в частности, не отобразятся ошибки сохранения), в этом случае следует использовать **KURS.saveAndNavigateToTab**

Определение функции:

KURS.saveAndNavigateToTab (tabEntityId, formClientId, tabFormMode)

Описание входных параметров:

- **tabEntityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**
- **formClientId** - идентификатор формы на странице. **НЕОБЯЗАТЕЛЬНЫЙ**
- **tabFormMode** - режим формы, принимает одно из значений: 'Edit' или 'View' (для режимов редактирования или просмотра соответственно). **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.saveAndNavigateToTab ('LIC_APPLICATIONS_Tab2');
```

2.1.6.36.KURS.runSqlAction – выполнение SQL-обработки

Описание:

Функция предназначена для выполнения SQL-обработки. Может быть использована при наступлении некоторого события, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Параметры recordId, additionalData, async, waitingOptions, onAsyncSuccess - необязательные.

Параметр async - признак асинхронной обработки, по умолчанию обработка СИНХРОННАЯ и функция возвращает данные сама.

Определение функции:

KURS.runSqlAction (actionId, entityId, recordId, additionalData, async, waitingOptions, onAsyncSuccess)

Описание входных параметров:

- **actionId** – идентификатор SQL-обработки. **ОБЯЗАТЕЛЬНЫЙ**
- **entityId** – идентификатор сущности, в рамках которой выполняется обработка. **ОБЯЗАТЕЛЬНЫЙ**
- **recordId** – текущий идентификатор записи (передается как @CurrentObjectId в SQL-обработку). **НЕОБЯЗАТЕЛЬНЫЙ**
- **additionalData** – дополнительные параметры в виде JSON-строки. **НЕОБЯЗАТЕЛЬНЫЙ**
- **async** - признак асинхронной обработки, по умолчанию обработка СИНХРОННАЯ и функция возвращает данные сама. **НЕОБЯЗАТЕЛЬНЫЙ**
- **waitingOptions** – настройка индикатора загрузки, только в АСИНХРОННОМ режиме. **НЕОБЯЗАТЕЛЬНЫЙ**

- **onAsyncSuccess** – функция, вызываемая при получении успешного ответа с сервера, только в АСИНХРОННОМ режиме. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра (в асинхронном режиме – параметр result):

- Значение в виде JSON с данными, для получения значения по конкретному полю можно использовать функцию KURS.getRecordFieldValue

Пример

```
var recordId = '123';

//Синхронный режим
var result = KURS.runSqlAction ('request_fns_school', 'SCHOOLS_Tab5', recordId);
if(!UTILS.isNullOrEmpty(result[0]) && !UTILS.isNullOrEmpty(result[0].ErrorMessage))
{
   KURS.showMessage ('Ошибка', result[0].ErrorMessage);
}

//Асинхронный режим
KURS.runSqlAction ('request_fns_school', 'SCHOOLS_Tab5', recordId, '', true, '', function (result)
{
  if(!UTILS.isNullOrEmpty(result[0]) && !UTILS.isNullOrEmpty(result[0].ErrorMessage))
  {
    KURS.showMessage ('Ошибка', result[0].ErrorMessage);
  }
});
```

2.1.6.37.KURS.runCustomAction – выполнение C#-обработки

Описание:

Функция предназначена для выполнения C#-обработки. Может быть использована при наступлении некоторого события, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Параметры entityId, recordId, additionalData, async, waitingOptions, onAsyncSuccess - необязательные.

Параметр async - признак асинхронной обработки, по умолчанию обработка СИНХРОННАЯ и функция возвращает данные сама.

Определение функции:

KURS.runCustomAction (actionId, entityId, recordId, additionalData, async, waitingOptions, onAsyncSuccess)

Описание входных параметров:

- **actionId** – идентификатор SQL-обработки. **ОБЯЗАТЕЛЬНЫЙ**
- **entityId** – идентификатор сущности, в рамках которой выполняется обработка. **НЕОБЯЗАТЕЛЬНЫЙ**
- **recordId** – текущий идентификатор записи (передается как @CurrentObjectId в SQL-обработку). **НЕОБЯЗАТЕЛЬНЫЙ**

- **additionalData** – дополнительные параметры в виде JSON-строки. **НЕОБЯЗАТЕЛЬНЫЙ**
- **async** - признак асинхронной обработки, по умолчанию обработка СИНХРОННАЯ и функция возвращает данные сама. **НЕОБЯЗАТЕЛЬНЫЙ**
- **waitingOptions** – настройка индикатора загрузки, только в АСИНХРОННОМ режиме. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onAsyncSuccess** – функция, вызываемая при получении успешного ответа с сервера, только в АСИНХРОННОМ режиме. **НЕОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра (в асинхронном режиме – параметр result):

- Значение в виде JSON с данными, для получения значения по конкретному полю можно использовать функцию KURS.getRecordFieldValue

Важно! Для определения c#-действий дополнительно требуется:

- создать класс, реализующий интерфейс KURS.WebForms.Components.Extendable.CustomActions.ICustomActionProcessor;
- зарегистрировать экземпляр этого класса в Global.asax в методе Application_Start, вызвав для этого метод KURS.WebForms.Components.Extendable.CustomActions.CustomActionProcessorsManager.RegisterProcessor (string actionId, ICustomActionProcessor processor);

Пример

```
var recordId = '123';

//Синхронный режим
var result = KURS.runCustomAction ('BuildPrimary', 'RPT_FORMS_Tab1', recordId);
KURS.showMessage ('Сообщение', 'Перестроение формы выполнено!', result.response);

//Асинхронный режим
KURS.runCustomAction ('BuildPrimary', 'RPT_FORMS_Tab1', recordId, null, true, {}, function (result)
{
  KURS.showMessage ('Сообщение', 'Перестроение формы выполнено!', result.response);
});
```

2.1.6.38.KURS.openPopupWindow – открытие popup-окна по URL

Описание:

Функция предназначена для открытия popup-окна по заданному URL по некоторому событию, например, по изменению состояния поля (JsOnChange), по нажатию кнопки (JsOnClick) или при загрузке формы (JsSection).

Определение функции:

KURS.openPopupWindow (url, title, cssClass, onClose)

Описание входных параметров:

- **url** – ссылка на страницу, которую нужно открыть в popup-окне. **ОБЯЗАТЕЛЬНЫЙ**
- **title** – заголовок popup-окна. **НЕОБЯЗАТЕЛЬНЫЙ**
- **cssClass** – класс, определяющий стиль popup-окна. **НЕОБЯЗАТЕЛЬНЫЙ**

- **onClose** – функция, выполняемая при закрытии popup-окна. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.openPopupWindow ('http://ya.ru', 'Яндекс', '', function()
{
   alert('closing');
});
```

2.1.6.39.KURS.openPopupWindowHtml – открытие popup-окна с заданным HTML

Описание:

Функция аналогична KURS.openPopupWindow, но отображает не данные по url, а переданный ей html, признак hasHeader управляет тем, отображать ли заголовок окна (в ряде случае - заголовок может уже присутствовать внутри переданного html).

Определение функции:

KURS.openPopupWindowHtml (html, hasHeader, title, cssClass, onClose)

Описание входных параметров:

- **html** – html-документ, который нужно отобразить в popup-окне. **ОБЯЗАТЕЛЬНЫЙ**
- **hasHeader** – признак «отображать заголовок окна». **НЕОБЯЗАТЕЛЬНЫЙ**
- **title** – заголовок popup-окна. **НЕОБЯЗАТЕЛЬНЫЙ**
- **cssClass** – класс, определяющий стиль popup-окна. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onClose** – функция, выполняемая при закрытии popup-окна. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```

```

2.1.6.40.KURS.showMessage – отображение сообщения пользователю

Описание:

Функция отображает popup-окно с сообщением и одной кнопкой. Кнопка просто закрывает окно, не выполняя никаких действий. Настраивается заголовок окна (title), два блока текста (message, detailedMessage). Платформенный аналог функции alert в JavaScript.

Определение функции:

KURS.showMessage (title, message, detailedMessage, onClose)

Описание входных параметров:

- **title** – заголовок popup-окна. **ОБЯЗАТЕЛЬНЫЙ**
- **message** – текст сообщения. **ОБЯЗАТЕЛЬНЫЙ**
- **detailedMessage** – текст дополнительного сообщения. **НЕОБЯЗАТЕЛЬНЫЙ**
- onClose – выполнение функции по закрытию окна сообщения

Пример

```
KURS.showMessage ('Сообщение', 'Перестроение формы выполнено!');
```

2.1.6.41.KURS.showUnobtrusiveMessage– отображение всплывающего сообщения пользователю

Описание:

Функция отображает popup-окно с сообщением и одной кнопкой. Кнопка просто закрывает окно, не выполняя никаких действий. Настраивается заголовок окна (title), два блока текста (message, detailedMessage). Платформенный аналог функции alert в JavaScript.

Определение функции:

KURS.showUnobtrusiveMessage (message, closeSeconds, cssClass)

Описание входных параметров:

- **message** – текст сообщения. **ОБЯЗАТЕЛЬНЫЙ**
- **closeSeconds** – длительность отображения сообщения. **НЕОБЯЗАТЕЛЬНЫЙ**
- **cssClass** – css-класс для стилизации сообщения

Пример

```
KURS.showMessage ('Сообщение', 'Перестроение формы выполнено!');
```

2.1.6.42.KURS.showDialog – отображение диалогового сообщения пользователю

Описание:

Функция отображает диалоговое popup-окно с сообщением и двумя кнопками. Кнопка «Нет» (текст кнопки настраивается параметром cancelText) просто закрывает окно, не выполняя никаких действий. Кнопка «Да» (текст кнопки настраивается параметром acceptText) выполняет действие, которое указано в параметре onAccept. Настраивается заголовок окна (title), два блока текста (message, detailedMessage), тексты кнопок (acceptText, cancelText). Платформенный аналог функции confirm в JavaScript.

Определение функции:

KURS.showDialog (title, message, detailedMessage, acceptText, cancelText, onAccept)

Описание входных параметров:

- **title** – заголовок popup-окна. **ОБЯЗАТЕЛЬНЫЙ**
- **message** – текст сообщения. **ОБЯЗАТЕЛЬНЫЙ**
- **detailedMessage** – текст дополнительного сообщения. **НЕОБЯЗАТЕЛЬНЫЙ**
- **acceptText**– текст кнопки принятия. **НЕОБЯЗАТЕЛЬНЫЙ**
- **cancelText** – текст кнопки отмены. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onAccept** – функция, выполняемая при нажатии кнопки принять. **ОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.showDialog ('Подтверждение действия', 'Вы уверены, что хотите заполнить форму по первичным данным?', '', 'Да', 'Нет', function()
{
//действия, которые нужно выполнить в случае нажатия положительной кнопки (Да)
});
```

2.1.6.43.KURS.showYesNoDialog – отображение диалогового сообщения пользователю

Описание:

Функция отображает диалоговое popup-окно с сообщением и двумя кнопками. Кнопка «Нет» просто закрывает окно, не выполняя никаких действий. Кнопка «Да» выполняет действие, которое указано в параметре onAccept. Настраивается заголовок окна (title), два блока текста (message, detailedMessage). Платформенный аналог функции confirm в JavaScript.

Определение функции:

KURS.showYesNoDialog (title, message, detailedMessage, onAccept)

Описание входных параметров:

- **title** – заголовок popup-окна. **ОБЯЗАТЕЛЬНЫЙ**
- **message** – текст сообщения. **ОБЯЗАТЕЛЬНЫЙ**
- **detailedMessage** – текст дополнительного сообщения. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onAccept** – функция, выполняемая при нажатии кнопки принять. **ОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.showYesNoDialog ('Подтверждение действия', 'Вы уверены, что хотите заполнить форму по первичным данным?', '', function()
{
//действия, которые нужно выполнить в случае нажатия положительной кнопки (Да)
});
```

2.1.6.44.KURS.showOkCancelDialog – отображение диалогового сообщения пользователю

Описание:

Функция отображает диалоговое popup-окно с сообщением и двумя кнопками. Кнопка «Отмена» просто закрывает окно, не выполняя никаких действий. Кнопка «Ок» выполняет действие, которое указано в параметре onAccept. Настраивается заголовок окна (title), два блока текста (message, detailedMessage). Платформенный аналог функции confirm в JavaScript.

Определение функции:

KURS.showOkCancelDialog (title, message, detailedMessage, onAccept)

Описание входных параметров:

- **title** – заголовок popup-окна. **ОБЯЗАТЕЛЬНЫЙ**
- **message** – текст сообщения. **ОБЯЗАТЕЛЬНЫЙ**
- **detailedMessage** – текст дополнительного сообщения. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onAccept** – функция, выполняемая при нажатии кнопки принять. **ОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.showOkCancelDialog ('Подтверждение действия', 'Вы уверены, что хотите заполнить форму по первичным данным?', '', function()
{
//действия, которые нужно выполнить в случае нажатия положительной кнопки (Да)
});
```

2.1.6.45.KURS.openSingleAddForm – открытие popup-окна формы создания

Описание:

Функция отображает форму создания в popup-окне.

Определение функции:

KURS.openPopupAddForm (entityId, ownerRecordIdField, ownerRecordId, ownerEntityId, onCloseOptions)

Описание входных параметров:

- **entityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**
- **ownerRecordIdField** - поле-ссылка на родительскую запись (используются для SubGrid). **НЕОБЯЗАТЕЛЬНЫЙ**
- **ownerRecordId** - идентификатор родительской записи. **НЕОБЯЗАТЕЛЬНЫЙ**
- **ownerEntityId** - идентификатор родительской сущности. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onCloseOptions** - объект, содержащий данные о необходимых действиях при закрытии окна. Поля: RelatedControlId - опциональный идентификатор элемента на странице, который нужно обновить; RelatedControlType - тип элемента (поддерживаемые значения: 'grid', 'picker'). **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```

```

2.1.6.46.KURS.openSingleEditForm – открытие popup-окна формы создания

Описание:

Функция отображает форму редактирования в popup-окне.

Определение функции:

KURS.openPopupEditForm (entityId, recordId, onCloseOptions)

Описание входных параметров:

- **entityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**

- **recordId** - идентификатор записи. **ОБЯЗАТЕЛЬНЫЙ**
- **onCloseOptions** - объект, содержащий данные о необходимых действиях при закрытии окна. Поля: RelatedControlId - опциональный идентификатор элемента на странице, который нужно обновить; RelatedControlType - тип элемента (поддерживаемые значения: 'grid', 'picker'). **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
```

2.1.6.47.KURS.openSingleViewForm – открытие popup-окна формы создания

Описание:

Функция отображает форму просмотра в popup-окне.

Определение функции:

KURS.openPopupViewForm (entityId, recordId, onCloseOptions)

Описание входных параметров:

- **entityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**
- **recordId** - идентификатор записи. **ОБЯЗАТЕЛЬНЫЙ**
- **onCloseOptions** - объект, содержащий данные о необходимых действиях при закрытии окна. Поля: RelatedControlId - опциональный идентификатор элемента на странице, который нужно обновить; RelatedControlType - тип элемента (поддерживаемые значения: 'grid', 'picker'). **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
```

2.1.6.48.KURS.openSingleList – открытие popup-окна списка

Описание:

Функция обновления списка по некоторому событию.

Определение функции:

KURS.openSingleList (entityId, filterString)

Описание входных параметров:

- **entityId** - идентификатор метаданных. **ОБЯЗАТЕЛЬНЫЙ**
- **filterString** - строка фильтра, аналогичная url-строке. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.openPopupList ('SearchOO_GRID',
'SortDir=Ascending&SortCol=vwm_SCHOOLS.Name&vwm_SCHOOLS.NAME_OR_INN_FILTER=990943
7740&vwm_SCHOOLS.REGIONFK=5&vwm_SCHOOLS.AKND_CHECK_PURPOSE_FILTER=3|4&ID_vwm_
```

```
SCHOOLS_ADDITIONAL.HAS_NOTFILLED_PRESCRIPTIONS=false&vwm_SCHOOLS.AKND_COUNT_DAYS
_FROM_FILTER_tovalue=101&vwm_SCHOOLS.PARENTFK=0C44556423C738225AE7BEB1D59FF5B0')
```

2.1.6.49.KURS.openPopupAddForm – открытие popup-окна формы создания

Описание:

Функция отображает форму создания в popup-окне.

Определение функции:

KURS.openPopupAddForm (entityId, ownerRecordIdField, ownerRecordId, ownerEntityId, onCloseOptions, onSave)

Описание входных параметров:

- **entityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**
- **ownerRecordIdField** - поле-ссылка на родительскую запись (используются для SubGrid). **НЕОБЯЗАТЕЛЬНЫЙ**
- **ownerRecordId** - идентификатор родительской записи. **НЕОБЯЗАТЕЛЬНЫЙ**
- **ownerEntityId** - идентификатор родительской сущности. **НЕОБЯЗАТЕЛЬНЫЙ**
- **onCloseOptions** - объект, содержащий данные о необходимых действиях при закрытии окна. Поля: RelatedControlId - опциональный идентификатор элемента на странице, который нужно обновить; RelatedControlType - тип элемента (поддерживаемые значения: 'grid', 'picker'). **НЕОБЯЗАТЕЛЬНЫЙ**
- **onSave**

Пример

```
```

2.1.6.50.KURS.openPopupEditForm – открытие popup-окна формы создания

Описание:

Функция отображает форму редактирования в popup-окне.

Определение функции:

KURS.openPopupEditForm (entityId, recordId, onCloseOptions, onSave)

Описание входных параметров:

- **entityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**
- **recordId** - идентификатор записи. **ОБЯЗАТЕЛЬНЫЙ**
- **onCloseOptions** - объект, содержащий данные о необходимых действиях при закрытии окна. Поля: RelatedControlId - опциональный идентификатор элемента на странице, который нужно обновить; RelatedControlType - тип элемента (поддерживаемые значения: 'grid', 'picker'). **НЕОБЯЗАТЕЛЬНЫЙ**
- **onSave**

2.1.6.51.KURS.openPopupViewForm – открытие popup-окна формы создания

Старое название метода - OpenPopupViewForm

Описание:

Функция отображает форму просмотра в popup-окне.

Определение функции:

KURS.openPopupViewForm (entityId, recordId, onCloseOptions)

Описание входных параметров:

- **entityId** - идентификатор сущности, форму которой требуется отобразить. **ОБЯЗАТЕЛЬНЫЙ**
- **recordId** - идентификатор записи. **ОБЯЗАТЕЛЬНЫЙ**
- **onCloseOptions** - объект, содержащий данные о необходимых действиях при закрытии окна. Поля: RelatedControlId - опциональный идентификатор элемента на странице, который нужно обновить; RelatedControlType - тип элемента (поддерживаемые значения: 'grid', 'picker'). **НЕОБЯЗАТЕЛЬНЫЙ**
- **onSave**

2.1.6.52.KURS.openPopupList – открытие popup-окна списка

Описание:

Функция обновления списка по некоторому событию.

Определение функции:

KURS.openPopupList (entityId, filterString)

Описание входных параметров:

- **entityId** - идентификатор метаданных. **ОБЯЗАТЕЛЬНЫЙ**
- **filterString** - строка фильтра, аналогичная url-строке. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.openPopupList ('SearchOO_GRID',
'SortDir=Ascending&SortCol=vwm_SCHOOLS.Name&vwm_SCHOOLS.NAME_OR_INN_FILTER=9909437740&vwm_SCHOOLS.REGIONFK=5&vwm_SCHOOLS.AKND_CHECK_PURPOSE_FILTER=3|4&ID_vwm_SCHOOLS_ADDITIONAL.HAS_NOTFILLED_PRESCRIPTIONS=false&vwm_SCHOOLS.AKND_COUNT_DAYS_FROM_FILTER_tovalue=101&vwm_SCHOOLS.PARENTFK=0C44556423C738225AE7BEB1D59FF5B0')
```

2.1.6.53.KURS.hidePopup– открытие popup-окна списка

Описание:

Функция обновления списка по некоторому событию.

Определение функции:

KURS.hidePopup ($popupElement)

Описание входных параметров:

- **$popupElement** - идентификатор метаданных. **ОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.openPopupList ('SearchOO_GRID',
'SortDir=Ascending&SortCol=vwm_SCHOOLS.Name&vwm_SCHOOLS.NAME_OR_INN_FILTER=9909437740&vwm_SCHOOLS.REGIONFK=5&vwm_SCHOOLS.AKND_CHECK_PURPOSE_FILTER=3|4&ID_vwm_SCHOOLS_ADDITIONAL.HAS_NOTFILLED_PRESCRIPTIONS=false&vwm_SCHOOLS.AKND_COUNT_DAYS_FROM_FILTER_tovalue=101&vwm_SCHOOLS.PARENTFK=0C44556423C738225AE7BEB1D59FF5B0')
```

2.1.6.54.KURS.generateReport – генерация платформенного отчета

Описание:

Функция предназначена для генерации платформенного отчета по нажатию кнопки (JsOnClick).

Определение функции:

KURS.generateReport (reportId, templateId, reportParameters, waitingOptions)

Описание входных параметров:

- **reportId** – идентификатор платформенного отчета. **ОБЯЗАТЕЛЬНЫЙ**
- **templateId** – идентификатор шаблона (какой шаблон использовать для выгрузки). **ОБЯЗАТЕЛЬНЫЙ**
- **reportParameters** – параметры отчета, задаются через JSON. **НЕОБЯЗАТЕЛЬНЫЙ**
- **waitingOptions** – настройка индикатора загрузки. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.generateReport ('INSPECTION_ORDER_REPORT', 'WORD', { RecordId: UTILS.getUrlParameter ('RecordId')}, { });
```

2.1.6.55.KURS.showWaiting – генерация платформенного отчета

Описание:

Функция предназначена для генерации платформенного отчета по нажатию кнопки (JsOnClick).

Определение функции:

KURS.showWaiting (title, message)

Описание входных параметров:

- **reportId** – идентификатор платформенного отчета. **ОБЯЗАТЕЛЬНЫЙ**
- **templateId** – идентификатор шаблона (какой шаблон использовать для выгрузки). **ОБЯЗАТЕЛЬНЫЙ**
- **reportParameters** – параметры отчета, задаются через JSON. **НЕОБЯЗАТЕЛЬНЫЙ**

- **waitingOptions** – настройка индикатора загрузки. **НЕОБЯЗАТЕЛЬНЫЙ**

Пример

```
KURS.generateReport ('INSPECTION_ORDER_REPORT', 'WORD', { RecordId: UTILS.getUrlParameter ('RecordId')}, { });
```

2.1.6.56.KURS. hideWaiting – генерация платформенного отчета

Описание:

Функция предназначена для генерации платформенного отчета по нажатию кнопки (JsOnClick).

Определение функции:

KURS.hideWaiting ()

Пример

```
KURS.generateReport ('INSPECTION_ORDER_REPORT', 'WORD', { RecordId: UTILS.getUrlParameter ('RecordId')}, { });
```

2.1.6.57.UTILS.isNullOrEmpty – проверка заполнения значения

Описание:

Функция может быть использована для проверки произвольного значения на заполнения.
Включает в себя следующие проверки

- Значение не NULL
- Значение не содержит пустой строки

Определение функции:

UTILS.isNullOrEmpty (value)

Описание входных параметров:

- **value** – произвольное значение. **ОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- значение типа bool (логическое) – true\false

Пример

```
var value = null;
if(UTILS.isNullOrEmpty(value))
{
   KURS.showMessage ('Ошибка', 'Значение не заполнено');
}
```

2.1.6.58.UTILS.stringEndsWith – проверка окончания строки

Описание:

Функция предназначена для проверки содержит ли конец строки определенное значение.

Определение функции:

UTILS.stringEndsWith (str, suffix)

Описание входных параметров:

- **str** – исходная строка. **ОБЯЗАТЕЛЬНЫЙ**
- **suffix** – строка для поиска. **ОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- значение типа bool (логическое) – true\false

Пример

```
```

2.1.6.59.UTILS.getUrlParameter – получение значения URL-параметра

Описание:

Функция предназначена для получения значения URL-параметра.

Определение функции:

UTILS.getUrlParameter (parameterName)

Описание входных параметров:

- **parameterName** – название URL-параметра. **ОБЯЗАТЕЛЬНЫЙ**

Описание выходного параметра:

- значение параметра типа string (строковое)

Пример

```
var recordId = UTILS.getUrlParameter ('RecordId');
```

2.1.6.60.afterGridRefresh – событие по обновлению списка

Описание:

Объявленная функция вызывается по событию обновления списка (например, было выполнено сохранение списка и сам список обновился).

Определение функции:

function afterGridRefresh(listid) { //действия которые необходимо выполнить при обновлении списка }

Описание входных параметров:

- **listid** – параметр передается системой автоматически - код сущности, список которой обновился. **ОБЯЗАТЕЛЬНЫЙ**
- **{}-** в теле функции указывается перечень действий, которые необходимо выполнить по этому событию. **ОБЯЗАТЕЛЬНЫЙ**

Пример

```
function afterGridRefresh(listid)
{
   if(listid == 'LIC_EDU_LEVELS')
   {
      var id = KURS.getListClientId('', 'LIC_EDU_PROGRAMS');
      KURS.refreshList(id);
   }
}
```

2.1.7. Общий вид XML-файла

```
<Entity id="id1">
    <References />
    <Data tableName="tableName1" connectionStringName="connectionStringName1"
newIdMode="None">
        <WhereSection>...</WhereSection>
        <SortSection>...</SortSection>
        ...
    </Data>
    <Filter>
        <FieldGroups>
            ...
        </FieldGroups>
    </Filter>
    <List pageSize="20" headerText="headerText1">
        <JsSection>...</JsSection>
        <ControlPanel position="Top" allowListSettings="true" allowExportToExcel="true"
allowExportToWord="false">
            ...
        </ControlPanel>
        <PagerPanel position="Both" />
        <ColumnGroups fitHeight="false">
            ...
        </ColumnGroups>
        <StyleConditions>
            ...
        </StyleConditions>
    </List>
    <Form>
        <JsSection>...</JsSection>
        <TabsControl>
            ...
        </TabsControl>
        <FieldGroups>
            ...
        </FieldGroups>
        <Validations>
            ...
        </Validations>
    </Form>
    <Fields>
        <Field fieldName="fieldName1" fieldType="String">
            <Data>
```

```
                <Dictionary tableName="tableName1" keyFieldName="keyFieldName1"
displayFieldName="displayFieldName1">
                    <Select>...</Select>
                </Dictionary>
                <SelectFragment>...</SelectFragment>
                <WhereFragment>...</WhereFragment>
                <SortFragment>...</SortFragment>
            </Data>
            <Filter filterMode="Default" caption="caption1" input="Text">
                <Picker entityId="entityId1" hasLink="false" />
                <Buttons>
                    ...
                </Buttons>
            </Filter>
            <List hidden="false" caption="caption1" order="1">
                <StyleConditions>
                    ...
                </StyleConditions>
            </List>
            <Form addMode="Default" editMode="Default" viewMode="Default"
caption="caption1" input="Text" defaultValue="defaultValue1" order="1"
lineBreakAfterInput="true" size="Normal" required="false" groupId="groupId1"
cssPrefix="cssPrefix1" formatSchema="formatSchema1" textRows="0" inputMask="inputMask1"
maskValidationEnabled="true" hideIfEmpty="false">
                <Picker entityId="entityId1" hasLink="false">
                    ...
                </Picker>
                <SubGrid fieldName="fieldName1" entityId="entityId1"
filterFieldName="filterFieldName1" />
                <ViewReport reportCode="reportCode1" showGenerateButton="false"
generateButtonText="generateButtonText1" />
                <File fileContentField="fileContentField1"
fileNameField="fileNameField1" />
                <Buttons>
                    ...
                </Buttons>
                <JsOnChange>...</JsOnChange>
            </Form>
        </Field>
        <Relation fieldName="fieldName1" relatedTableName="relatedTableName1"
relatedTableField="relatedTableField1" relatedTableAlias="relatedTableAlias1"
mustExist="true">
            <JoinCondition>...</JoinCondition>
            <Field fieldName="fieldName1" fieldType="String" />
        </Relation>
    </Fields>
</Entity>
```

2.2. Описание метаданных модуля «Отчетность»

2.2.1. Общее описание

Описание отчетов состоит из двух частей:

- Метаданные отчетов
- Шаблоны отчетов

Метаданные отчетов хранятся в файлах формата XML в заранее заданной папке внутри папки приложения в подпапке Metadata.

Шаблоны отчетов хранятся в файлах формата Excel XML, DOCX, REPY (в зависимости от типа используемого шаблона) в заранее заданной папке внутри папки приложения в подпапке Templates.

Относительный путь (относительно корневой папки приложения – папки, на которую настроен IIS) к папке, содержащей метаданные и шаблоны, задается в конфигурационном файле приложения в секции KURSConfiguration в разделе MetadataLocations с помощью ключа reports.

Пример

```
<KURSConfiguration>
    <MetadataLocations entities="~/_Metadata" accessControl="~/_SystemMetadata"
reports="~/_Reports" />
```

Каждый отчет должна быть описан в отдельном файле метаданных, один отчет может быть привязан к нескольким шаблонам.

2.2.2. Структура метаданных отчетов

2.2.2.1. Report - Отчет

Комплексный тип. Корневой элемент. Содержит описание отчета.

Дочерние элементы:

- Templates (1..1) – используемые шаблоны для выгрузки отчета
- Sources (1..1) – источники данных для выгрузки отчета
- Input (0..1) – глобальные настройки фильтра

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
Id	String	Уникальный ключ отчета (желательно совпадение с названием, содержащим его, файла XML)	Обязательный
name	String	Смысловое наименование отчета	Необязательный

Пример

```
<Report id="USERS_REPORT" name="Активность пользователей">
```

2.2.2.2. Report/Templates – Шаблоны отчета

Элемент комплексного типа. Раздел описания используемых в отчете шаблонов.

Дочерние элементы:

- Template (1..N) – используемые шаблоны для выгрузки отчета

Пример

```
<Templates>
    <Template id="WORD" name="Выгрузка в Word" type="Word"
templateFileName="TEMPLATES_4_9_SUM_WORD.docx"/>
    <Template id="EXCEL" name="Выгрузка в Excel" type="Excel"
templateFileName="TEMPLATES_4_9_SUM_EXCEL.xml"/>
</Templates>
```

2.2.2.3. Report/Templates/Template – Шаблон отчета

Элемент простого типа. Раздел содержит описание шаблона.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
Id	String	Уникальный ключ шаблона отчета	Обязательный
name	String	Наименование шаблона отчета	Обязательный
type	ReportTypesEnum	Тип используемого шаблона	Обязательный
templateFileName	String	Путь к файлу шаблона	Необязательный

Пример

```
<Template id="WORD" name="Выгрузка в Word" type="Word"
templateFileName="TEMPLATES_4_9_SUM_WORD.docx"/>
```

2.2.2.4. Report/Templates/Template/TextTemplate – Текстовый шаблон отчета

Элемент строкового типа. Внутри разделе задается текстовый шаблон выгрузки отчета.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
Id	String	Уникальный ключ шаблона отчета	Обязательный
name	String	Наименование шаблона отчета	Обязательный
type	ReportTypesEnum	Тип используемого шаблона	Обязательный
templateFileName	String	Путь к файлу шаблона	Необязательный

Пример

```
<Templates>
    <Template id="WORD" name="Выгрузка в Word" type="Word"
templateFileName="TEMPLATES_4_9_SUM_WORD.docx"/>
    <Template id="EXCEL" name="Выгрузка в Excel" type="Excel"
templateFileName="TEMPLATES_4_9_SUM_EXCEL.xml"/>
</Templates>
```

2.2.2.5. Report/Sources– Источники отчета

Элемент комплексного типа. В разделе задаются источники данных для получения выборок.

Дочерние элементы:

– Source (1..N) – источник данных

Пример

```
<Sources>
    <Source id="AOI1" name="Сведения об образовательной организации"
connectionStringName="Test">
        <Query>
            <![CDATA[
SELECT
s.FULL_NAME as "SNAME",
(case when s.DELETED = 0 then N'Да' else N'Нет' end) as "IS_OUTDATED",
reg.NAME as "REGION",
s.LAW_ADDRESS as "LAW_ADDRESS",
s.OGRN as "GOSREGNUM",
s.INN as "INN",
(N'Лицензия № ') as "LICENSE",
(N'Действует')  AS "LICENSE_STATUS",
(N'Свидетельство об аккредитации № ') as "CERTIFICATE",
(N'Действует') AS "CERTIFICATE_STATUS"
FROM SCHOOLS s
LEFT JOIN REGIONS reg ON reg.ID=s.REGIONS_FK
WHERE s.ID = @SchoolId
            ]]>
        </Query>
        <Output>
            <Parameter fieldName="SNAME" caption="SNAME" visible="true" formatSchema=""
/>
            <Parameter fieldName="IS_OUTDATED" caption="IS_OUTDATED" visible="true"
formatSchema ="" />
            <Parameter fieldName="REGION" caption="REGION" visible="true" formatSchema
="" />
            <Parameter fieldName="LAW_ADDRESS" caption="LAW_ADDRESS" visible="true"
formatSchema ="" />
            <Parameter fieldName="GOSREGNUM" caption="GOSREGNUM" visible="true"
formatSchema ="" />
            <Parameter fieldName="INN" caption="INN" visible="true" formatSchema ="" />
            <Parameter fieldName="LICENSE" caption="LICENSE" visible="true"
formatSchema ="" />
            <Parameter fieldName="LICENSE_STATUS" caption="LICENSE_STATUS"
visible="true" formatSchema ="" />
            <Parameter fieldName="CERTIFICATE" caption="CERTIFICATE" visible="true"
formatSchema ="" />
        </Output>
    </Source>
</Sources>
```

2.2.2.6. Report/Sources/Source – Источник отчета

Элемент комплексного типа. В разделе описывается источник данных.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
Id	String	Идентификатор источника данных	Обязательный
name	String	Наименование источника данных	Обязательный
connectionStringName	String	Наименование строки подключения к БД (из конфигурационного файла приложения)	Необязательный

Пример

```
<Source id="AOI1" name="Сведения об образовательной организации"
connectionStringName="Test">
    <Query>
        <![CDATA[
SELECT
s.FULL_NAME as "SNAME",
(case when s.DELETED = 0 then N'Да' else N'Нет' end) as "IS_OUTDATED",
reg.NAME as "REGION",
s.LAW_ADDRESS as "LAW_ADDRESS",
s.OGRN as "GOSREGNUM",
s.INN as "INN",
(N'Лицензия № ') as "LICENSE",
(N'Действует')  AS "LICENSE_STATUS",
(N'Свидетельство об аккредитации № ') as "CERTIFICATE",
(N'Действует') AS "CERTIFICATE_STATUS"
FROM SCHOOLS s
LEFT JOIN REGIONS reg ON reg.ID=s.REGIONS_FK
WHERE s.ID = @SchoolId
        ]]>
    </Query>
    <Output>
        <Parameter fieldName="SNAME" caption="SNAME" visible="true" formatSchema="" />
        <Parameter fieldName="IS_OUTDATED" caption="IS_OUTDATED" visible="true"
formatSchema ="" />
        <Parameter fieldName="REGION" caption="REGION" visible="true" formatSchema =""
/>
        <Parameter fieldName="LAW_ADDRESS" caption="LAW_ADDRESS" visible="true"
formatSchema ="" />
        <Parameter fieldName="GOSREGNUM" caption="GOSREGNUM" visible="true"
formatSchema ="" />
        <Parameter fieldName="INN" caption="INN" visible="true" formatSchema ="" />
        <Parameter fieldName="LICENSE" caption="LICENSE" visible="true" formatSchema
="" />
        <Parameter fieldName="LICENSE_STATUS" caption="LICENSE_STATUS" visible="true"
formatSchema ="" />
        <Parameter fieldName="CERTIFICATE" caption="CERTIFICATE" visible="true"
formatSchema ="" />
    </Output>
</Source>
```

2.2.2.7. Report/Sources/Source/Query– SQL-выражение для получения выборки данных

Элемент строкового типа. Содержит SQL-выражение для получения выборки данных.

Пример

```
<Query>
   <![CDATA[
SELECT
s.FULL_NAME as "SNAME",
(case when s.DELETED = 0 then N'Да' else N'Нет' end) as "IS_OUTDATED",
reg.NAME as "REGION",
s.LAW_ADDRESS as "LAW_ADDRESS",
s.OGRN as "GOSREGNUM",
s.INN as "INN",
(N'Лицензия № ') as "LICENSE",
```

```
(N'Действует')  AS "LICENSE_STATUS",
(N'Свидетельство об аккредитации № ') as "CERTIFICATE",
(N'Действует') AS "CERTIFICATE_STATUS"
FROM SCHOOLS s
LEFT JOIN REGIONS reg ON reg.ID=s.REGIONS_FK
WHERE s.ID = @SchoolId
   ]]>
</Query>
```

2.2.2.8. Report/Sources/Source/Output – Настройка выходных параметров

Элемент комплексного типа. В разделе описываются выходные параметры выборки данных. Количество параметров должно быть равно количество столбцов, получаемых в результате выполнения SQL-запроса.

Пример

```
<Output>
    <Parameter fieldName="SNAME" caption="SNAME" visible="true" />
    <Parameter fieldName="IS_OUTDATED" caption="IS_OUTDATED" visible="true" />
    <Parameter fieldName="REGION" caption="REGION" visible="true" />
    <Parameter fieldName="LAW_ADDRESS" caption="LAW_ADDRESS" visible="true" />
    <Parameter fieldName="GOSREGNUM" caption="GOSREGNUM" visible="true" />
    <Parameter fieldName="INN" caption="INN" visible="true" />
    <Parameter fieldName="LICENSE" caption="LICENSE" visible="true" />
    <Parameter fieldName="LICENSE_STATUS" caption="LICENSE_STATUS" visible="true" />
    <Parameter fieldName="CERTIFICATE" caption="CERTIFICATE" visible="true" />
</Output>
```

2.2.2.9. Report/Sources/Source/Output/Parameter – Выходной параметр

Элемент простого типа. В разделе описывается выходной параметр.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
fieldName	String	Идентификатор параметры (должен совпадать с названием\псевдонимом поля из SQL-выборки	Обязательный
caption	String	Заголовок поля (атрибут используется только для типа шаблона Html)	Обязательный
visible	Boolean	Видимый (атрибут используется только для типа шаблона Html)	Обязательный
formatSchema	String	Путь к файлу шаблона	Необязательный

Пример

```
<Parameter fieldName="LAW_ADDRESS" caption="LAW_ADDRESS" visible="true" />
```

2.2.2.10.Report/Input– Входные параметры

Элемент комплексного типа. В разделе описываются входные параметры генерации отчета. Указанные в этом блоке параметры передаются в SQL-запрос (с добавлением префикса - @)

Дочерние элементы:

– Parameter (1..N) – используемые шаблоны для выгрузки отчета

Пример

```
<Input>
    <Parameter fieldName="REGIONS" caption="Регион" input="MultiPicker" size="Long"
lineBreakAfterInput="true">
        <Picker entityId="REGIONS"/>
    </Parameter>
    <Parameter fieldName="PERIOD_YEAR" caption="Год" input="DropDown" size="Normal"
lineBreakAfterInput="true" >
        <Dictionary connectionStringName="AKNDPP_FEDERAL" tableName="FORM_4_9"
keyFieldName="ID" displayFieldName="NAME" orderFieldName="NAME DESC"
cacheDurationInSeconds="3600" cacheByUser="false" >
            <Select>
                <![CDATA[
select DISTINCT YEAR(DATE) as "ID", CONVERT(varchar(100), YEAR(DATE)) as "NAME" FROM
dbo.FORM_4_9
            ]]>
            </Select>
        </Dictionary>
    </Parameter>
    <Parameter fieldName="REPORT_PERIOD" caption="Период предоставления отчетности"
input="DropDown" size="Normal" lineBreakAfterInput="true">
        <Dictionary connectionStringName="AKNDPP_FEDERAL" tableName="ReportPeriods"
keyFieldName="ID" displayFieldName="NAME" orderFieldName="CAPTION"
cacheDurationInSeconds="3600" cacheByUser="false">
            <Select>
                <![CDATA[
select ID, CAPTION AS "NAME" FROM dbo.ReportPeriods
            ]]>
            </Select>
        </Dictionary>
    </Parameter>
</Input>
```

2.2.2.11.Report/Input/Parameter– Входной параметр

Элемент комплексного типа. Раздел содержит описание входного параметра.

Дочерние элементы:

– Dictionary (0..1) – справочник (для выбора значения из выпадающего списка)

– Picker (0..1) – выбор значения их списка

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
fieldName	String	Наименование входного параметра (в тело SQL запроса передается как @ + fieldName)	Обязательный

caption	String	Задает заголовок	Обязательный
input	FormInputTypesEnum	Способ ввода данных	Обязательный
defaultValue	String	Значение по умолчанию	Необязательный
order	unsignedInt	Порядковый номер поля ввода (по умолчанию поля появляются в порядке их объявления)	Необязательный
lineBreakAfterInput	Boolean	Включает ввод с новой строки следующего поля (по умолчанию – false, тогда следующее поля ввода появится на той же строке что и текущее)	Необязательный
size	InputSizesEnum	Ширина поля на форме (по умолчанию – Normal)	Необязательный
cssPrefix	String	Настраивает css для отдельного поля формы	Необязательный

Пример

```
<Parameter fieldName="REGIONS" caption="Регион" input="MultiPicker" size="Long"
lineBreakAfterInput="true">
    <Picker entityId="REGIONS"/>
</Parameter>
```

2.2.2.12.Report/Input/Parameter/Dictionary - Справочник

Элемент комплексного типа. Раздел описывает справочник, который необходимо использовать при работе с параметров (ввод данных из выпадающего списка).

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
tableName	String	Название используемой в качестве справочника таблицы в БД	Обязательный
keyFieldName	String	Название ключевого поля	Обязательный
displayFieldName	String	Название отображаемого поля	Обязательный
codeFieldName	String	Название кодового поля (для внутренних связей)	Необязательный
orderFieldName	String	Название поля сортировки	Необязательный
alias	String	Псевдоним выборки данных справочника	Необязательный
cacheDurationInSeconds	UnsignedInt	Длительность хранения кеша. 0 – кеш не используется	Необязательный
cacheByUser	Boolean	Требуется кешировать данные в рамках пользователя	Необязательный

Пример

```
<Dictionary tableName="SCHOOL_TYPES" keyFieldName="ID" displayFieldName="NAME"
orderFieldName="NAME" cacheDurationInSeconds="3600" cacheByUser="false" >
    <Select>
        <![CDATA[
select ID ,NAME  from SCHOOL_TYPES  where NOT_TRUE = 0 order by NAME
        ]]>
    </Select>
</Dictionary>
```

2.2.2.13.Report/Input/Parameter/Dictionary/Select – SQL-выражение для получения справочника

Пустой элемент типа String. В этом разделе задается SQL-выражение для получения выборки данных, используемой в качестве справочника.

Пример

```
<Select>
   <![CDATA[
select ID ,NAME  from SCHOOL_TYPES  where NOT_TRUE = 0 order by NAME
   ]]>
</Select>
```

2.2.2.14.Report/Input/Parameter/Picker – Выбор из списка

Элемент комплексного типа. Выбор значения или значений из списка.

Атрибуты:

Имя атрибута	Тип Атрибута	Комментарий	Встречаемость
entityId	String	Наименование сущности, используемой для выбора записи	Обязательный

Пример

```
<Parameter fieldName="REGIONS" caption="Регион" input="MultiPicker" size="Long"
lineBreakAfterInput="true">
    <Picker entityId="REGIONS"/>
</Parameter>
```

2.2.3. Справочные элементы

2.2.3.1. ReportTypesEnum – Типы шаблонов отчета

Элемент типа string, не содержит дочерних элементов и атрибутов. Может принимать одно из следующих значений:

- Html – выгрузка в HTML-формат (без файла шаблона)
- Text – использование текстового шаблона (шаблон задается внутри метаданных)
- GridSimple – выгрузка в Excel без форматирования (без файла шаблона)

- GridFormatted – использование Excel-шаблона простого формата (шаблон задается в отдельном файле)
- Word – выгрузка в WORD (шаблон задается в отдельном файле)
- Excel – выгрузка в EXCEL (гибкая разметка, шаблон задается в отдельном файле)

2.2.3.2. InputParameterTypesEnum – типы элементов управления (способы ввода)

Элемент типа string, не содержит дочерних элементов и атрибутов. Задает один из следующих типов форматов, вводимых данных в фильтре:

- Text – текстовый ввод
- Boolean –выпадающий список из трех вариантов: да, нет, любой
- StrictBoolean – элемент управления «CheckBox»
- RadioBoolean – элемент управления «радиокнопки», доступны варианты: да, нет, все
- Date – выбор даты (без времени)
- DateTime – выбор даты и времени
- DropDown – выпадающий список
- MultiDropDown – выпадающий список с множественным выбором значений
- Number – ввод числа
- NumbersRange – диапазон нескольких чисел
- Picker – выбор значения из всплывающего окна со списком
- MultiPicker – множественный выбор значения из всплывающего окна со списком

2.2.3.3. InputParameterSizesEnum – размеры элементов управления

Простой элемент текстового типа. Задает длину поля ввода на форме. Принимает одно из трех значений:

- Long – длинный
- NormalLong – средне-длинный
- Normal – нормальный
- ShortNormal – средне-короткий
- Short – короткий

2.2.4. Язык разметки в файлах шаблонов

Важно! Все элементы языка разметки должны быть написаны в верхнем регистре.

2.2.4.1. Вывод скалярного и векторного значения

В шаблонах типа Html, Excel, Word доступны следующие параметры разметки:

- [[<источник>.<поле источника>]] - вывод одиночного значения (значение из первой строки результирующей таблицы запроса), пример [[BUILDERS.REGION]] .
- [[REPEAT:<источник>.<поле источника>]] - вывод всех значений в столбец (весь столбец результирующей таблицы запроса), пример [[REPEAT:BUILDERS.NAME]] .

Поддерживаются следующие функциональные возможности:

- Возможность выводить данные из нескольких источников на одном листе (источники и поля указываются в разметке нового формата).
- Возможность выводить одиночные значения в произвольных ячейках листа ([[<источник>.<поле источника>]] может быть определено в любой ячейке).
- Возможность выводить одиночные значения в ячейках со статичным текстом, пример ячейки "Наименование района: [[BUILDERS.РАЙОН]]".
- Копирование формул в строках вывода всех значений (строки с [[REPEAT:<источник>.<поле источника>]]).
- Объединение ячеек вывода одиночного значения (ячейки с [[<источник>.<поле источника>]], без REPEAT) в строках вывода всех значений (строки с [[REPEAT:<источник>.<поле источника>]]).
- Для переноса данных в ячейках Excel можно использовать команду !NEWLINE! в Sql-запросе.

2.2.4.2. Группировка данных (только Excel)

Язык разметки:

- [[GROUP_BEGIN:GROUPBY(<источник>.<поле источника>)]] - обозначение начала группы, в части GROUPBY(<источник>.<поле источника>) указано поле источника, по которому необходимо группировать данные, пример [[GROUP_BEGIN:GROUPBY(BUILDERS.РАЙОН)]] .
- [[GROUP_END]] - обозначение окончания группы.

Поддерживаются следующие функциональные возможности:

- Возможность выводить название группы в произвольной ячейке внутри группы. Название группы выводится с использованием обозначения [[<источник>.<поле источника>]] (вывод одиночного значения, если в
- строке с [[REPEAT:<источник>.<поле источника>]] - объединение ячеек). К примеру такое поле может располагаться сверху над списком или отсутствовать вообще.
- Для ячеек внутри группы с обозначением [[<источник>.<поле источника>]] выводить первое значение с фильтром по текущей группе (у каждой группы свое значение, если источник другой - фильтрацию не

– применять).
– Для ячеек внутри группы с обозначением [[REPEAT:<источник>.<поле источника>]] выводить все значения с фильтром по текущей группе в столбец (если источник другой - фильтрацию не применять).

2.2.4.3. Параметры для функций (только Excel)

Язык разметки:

– =<имя функции>(GROUP_C<индекс столбца>) - параметр GROUP_C<индекс столбца> для промежуточных функций в группах, пример СУММ(GROUP_C12) преобразуется в СУММ(R[-3]C12:R[-1]C12) .
– =<имя функции>(TOTAL_C<индекс столбца>) - параметр TOTAL_C<индекс столбца>, пример СУММ(TOTAL_C12) преобразуется в СУММ(R[-23]C12:R[-21]C12;R[-15]C12:R[-13]C12;R[-7]C12:R[-5]C12) .

Поддерживаются следующие функциональные возможности:

– Вместо параметра GROUP_C<индекс столбца> в функциях подставляется диапазон строк группы (только строки с [[REPEAT:<источник>.<поле источника>]] в группе)
– Вместо параметра TOTAL_C<индекс столбца> в функциях подставляется диапазон всех выводимых строк (строки с [[REPEAT:<источник>.<поле источника>]] во всем отчете)
– Возможность использования параметра TOTAL_C<индекс столбца> в отчетах без группировки.
– Поддержка написаний функций как на русском, так и на английском языках.
– Возможность использования любых функций Excel (простых, сложных и составных), пример СУММ(GROUP_C12) - СРЗНАЧ(GROUP_C11) .

2.2.4.4. Условное скрытие блоков (только Word)

Язык разметки:

– [[HIDEIF_BEGIN:<критерий сравнения>(<идентификатор источника>)]] – обозначения начала блока, который требуется скрывать по условию, например [[HIDEIF_BEGIN:NOROWS(SOURCE1)]]
– [[HIDEIF_END]] – обозначение окончания блока

Поддерживаются следующие функциональные возможности:

– Скрываемый по условию блок информации должен быть помещен между тегами HIDEIF_BEGIN и HIDEIF_END

Поддерживаются следующие критерии сравнения:

– NOROWS – ноль строк данных в источнике

2.2.4.5. Клонирование блоков (только Word)

Язык разметки:

– [[CLONE_BEGIN:<идентификатор источника>.<идентификатор поля>]] – обозначения начала блока, который требуется скрывать по условию, например [[CLONE_BEGIN:SUP.ID]]
– [[CLONE_END]] – обозначение окончания клонируемого блока
– [[<идентификатор источника>.<наименование поля>]] (ID=[[<идентификатор клонируемого блока источника>.<наименование поля>]]) – вывод скалярного значения другого источника внутри клонируемого блока (применяется фильтрация выборки второго источника), например [[SUP.NUMBER]] (ID=[[SUP.ID]])
– [[REPEAT:<идентификатор источника 2>.<наименование поля>]][[FILTER(<идентификатор источника 1>,<идентификатор источника 1>.<наименование поля>)]] – вывод скалярного значения другого источника внутри клонируемого блока (применяется фильтрация выборки второго источника), например [[REPEAT:SUP_EDU01_1.N]][[FILTER(SUP_EDU01_1,SUP_EDU01_1.GROUP_ID)]]
– [[HIDEIF_BEGIN:NOROWS_FILTER(<идентификатор источника 1>,<идентификатор источника 1>.<наименование поля>)]] - Условное скрытие блоков внутри клонируемого блока на основе данных другого источника, например [[HIDEIF_BEGIN:NOROWS_FILTER(SUP_EDU01_1,SUP_EDU01_1.GROUP_ID)]]

Поддерживаются следующие функциональные возможности:

– Клонируемый блок информации должен быть помещен между тегами CLONE_BEGIN и CLONE_END
– Количество блоков-клонов зависит от количество уникальных значений в параметре (идентификатор поля), заданном в теге CLONE_BEGIN
– Возможно использовать вложенные клонируемые блоки

Пример

[[CLONE_BEGIN:SUP.ID]]
Приложение № [[SUP.N]] (ID=[[SUP.ID]])
Уровни образования
[[REPEAT:LEV.ID]] (SUP_FK=[[REPEAT:LEV.SUP_FK]])
Образовательные программы
[[REPEAT:PROG.ID]][[FILTER(PROG.SUP_FK)]]
[[CLONE_END]]

2.2.5. Шаблоны отчетов

2.2.5.1. Excel-шаблоны с гибкой настройкой

Создание шаблона происходит в среде MS Excel (или в аналогичной среде по работе с электронными таблицами).

Шаблон Excel-формата должен быть сохранен с расширением .xmla, для этого сохраняем файл Excel в формате Таблица XML 2003 (.xml), после этого вручную переименовываем расширение.

2.2.5.2. Word-шаблоны

Создание шаблона происходит в среде MS Word с расширением DOCX (или в аналогичной среде по работе с Word-документами).

2.2.5.3. Текстовые шаблоны

Шаблон задается внутри метаданных в разделе TextTemplate.

2.2.6. Общий вид XML-файла

```
<Report id="id1" name="name1">
    <Templates>
        <Template id="id2" name="name2" type="Text"
templateFileName="templateFileName2">
            <TextTemplate>...</TextTemplate>
        </Template>
    </Templates>
    <Sources>
        <Source id="id1" name="name1" connectionStringName="connectionStringName1">
            <Query>...</Query>
            <Output>
                <Parameter fieldName="fieldName1" caption="caption1" hidden="false"
formatSchema="formatSchema1" />
            </Output>
        </Source>
    </Sources>
    <Input>
        <Parameter fieldName="fieldName1" caption="caption1" input="Text"
defaultValue="defaultValue1" size="Normal" lineBreakAfterInput="true"
cssPrefix="cssPrefix1">
            <Dictionary connectionStringName="connectionStringName1"
tableName="tableName1" keyFieldName="keyFieldName1"
displayFieldName="displayFieldName1" codeFieldName="codeFieldName1"
orderFieldName="orderFieldName1" alias="alias1" cacheDurationInSeconds="0"
cacheByUser="false">
                <Select>...</Select>
            </Dictionary>
            <Picker entityId="entityId1" />
        </Parameter>
    </Input>
</Report>
```

2.3. Описание метаданных модуля «Управление доступом»

2.3.1. Общее описание

Описание ролевой модели состоит из следующих частей:

- Описание допустимых действий в системе – метаданные действий
- Описание интерфейсов системы – метаданные интерфейсов
- Описание меню системы – метаданные меню
- Метаданные ролей пользователей системы – метаданные ролей

Метаданные ролевой модели хранятся в файлах формата XML в заранее заданной папке внутри папки приложения:

- Actions.xml – файл с метаданными действий
- Interfaces.xml – файл с метаданными интерфейсов
- Menu.xml – файл с метаданными меню
- Roles.xml – файл с метаданными ролей

Относительный путь (относительно корневой папки приложения – папки, на которую настроен IIS) к папке, содержащей метаданные ролевой модели задается в конфигурационном файле приложения в секции KURSConfiguration в разделе MetadataLocations с помощью ключа accessControl.

Пример

```
<KURSConfiguration>
    <MetadataLocations entities="~/_Metadata" accessControl="~/_SystemMetadata"
reports="~/_Reports" />
```

2.3.2. Описание допустимых действий в системе

2.3.2.1. Структура метаданных

2.3.2.1.1. Actions – Допустимые действия в системе

Комплексный тип. Корневой элемент. Содержит описание допустимых в системе действий.

Дочерние элементы:

- Action (0..N) – допустимое действие

Пример

```
<Actions>
    <Action id="ReportsAdministration" name="Настройка отчетов" />
    <Action id="ViewRegionMonitoring" name="Региональный мониторинг" />
    <Action id="AccessControlAdministration" name="Управление доступом" />
    <Action id="CatalogsAdministration" name="Системные справочники" />
    <Action id="ViewAdminReports" name="Отчеты администратора" />
    <Action id="ViewFederalMonitoring" name="Федеральный мониторинг" />
    <Action id="ViewSupportMonitoring" name="Мониторинг службы поддержки" />
```

```
    <Action id="Work" name="Работа с приложением" />
    <Action id="Admin" name="Мониторинг(Администратор)" />
</Actions>
```

2.3.2.1.2. Actions/Action – Допустимое действие

Элемент комплексного типа. Раздел описания допустимого действия в системе.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
Id	String	Уникальный ключ действия	Обязательный
name	String	Наименование действия	Обязательный

Пример

```
<Action id="ViewRegionMonitoring" name="Региональный мониторинг" />
```

2.3.3. Описание интерфейсов системы

2.3.3.1. Структура метаданных

2.3.3.1.1. Interfaces – Интерфейсы системы

Комплексный тип. Корневой элемент. Содержит описание доступных в системе интерфейсов.

Дочерние элементы:

– Interface (0..N) – интерфейс системы

Пример

```
<Interfaces>
    <Interface id="Users_List" type="MDGrid" actionId="Administration_Users">
        <MDGrid entityId="Users" />
    </Interface>
    <Interface id="Users_Form" type="MDForm" actionId="Administration_Users">
        <MDForm entityId="Users" />
    </Interface>
    <Interface id ="LICENSE_REPORT_DEBUG" type="Report" >
        <Report reportId="LICENSE_REPORT_DEBUG" />
    </Interface>
    <Interface id="SVOD_REPORT" type="Url" actionId="RptSigned">
        <Url value="../SSRS/SSRSReport.aspx" />
    </Interface>
</Interfaces>
```

2.3.3.1.2. Interfaces/Interface – Интерфейс системы

Элемент комплексного типа. Раздел описания допустимого действия в системе.

Допустим один из следующих дочерних элементов:

– MDGrid (0..1) – платформенный список

– MDForm (0..1) – платформенная форма

– Report (0..1) – платформенный отчет

– Url (0..1) – прикладной интерфейс

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Уникальный ключ действия	Обязательный
type	InterfaceTypesEnum	Тип интерфейса	Обязательный
actionId	String	Идентификатор действия (привязка интерфейса к допустимому действию в системе)	Необязательный

Пример

```
<Interface id="Users_List" type="MDGrid" actionId="Administration_Users">
    <MDGrid entityId="Users" />
</Interface>
```

2.3.3.1.3. Interfaces/Interface/MDGrid – Платформенный список

Элемент комплексного типа. Раздел описания допустимого действия в системе.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
entityId	String	Идентификатор сущности	Обязательный

Пример

```
<Interface id="Users_List" type="MDGrid" actionId="Administration_Users">
    <MDGrid entityId="Users" />
</Interface>
```

2.3.3.1.4. Interfaces/Interface/MDForm – Платформенная форма

Элемент комплексного типа. Раздел описания допустимого действия в системе.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
entityId	String	Идентификатор сущности	Обязательный

Пример

```
<Interface id="Users_Form" type="MDForm" actionId="Administration_Users">
    <MDForm entityId="Users" />
</Interface>
```

2.3.3.1.5. Interfaces/Interface/Report – Платформенный отчет

Элемент комплексного типа. Раздел описания допустимого действия в системе.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
reportId	String	Идентификатор отчета	Обязательный

Пример

```
<Interface id ="LICENSE_REPORT" type="Report" >
    <Report reportId="LICENSE_REPORT" />
</Interface>
```

2.3.3.1.6. Interfaces/Interface/Url – Прикладной интерфейс

Элемент комплексного типа. Раздел описания допустимого действия в системе.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
value	String	Адрес страницы, к которой необходимо предоставить доступ	Обязательный

Пример

```
<Interface id="SVOD_REPORT" type="Url" actionId="RptSigned">
    <Url value="../SSRS/SSRSReport.aspx" />
</Interface>
```

2.3.4. Справочные элементы

2.3.4.1. InterfaceTypesEnum – Типы интерфейсов

Элемент типа string, не содержит дочерних элементов и атрибутов. Может принимать одно из следующих значений:

- MDGrid – платформенный список (таблица)
- MDForm – платформенная форма просмотра\ввода данных
- Report – платформенный отчет (выгрузка данных)
- Url – прикладной (не платформенный) интерфейс, определяется ссылкой на страницу

2.3.5. Описание меню системы

2.3.5.1. Структура метаданных

2.3.5.1.1. Menu – Меню системы

Комплексный тип. Корневой элемент. Содержит описание меню системы. Порядок отображения элементов меню однозначно задается последовательностью описания элементов меню в метаданных.

Дочерние элементы:

- MenuItem (0..N) – элемент меню

Пример

```
<Menu>
    <MenuItem id="MonitoringOperatorView" name="Детальные сведения по всем регионам"
actionId="ViewSupportMonitoring" interfaceId="MonitoringOperatorView_List"
tooltip="Мониторинг" />
    <MenuItem id="Administration" name="Администрирование" actionId="Administration"
tooltip="Администрирование">
        <MenuItem id="AllUsers" name="Пользователи" actionId="Administration_Users"
interfaceId="Users_List" tooltip="Пользователи" />
        <MenuItem id="ActionsLog_List" name="Журнал событий системы"
actionId="Administration_Users" interfaceId="ActionsLog_List" tooltip="" />
        <MenuItem id="ReportActivity" name="Активность пользователей"
actionId="Administration_Users" interfaceId="USERS_REPORT" tooltip="" />
    </MenuItem>
</Menu>
```

2.3.5.1.2. Menu/MenuItem – Элемент меню

Элемент комплексного типа. Раздел описания элемента меню. Элемент меню может выступать в качестве родителя для вложенных элементов меню.

Дочерние элементы:

- MenuItem (0..N) – вложенный элемент меню

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Уникальный ключ элемента меню	Обязательный
name	String	Отображаемое наименование элемента меню	Обязательный
actionId	String	Идентификатор действия (привязка элемента меню к допустимому действию в системе)	Необязательный
interfaceId	String	Идентификатор интерфейса (привязка интерфейса к элементу меню)	Необязательный
tooltip	String	Текст всплывающей подсказки при наведении курсора мыши на элемент меню	Необязательный

Пример

```
<MenuItem id="Administration" name="Администрирование" actionId="Administration"
tooltip="Администрирование">
    <MenuItem id="AllUsers" name="Пользователи" actionId="Administration_Users"
interfaceId="Users_List" tooltip="Пользователи" />
    <MenuItem id="RegUsers" name="Региональные пользователи"
actionId="Administration_Users" interfaceId="Users_Regional_List" tooltip="" />
    <MenuItem id="FedUsers" name="Федеральные пользователи"
actionId="Administration_Users" interfaceId="Users_Federal_List" tooltip="" />
    <MenuItem id="OoUsers" name="Пользователи ОО" actionId="Administration_Users"
interfaceId="APPLICANT_USER_List"  tooltip="" />
    <MenuItem id="ActionsLog_List" name="Журнал событий системы"
actionId="Administration_Users" interfaceId="ActionsLog_List" tooltip="" />
    <MenuItem id="ReportActivity" name="Активность пользователей"
actionId="Administration_Users" interfaceId="USERS_REPORT" tooltip="" />
</MenuItem>
```

2.3.6. Описание ролей пользователей

2.3.6.1. Структура метаданных

2.3.6.1.1. Roles – Роли пользователей

Комплексный тип. Корневой элемент. Содержит описание доступных в системе ролей.

Дочерние элементы:

- Role (0..N) – роль пользователя

Пример

```
<Roles>
    <Role id="Admin" name="Администратор" defaultInterfaceId="Users_List">
        <Access actionId="Work" canCreate="true" canRead="true" canUpdate="true"
canDelete="true" />
        <Access actionId="CatalogsAdministration" canCreate="true" canRead="true"
canUpdate="true" canDelete="true" />
        <Access actionId="AccessControlAdministration" canCreate="true" canRead="true"
canUpdate="true" canDelete="true" />
        <Access actionId="ReportsAdministration" canCreate="true" canRead="true"
canUpdate="true" canDelete="true" />
        <Access actionId="Admin" canCreate="true" canRead="true" canUpdate="true"
canDelete="true" />
    </Role>
    <Role id="RegionalMonitoring" name="Региональный мониторинг"
defaultInterfaceId="MonitoringRegionsView_List">
        <Access actionId="Work" canCreate="true" canRead="true" canUpdate="true"
canDelete="true" />
        <Access actionId="ViewRegionMonitoring" canCreate="false" canRead="true"
canUpdate="false" canDelete="false" />
    </Role>
    <Role id="SupportMonitoring" name="Мониторинг службы поддержки"
defaultInterfaceId="MonitoringOperatorView_List">
        <Access actionId="Work" canCreate="true" canRead="true" canUpdate="true"
canDelete="true" />
        <Access actionId="ViewSupportMonitoring" canCreate="true" canRead="true"
canUpdate="true" canDelete="true" />
    </Role>
</Roles>
```

2.3.6.1.2. Roles/Role – Элемент меню

Элемент комплексного типа. Раздел описания роли пользователя.

Дочерние элементы:

- Access (1..N) – доступ к действию

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
id	String	Уникальный ключ элемента меню	Обязательный
name	String	Отображаемое наименование элемента меню	Обязательный

defaultInterfaceId	String	Идентификатор интерфейса, отображаемого при входе в систему после авторизации	Необязательный

Пример

```
<Role id="RegionalMonitoring" name="Региональный мониторинг"
defaultInterfaceId="MonitoringRegionsView_List">
    <Access actionId="Work" canCreate="true" canRead="true" canUpdate="true"
canDelete="true" />
    <Access actionId="ViewRegionMonitoring" canCreate="false" canRead="true"
canUpdate="false" canDelete="false" />
</Role>
```

2.3.6.1.3. Roles/Role/Access – Доступ к действию

Элемент комплексного типа. Раздел описания роли пользователя.

Атрибуты:

Имя атрибута	Тип атрибута	Комментарий	Встречаемость
actionId	String	Идентификатор допустимого действия	Обязательный
canCreate	String	Разрешить создание (по умолчанию разрешено)	Необязательный
canRead	String	Разрешить чтение (по умолчанию разрешено)	Необязательный
canUpdate	String	Разрешить обновление (по умолчанию разрешено)	Необязательный
canDelete	String	Разрешить удаление (по умолчанию разрешено)	Необязательный

Пример

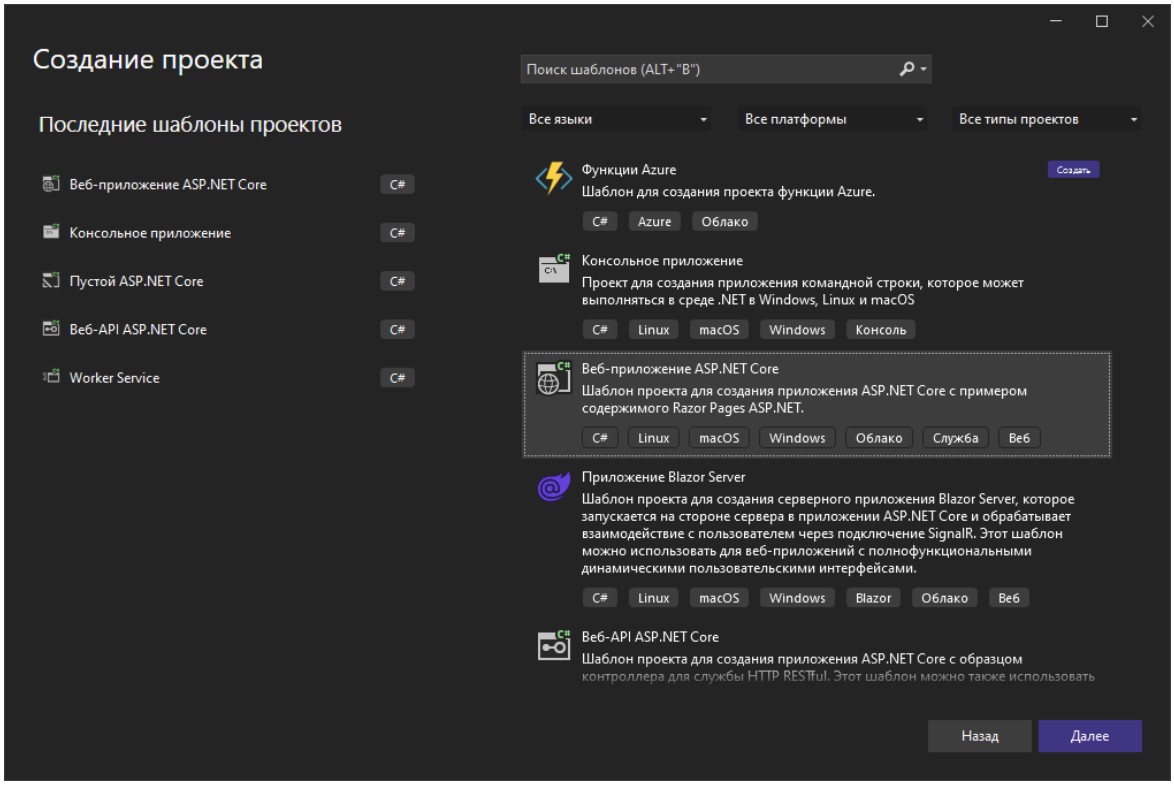
```
<Access actionId="AccessControlAdministration" canCreate="true" canRead="true"
canUpdate="true" canDelete="true" />
```

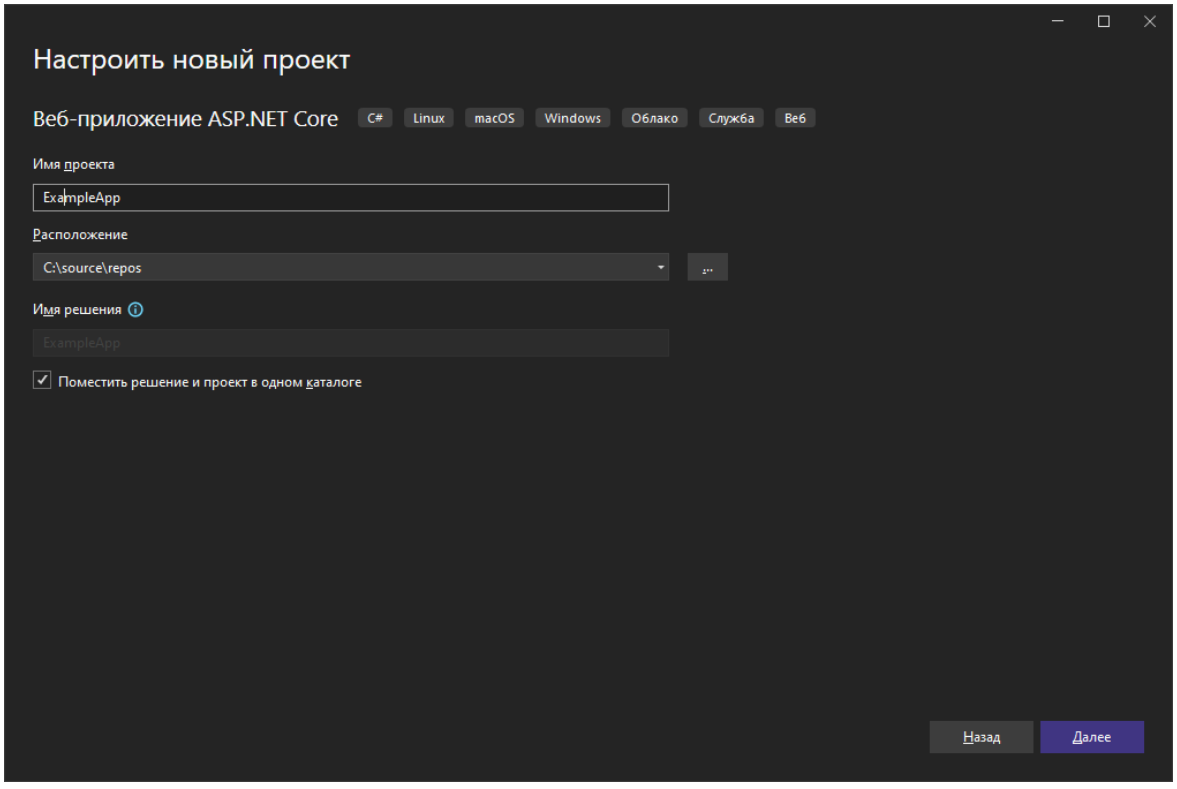
3. ИНСТРУКЦИЯ ПО УСТАНОВКЕ И ЗАПУСКУ ПРОГРАММЫ

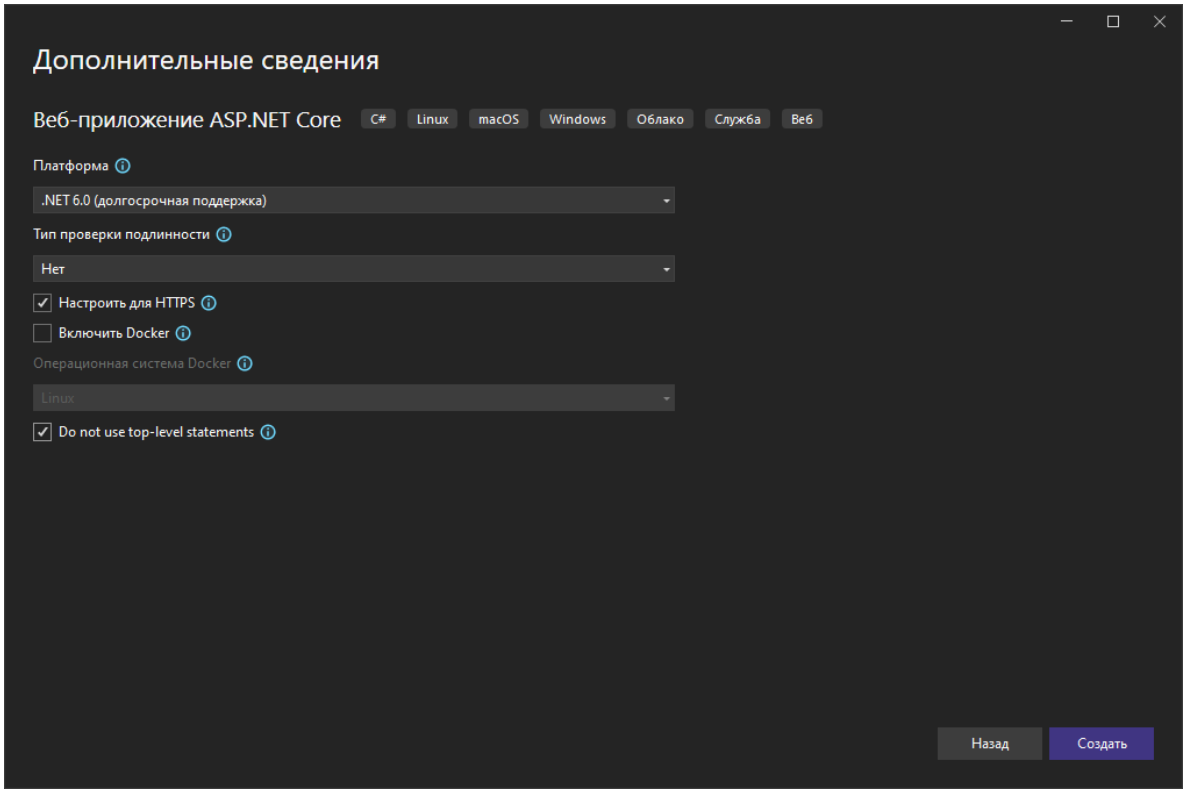
Данное руководство описывается на примере шаблона проекта приложения ASP.NET Core с примером содержимого Razor Pages ASP.NET:

3.1. Создание веб-приложения прикладной системы

Шаг 1: Необходимо создать веб-приложение ASP.NET Core





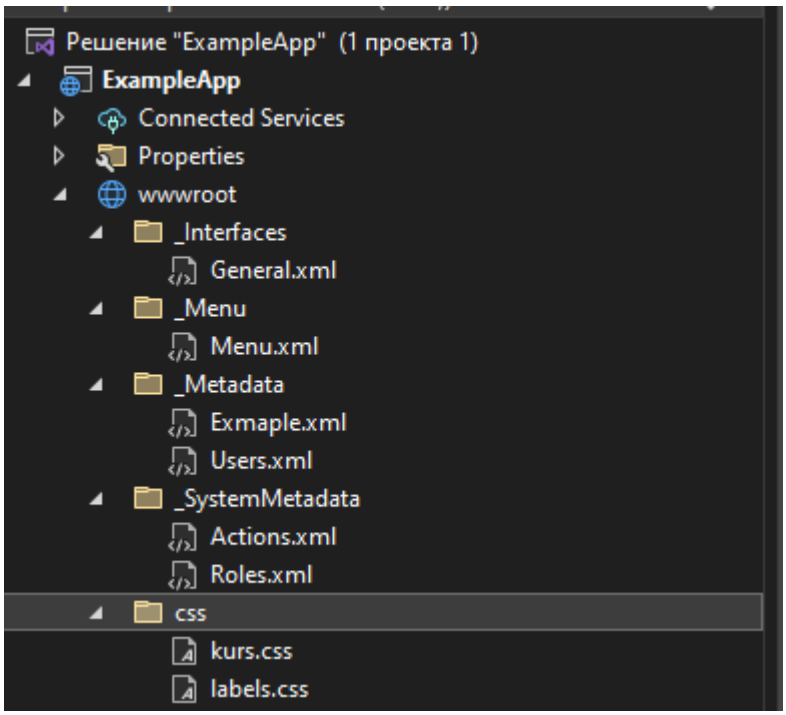


3.2. Наполнение веб-приложения метаданными КУРС

Шаг 2: наполнение проекта обязательными файлами

Для запуска приложения необходимо некоторые обязательные файлы:

1) В корневой каталог добавить обязательные метаданные:
   - Roles.xml
   - Actions.xml
   - Interfaces.xml
   - Menu.xml
   - Users.xml
2) В корневой каталог добавить css-файлы:
   - kurs.css
   - labels.css



3.3. Инициализация компонентов КУРС в веб-приложении прикладной системы

Шаг 3: настройка параметров запуска приложения

В Program.cs необходимо установить класс Startup в качестве стартового посредством использования метода UseStartup()

В Startup.cs необходимо зарегистрировать сервисы и настроить обработку запросов:

Метод ConfigureServices(IServiceCollection services):

- объекту CookiePolicyOptions установить:
  - CheckConsentNeeded = context => true;
  - MinimumSameSitePolicy = SameSiteMode.None;
- добавить сервисы MVC:
  - services.AddMvc(options => { options.EnableEndpointRouting = false; });
  - services.AddMvc().AddRazorPagesOptions(options => { options.Conventions.AddPageRoute("/Default", ""); });
- добавить контекст выполнения
  - services.AddHttpContextAccessor();
- добавить объекты сервисов:
  - services.AddSingleton<Config>();
  - services.AddSingleton<WebConfig>();
  - services.AddSingleton<WebContext>();

- при необходимости настроить параметры Kestrel и IIS
  - options.Limits.MaxRequestBodySize = 1073741824;
  - options.AllowSynchronousIO = true;

В методе Configure(IApplicationBuilder app, IWebHostEnvironment env):

- добавить возможность использования статических файлов:
  - app.UseStaticFiles();
- добавить компонент CookiePolicyMiddleware:
  - app.UseCookiePolicy();
- добавить компонент RouterMiddleware:
  - app.UseMvc();
- получить доступ к текущему HttpContext:
  - var httpContextAccessor = app.ApplicationServices.GetRequiredService<IHttpContextAccessor>();
- инициализировать конфигурации приложения:

Для обработки конфигурации необходимо реализовать классы Config, WebConfig, затем провести инициализацию:

- var config = app.ApplicationServices.GetRequiredService<Config>();
- var webConfig = app.ApplicationServices.GetRequiredService<WebConfig>();
- KURS.Core.Configuration.KURSConfig.Init(() => httpContextAccessor.HttpContext == null ? config : httpContextAccessor.HttpContext.RequestServices.GetRequiredService<Config>());
- KURS.Core.Web.Configuration.KURSWebConfig.Init(() => httpContextAccessor.HttpContext == null ? webConfig : httpContextAccessor.HttpContext.RequestServices.GetRequiredService<WebConfig>());
-

- инициализировать контекст путей веб-приложения

Для инициализации контекста путей необходимо реализовать класс WebContext, затем провести инициализацию:

- var webContext = app.ApplicationServices.GetRequiredService<WebContext>();
- KURS.Core.Web.KURSWebContext.Init(() => httpContextAccessor.HttpContext == null ? webContext :

httpContextAccessor.HttpContext.RequestServices.GetRequiredService<WebContext>());

- KURS.Core.KURSOperationContext.Init(() => httpContextAccessor.HttpContext == null ? webContext : httpContextAccessor.HttpContext.RequestServices.GetRequiredService<WebContext>());

– инициализировать другие необходимые классы и подключить базы данных
  - KURS.Core.Web.KURSWebContext.Init()
  - KURS.Core.KURSOperationContext.Init()
  - KURS.Core.Web.KURSResourcesManager.Init();
  - KURS.Core.Web.Security.AuthHelper.Init();
  - KURS.Core.Common.DataAccess.DBObjectsFactory.InjectConnectionObject();
  - KURS.Core.Metadata.Reporting.ReportingManager.Init();
  - KURS.Core.Metadata.AccessControl.AccessControlManager.Init(KURS.Core.Metadata.Entities.MetadataManager.Init();
  - Конструктор ролей: KURS.Core.Web.Controls.ControlFactory.RegisterConstructor<KURS.Core.Web.Controls.Metadata.MetadataCustomField< KURS.Core.Web.Models.Metadata.Forms.FormFieldModel>, KURS.Core.Web.Models.Metadata.Forms.FormFieldModel>(() => { return new KURS.Core.Web.Controls.Metadata.UserRolesCustomField<KURS.Core.Web.Models.Metadata.Forms.FormFieldModel> (); });
  - KURS.Core.Web.Extendable.Notifications.ActionsInitializer.Init(null);
  - Указать пути к папкам с метаданными
  - Управление пользователями: ServicesManager.RegisterService<UniversalEntity>("Users", new UserMetadataService("Users"));
  - SessionEntityService.InitAll();

3.4. Описание файла конфигурации приложения appsettings.json

Файл конфигурации должен содержать в себе секцию “KURS”, состоящую из следующих частей:

- UI
- Notifications
- WebAuth

- Log
- Metadata
- Files
- Database
- Security
- Smtp
- Format

Описание подсекций:

UI. Настройки интерфейса системы:

– WaitingDelaySec – устанавливает задержку (в секундах) перед показом анимации загрузки

– NoticeHideDelaySec – устанавливает время (в секундах), в течение которого отображаются всплывающие сообщения

Notifications. Настройки подсистемы уведомлений пользователей:

– Enabled – определяет, работает ли подсистема уведомлений пользователя

– CheckIntervalSec – устанавливает период опроса сервера для получения новых уведомлений (в секундах)

WebAuth. Настройки авторизации пользователей:

– CookieName – наименование файла cookie

– TokenLifetimeHours – время жизни токена авторизации (в часах)

– TokenRefreshHours – время обновления токена при активности пользователя

– TokenAESKey – ключ алгоритма AES-шифрования

Log. Настройки логирования событий системы:

– LogInOut – определяет, записывать ли в лог вход/выход из Системы

– LogCUD - определяет, записывать ли в лог создание, изменение, удаление какой-либо записи из базы данных (стандартными кнопками)

– LogCUDErrors - определяет, записывать ли в лог ошибки при создании, изменении, удалении какой-либо записи из базы данных (стандартными кнопками)

– LogReports - определяет, записывать ли в лог факт выгрузки отчёта из Системы

– LogNotifications – определяет, записывать ли в лог факт отправки уведомления

– LogUncaughtExceptions – определяет, записывать ли в лог необработанные ошибки

– LogButtons – определяет, записывать ли в лог факт нажатия на кнопку

– IncludeExceptionStackTrace – определяет, включать ли в ошибку трассировку стека

– ConnectionStringName – наименование строки подключения для записи лога в базу данных

Metadata. Настройки использования метаданных:

– StorageMode – способ хранения метаданных приложения. Принимает параметры *FileSystem, Database*
– EntitiesPath – путь к папке с основными файлами метаданных Системы
– InterfacesPath – путь к папке с описанием интерфейсов Системы
– MenuPath – путь к папке с файлом описания меню Системы
– AccessControlPath – путь к папке с файлами описания ролевой модели Системы
– ReportsPath – путь к папке с файлами метаданных отчётов Системы

Files. Настройки файлового хранилища:

– MaxSizeMB – ограничение на размер загружаемого файла в одно поле (в Мегабайтах)
– Mode – режим работы файлового хранилища. Принимает значения *FileSystem, Database*
– StoragePath – путь к основному файловому хранилищу Системы
– UploadStoragePath – путь к временному файловому хранилищу Системы

Database. Настройки баз данных:

– OLAPCommandTimeout
– OLTPCommandTimeout
– ConnectionStrings – массив строк подключений к базам данных:
  o Name – наименование строки подключения
  o Value – строка подключения в соответствии с используемой СУБД
  o Provider – провайдер базы данных

Security. Настройки безопасности учётных записей и приложения:

– MaxLoginAttempts – максимальное число попыток неудачного входа в систему
– BlockOnMaxAttemptsForMinutes – время блокировки учётной записи после достижения максимально числа попыток входа в систему (в минутах)
– PasswordLifetimeDays – время жизни пароля учётной записи (в днях)
– BlockOnInactivityDays -
– UseStrongPasswords – определяет, использовать ли усиленные пароли
– UseTemporaryPasswords – определяет, использовать ли временные пароли
– AllowResetPassword – определяет, можно ли сбросить пароль
– ResetPasswordTitle – заголовок формы сброса пароля пользователя
– ResetPasswordMessage – сообщение при сбросе пароля пользователя

- ShowExceptionDetails – определяет, показывать ли полную ошибку в окне Системы
- CacheReloadKey – ключ для обновления метаданных в кэше

Smtp. Настройки почтового сервера:

- Host – адрес почтового сервера
- Port – порт для подключения к почтовому серверу
- EnableSsl – определяет, использовать ли SSL-шифрование
- UserName – логин учётной записи
- From – почтовое имя отправителя
- Password – пароль учётной записи

Format. Настройки отображения типов данных:

- Schemas – массив используемых форматов отображения
  - Name – наименование формата
  - BooleanFormat – блок настроек отображению булевых значений
    - TrueText – отображаемый текст для значения true
    - FalseText – отображаемый текст для значения false
  - BoundaryValuesFormat – блок настроек граничных значений
    - NullText – отображаемый текст для NULL-значения
    - ZeroText – отображаемый тест
    - EmptyStringText – отображаемый текст для пустой строки
  - DateFormat – блок настроек отображений формата дат
    - Format – формат отображения даты. Принимает значения *Short, Long*
    - IncludeTime – определяет, отображать ли время
  - LongStringFormat – блок настроек для больших текстовых значений
    - truncateToLength – определяет, сколько символов необходимо оставить с начала строки. Если 0 – то значение не обрезается