Инновационные разработки

Знакомство с разработками ТПУ

Для ускорения разработки корпоративных приложений - для сотрудников УЦ были разработаны сервисы по созданию, настройки, валидации и генерации данных на основе параметрических SQL запросов. В связке с приложениями и сквозной авторизацией - получились следующие программные компоненты.

1. Динамические API методы для получения данных в режиме только-чтение.
2. Динамические формы ввода данных (добавление, редактирование).
3. Динамические группы форм ввода (заявки на создание документов, опросы, ввод сложных форм).
4. Формирование нагруженных отчётов в параллельных потоках на отдельном сервере базы данных.
5. Динамические правила доступа к контенту на основе персонализированных параметров.

Сущность (Entity)

Описывает базовое понятия работы с объектами

Общие сведения

Сущности (Entity) необходимы для описания **моделей в контексте ORM (Object-Relational Mapping) — это класс, который представляет таблицу в базе данных**. Каждая модель содержит поля, соответствующие столбцам таблицы. Эти поля определяют тип данных, валидацию и другие свойства для хранения, извлечения и обновления данных.

Создание сущностей возможно через интерфейс по ссылке https://service2.tpu.ru/entity/index.html

Сущность имеет следующие свойства:

1. **Параметры** (название и тип данных)
2. **Класс реализации** (ссылка на класс PHP, который описывают данную модуль в виде **ORM**)
3. Ссылки на **синхронизаторы**, которые отслеживают изменения в базе данных и производят независимую синхронизацию данных с внешними приложениями по средствам передачи JSON пакета на API приложения.

> При изменении структуры таблицы в базе данных, нужно в обязательном порядке очистить кэш данной сущности в приложении для корректной работы сервисов ТПУ.

Динамические источники данных (DID)

Динамический источник данных - это SQL запрос с параметрами, который может возвращать разные наборы данных в зависимости от указанных настроек. Результат работы будет использован практически во всех динамических компонентах.

Общие сведения о динамическом источнике данных

Динамические источники данных с разными типами имеют разное назначение и тип возвращаемого значения. В динамическом источнике данных можно указать параметры, которые будет переданы внутрь SQL при выполнении. Динамический источник данных может экспортировать динамические колонки, сгруппированные по категориям, а так же есть фильтры для выборки данных.

Типы динамических источников данных

1. **Валидатор** - используется для проверки системных объектов на наличие ошибок в данных или логики заполнения
2. **Кортеж** - возвращает набор данных 1 строку с несколькими колонками
3. **Личность** - возвращает одну колонку в которой находится ИД личности
4. **Логическое условие** - возвращает значение 0 или 1 в результате выполнения SQL запроса
5. **Массив** - возвращает произвольный набор данных из M строк и N столбцов
6. **Отчет** - разновидность типа Массив, но используется для построения динамических отчетов
7. **Скалярное** значение - возвращает скалярное значение из базы данных
8. **Список** - используется для динамических выпадающих списков
9. **Студент** - возвращает ИД личного дела студента, ИД личности и ИД группы, используется в целевых аудиториях и списках визирующих лиц
10. **Сотрудник** - возвращает ИД личного дела сотрудника, ИД личности, ИД должности и ИД подразделения, используется в целевых аудиториях и списках визирующих лиц
11. **Классификатор** - используется для работы через API со справочниками, возвращает ИД, полное название, может содержать краткое значение и статус, а так же могут присутствовать другие опциональные атрибуты.

Динамические источники данных используются везде, где нужно применять быстро меняющуюся логику приложения.

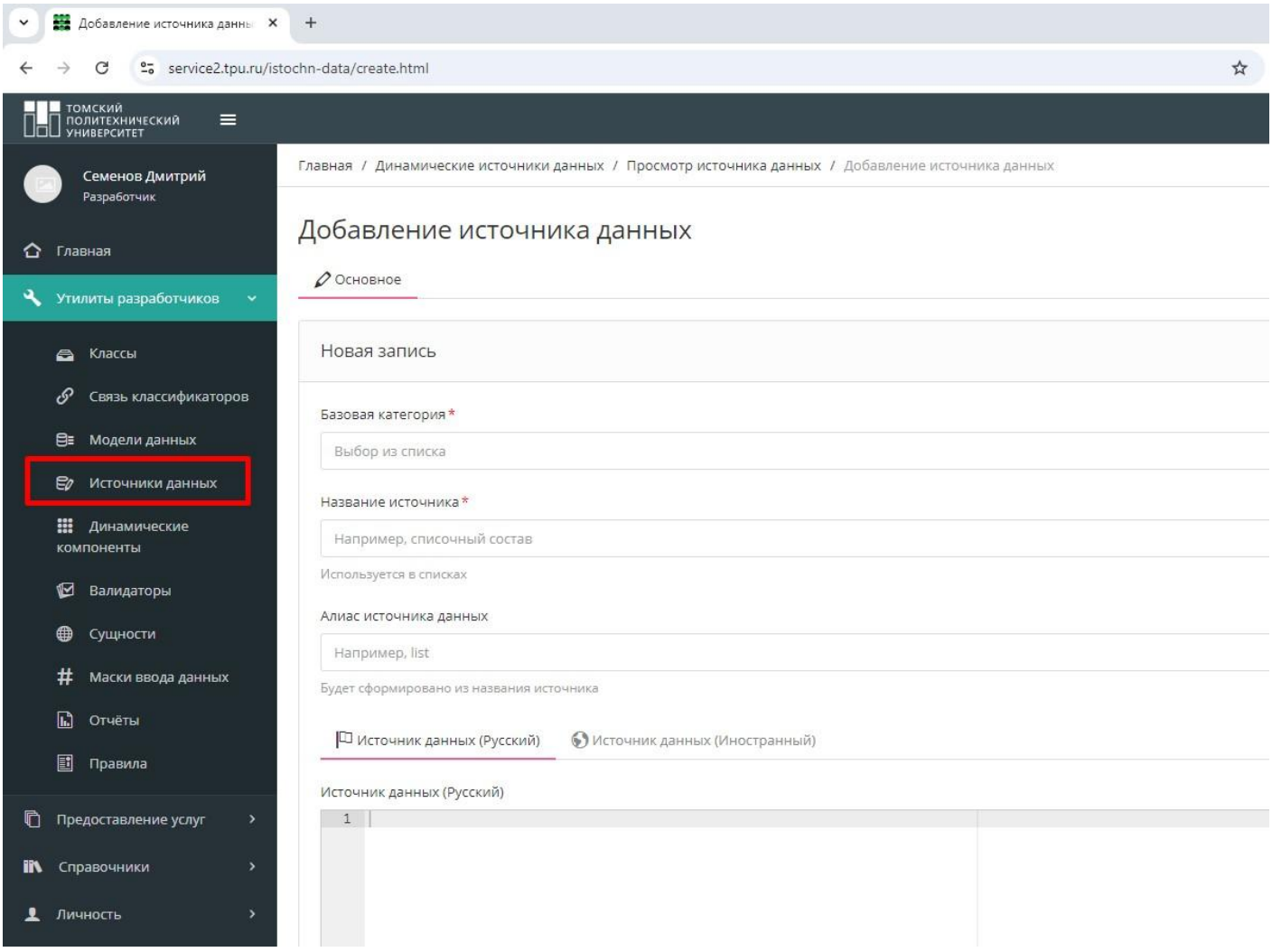
- Сформировать список визирующих лиц
- Определить видимость элементов ввода данных на форме

- Экспортировать данные во внешнюю систему
- Проверить данные на наличие ошибок
- Сформировать бланк документа

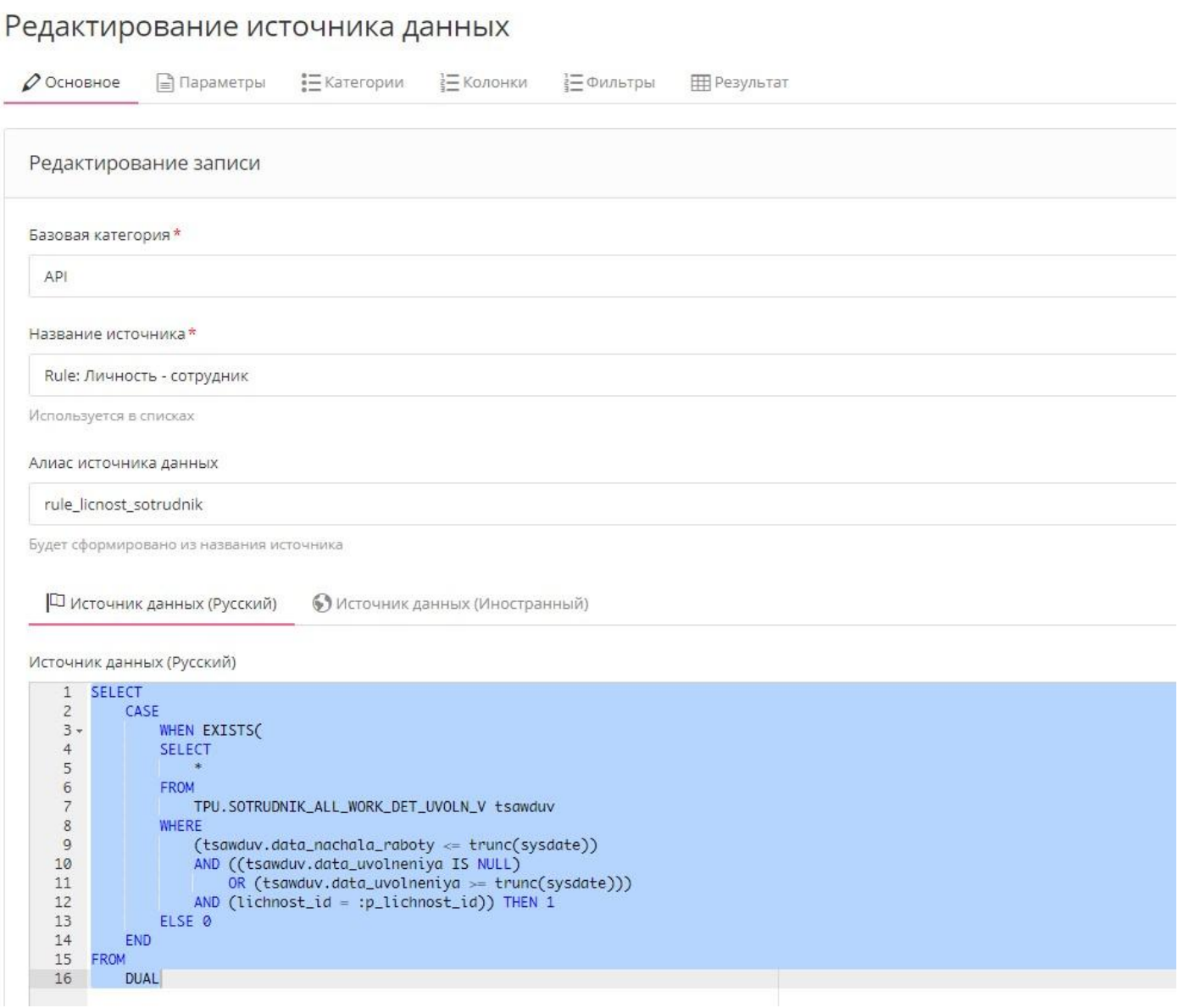
> Динамические источники данных поддерживают мультиязычность, склонение по падежам и вложенную сортировку.

Создание динамического источника

Для добавления динамического источника данных требуется открыть раздел "Утилиты разработчиков" и выбрать меню "Источники данных"



Далее необходимо указать основные параметры динамического источника данных, и указать сам SQL запрос



Пример динамического источника данных

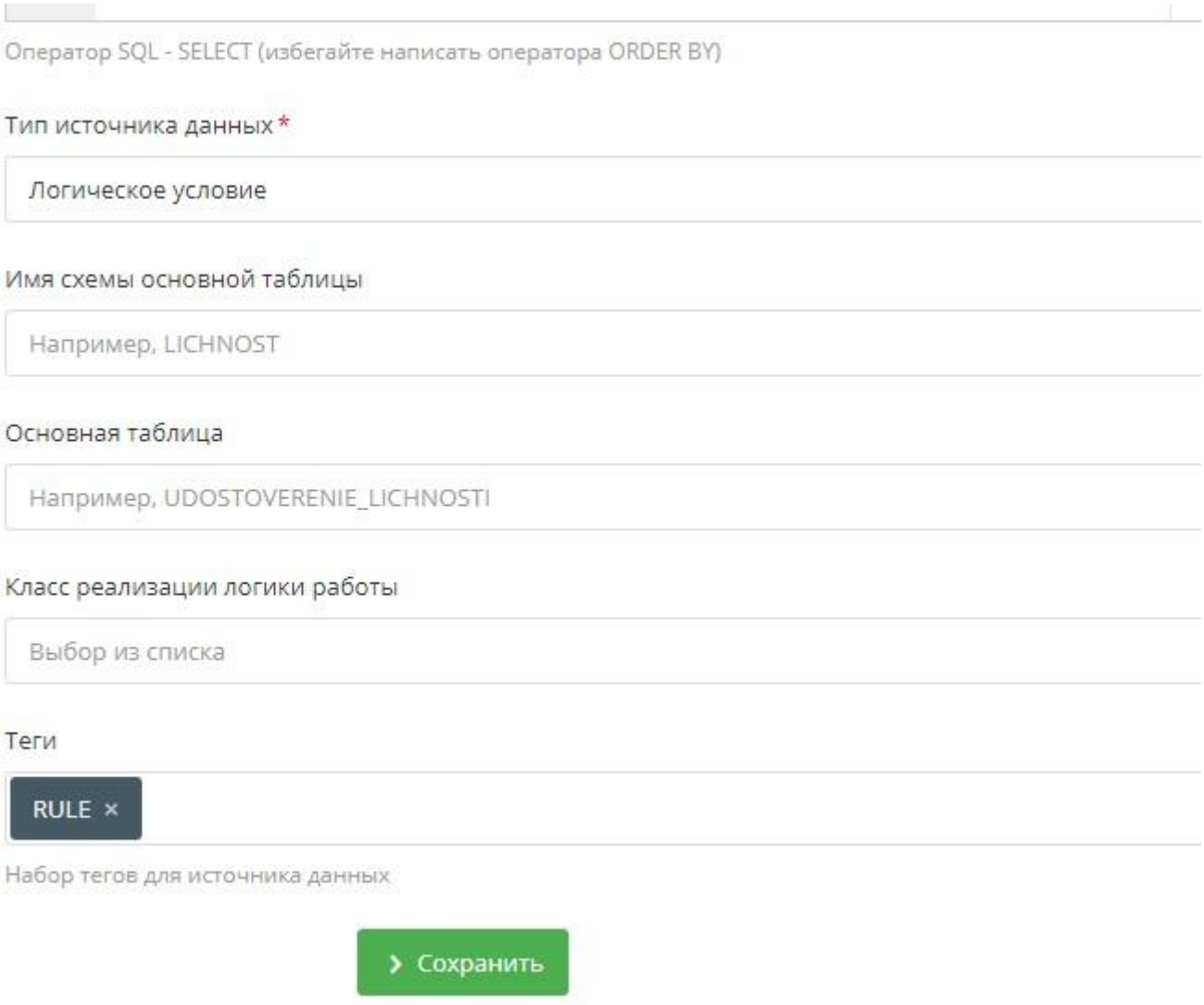
> Все **параметр** используемые в SQL запрос имеют префикс **p_**

```
SELECT
□CASE
□□WHEN EXISTS(
□□SELECT
□□□*
□□FROM
□□□TPU.SOTRUDNIK_ALL_WORK_DET_UVOLN_V tsawduv
□□WHERE
□□□(tsawduv.data_nachala_raboty <= trunc(sysdate))
```

```
□□□AND ((tsawduv.data_uvolneniya IS NULL)
□□□□OR (tsawduv.data_uvolneniya >= trunc(sysdate)))
□□□AND (lichnost_id = :p_lichnost_id)) THEN 1
□□ELSE 0
□END
FROM
□DUAL
```

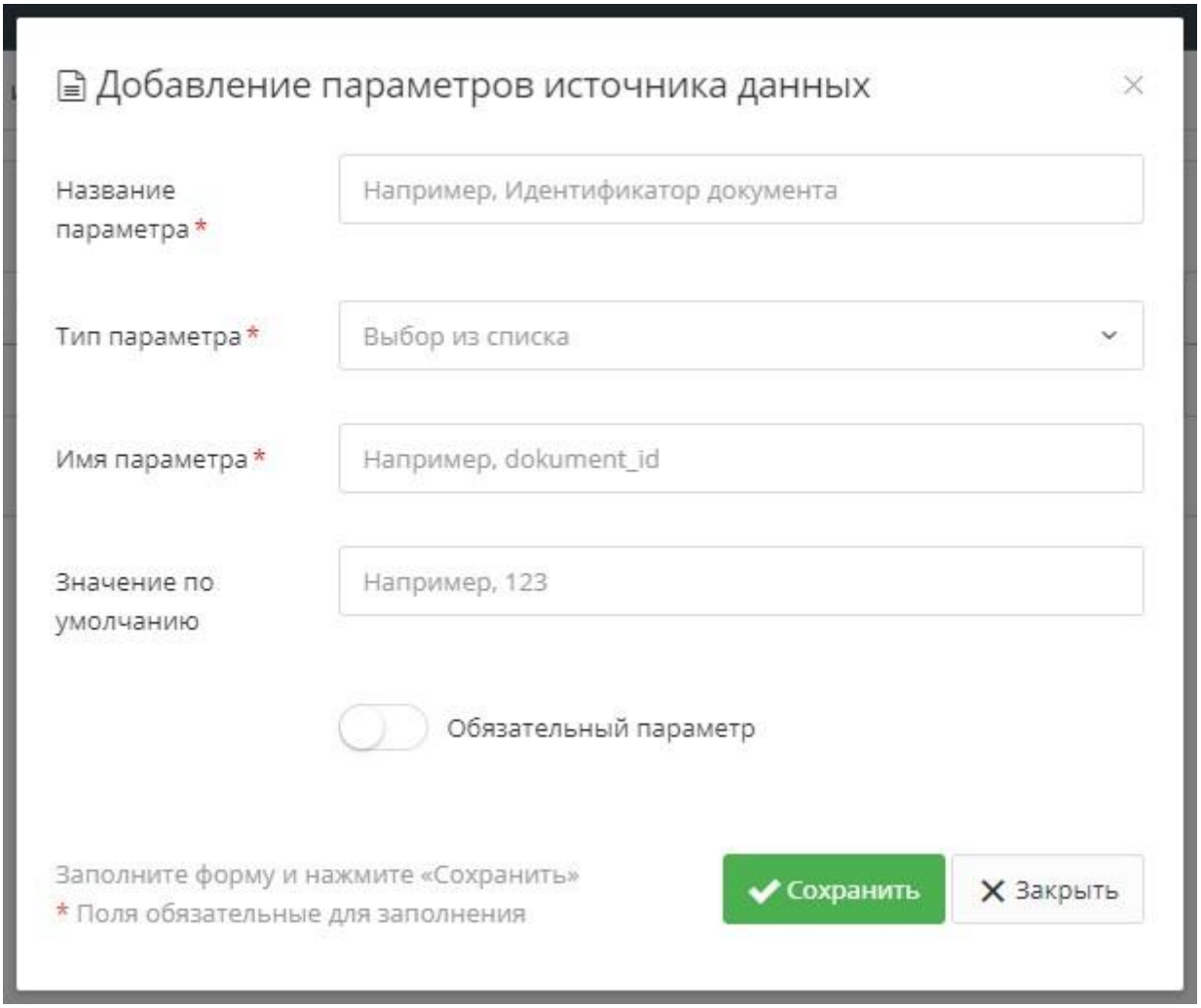
В нижней части формы необходимо указать тип источника данных, для динамических источников данных типа Список - нужно указать имя таблицы и схемы основной таблиц, идентификатор которой объявлен как Первичный ключ.

Можно указать теги, для поиска динамических источников данных по предметной области.



Параметры динамического источника данных

В данном разделе указываются параметры, которые могут быть использованы при формировании SQL запроса



> Название параметра **lichnost_id**, а в SQL запросе нужно указывать **:p_lichnost_id**

Количество параметров в SQL запросе должно совпадать с количеством объявленных параметров, а так же должен совпадать тип данных указанных параметров.

> Пример

```
SELECT
  CASE
    WHEN EXISTS(
    SELECT
      *
    FROM
      TPU.SOTRUDNIK_ALL_WORK_DET_UVOLN_V tsawduv
    WHERE
      (tsawduv.data_nachala_raboty <= trunc(sysdate))
      AND ((tsawduv.data_uvolneniya IS NULL)
        OR (tsawduv.data_uvolneniya >= trunc(sysdate)))
      AND (lichnost_id = :p_lichnost_id)) THEN 1
    ELSE 0
  END
FROM
  DUAL
```

При выполнении с параметром ***lichnost_id*** = ***12345*** будет заменён на следующий запрос

```
SELECT
  CASE
    WHEN EXISTS(
    SELECT
      *
    FROM
      TPU.SOTRUDNIK_ALL_WORK_DET_UVOLN_V tsawduv
    WHERE
      (tsawduv.data_nachala_raboty <= trunc(sysdate))
      AND ((tsawduv.data_uvolneniya IS NULL)
        OR (tsawduv.data_uvolneniya >= trunc(sysdate)))
      AND (lichnost_id = 12345)) THEN 1
    ELSE 0
  END
FROM
  DUAL
```

Динамический источник данных - тип личность

Динамический источник данных должен возвращать только одну колонку и может возвращать более одной строки.

Значение в этой колонке должны ссылать на сущность личность

```
SELECT COALESCE (ruk.lichnost_id, zam.lichnost_id) AS lichnost_id
  FROM obuchenie.vibor_studenta_protokol vp
       LEFT JOIN TPU_SYSTEM.rukovoditel_podrazdeleniya_mv ruk
         ON ruk.podrazdelenie_id = vp.obuch_podrazdel_id
       LEFT JOIN TPU_SYSTEM.zam_rukovod_podrazdeleniya_mv zam
         ON zam.podrazdelenie_id = vp.obuch_podrazdel_id
 WHERE vp.protokol_id = :p_dokument_id
```

> Этот динамический источник данных чаще всего используется в динамическом списке визирующих лиц, поэтому должен иметь как минимум один обязательный параметр **dokument_id**

Классификаторы (Classifier)

В этой книге описан принцип работы со справочниками, получение данных, использование во внешних системах.

Классификаторы (Classifier)

Классификатор — систематизированный перечень наименованных объектов, каждому из которых в соответствие дан уникальный код (**Идентификатор**). Классификация объектов производится согласно правилам распределения заданного множества объектов на подмножества (классификационные группировки) в соответствии с установленными признаками их различия или сходства.

Классификатор имеет следующую структуру:

Атрибут	Назначение	Тип данных	Обязательность
id	Уникальный идентификатор	Целый	Да
short_name	Краткое название	Строковый	Нет
full_name	Полное название	Строковый	Да
rec_status	Статус записи	Целый	Да

Для получения списка классификаторов, нужно вызвать следующий метод API, указав ваш публичный ключ приложения:

> **https://api.tpu.ru/v2/classifier/list**

Для получения значений конкретного справочника, нужно вызывать следующий метод API, указав ваш публичный ключ приложения и ***алиас*** справочника (***алиас*** получили при вызове предыдущего метода.)

> **https://api.tpu.ru/v2/classifier/<alias>**

Основное правило классификаторов:

>

> Значения full_name не могут изменятся у существующих записей, то есть если требуется новая формулировка - то старое значение классификатора переводится в архив и добавляется новое значение с новым уникальным Идентификатором.

Модели данных (EnityModel)

В данной главе описаны базовые понятие что такое модель данных, структура модели и принцип работы с ними через API.

Модель данных может использоваться:

1. во внешних приложениях, как основной источник данных;
2. в системе синхронизации, как источник данных для описания объектов;
3. при формировании выходных документов, как источник данных для шаблонов.

Создание модели данных

Модель данных представляет собой описание объекта предметной области, которого можно представить в виде структуры в формате JSON. Все узлы структуры должны быть названы латинскими буквами, слова без пробелом в нижнем регистре.

Для создания новой **модели данных** перейдите в раздел разработчика в проекте Сервисы ТПУ

> Вы видите только модели данных, которые вы создали (авторы) или с вами поделились другие разработчики.

В правом верхнем углу нажмите кнопку "Добавить" и заполните все основные поля.

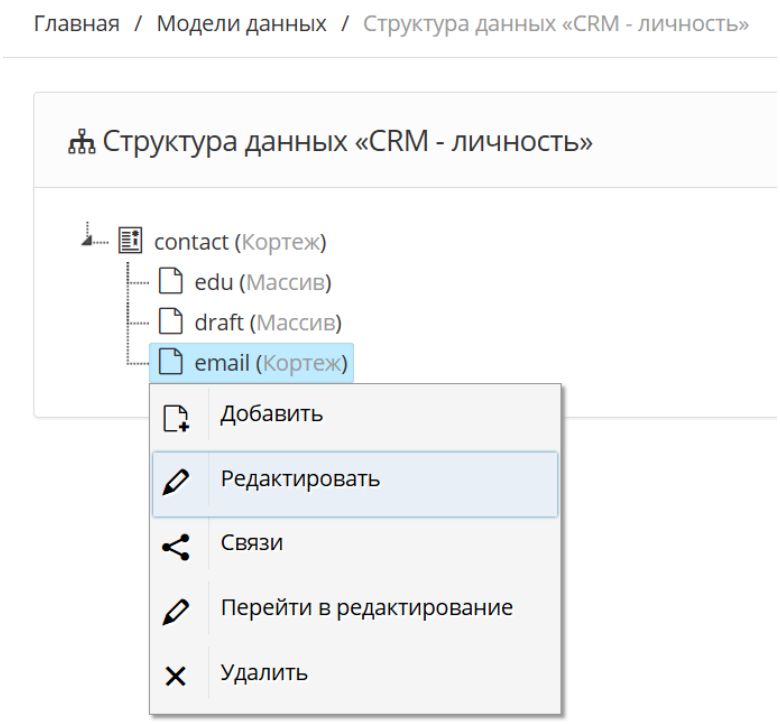
Параметр	Значение
Название	Краткое название модели данных
Алиас	Уникальный алиас модели, будет использовать в API и в синхронизации
Описание	Полное описание модели данных
Вид доступа к модели данных	Уровень доступа 1. Всем 2. Только пользователям (требуется авторизация) 3. Только корпоративным пользователям (действующий студент или сотрудник)
Время жизни кеша в секундах	Время жизни кэша сформированного объекта модели данных в памяти, для ускорения работы. > Если кэширование не требуется - значение в этом поле нужно указать 0.
Приложения модели	Перечень приложений, которые могут обращаться к указанной модели данных. > Каждое приложение должно передавать свой публичный ключ доступа.

Параметр	Значение
Источник для корневого узла	По умолчанию для структуры модели данных создается древовидное представление с корневым элементом (root). Данное значение в дальнейшем можно будет поменять. > У одного объекта модели данных может быть только один корневой элемент.

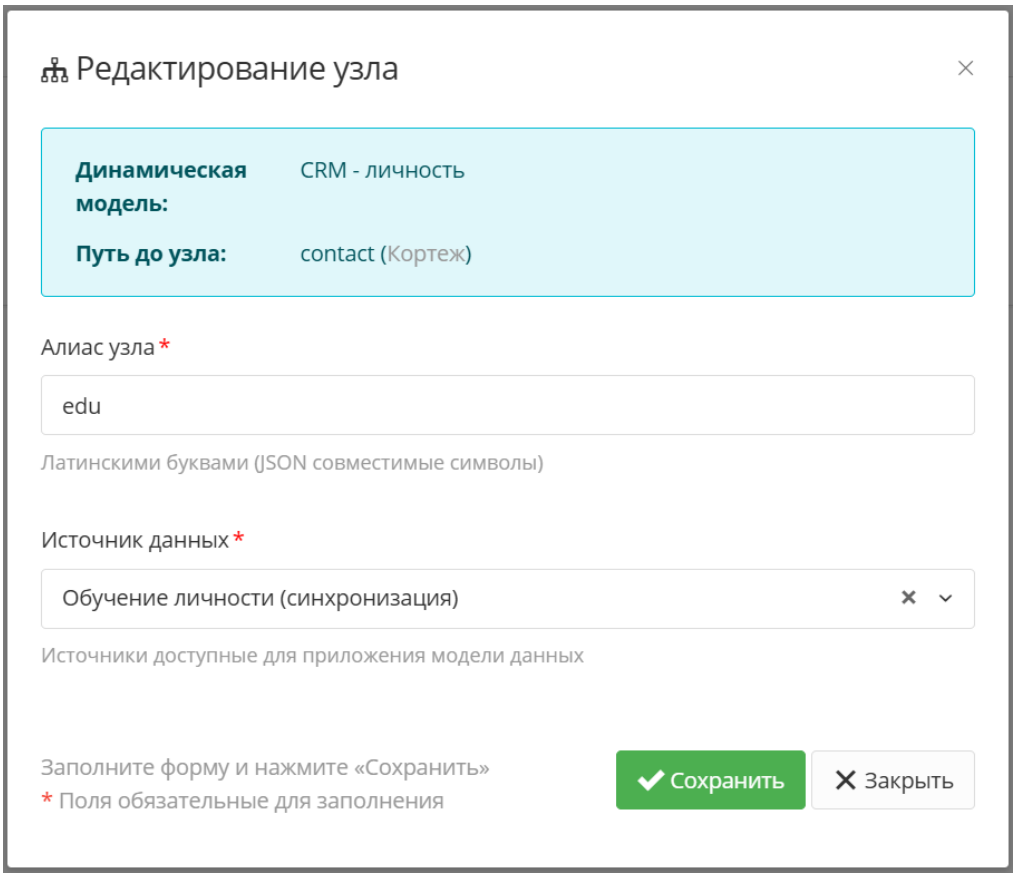
В качестве источников данных для узлов структуры могут быть использованы только 3 типа динамических источников данных.

1. Кортеж
2. Массив
3. Список

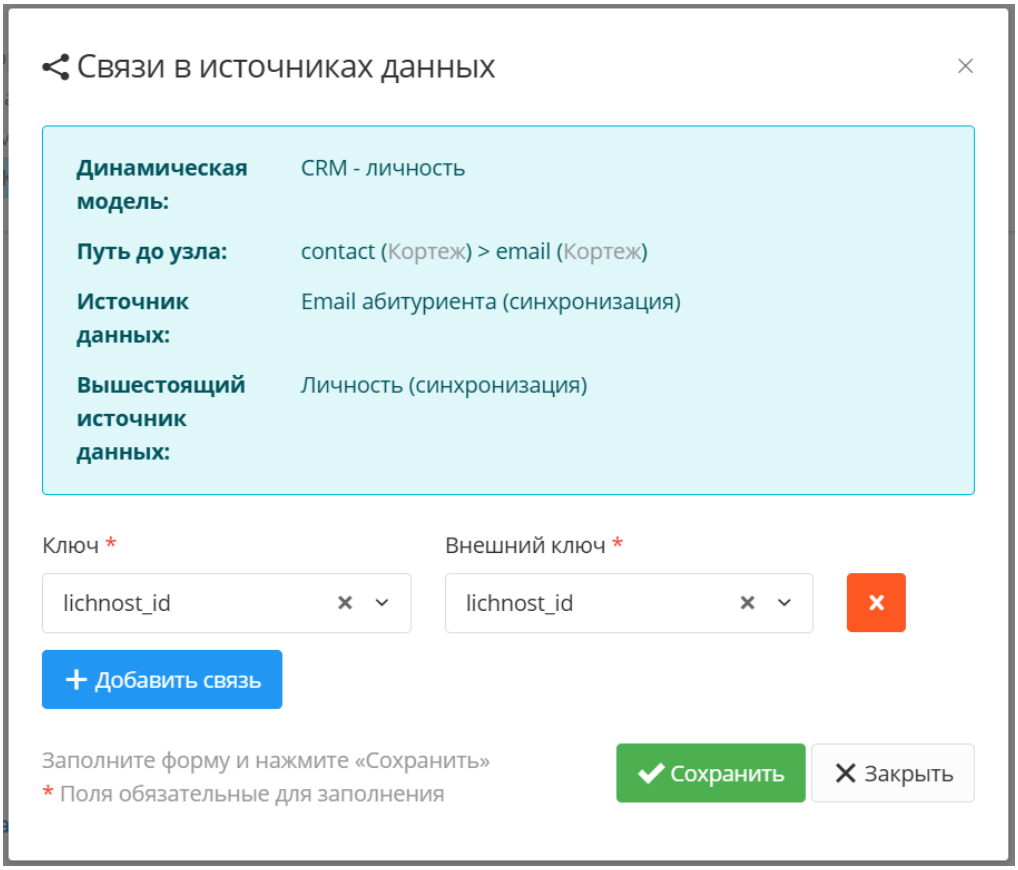
Редактирование описания структуры модели данных осуществляется через контекстное меню в режиме редактирования. Для добавления / редактирование / удаления узлов - требуется воспользоваться соответствующими элементами меню.



При редактировании узла требуется указать алиас (должен быть обязательно совместим со стандартом JSON) и источник данных (должен быть подготовлен заранее)



Для связи дочерних элементов с родительскими требуется указать хотя бы один внешний ключ, пересечение может быть организовано по составному ключу.



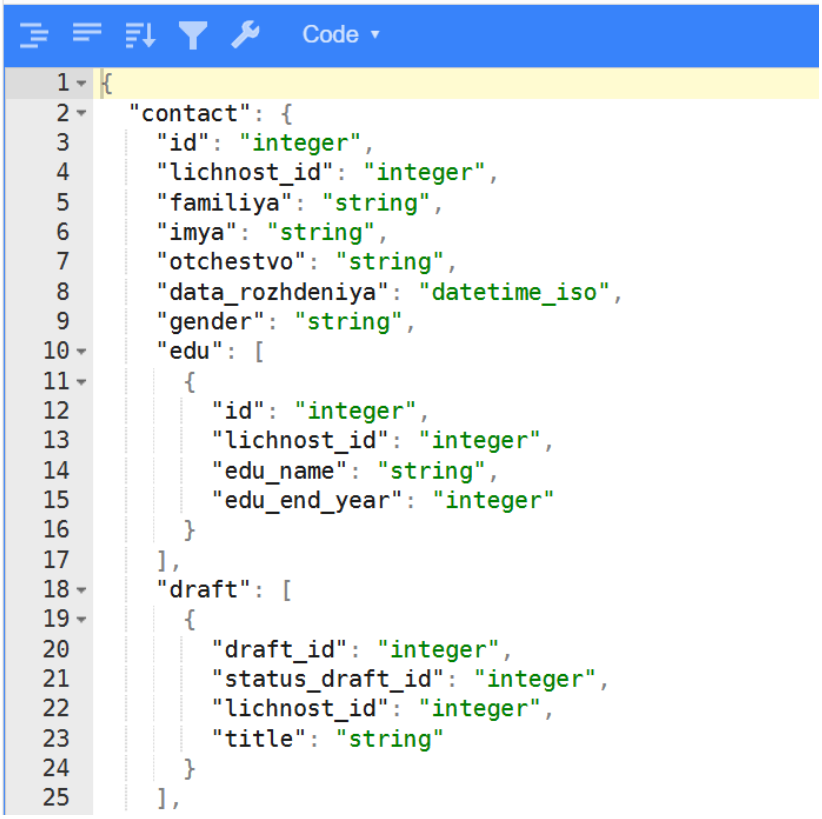
Для оперативной корректировки источника данных в контекстном меню есть ссылка на переход в режим редактирование. Источник данных можно редактировать, если вы его автор или с вами поделились доступом.

Предварительный просмотр

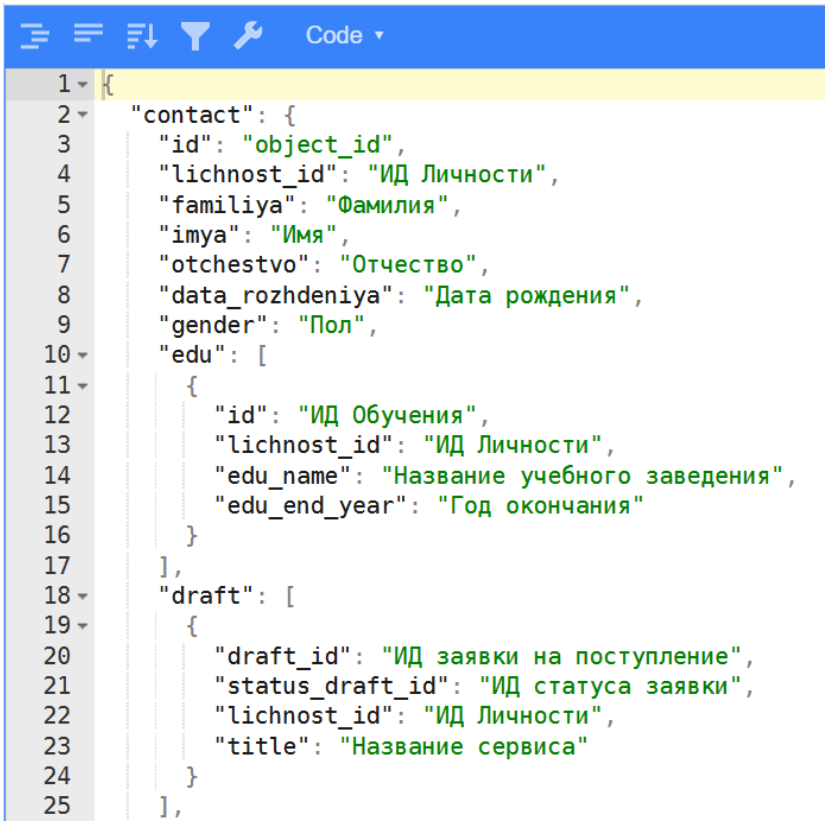
После создания описание модели данных, для проверки работы можно перейти в режим предварительный просмотр.

Предварительный просмотр формирует 3 различных описания.

1. Структура (тут представлены все экспортируемые поля с указанием типа данных)

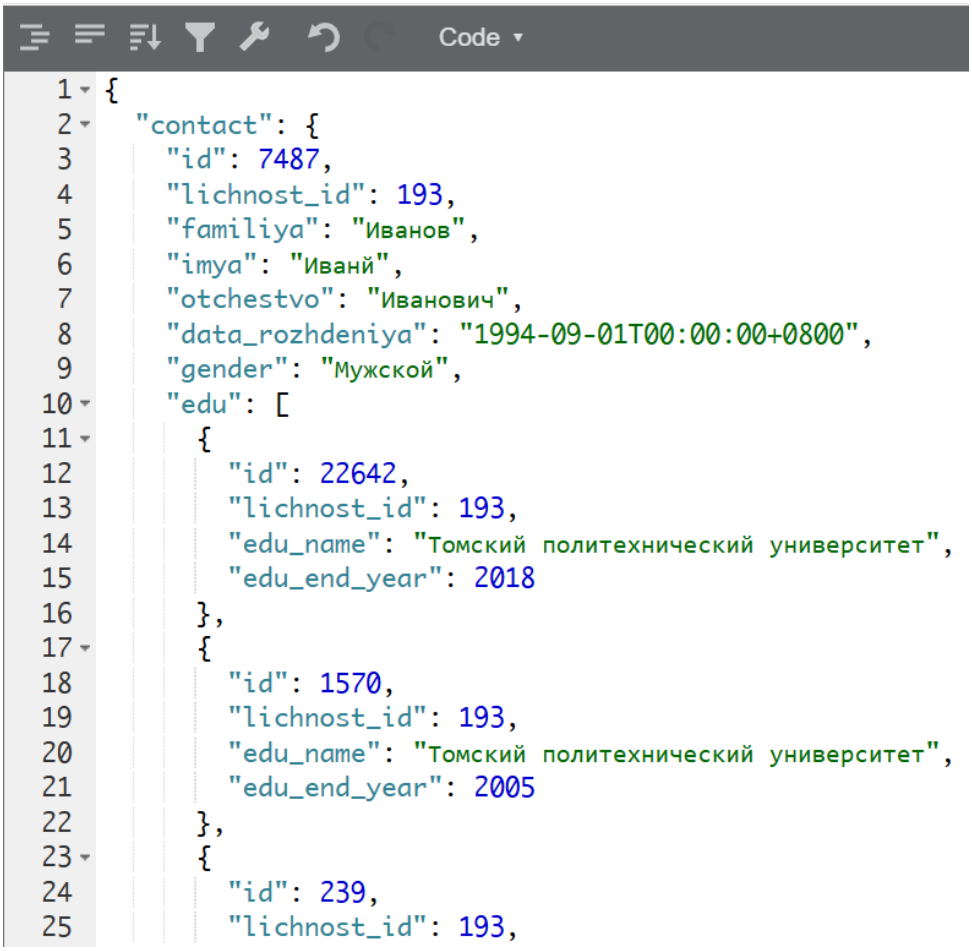


2. Комментарии (в этом режиме все поля содержат описательную часть, комментарий к назначению данного поля)



```
{
  "contact": {
    "id": "object_id",
    "lichnost_id": "ИД Личности",
    "familiya": "Фамилия",
    "imya": "Имя",
    "otchestvo": "Отчество",
    "data_rozhdeniya": "Дата рождения",
    "gender": "Пол",
    "edu": [
      {
        "id": "ИД Обучения",
        "lichnost_id": "ИД Личности",
        "edu_name": "Название учебного заведения",
        "edu_end_year": "Год окончания"
      }
    ],
    "draft": [
      {
        "draft_id": "ИД заявки на поступление",
        "status_draft_id": "ИД статуса заявки",
        "lichnost_id": "ИД Личности",
        "title": "Название сервиса"
      }
    ],
```

3. Пример (Загруженный объект по умолчанию из базы, с реальными значениями)



```
{
  "contact": {
    "id": 7487,
    "lichnost_id": 193,
    "familiya": "Иванов",
    "imya": "Иваń",
    "otchestvo": "Иванович",
    "data_rozhdeniya": "1994-09-01T00:00:00+0800",
    "gender": "Мужской",
    "edu": [
      {
        "id": 22642,
        "lichnost_id": 193,
        "edu_name": "Томский политехнический университет",
        "edu_end_year": 2018
      },
      {
        "id": 1570,
        "lichnost_id": 193,
        "edu_name": "Томский политехнический университет",
        "edu_end_year": 2005
      },
      {
        "id": 239,
        "lichnost_id": 193,
```

После проверки модели данных, её можно использовать через API интеграцию или получать данные через сервер Синхронизации, а так же использовать как источник данных при

формировании документов для СОУД.

по API можно запросить:

1. список объектов
2. один объект по первичному ключу

Через сервер синхронизации:

1. при изменении свойств объект - будет вызван API во внешней системе, куда будет передан один объект или список объектов.

При создании шаблона документа можно указать модель данных, как основной источник данных при формировании документа в формате PDF (шаблон может быть в формате Word или LaTeX).

Работа с моделью данных через API

С моделями данных можно работать через API только на чтение.

Для получения списка доступных моделей данных, необходимо вызвать следующий метод API, указав публичный ключ приложения.

> https:://api.tpu.ru/v2/entity

Для получения структуры/описания модели данных можно использовать следующие методы API.

> https://api.tpu.ru/v2/entity/structure/<id>

Для получения возможного списка параметров, которые можно использовать в модели данных (*данный список может быть пуст*)

> https://api.tpu.ru/v2/entity/parametr/<id>

Для получения возможных фильтров, которые используются при работе с корневым элементом модели данных.

> https:://api.tpu.ru/v2/entity/filter/<id>

> Фильтры доступны только при работе с моделями данных организованных на источниках данных типа массив.

Сортировка выборки данных

> sort=[direction][column]

Для обратной сортировки добавьте минус перед названием столбца

Пример:

```
?sort=title,-created
```

Фильтрация выборки данных:

> filter[type][op]=value

Допустимые сравнения:

Оператор	Сравнение	Описание
eq	equal (=)	Эквивалентно (равно) указанному значению
like	text search	Поиск по тексту
gte	>=	Больше и равно
lte	<=	Меньше и равно
gt	>	Больше
lt	<	Меньше

Пример:

```
?filter[created][eq]=2019-07-16
```

Все параметры можно комбинировать

Модель данных для печати бланков

Правила системных объектов (Rule)

Для получения доступа к системным объектам авторизованный пользователь должен пройти проверку правил. Правила создает разработчик ЕИС ТПУ. Если будет пройдена хотя бы одна проверка - доступ пользователя к системного объекту будет разрешён, в противном случае доступ будет запрещён.

Введение

Для получения доступа к системным объектам авторизованный пользователь должен пройти проверку правил. Правила создает разработчик ЕИС ТПУ. Если будет пройдена хотя бы одна проверка - доступ пользователя к системного объекту будет разрешён, в противном случае доступ будет запрещён.

Динамический список визирующих лиц (SVL)

Описание механизма формирования произвольного списка визирующих лиц для внутреннего документа оборота. Предварительный список визирования может быть использован при создании следующих документов: приказ, служебная записка, договор, проект и другие.

Введение

Список визирующих лиц (СВЛ) - это компонент связывающих правила формирования списка подписантов по определённым правилам, которые можно выразить через оператор where SQL

Синхронизация

Настройка базы данных Oracle для синхронизации

Для подключения синхронизации, требуется создать триггер на основную таблицу, которую нужно отслеживать.

Ниже приведён шаблон PL/SQL кода для создания требуемого триггера.

Параметр	Описание	Пример
schema_name	Название схемы базы данных	*ORGANIZACIYA*
table_name	Название таблицы в схеме базы данных	*ORGANIZACIYA*

```
CREATE OR REPLACE TRIGGER <:schema_name>.<:table_name>_SYNC AFTER DELETE OR INSERT OR UPDATE
ON <:table_name>
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
BEGIN
  IF INSERTING
  THEN
    OTCHET.sync (:NEW.id,
                 '<:schema_name>',
                 '<:table_name>',
                 1);
  ELSIF UPDATING
  THEN
    OTCHET.sync (:OLD.id,
                 '<:schema_name>',
                 '<:table_name>',
                 3);
  ELSIF DELETING
  THEN
    OTCHET.sync (:OLD.id,
                 '<:schema_name>',
```

```
                '<:table_name>',
                2);
  END IF;
EXCEPTION
  WHEN OTHERS
  THEN
    NULL;
END;
/
```

> При выполнении данного запроса в базе данных, замените переменные вашими значениями.

> Если требуется более сложная синхронизация, то можно использовать ваши '***<:schema_name>***' и '***<:table_name>***' в других **процедурах** или **триггерах**, но **primary_key** должен быть первичным ключом именно той таблицы, которую описывает данная **сущность**.

Генерация отчётов

В данной главе описан процесс создания отчётов, от написания динамических источников и до подключения виджета.