РУКОВОДСТВО АДМИНИСТРАТОРА

платформы виртуализации данных АрхиГраф.MDM

Содержание

Основные сведения

АрхиГраф.MDM – платформа виртуализации данных, которая может быть использована и в качестве классической MDM-системы. Общая архитектура платформы показана на рис. 1.1.

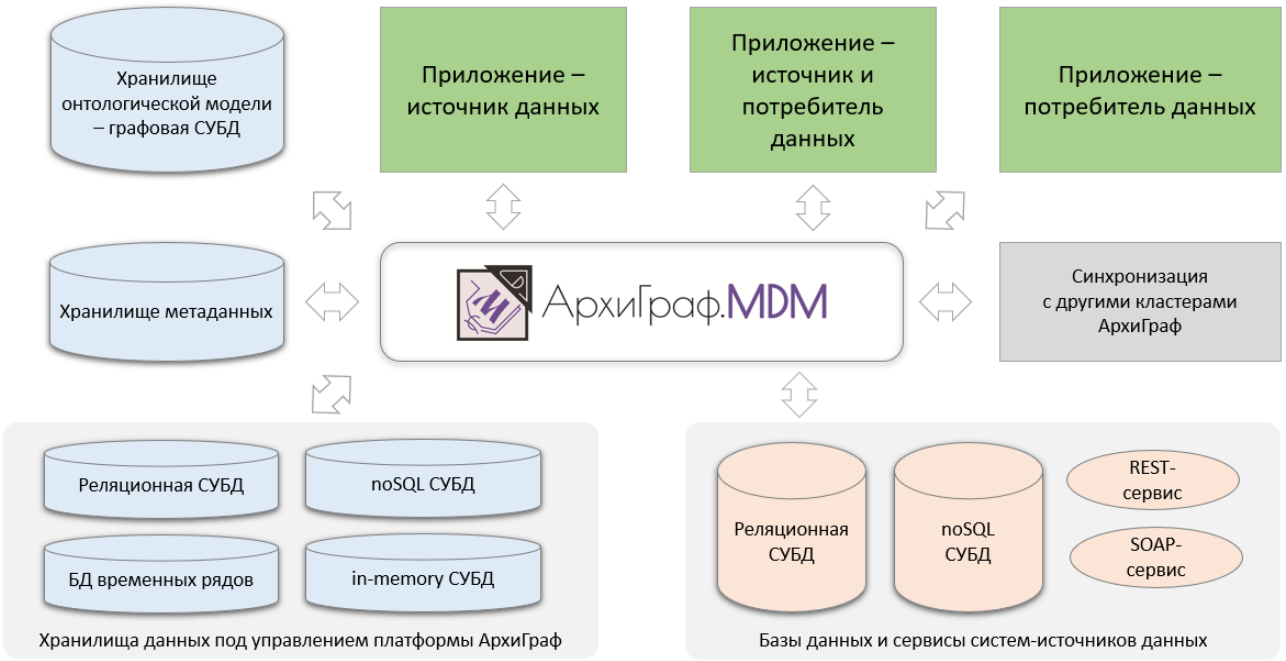

Рис. 1.1. Архитектура АрхиГраф.MDM

Платформа АрхиГраф.MDM является промежуточным слоем (middleware), предоставляющим сервисы для работы с данными, расположенными во множестве различных источников как под управлением платформы, так и вне ее. Сервисы доступны как в синхронном варианте (HTTP-запросы, REST-сервисы), так и в асинхронном (обмен через очереди Kafka, RabbitMQ).

АрхиГраф.MDM использует онтологическую модель для описания структуры обрабатываемых данных. Онтологическая модель размещена в графовой СУБД (продукт класса RDF Triple Store), тогда как сами данные располагаются в различных базах данных, подключенных к платформе.

Создание и редактирование онтологической модели выполняется при помощи редактора АрхиГраф.Мир[1]. Наполнение данными может выполняться вручную при помощи АрхиГраф.Мир или с помощью специально разработанных пользовательских интерфейсов, а также средствами автоматизации обмена информацией, использующими API АрхиГраф.MDM[2].

Задачи администрирования АрхиГраф.MDM состоят в следующем:

– настройка «точек доступа» – отдельных пространств данных, каждое из которых включает онтологическую модель и набор хранилищ информации;

[1] Подробнее о редакторе онтологий АрхиГраф.Мир на сайте: https://trinidata.ru/archigraph_mir.htm
[2] См. описание API АрхиГраф.MDM: https://trinidata.ru/files/ArchiGraphAPI.pdf

– конфигурирование хранилищ данных и управление распределением объектов данных, относящихся к различным классам, между хранилищами;

– создание учетных записей для клиентов MDM и настройке прав доступа на уровне классов и атрибутов онтологической модели;

– настройка подписок на изменения модели данных;

– настройка фоновых заданий, прослушиваемых очередей, констант и лицензий.

В АрхиГраф.MDM предусмотрены два инструмента конфигурирования: визуальный конфигуратор и консольная утилита mdmctl, предназначенная в первую очередь для создания сценариев автоматизированного развертывания.

Для понимания принципов работы с содержимым баз данных под управлением АрхиГраф.MDM желательно познакомиться с концепциями семантического (онтологического) моделирования. В качестве введения в эту проблематику рекомендуем материалы, разработанные компанией ТриниДата:

– Методическое пособие «Введение в онтологическое моделирование» https://trinidata.ru/files/SemanticIntro.pdf,

– Корпоративные автоматизированные системы на основе онтологических моделей: книга рецептов https://trinidata.ru/files/CookBook.pdf,

– Руководство по созданию и поддержке семантической модели https://trinidata.ru/files/SemanticModelDesign.pdf.

Визуальный интерфейс конфигурирования АрхиГраф.MDM

Интерфейс конфигуратора включает в себя рабочее пространство, на главной странице которого находится раздел «Точки доступа». На рисунке 2.1 представлены основные элементы интерфейса, идентичные для всех разделов приложения.

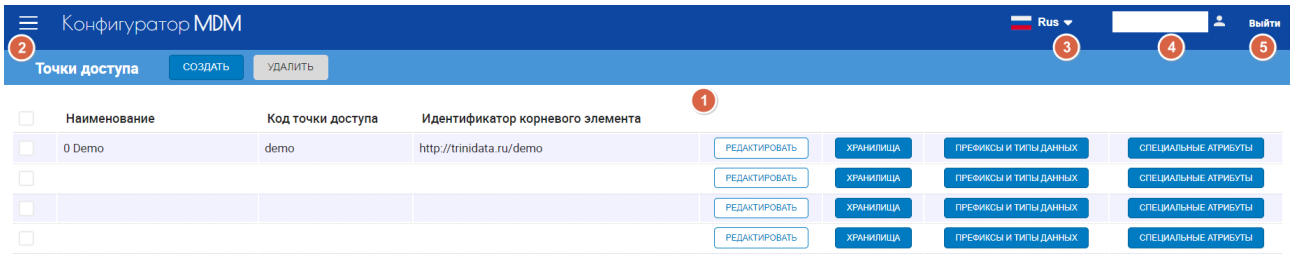

Рис. 2.1. Элементы графического интерфейса Конфигуратора АрхиГраф.MDM

1 – страница открытого раздела;

2 – кнопка выпадающего меню, при нажатии на которую произойдет раскрытие списка навигации по разделам интерфейса Конфигуратора АрхиГраф.MDM;

3 – переключатель языка интерфейса;

4 – имя авторизованного пользователя;

5 – кнопка выхода из системы.

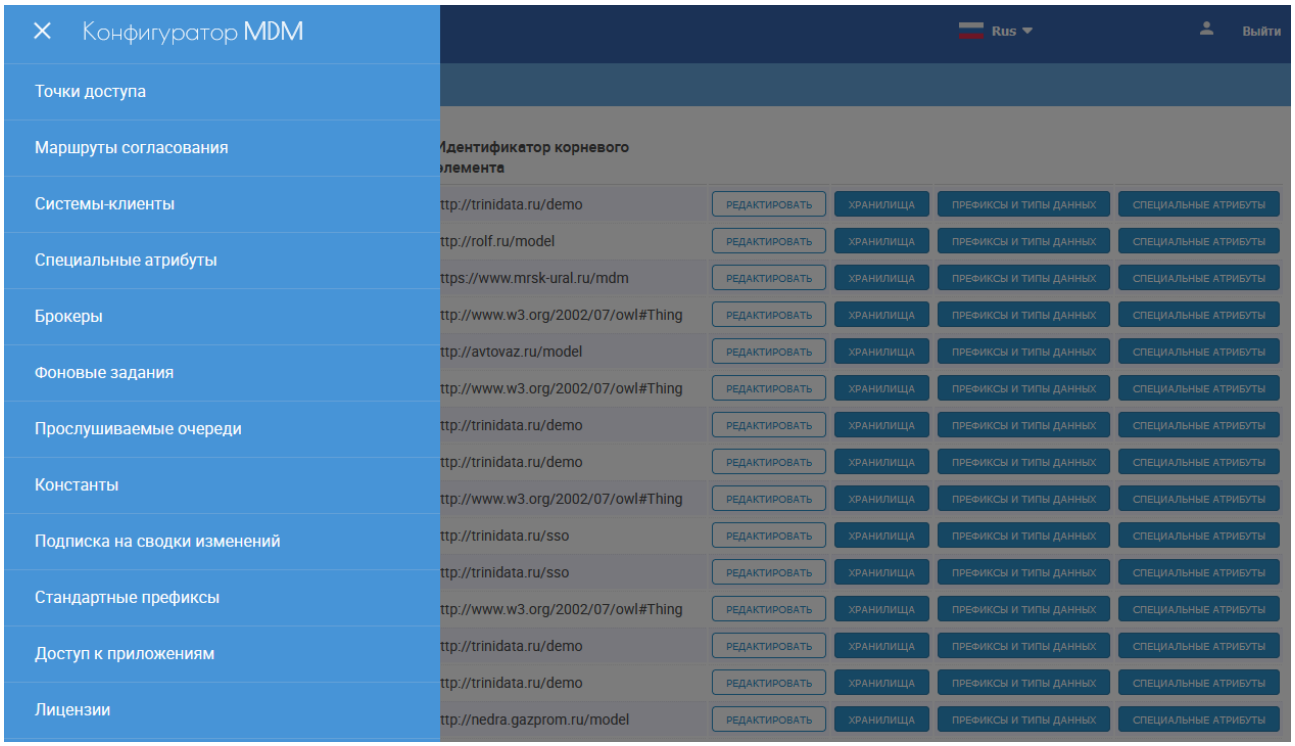

Рис. 2.2. Выпадающее меню разделов интерфейса Конфигуратора АрхиГраф.MDM

Управление точками доступа

Точка доступа (Endpoint) является объектом адресации запросов, которые выполняются к АрхиГраф.MDM с использованием API. В логическом смысле точка доступа – ресурс, содержащий модель данных и данные, относящиеся к одному домену.

Страница управления точками доступа имеет следующий вид:

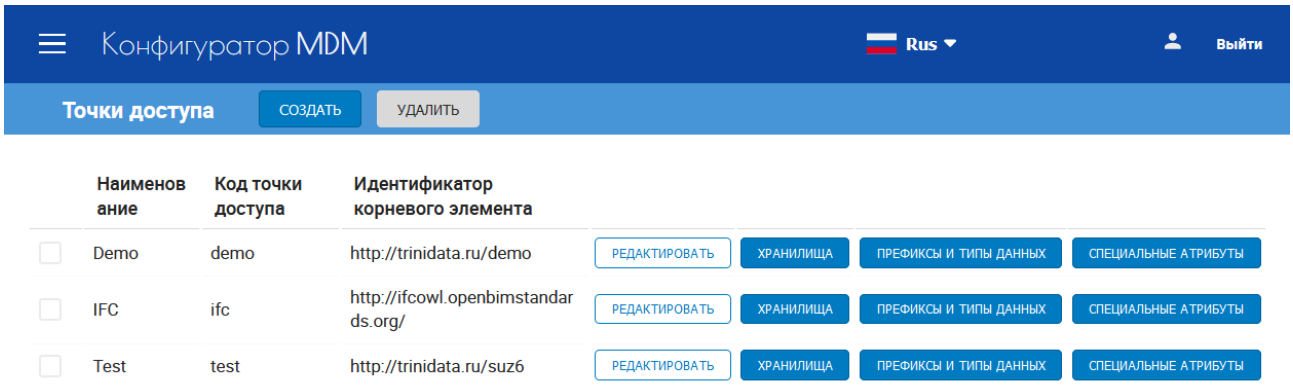

Рис. 3.1. Управление точками доступа

Основные свойства каждой из них:

- читаемое наименование,
- код, используемый при обращении через API,
- идентификатор корневого элемента онтологии (его подклассами должны быть классы верхнего уровня онтологии, рекомендуем использовать http://www.w3.org/2002/07/owl#Thing; указывать идентификатор нужно в полном виде, не сокращая префикс).

Точки доступа ► Создать/Редактировать

Отметив одну или несколько точек доступа с помощью установки флагов в чекбоксах и нажав кнопку «удалить», можно удалить точки доступа. Нажатие кнопки «Создать» открывает диалоговое окно создания новой точки доступа, в котором необходимо ввести значения основных свойств точки доступа.

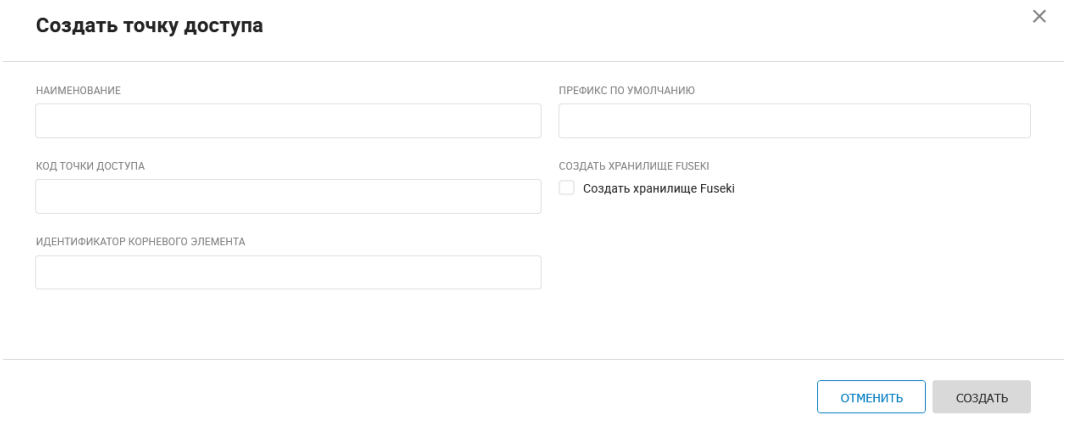

Рис. 3.2. Диалоговое окно создания новой точки доступа

Значения основных свойств можно изменить для любой существующей точки доступа, нажав кнопку «Редактировать».

Точки доступа ► Хранилища

Нажатие на кнопку «Хранилища» открывает страницу настройки хранилища для каждой точки доступа. Принципы их настройки описаны в разделе «Настройка хранилищ».

Точки доступа ► Префиксы и типы данных

Нажатие на кнопку «Префиксы и типы данных» открывает страницу настройки набора префиксов, которые для этой точки доступа будет поддерживать редактор АрхиГраф.Мир. Дополнительно на данной странице настраивается префикс по умолчанию, который поддерживает MDM и набор типов данных, специфических для этой точки доступа.

Область работы с префиксами и типами данных выглядит так:

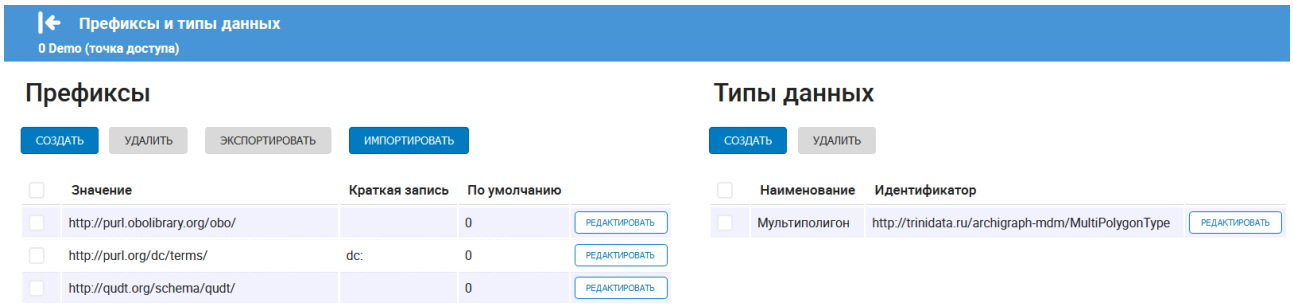

Рис. 3.3. Список префиксов и типов данных, определенных для точки доступа

Кнопки «Создать», «Редактировать» и «Импортировать» активны по умолчанию. Чтобы активировать кнопки «Удалить» и «Экспортировать» необходимо выбрать одну или несколько точек доступа (или типов данных) с помощью установки флагов в чекбоксах. Импорт и экспорт данных осуществляется в формате json.

Точки доступа ► Префиксы и типы данных ► Префиксы ► Создать/Редактировать

Префикс, для которого в списке установлено значение равное 1, является префиксом по умолчанию для данной точки доступа. Это означает, что MDM будет сокращать этот префикс при работе через API. Например, если префикс по умолчанию – http://trinidata.ru/demo/, то идентификаторы элементов онтологии http://trinidata.ru/demo/Персона и Персона будут эквивалентны при обращении к API MDM и при работе в редакторе АрхиГраф.Мир.

Остальные префиксы имеют значение только для АрхиГраф.Мир. Редактор онтологии предлагает выбрать один из этих префиксов при создании любого элемента онтологии.

Дополнительно префиксы можно экспортировать, импортировать и удалять.

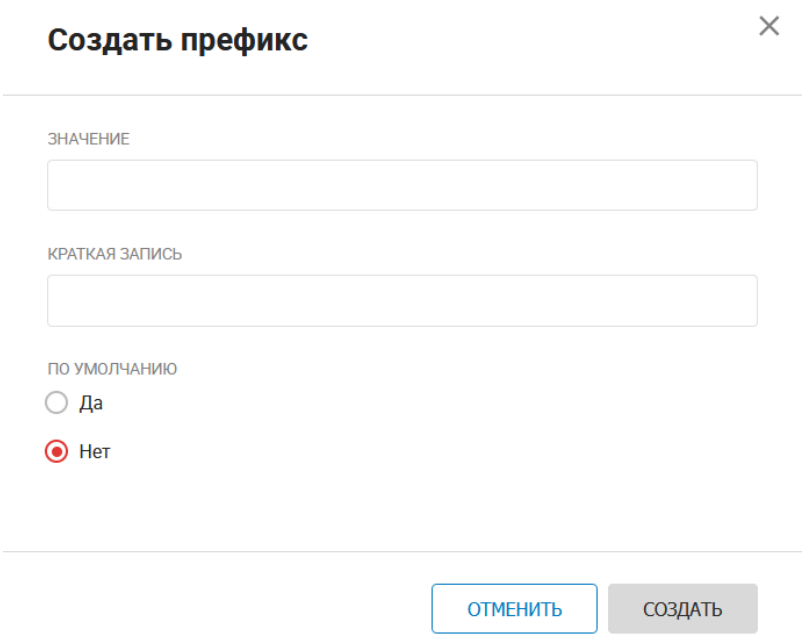

Рис. 3.4. Диалоговое окно создания префикса

Точки доступа ► Префиксы и типы данных ► Типы данных

Настройка типов данных заключается в указании наименования и идентификатора для каждого типа. Например, если указать наименование «Any URI» и идентификатор xsd:anyURI, то после этого можно будет присваивать литеральным атрибутам онтологии такой тип данных.

После создания или редактирования объектов данных разделов в интерфейсе Конфигуратора АрхиГраф.MDM появится всплывающая подсказка с текстом «Для применения изменений необходимо обновить кэш». При нажатии на нее произойдет обновление кэша.

Точки доступа ► Специальные атрибуты

Нажатие на кнопку «Специальные атрибуты» в разделе «Точки доступа» открывает страницу, на которой можно управлять набором атрибутов, определенных вне онтологии. Эта возможность полезна, например, для создания атрибутов, значениями которых могут обладать классы или свойства. Вид диалогового окна создания специального атрибута показан на следующем рисунке.

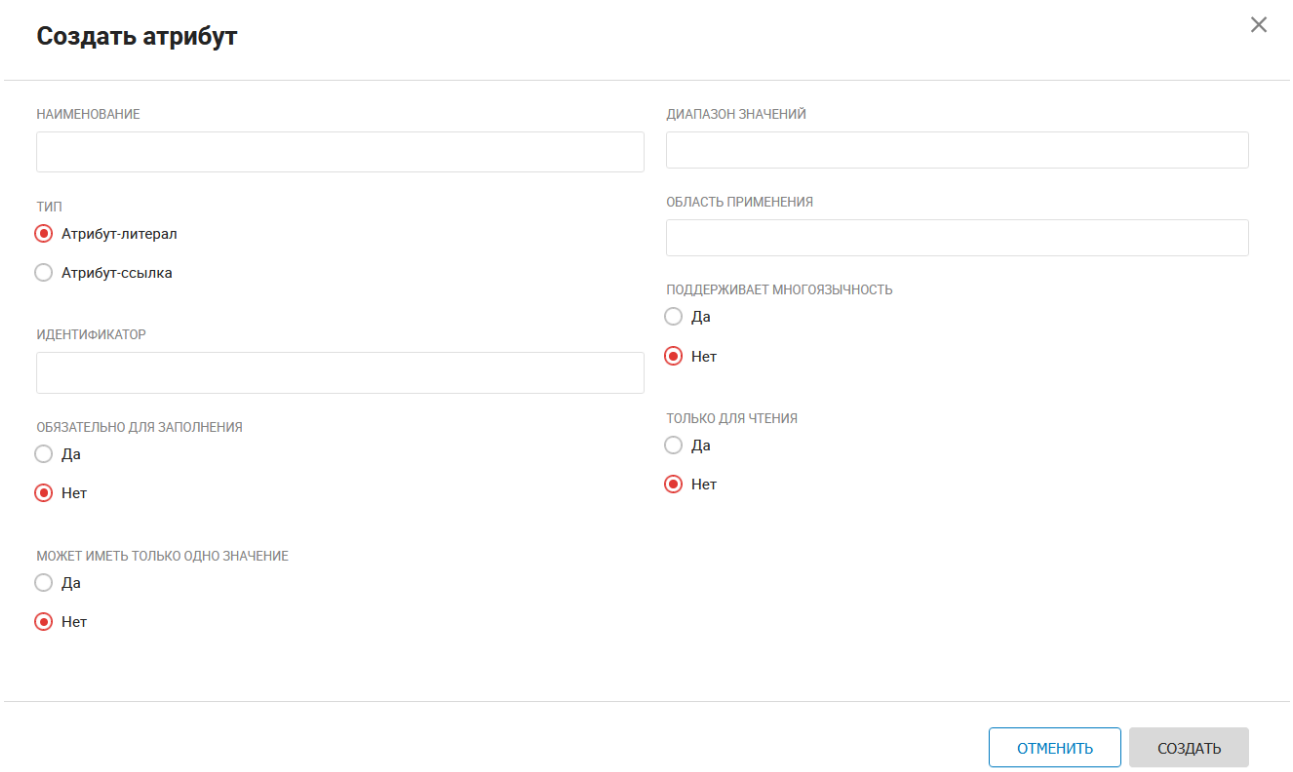

Рис. 3.5. Диалоговое окно создания специального атрибута

Поля свойств атрибута имеют следующий смысл:

- «Наименование» – читаемое название атрибута
- «Тип» – атрибут-литерал (owl:DatatypeProperty) или атрибут-ссылка (owl:ObjectProperty).
- «Идентификатор» – полный URI, идентификатор создаваемого атрибута.
- «Область применения» – класс или тип сущности, к которому применим атрибут (например, выберите owl:Class, чтобы создать атрибут, применимый ко всем классам онтологии).
- «Диапазон значений» – для атрибутов-ссылок это один или несколько классов онтологии, из числа объектов которых выбираются значения для данного свойства. Для атрибутов-литералов это один из доступных типов данных xsd.
- «Поддерживает многоязычность», «Обязательно для заполнения», «Может иметь только одно значение» (для каждого объекта), «Только для чтения» – дополнительные свойства атрибута.

Настройка хранилищ

Хранилище – это таблица или коллекция базы данных, в которой размещается информация об объектах определенного класса(-ов) онтологии. Чтение и запись в хранилища выполняет MDM, они не должны быть доступны напрямую для других программных компонентов.

Хранилища настраиваются отдельно для каждой точки доступа, переход к их настройке происходит нажатием кнопки «Хранилища» в разделе «Точки доступа». Список хранилищ точки доступа имеет такой вид:

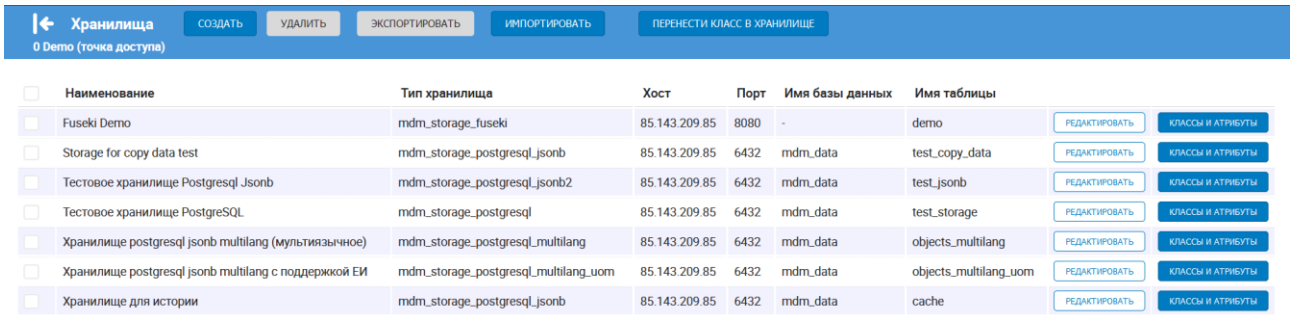

Рис. 4.1. Список хранилищ для точки доступа

В списке отображаются основные свойства каждого хранилища, а также кнопки «Редактировать» и «Классы и атрибуты».

Точки доступа ► Хранилища ► Создать/Редактировать

Свойства хранилища:

- «Тип хранилища» – выбирается один из типов, поддерживаемых MDM. Фактически это выбор внутреннего «драйвера» для подключения к базе данных определенного типа.
- «Наименование» – читаемое название для отображения в списке
- «Хост», «Порт», «Логин», «Пароль», «Имя базы данных», «Имя таблицы» – реквизиты подключения к СУБД.
- «Путь к сервису чтения/записи» – специфичные реквизиты для случая, когда хранилище является точкой доступа SPARQL. Значения этих полей обычно имеют вид «/fuseki/[dataset]/query» и «/fuseki/[dataset]/update» (для Apache Fuseki). В этом случае поле «Имя базы данных» надо оставить пустым, а в поле «Имя таблицы» указать имя датасета.
- «Приоритет» – число, определяющее порядок, в котором MDM ищет объект по идентификатору в хранилищах в случае, если класс объекта не известен. Хранилища с наибольшим значением приоритета просматриваются первыми.
- «Хранить историю изменения» - флаг, позволяющий отключить запись истории изменений, если она занимает много памяти, например для хранилищ временных рядов.
- «Допускать применение правил логического вывода» - флаг, позволяющий отключить применение правил логического вывода и ускорить работу хранилища.

- «Аналитическое» - флаг, устанавливающийся для аналитических хранилищ, для которых доступны только запросы поиска по классу (т. е. запросы поиска объекта по идентификатору не доступны).
- «Дополнительные параметры (json)» - значение в формате json, которое позволяет ввести для некоторых типов хранилищ (например, MongoBD) дополнительные настройки, например задать тип сортировки.
- «Кэшировать названия объектов» – флаг, определяющий, что в данном хранилище для ссылок на другие объекты сохраняются читаемые названия этих объектов, а не только идентификаторы. Кэширование названий существенно ускоряет извлечение данных из хранилищ. Для хранилищ PostgreSQL хорошей практикой является простановка значения «Да».
- «Не переименовывать кэшированные наименования значений атрибутов-ссылок» - флаг, в котором по умолчанию выставляется «Нет». Если для параметра будет проставлен флаг «Да», то для хранилищ, у которых в предыдущем параметре «Кэшировать названия объектов» установлено «Да», будет создана фоновая задача на изменение читаемого наименования объектов, являющихся значениями атрибутов-ссылок.

Диалоговое окно создания/редактирования хранилища выглядит так:

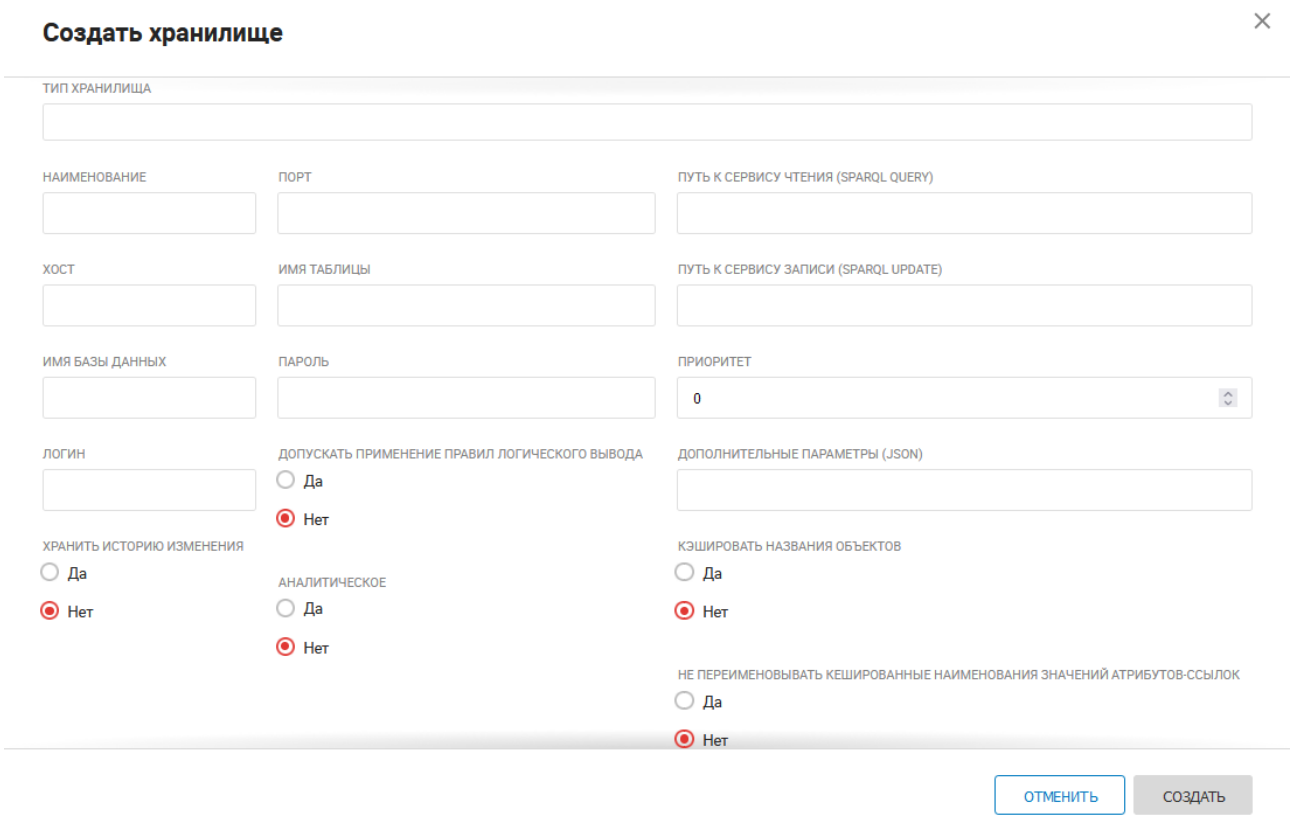

Рис. 4.2. Окно создания/редактирования хранилища

Точки доступа ► Хранилища ► Групповое редактирование

Чтобы одновременно отредактировать несколько хранилищ (задать им идентичные значения свойств) необходимо выбрать их с помощью установки флагов в чекбоксах и нажать кнопку «Групповое редактирование». Далее откроется диалоговое окно, в котором можно установить флаг «Сбросить значение» или ввести новое значение для любого свойства. После внесения необходимых изменений для их сохранения необходимо нажать на кнопку «Редактировать» в нижней части диалогового окна.

Точки доступа ► Хранилища ► Классы и атрибуты

После того, как основные свойства хранилища настроены, нужно указать, объекты каких классов будут в нем храниться, и как свойства объектов будут спроецированы в структуру данных хранилища. Для этого предназначена вкладка «Классы и атрибуты», которая имеет следующий вид:

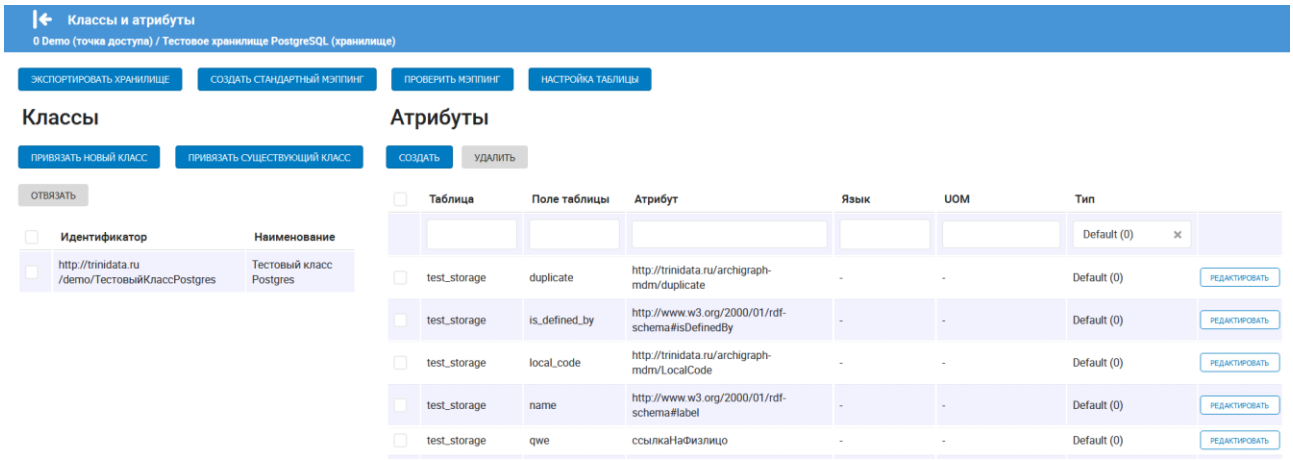

Рис. 4.3. Вкладка «Классы и атрибуты» в свойствах хранилища

Если датасет уже создан, то в рабочем пространстве раздела будут выведены кнопки «Проверить мэппинг» и «Настройка таблицы». Если же датасет для данного хранилища пока не существует, то вместо описанных двух кнопок в интерфейсе появится опция «Создать датасет».

Точки доступа ► Хранилища ► Классы и атрибуты ► Привязать новый класс

Кнопка «Привязать новый класс» позволяет создать новый класс, не переходя в интерфейс АрхиГраф.Мир, и сразу привязать его. К хранилищу можно привязать любое количество классов, но рекомендуется делать это только в случае, если объекты данных классов имеют похожие наборы свойств. В этом случае можно создать в хранилище индексы по столбцам/элементам коллекций, которые хранят значения свойств, по значениям которых наиболее часто будут выбираться объекты из хранилища.

Точки доступа ► Хранилища ► Классы и атрибуты ► Привязать существующий класс

Кнопка «Привязать существующий класс» позволяет выбрать класс онтологии и привязать его к данному хранилищу.

Точки доступа ► Хранилища ► Классы и атрибуты ► Отвязать

Чтобы отвязать класс от хранилища необходимо выбрать один или несколько классов с помощью установки флагов в чекбоксах и нажать кнопку «Отвязать».

Точки доступа ► Хранилища ► Классы и атрибуты ► Экспортировать хранилище

Кнопка «Экспортировать хранилище» позволяет произвести экспорт хранилища в формате json.

Точки доступа ► Хранилища ► Классы и атрибуты ► Создать стандартный мэппинг

В разделе «Атрибуты» выполняется настройка соответствия свойств онтологической модели столбцам/элементам коллекции хранилища. Если область таблицы атрибутов пуста, то можно нажать кнопку «Создать стандартный мэппинг» и задать в настройках название таблицы. Для jsonb хранилищ можно создать минимальный мэппинг. В него входит uri и data. В стандартный мэппинг входят все стандартные атрибуты объектов.

Если для объектов в АрхиГраф.Мир будет использоваться многозначность их типа (объекты будут принадлежать нескольким классам одновременно), то для «Тип столбцов Type/Type_label» необходимо выбрать значение «Массив», в ином случае – «Строка».

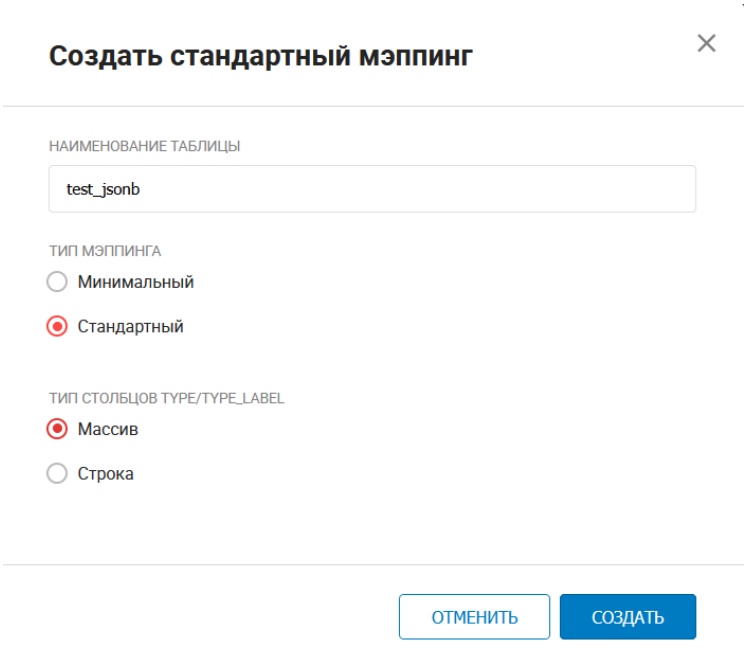

Рис. 4.4. Диалоговое окно создания стандартного и минимального мэппинга

Точки доступа ► Хранилища ► Классы и атрибуты ► Проверить мэппинг

Опция «Проверить мэппинг» используется для проверки соответствия столбцов в таблице хранилища согласно мэппингу. При отсутствии столбцов система выведет их перечень в окне «Результат проверки».

Точки доступа ► Хранилища ► Классы и атрибуты ► Настройка таблицы

Раздел «Настройка таблицы» позволяет создавать или удалять столбцы и индексы у таблицы. Если таблица для хранилища отсутствует, то произойдет автоматическое создание пустой таблицы без столбцов.

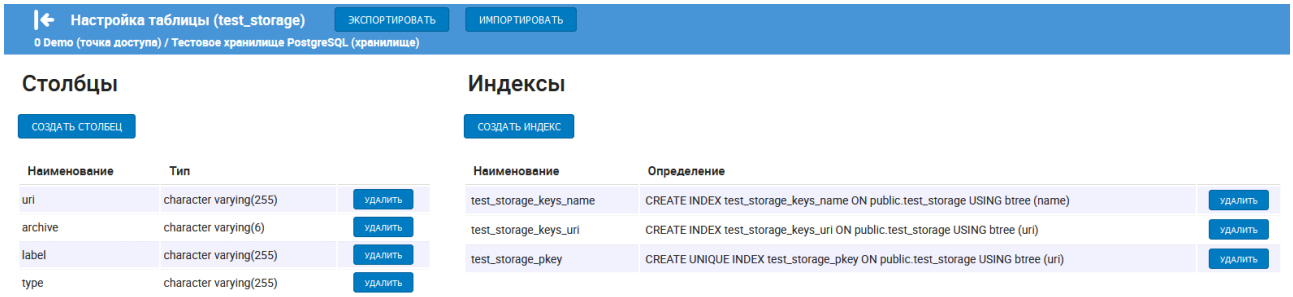

Рис. 4.5. Вкладка «Настройка таблицы»

Точки доступа ► Хранилища ► Классы и атрибуты ► Настройка таблицы ► Создать столбец

В разделе «Настройка таблицы» выполняется создание столбцов.

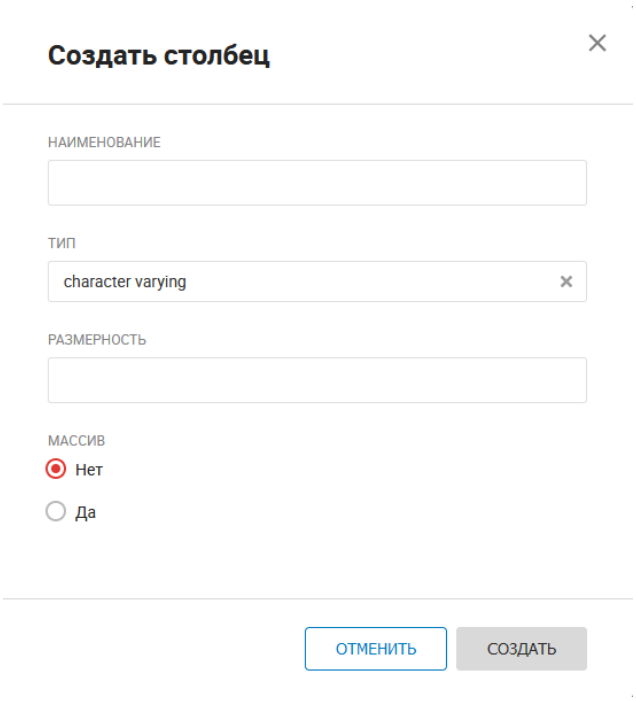

Рис. 4.6. Диалоговое окно создания столбца

Свойства столбца:

- «Наименование» – читаемое название для отображения в списке.
- «Тип» – тип данных, хранящихся в столбце.
- «Размерность» - длина строки, для символьного типа данных. Является необязательным для заполнения свойством.
- «Массив» - свойство для хранения в базе данных многозначных атрибутов. Для всех типов данных из свойства «Тип» может быть выбран и «Да» и «Нет».

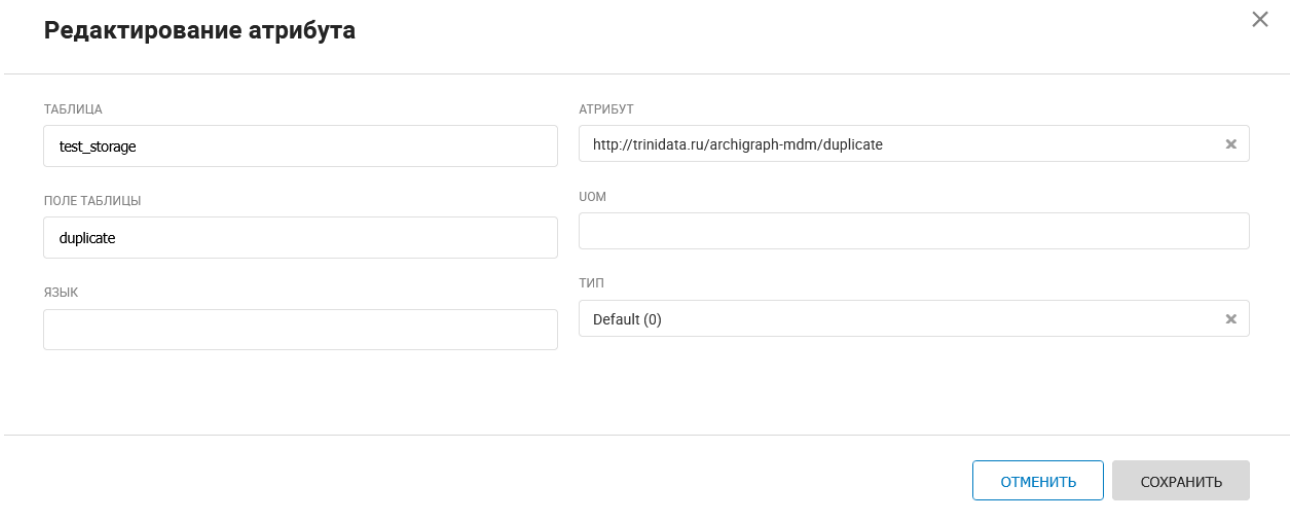

Рис. 4.7. Окно редактирования способа хранения атрибута онтологии в хранилище

Для хранилищ MongoDB и различных видов точек доступа SPARQL настраивать способ хранения атрибутов не требуется. Для хранилищ типа postgresql_jsonb и postgresql всегда должен быть создан следующий минимальный набор столбцов в таблице и правил соответствия:

Поле таблицы	Тип поля	Атрибут онтологии
uri	character varying(255) NOT NULL (первичный ключ)	uri
data	jsonb (этот столбец нужен только для хранилища postgresql_jsonb)	data
type	character varying(255)[]	http://www.w3.org/1999/02/22-rdf-syntax-ns#type
archive	character varying(8)	http://trinidata.ru/archigraph-mdm/archive
name	character varying(512)	http://www.w3.org/2000/01/rdf-schema#label
local_code	character varying(255)	http://trinidata.ru/archigraph-mdm/LocalCode
source_system	character varying(255)	http://trinidata.ru/archigraph-mdm/SourceSystem
is_defined_by	character varying(255)[]	http://www.w3.org/2000/01/rdf-schema#isDefinedBy
duplicate	character varying(255)[]	http://trinidata.ru/archigraph-mdm/duplicate

-

Кроме этих столбцов, могут быть созданы и столбцы для хранения любых других свойств объектов. Для хранилища типа postgresql создание описанных выше столбцов обязательно, для хранилища типа postgresql_jsonb – опционально: для него имеет смысл создавать столбцы для тех свойств, по значениям которых в Postgres нужно построить индексы для ускорения выборки данных. Значения всех свойств в любом случае будут храниться в столбце data.

Точки доступа ► Хранилища ► Классы и атрибуты ► Настройка таблицы ► Создать индекс

При нажатии на кнопку «Создать индекс» появится диалоговое окно, в котором необходимо задать тип индекса и указать столбец, к которому он будет применим.

Точки доступа ► Хранилища ► Классы и атрибуты ► Атрибуты ► Создать/Редактировать

Элементы настройки при создании и редактировании мэппинга для атрибутов схожи. Диалоговое окно редактирования выглядит так, как показано ниже на рис. 4.7.

Для хранилищ реляционных СУБД в поле «Таблица» вводится имя таблицы в СУБД хранилища, «Поле таблицы» – имя поля таблицы БД, в которое происходит сохранение значений атрибута, в поле «Атрибут» выбирается атрибут онтологической модели, значение которого сохраняется в указанное поле.

Для хранилищ документ-ориентированных и графовых СУБД создание правил соответствия свойств модели элементам коллекции хранилища, как правило, не требуется.

В хранилищах типа postgresql_jsonb хранение всех свойств объекта происходит в JSON-коллекции, помещаемой в поле data таблицы БД, которое имеет тип jsonb. Вместе с тем, в целях повышения эффективности индексирования значения отдельных свойств могут быть вынесены в отдельные столбцы СУБД. При этом по умолчанию те же значения продолжают сохраняться и в поле data типа jsonb.

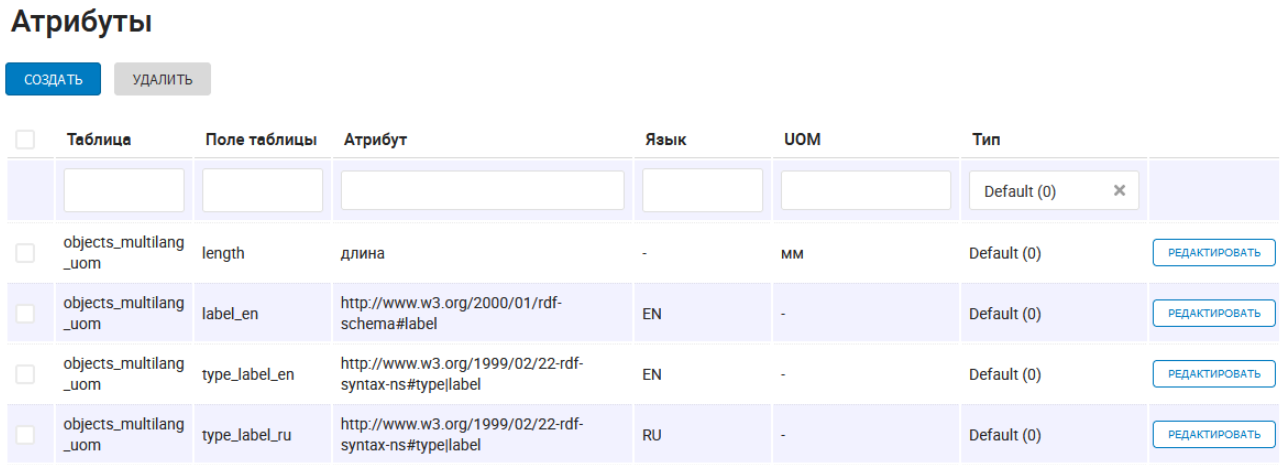

Рис. 4.8. Пример настройки атрибута для мультиязычных данных

Если в значении атрибута необходимо хранить мультиязычные данные, то для атрибута нужно создать несколько экземпляров настроек с заполнением свойства «Язык». Например, в примере на рис. 4.8. для атрибута http://www.w3.org/1999/02/22-rdf-syntax-ns#type|label

указано два языка: «RU» и «EN». В таком случае свойство «Поле таблицы» имеет разные идентификаторы.

Аналогичная настройка требуется, если в столбцах базы данных хранятся значения одного атрибута с различными единицами измерения. В таком случае аналогично примеру, описанному выше, необходимо создать несколько экземпляров настроек с указанием идентичного атрибута, а поле «UOM» заполнить идентификаторами требуемых единиц измерения.

При настройке поля «Тип» необходимо выбрать один из типов мэппинга: Default (0) – стандартный, Column only (1) – для сохранения значений атрибута только в заданный столбец без занесения в столбец data, Large Objects (2) – используется для файлов, JSONB (3) – используется, при большом объеме данных в хранилище, улучшает производительность.

Автоматическая настройка хранилищ

Точки доступа ► Хранилища ► Перенести класс в хранилище

Кнопка «Перенести класс в хранилище» на странице «Хранилища» позволяет выполнить операцию создания и настройки хранилища типа postgresql или postgresql_jsonb в полуавтоматическом режиме. После нажатия на эту кнопку «Перенести класс в хранилище» появляется диалоговое окно, в котором пользователь выбирает наименование хранилища и вводит идентификатор класса модели.

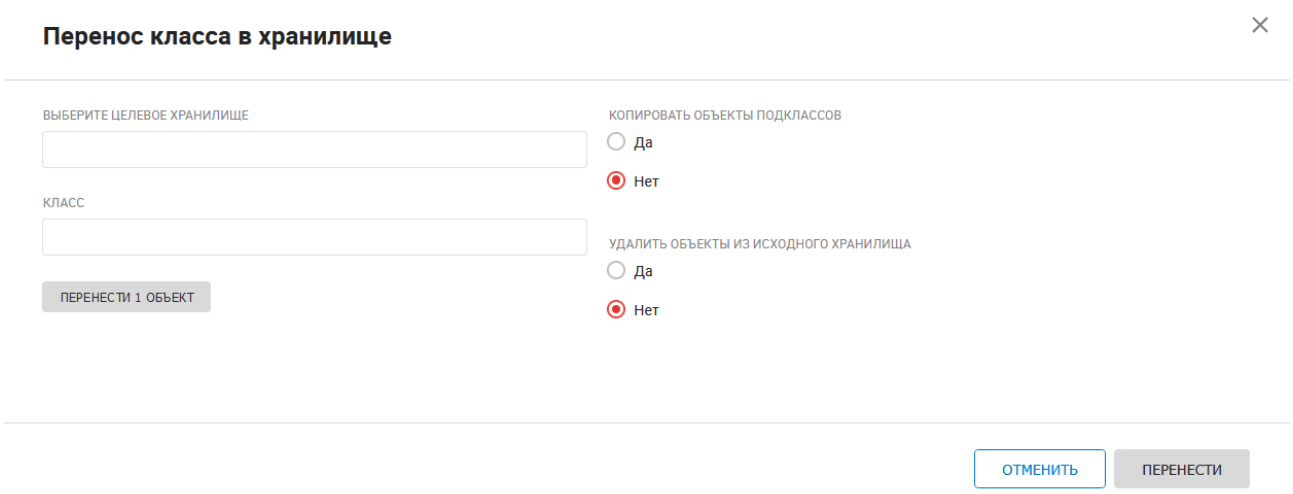

Рис. 4.9. Диалоговое окно переноса класса в хранилище

Если нужно сразу скопировать или перенести уже существующие в MDM объекты данного класса в создаваемое хранилище, пользователь отмечает переключатели «Копировать объекты подклассов» и/или «Удалить объекты из исходного хранилища».

Настройка маршрутов согласования

В разделе «Маршруты согласования» производится настройка последовательностей подтверждения изменений, которые пользователи вносят в атрибуты объектов модели.

Маршрут согласования создается для точки доступа.

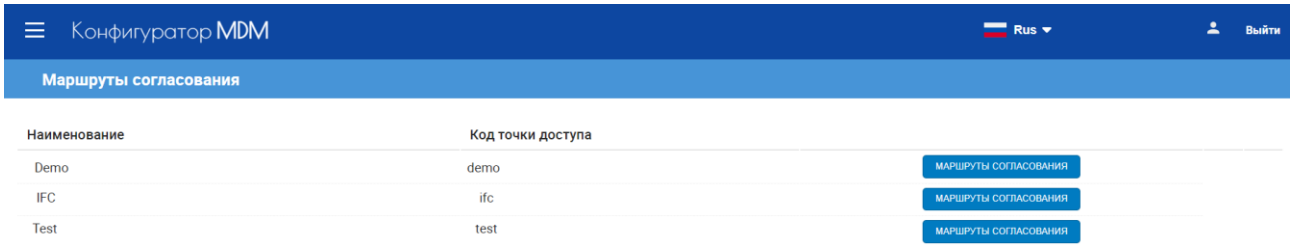

Рис. 5.1. Список точек доступа с маршрутами согласования

Маршруты согласования ► Маршруты согласования ► Создать/редактировать

В диалоговом окне создания/редактирования маршрута согласования нужно указать следующие свойства:

- Наименование – читаемое название маршрута
- Класс(ы) объектов – классы объектов, для которых определен маршрут согласования
- Применять при создании/изменении/удалении – признаки использования маршрута для соответствующих действий с объектами класса
- Активен – позволяет временно отключить маршрут для объектов данного класса.

Дополнительно маршруты согласования можно удалять, экспортировать и импортировать.

Для созданного маршрута необходимо выбрать участников, которые должны согласовывать изменения объектов указанного класса через АрхиГраф.Мир. Для перехода к редактированию структуры маршрута нужно нажать кнопку «Маршрут».

Маршруты согласования ► Маршруты согласования ► Маршрут ► Добавить сущность

В качестве участников маршрута нажатием на кнопку «Добавить сущность» могут быть добавлены:

Система-источник – информационная система, клиент АрхиГраф.MDM. В этом случае согласование сможет выполнять любой пользователь, работающий с MDM от имени соответствующей системы.

Логин – пользователь, работающий с АрхиГраф.Мир и авторизованный с указанным логином KeyCloak.

Имя пользователя – пользователь, работающий с АрхиГраф.Мир и авторизованный в KeyCloak, причем его учетная запись имеет указанное имя.

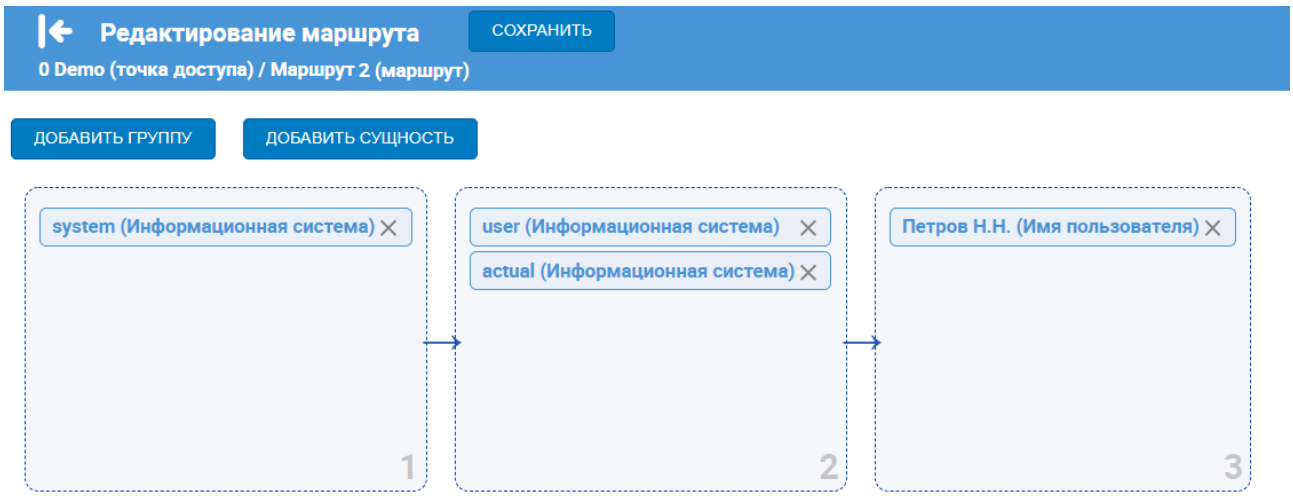

Рис. 5.2. Редактирование маршрута согласования

После создания маршрута его участников можно перемещать между группами путем перетаскивания, удалять участников нажатием на крестик в блоке с именем участника. По окончании редактирования маршрута его необходимо сохранить.

Маршруты согласования ► Маршруты согласования ► Маршрут ► Добавить группу

Участники могут объединяться в группы, если от них требуется проводить согласование в параллельном режиме – для этого нужно нажать кнопку «Добавить группу». Для группы можно выбрать режим (логическую операцию) согласования участников: «И» или «ИЛИ», по умолчанию проставляется режим «И».

Настройка систем-клиентов

В разделе «Системы-клиенты» конфигуратора MDM выполняется настройка учетных записей, под которыми автоматизированные системы подключаются к API MDM. На стартовой странице этого раздела отображается список точек доступа.

Системы-клиенты ► Системы-клиенты

После выбора точки доступа и перехода в подраздел «Системы-клиенты» отображается список зарегистрированных для нее систем-клиентов, который имеет такой вид:

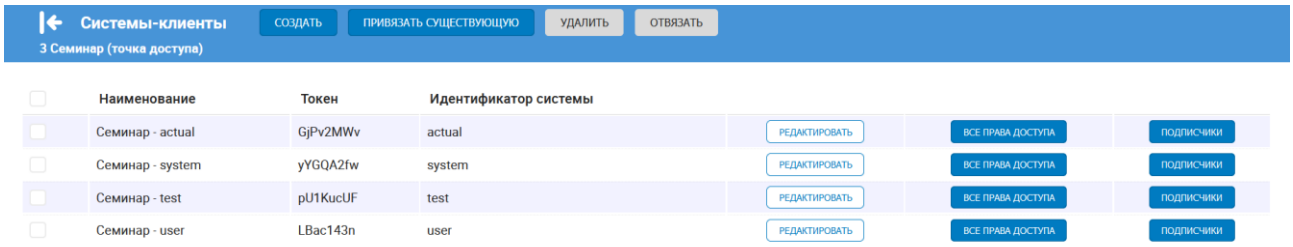

Рис. 6.1. Список систем-клиентов для точки доступа

Системы-клиенты ► Системы-клиенты ► Создать/Редактировать

В диалоге создания/редактирования системы нужно указать следующие свойства:

- Наименование – читаемое название
- Токен – опционально используется для дополнительного подтверждения аутентичности обращающейся системы, может передаваться в поле Token в обращениях к API.
- Идентификатор системы – используется при обращении через API, указывается в поле Originator всех запросов

Дополнительно есть возможность привязать и отвязать существующую систему-клиента.

Системы-клиенты ► Системы-клиенты ► Все права доступа ► Права доступа к классам ► Создать/Редактировать

По умолчанию каждая система-клиент имеет неограниченный доступ ко всем данным в точке доступа. Эти права можно ограничить. При нажатии на кнопку «Все права доступа» будет осуществлен переход в раздел для настройки прав доступа к классам, атрибутам и прав систем-источников.

Права доступа к классам отображены в виде списка классов, к которым системе-клиенту разрешен доступ. Нажатие на кнопку «Создать» открывает диалоговое окно, в котором необходимо выбрать класс для настройки доступа и установить переключатели:

- Чтение – флаг выбора может ли система-клиент читать объекты данного класса.
- Запись – флаг выбора может ли система-клиент создавать, изменять, удалять объекты данного класса.

- Запись с подтверждением – флаг выбора может ли система-клиент создавать запросы на создание, изменение, удаление объектов данного класса, которые должен подтвердить или отклонить пользователь, работающий от имени системы с более высокими правами доступа.

Если все три переключателя установить в положение «Нет», данной системе будет запрещен доступ ко всем объектам выбранного класса.

Подклассы наследуют уровень доступа, установленный для надкласса.

При редактировании прав доступа к классу можно дополнительно установить переключатели:

- Создание - система-клиент может создавать объекты данного класса.
- Изменение - система-клиент может изменять объекты данного класса.
- Удаление - система-клиент может удалять объекты данного класса.

Дополнительно есть возможность экспортировать и импортировать права доступа.

Системы-клиенты ► Системы-клиенты ► Все права доступа ► Права доступа к классам ► Условия

При нажатии на кнопку «Условия» откроется диалоговое окно «Редактирование условий». При нажатии на иконку «Плюс» будет открыто дополнительное окно с возможностью настройки атрибута для условия, тип операции и значение.

Типы операций:

- equal – равно значению;
- notequal – не равно значению;
- exists – значение атрибута существует (заполнено);
- notexist – значение атрибута не существует (не заполнено);
- contains – значение содержит;
- iequal – равно значению (целому числу);
- more – значение больше чем;
- less – значение меньше чем;
- moreorequal – значение больше или равно;
- lessorequal – значение меньше или равно.

Системы-клиенты ► Системы-клиенты ► Все права доступа ► Права доступа к атрибутам

В подразделе «Права доступа к атрибутам» настраиваются права доступа системы-клиента к значениям свойств объектов онтологии:

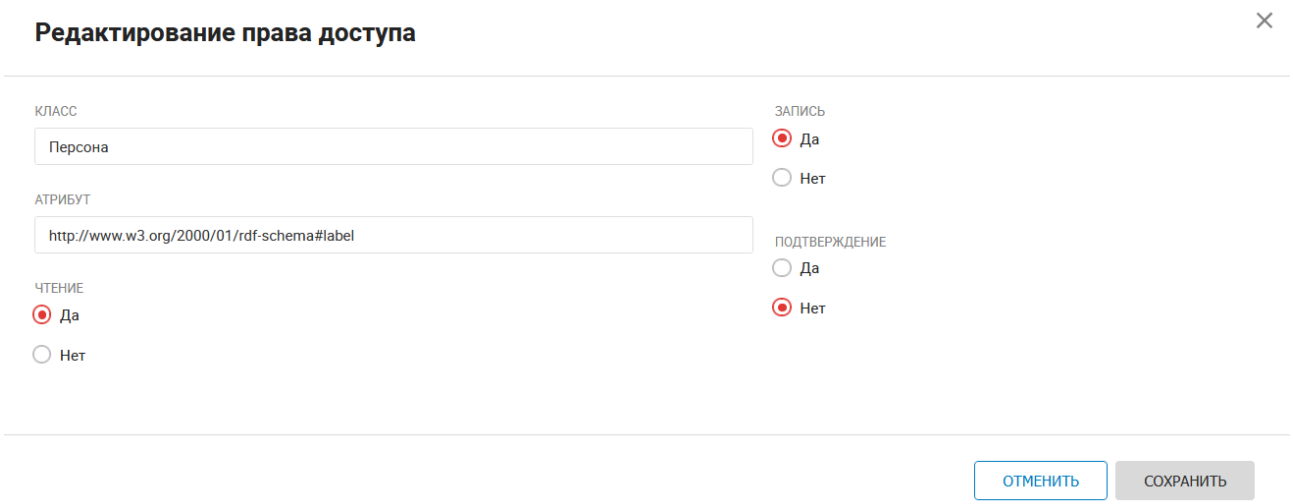

Рис. 6.2. Диалоговое окно настройки прав доступа к атрибутам (свойствам объекта онтологии).

Возможно, например, настроить права доступа какой-либо системы к объектам класса «Персона» как «Запись», но для атрибута rdfs:label установить доступ «Чтение». Тогда система-клиент сможет изменять значения всех атрибутов Персоны, кроме rdfs:label, который будет доступен ей только на чтение.

Права доступа к атрибутам настраиваются в контексте класса; таким образом, системе-клиенту может быть доступен только для чтения атрибут rdfs:label объектов класса Персона, но доступен на запись атрибут rdfs:label объектов класса Контрагент.

Системы-клиенты ► Системы-клиенты ► Все права доступа ► Права доступа систем-источников

В подразделе «Права доступа систем-источников» осуществляются настройки при необходимости разделения прав доступа для подсистем одной системы.

Системы-клиенты ► Системы-клиенты ► Подписчики ► Создать/Редактировать

На вкладке «Подписчики» можно просмотреть и настроить набор подписок для данной системы-клиента (установить подписки может и само приложение с помощью API MDM). На стартовом экране этой вкладки отображаются реквизиты очереди, через которую система-клиент хочет получать от MDM по подписке информацию о создании/изменении/удалении объектов определенных классов.

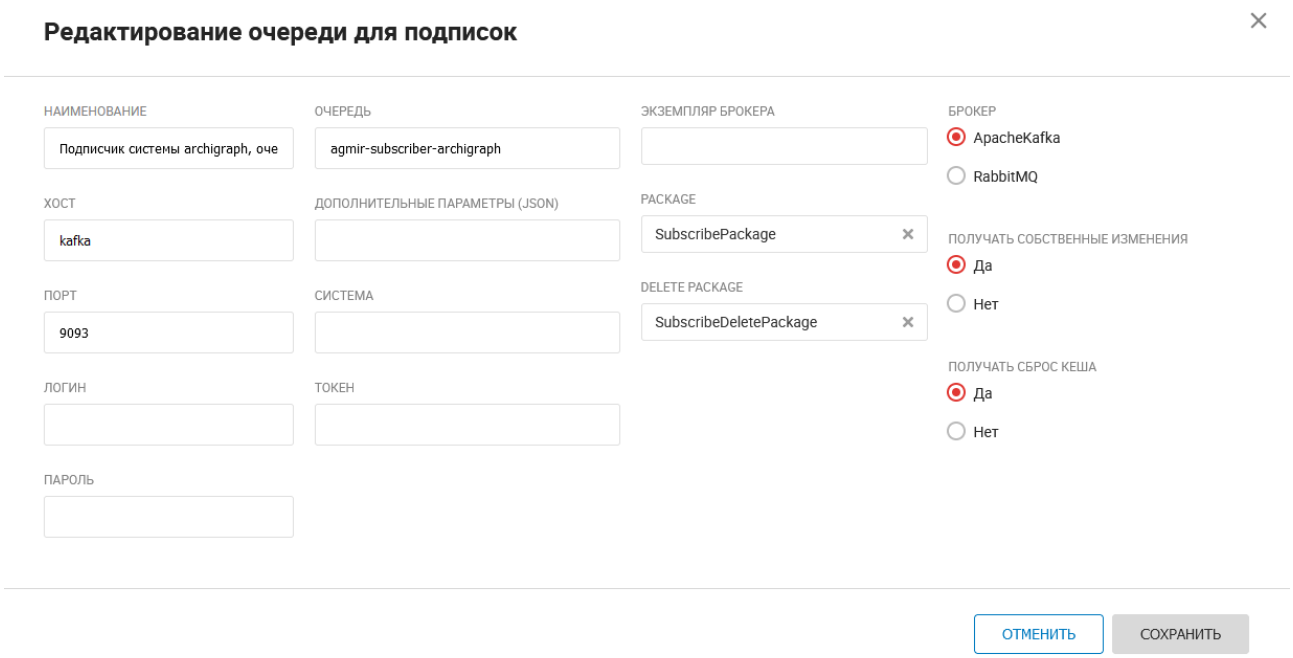

Рис. 6.3. Реквизиты подписки

В диалоговом окне нужно ввести читаемое наименование конфигурации, выбрать один из поддерживаемых типов очередей обмена сообщениями (RabbitMQ или Kafka), а также указать реквизиты подключения к очереди: хост, порт, логин, пароль, имя очереди/топика, дополнительные параметры (json), система, токен, экземпляр брокера, Package, Delete package и установить переключатель необходимо ли получать собственные изменения.

Дополнительно подписчиков можно удалять, экспортировать и импортировать.

Системы-клиенты ▶ Системы-клиенты ▶ Подписчики ▶ Подписки

При нажатии на кнопку «Подписки» будет отображен список классов, информация об изменении объектов которых поступает системе-клиенту через эту очередь. Окно добавления/редактирования подписки имеет такой вид:

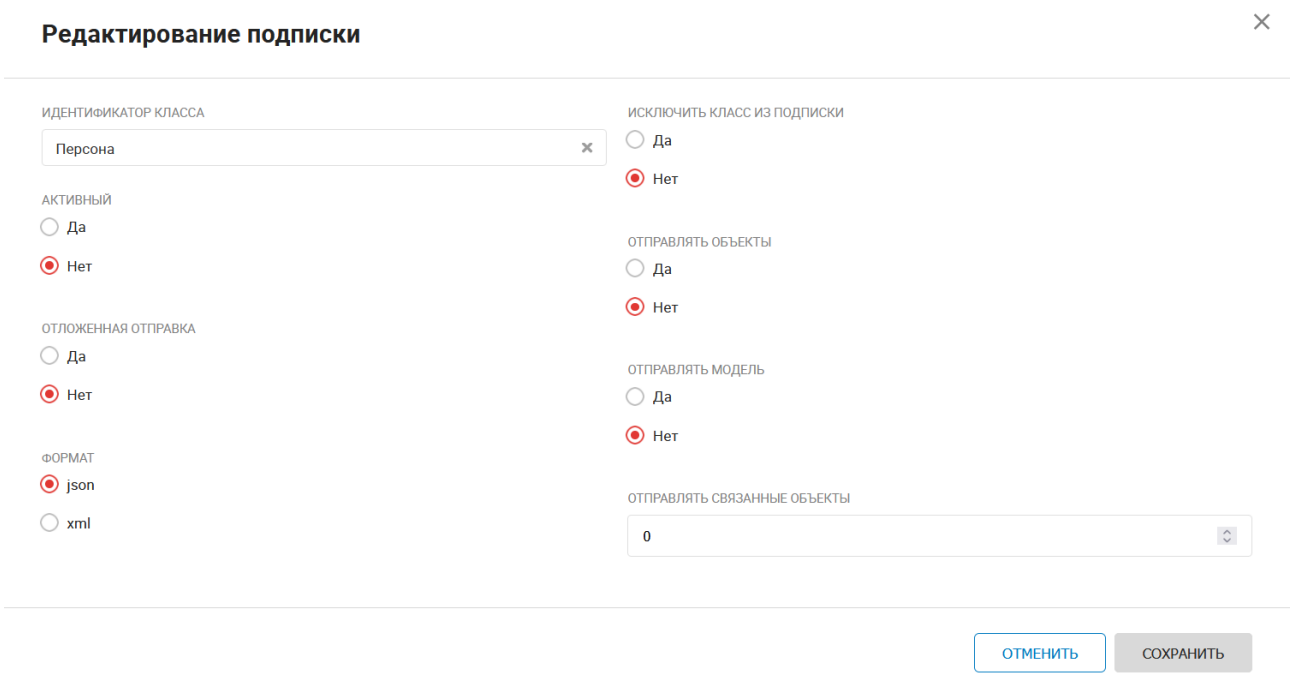

Рис. 6.4. Окно свойств подписки на определенный класс

В этом окне необходимо выбрать идентификатор класса онтологии, а затем установить переключатели:

- Активный – позволяет временно отключить подписку на объекты данного класса
- Отложенная отправка – если установлен, то отправка сообщения происходит не в момент изменения объекта в MDM, а пакетами с определенной периодичностью
- Формат отправляемых сообщений – XML или JSON
- Исключить класс из подписки – возможен вариант, когда устанавливается подписка на надкласс, а затем исключается подписка на один или несколько его подклассов
- Отправлять объекты – если установлен, то MDM отправляет уведомления об изменении объектов выбранного класса
- Отправлять модель – если установлен, то MDM отправляет уведомления об изменении модели (набора применимых атрибутов, свойств самого класса) выбранного класса
- Отправлять связанные объекты – если установлен, то MDM отправляет не только сам изменившийся объект, но и связанные с ним объекты. В значении данного свойства указывается число, которое указывает на глубину связности. Например, при установке значения «1» подписка будет осуществляться на указанный класс и дополнительные объекты, на которые ссылаются атрибутами-ссылками объекты этого класса. Если будет указано «2», то глубина увеличится на один уровень и к подписке присоединяться объекты, на которые ссылаются дополнительные объекты и т. д.

Системы-клиенты ▶ Системы-клиенты ▶ Подписчики ▶ Подписки ▶ Условия и группы

На вкладке «Условия и группы» можно настроить логические условия, по которым происходит отбор объектов, отправляемых по подписке (если условия не настроены, то отправляются любые измененные объекты класса).

В дополнительных условиях подписки «Группа» - целое число. По умолчанию условия с одинаковой группой объединяются между собой логической операцией «ИЛИ». Группы с одинаковым значением свойства «Группа» между собой объединяются логическим «И». Условия без указания группы (или группа = 0) между собой и между группами по умолчанию объединяются логическим «И». Операция сравнения included используется для задания сложных логических условий.

Системы-клиенты ▶ Системы-клиенты ▶ Подписчики ▶ Подписки ▶ Связанные объекты

На вкладке «Связанные объекты» можно указать, объекты каких именно классов должны отправляться вместе с измененным объектом, если установлен переключатель «Отправлять связанные объекты». Также необходимо выбрать свойство, которым измененный объект ссылается на связанный объект для отправки.

Если несколько систем-клиентов с одинаковым идентификатором используются в разных точках доступа, то нет необходимости для каждой из них создавать отдельную запись. Для этого данной функции необходимо нажать кнопку «Привязать существующую». Кнопка «Отвязать» используется если необходимо отвязать систему от указанной точки, а «Удалить» - если необходимо удалить систему из всех точек доступа.

Системы-клиенты ▶ Права доступа

В разделе представлен инструмент для просмотра прав доступа конкретного класса или атрибута. Необходимо установить флаг выбора класса или атрибута, и в поле поиска ввести минимум два символа идентификатора класса/атрибута. После выбора идентификатора необходимо нажать на кнопку «Перейти». Страница просмотра и редактирования прав доступа атрибута являетсяФилиаломОрганизации указаны на рис. 6.5.

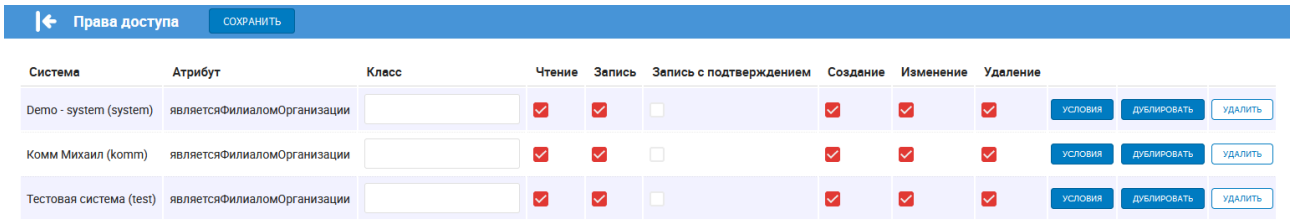

Рис. 6.5. Страница просмотра и редактирования прав доступа атрибута

Системы-клиенты ▶ Активные подписки ▶ Подписчик

Раздел аналогичен пути «Системы-клиенты ▶ Системы-клиенты ▶ Подписчики ▶ Подписки».

Системы-клиенты ▶ Активные подписки ▶ Подписчик ▶ Условия и группы

Раздел аналогичен пути «Системы-клиенты ▶ Системы-клиенты ▶ Подписчики ▶ Подписки ▶ Условия и группы».

Системы-клиенты ▶ Активные подписки ▶ Подписчик ▶ Связанные объекты

Раздел аналогичен пути «Системы-клиенты ▶ Системы-клиенты ▶ Подписчики ▶ Подписки ▶ Связанные объекты»

Настройка специальных атрибутов

Специальные атрибуты ► Создать/редактировать

В разделе «Специальные атрибуты» настраивается набор атрибутов для всех точек доступа.

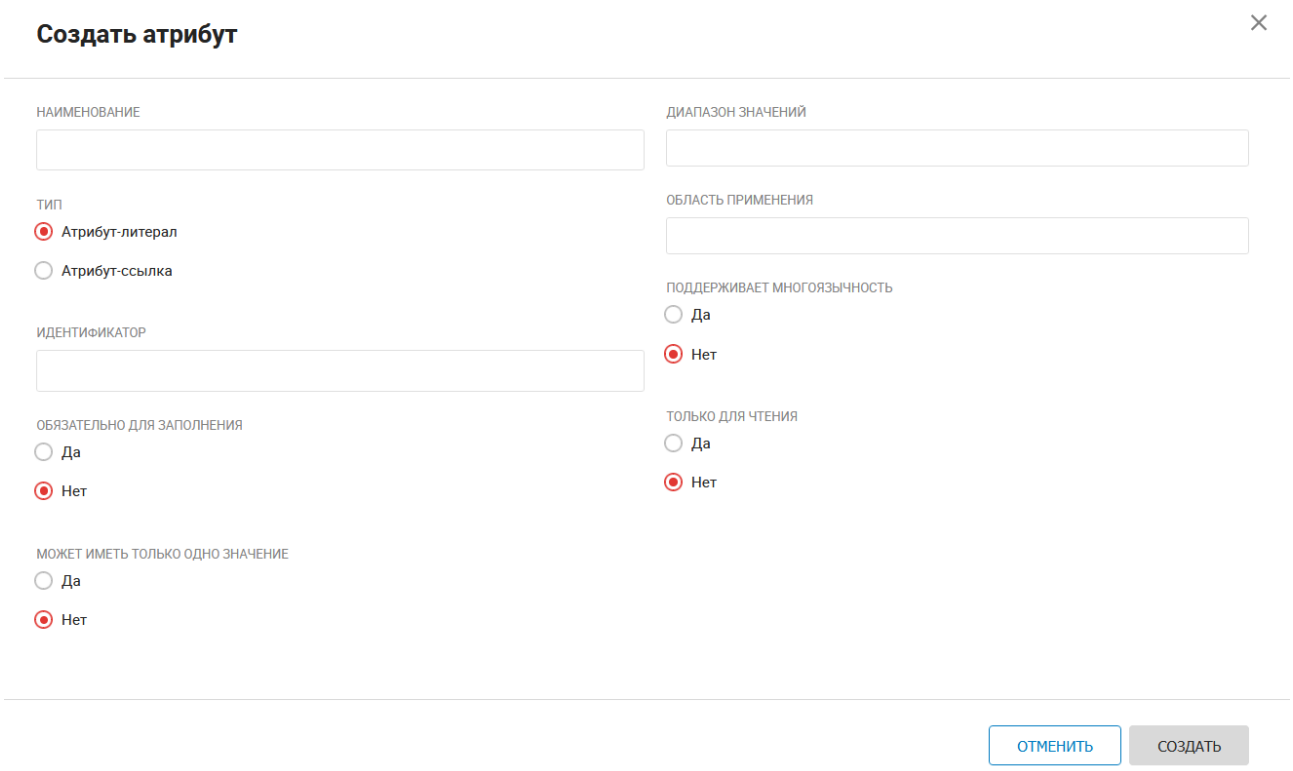

Рис. 6.6. Диалоговое окно создания специального атрибута

Поля свойств атрибута имеют следующий смысл:

- «Наименование» – читаемое название атрибута
- «Тип» – атрибут-литерал (owl:DatatypeProperty) или атрибут-ссылка (owl:ObjectProperty).
- «Идентификатор» – полный URI, идентификатор создаваемого атрибута.
- «Область применения» – класс или тип сущности, к которому применим атрибут (например, выберите owl:Class, чтобы создать атрибут, применимый ко всем классам онтологии).
- «Диапазон значений» – для атрибутов-ссылок это один или несколько классов онтологии, из числа объектов которых выбираются значения для данного свойства. Для атрибутов-литералов это один из доступных типов данных xsd.
- «Поддерживает многоязычность», «Обязательно для заполнения», «Может иметь только одно значение» (для каждого объекта), «Только для чтения» – дополнительные свойства атрибута.

При редактировании специального атрибута появляется дополнительное поле «Редактировать параметры». В нем при нажатии на символ + можно настроить значение любого нестандартного атрибута для стандартного атрибута.

Настройка брокеров

Брокеры ► Создать/редактировать

При нажатии на кнопки «Создать» или «Редактировать» будет раскрыто диалоговое окно настройки свойств брокера.

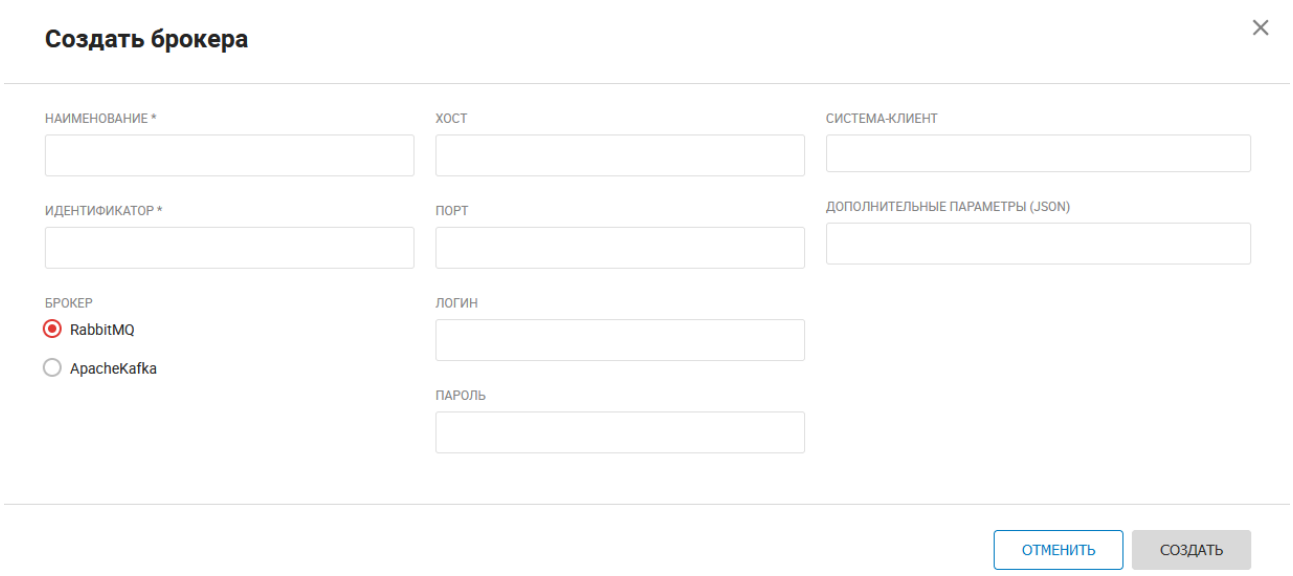

Рис. 6.7. Окно свойств брокера

Свойства брокера:

- «Наименование» – читаемое название для отображения в списке.
- «Идентификатор» –код, который используется для оформления подписки запросом к АрхиГраф.MDM с использованием данного брокера.
- «Брокер» - тип программного брокера сообщений.
- «Хост», «Порт», «Логин», «Пароль» – реквизиты подключения к серверу.
- «Система-клиент» - система-источник, для которого настраивается брокер.
- «Дополнительные параметры (json)» - значение в формате json, которое позволяет задать дополнительные настройки.

Дополнительно брокеры можно удалять, экспортировать и импортировать.

Настройка фоновых заданий

Фоновые задания ► Создать/редактировать

На вкладке «Фоновые задания» можно указать

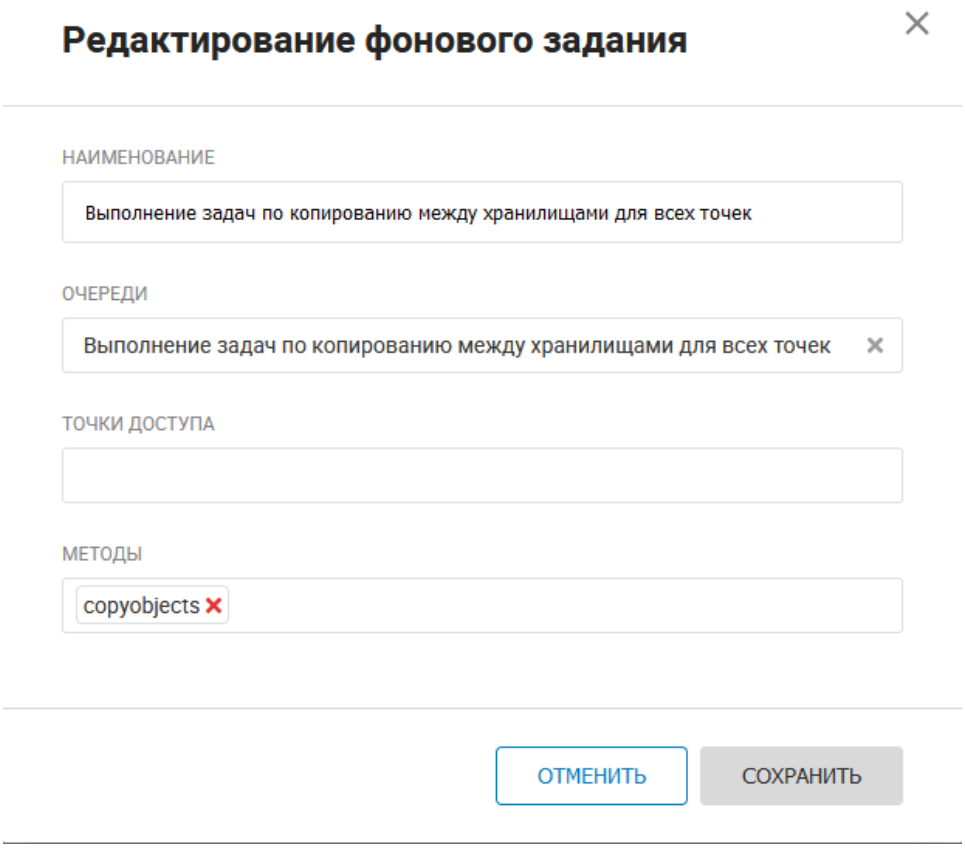

Рис. 7.1. Настройка фоновых заданий

В диалоге создания/редактирования фоновых заданий нужно указать следующие свойства:

- Наименование – читаемое название.
- Очереди – прослушиваемые очереди, настраиваемые в разделе Конфигуратора АрхиГраф.MDM «Прослушиваемые очереди». В данном свойстве необходимо выбрать одну очередь из выпадающего списка.
- Точки доступа – точки доступа, для которых настраивается фоновое задание. Если оставить этот параметр незаполненным, то задание будет выполняться для всех точек.
- Методы – методы, применяемые в задании.

  Варианты методов:
  - delayedexecution – отложенное исполнение,
  - copyobjects – копирование объектов,
  - renameattrvalue – актуализировать читаемое наименование объекта в атрибутах-ссылках при его переименовании,

- changereferance – при слиянии дубликатов заменить все ссылки на объект ссылкой на его дубликат,
- deleteattrvalue - удалить ссылки на объект,
- deleteattr - удалить все значения атрибута.

Настройка прослушиваемых очередей

Прослушиваемые очереди ► Создать/редактировать

АрхиГраф.MDM может получать запросы от систем-клиентов через очереди в асинхронном режиме, и отправлять ответы на эти запросы также через очереди. В разделе «Прослушиваемые очереди» конфигуратора можно настроить пары таких очередей. Интерфейс раздела имеет такой вид:

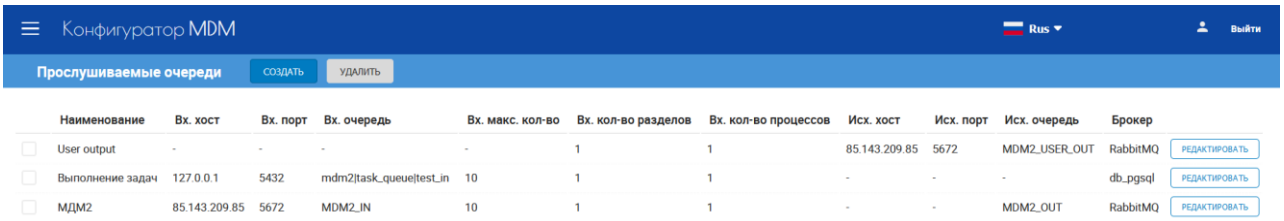

Рис. 8.1. Настройка прослушиваемых очередей

Для каждой пары очередей указываются следующие настройки:

- «Наименование» – читаемое название для отображения в интерфейсе
- «Брокер» – тип менеджера очередей, RabbitMQ,ApacheKafka или db_pgsql.
- Входящие и исходящие: хост, порт, логин, пароль, очередь – реквизиты для подключения MDM к входящей и исходящей очереди
- Вх. макс. кол-во, Вх./Исх. кол-во разделов, кол-во процессов – настройки очередей, определяющие режим чтения и отправки сообщений, в том числе количество процессов MDM для обработки сообщений определенной очереди
- «Дополнительные параметры (JSON)» – значение в формате json, которое позволяет задать дополнительные настройки.
- «Экземпляр брокера» - элемент из списка тех брокеров, которые заданы в разделе «Брокер.

Дополнительно прослушиваемые очереди можно удалять, экспортировать и импортировать.

Настройка констант

Раздел «Константы» Конфигуратора MDM позволяет настроить значения переменных, определяющих общие настройки работы системы. Раздел имеет следующий вид:

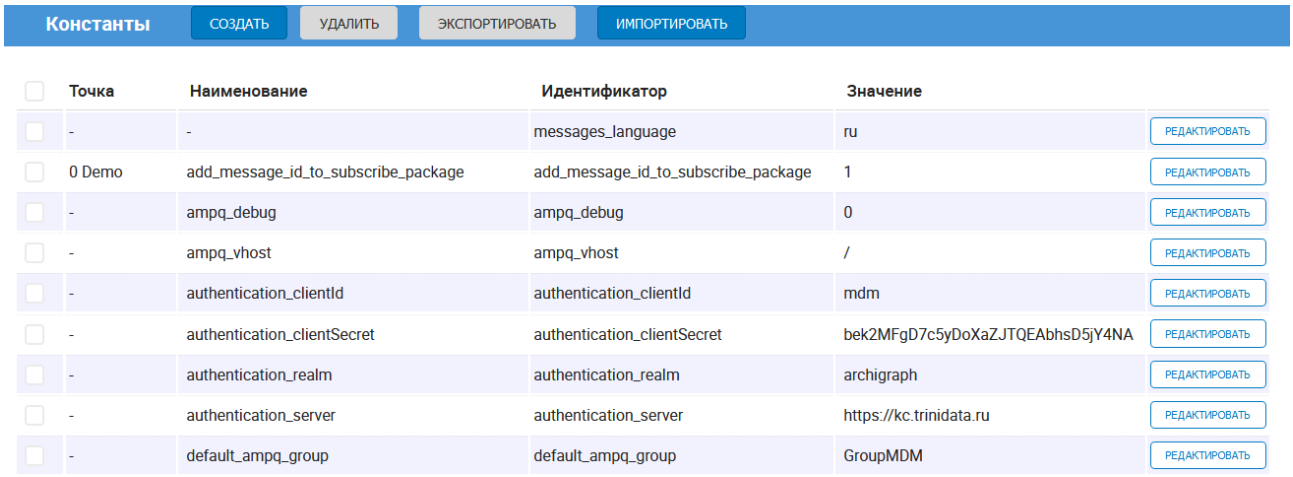

Рис. 9.1. Раздел «Константы» Конфигуратора MDM

При создании константы необходимо указать ее точку доступа, наименование, идентификатор и значение. Дополнительно есть возможность экспортировать и импортировать константы.

При редактировании константы есть возможность восстановить значение константы по умолчанию.

Настройка подписки на сводки изменений

Подписка на сводки изменений ► Создать/редактировать

Раздел «Подписка на сводки изменений» позволяет подписываться на изменения объектов выбранных классов через электронную почту.

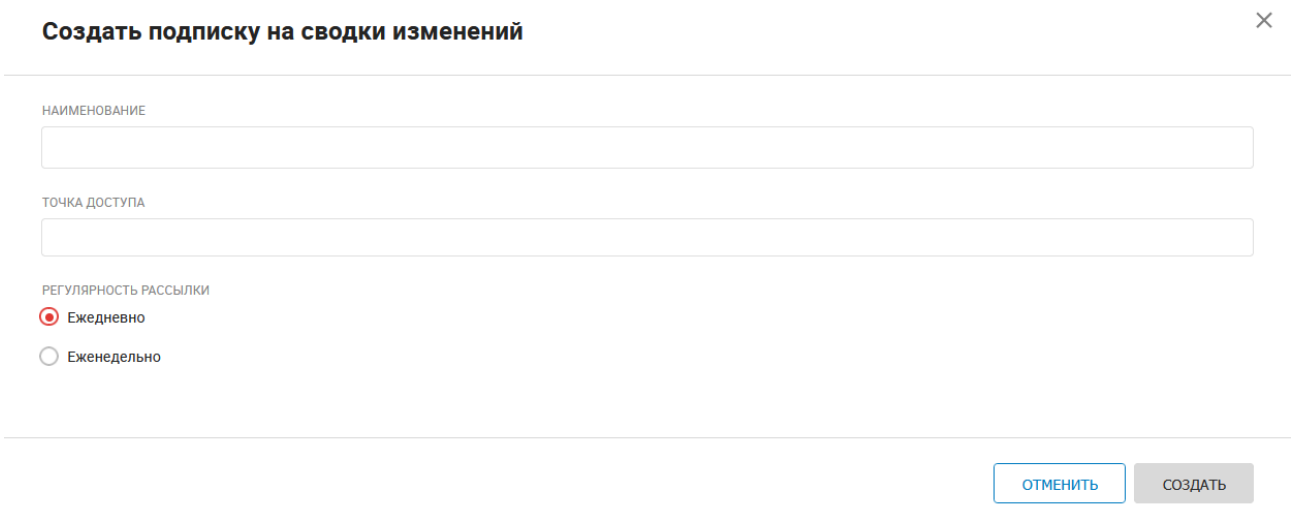

Рис. 10.1. Диалоговое окно создания подписки

Для создания подписки необходимо указать наименование подписчика, точку доступа и регулярность рассылки.

Подписка на сводки изменений ► Классы и подписчики

После создания подписки необходимо перейти к настройке классов указанной точки доступа и электронной почты получателя рассылки.

Настройка стандартных префиксов

Стандартные префиксы ► Создать/редактировать

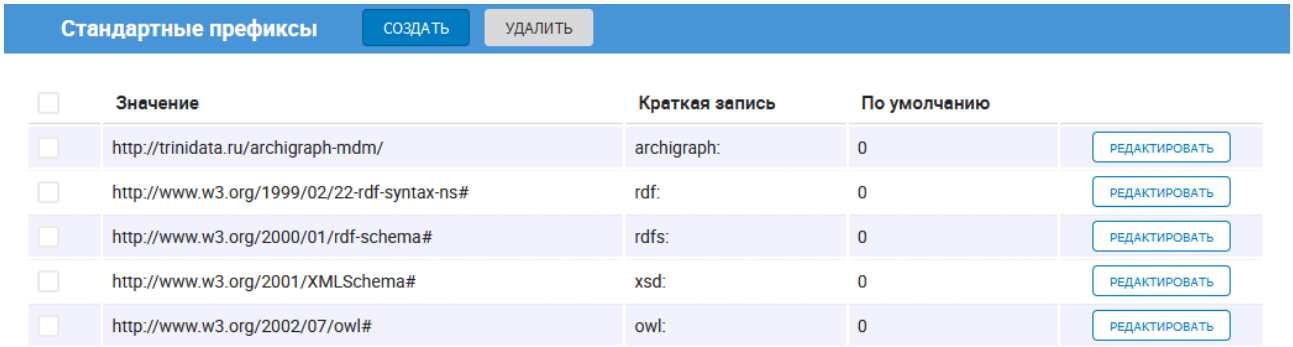

Рис. 11.1. Раздел «Стандартные префиксы» Конфигуратора MDM

Для настройки префикса необходимо указать его значение в формате URI, краткую запись и является ли он префиксом по умолчанию. Если для префикса указано «По умолчанию» равное «Да», то это означает, что АрхиГраф.MDM будет сокращать этот префикс при работе через API.

Настройка доступа к приложениям

Доступ к приложениям ► Сохранить

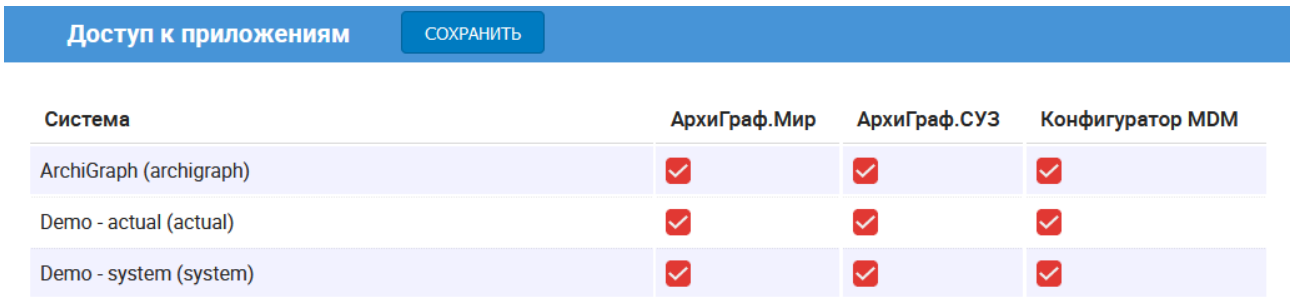

Рис. 12.1. Раздел «Доступ к приложениям» Конфигуратора MDM

Для настройки доступа систем-клиентов к приложениям необходимо проставить флаги для необходимых приложений и нажать кнопку «Сохранить».

Дополнительно для Конфигуратора MDM существует возможность настройки переменной окружения `ALLOWED_GROUPS`, которая предоставляет доступ системе при совпадении настроек группы Keycloak.

Лицензии

Раздел «Лицензии» предназначен для настройки лицензий АрхиГраф.MDM, АрхиГраф.Мир, АрхиГраф.СУЗ. Для каждого приложения необходимо ввести значения «Ключ запрос» и «Ключ ответ».

Настройка АрхиГраф.MDM с помощью утилиты mdmctl

Утилита служит инструментом настройки параметров АрхиГраф.MDM, таких как точка доступа, хранилище, система, классы, языки и т. д.

Во всех командах есть необязательный параметр **`--debug yes`**, который выводит sql запрос. На вспомогательные запросы это не распространяется.

Команда:

```
mdmctl help
```

выводит краткую справку по доступным командам.

Точка доступа (endpoint)

Создание точки доступа

Команда:

```
mdmctl add endpoint code_endpoint
```

где:

- code_endpoint - код создаваемой точки доступа. Является обязательным параметром и создает точку доступа с указанным кодом, пустым именем и английским языком по умолчанию.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- name - задает имя точки доступа для удобства идентификации. Так же с именем через знак "@" идет код языка, если язык не указывается, по умолчанию будет установлен английский,
- prefix — задает префикс для точки доступа,
- root — задает корень точки доступа,
- storage — задает хранилище данных. В этом параметре передается имя хранилища без указания языка,
- model_storage - задает хранилище модели. В этом параметре передается имя хранилища без указания языка.

Пример набора параметров для создания точки доступа:

```
mdmctl add endpoint Demo --name "Точка Demo"@ru --prefix http://trinidata.ru/demo/ --root http://trinidata.ru/demo
```

Удаление точки доступа

Команда:

```
mdmctl delete endpoint code_endpoint
```

где:

- code_endpoint - код удаляемой точки доступа.

Точка доступа не удаляется безвозвратно, а помечается специальным атрибутом и считается удаленной для системы.

Обновление точки доступа

Команда:

```
mdmctl update endpoint code_endpoint
```

где:

- code_endpoint - код обновляемой точки доступа.

Без введения дополнительных параметров обновления не произойдет. Для изменения необходимо задействовать хотя бы один дополнительный параметр со значением.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- name — задает имя для точки доступа. Если точек доступа с таким именем несколько (один код и одно имя, но разные языки) обновятся все точки доступа, сменится только имя. Если задаем имя и язык (например: mdmctl update endpoint demo-point@en --name demo) обновится имя только для точки доступа с кодом языка «en».
- prefix — меняет префикс для точки доступа,
- root — меняет корень для точки доступа,
- model_storage — меняет хранилище моделей для точки доступа,
- storage — меняет хранилище данных для точки доступа,
- code — меняет код точки доступа. Если код задан с языком (например, **`mdmctl update endpoint demo@en --code Demo1`**) то команда заменит код только для точки доступа, с сочетанием **`код@язык`**.

Обновление префикса по умолчанию (альтернативный вариант)

Команда:

```
mdmctl update prefix_default code_endpoint new_prefix
```

где:

- code_endpoint - код обновляемой точки доступа,
- new_prefix — новый префикс по умолчанию для выбранной точки доступа.

Просмотр списка точек доступа

Команда**:**

```
mdmctl show endpoint
```

выведет свойства всех точек доступа,

```
mdmctl show endpoint Demo
```

покажет свойство точки доступа с кодом «Demo».

Просмотр префикса по умолчанию для точки доступа

Команда:

```
mdmctl show prefix_default Demo
```

покажет свойство точки доступа с кодом «Demo».

Хранилище (storage)

Создание хранилища

Команда:

```
mdmctl add storage storage_type storage_name
```

где:

- storage_type – тип хранилища (варианты: fuseki, allegrograph, blazegraph, mongodb, mongo, postgresql, postgresql_jsonb),
- storage_name – имя хранилища.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- host — имя сервера,
- port — порт на сервере,
- login — логин для доступа к базе,
- password — пароль для доступа к базе (нельзя использовать восклицательный знак и кавычки (и одинарные, и двойные)),
- query — путь к SPARQL-сервису чтения данных,
- update — путь к SPARQL-сервису изменения данных,
- dbname — имя базы данных, которая используется как хранилище,
- table — указывает конкретную таблицу с данными,
- history — если true, то определяет, что хранилище сохраняет историю изменения свойств объектов,
- logic — если true, то определяет, что хранилище поддерживает логические операции.

Пример команды с параметрами для Fuseki:

```
mdmctl add storage fuseki "Хранилище модели Demo" --host 127.0.0.1 --port 8080 --login mdm --password test --query /fuseki/demo/query --update /fuseki/demo/update
```

где:

- fuseki – тип создаваемого хранилища,
- Хранилище модели Demo – его название,
- 127.0.0.1 – IP-адрес хоста,
- 8080 – порт,
- mdm – логин для HTTP-авторизации,
- test – пароль для HTTP-авторизации,
- /fuseki/demo/query – путь к SPARQL-сервису чтения данных,
- /fuseki/demo/update – путь к SPARQL-сервису изменения данных,
- logic — если true, то определяет, что хранилище поддерживает логические операции.

Пример команды с параметрами для Postgres:

```
mdmctl add storage postgresql_jsonb "Хранилище данных Demo Postgres" --host 127.0.0.1 --login mdm --password test --dbname mdm_data --table test --history true --logic true
```

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- dbname mdm_data – таким образом задается имя базы данных, которая используется как хранилище,
- table test – указывает конкретную таблицу с данными,
- history true – определяет, что хранилище сохраняет историю изменения свойств объектов,
- logic true – определяет, что хранилище поддерживает логические операции.

Удаление хранилища

Команда:

```
mdmctl bind storage "Хранилище данных Demo Postgres" Demo
```

где:

- Хранилище данных Demo Postgres — пример имени удаляемого хранилища.

Присоединение хранилища данных к точке доступа

Команда:

```
mdmctl bind storage "Хранилище данных Demo Postgres" Demo
```

где:

- Хранилище данных Demo Postgres — пример имени хранилища
- Demo — пример кода точки доступа.

Присоединение хранилища моделей к точке доступа

Команда:

```
mdmctl bind model_storage "Хранилище модели Demo" Demo
```

где:

- Хранилище модели Demo — пример имени хранилища,
- Demo — пример кода точки доступа.

Отвязывание хранилища от точки доступа

Команда:

```
mdmctl unbind storage "Хранилище данных Demo Postgres" Demo
```

где:

- Хранилище модели Demo — пример имени хранилища,
- Demo — пример кода точки доступа.

Обновление хранилища

Команда:

```
mdmctl update storage "storage_name"
```

где:

- storage_name — имя хранилища.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- name — новое имя хранилища (без @ и кода языка),
- host — имя сервера,
- port — порт на сервере,
- login — логин для доступа к базе,
- password — пароль для доступа к базе,
- query_path — путь к SPARQL-сервису чтения данных,
- update_path — путь к SPARQL-сервису изменения данных ,
- dbname — имя базы данных, которая используется как хранилище,
- tab — указывает конкретную таблицу с данными,
- history — если true, то определяет, что хранилище сохраняет историю изменения свойств объектов,
- logic — если true, то определяет, что хранилище поддерживает логические операции.

Для обновления необходимо добавить хотя бы один дополнительный параметр со значением.

Пример команды:

```
mdmctl update storage "Хранилище модели Demo" --name "Хранилище данных demo" --dbname stor_demo
```

изменит хранилище с именем "Хранилище модели Demo". Имя изменится на "Хранилище данных demo", а имя базы данных станет "stor_demo".

Удаление хранилища

Команда:

```
mdmctl delete storage "Хранилище данных Demo Postgres"
```

где:

- Хранилище данных Demo Postgres — пример имени хранилища.

Просмотр списка хранилищ

Команда:

```
mdmctl show storage
```

покажет все существующие хранилища,

```
mdmctl show storage "Хранилище данных Demo Postgres"
```

покажет свойства хранилища с именем "Хранилище данных Demo Postgres".

Языки

Создание языка

Команда:

```
mdmctl add language it "Italiano"
```

где:

- it – пример кода языка (итальянский),
- Italiano – пример наименования языка.

Просмотр списка языков

Команда:

```
mdmctl show language
```

выводит информацию по всем языкам,

```
mdmctl show language ru
```

выводит информацию по языку с кодом ru.

Удаление языка

Команда:

```
mdmctl delete language it
```

где:

- it – пример кода языка.

Прослушиваемые очереди

Создание прослушиваемой очереди в настройках MDM

Команда:

```
mdmctl add queue
```

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- name — имя прослушиваемой очереди,
- in — название входящей очереди,
- out — название исходящей очереди,
- input_host — хост входящей очереди,
- output_host — хост исходящей очереди,
- input_port — порт входящей очереди,
- output_port — порт исходящей очереди,
- input_login — логин входящей очереди,
- output_login — логин исходящей очереди,
- input_password — пароль входящей очереди,
- output_password — пароль исходящей очереди,
- broker — брокер очередей (например, RabbitMQ).

Для обновления необходимо добавить хотя бы один дополнительный параметр со значением.

Пример команды:

```
mdmctl add queue --name demo --in SYNCH_QUEUE_IN --out SYNCH_QUEUE_OUT --input_host 127.0.0.1 --input_port 5672 --broker RabbitMQ --input_login admin --input_password mypass --output_host 127.0.0.1 --output_port 5672 --output_login admin --output_password mypass
```

Если не задать имя оно формируется из параметров **--in** и **--out**, например, из команды выше получилось бы "SYNCH_QUEUE_IN to SYNCH_QUEUE_OUT".

Просмотр прослушиваемых очередей

Команда:

```
mdmctl show queue
```

выводит свойства всех очередей,

```
mdmctl show queue demo
```

выводит свойство очереди с именем demo.

Обновление очереди

Команда:

```
mdmctl update queue "queue_name"
```

где:

- "queue_name" имя очереди.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- name — имя прослушиваемой очереди,
- input_queue — название входящей очереди,
- out_queue — название исходящей очереди,
- input_host — хост входящей очереди,
- output_host — хост исходящей очереди,
- input_port — порт входящей очереди,
- output_port — порт исходящей очереди,
- input_login — логин входящей очереди,
- output_login — логин исходящей очереди,
- input_password — пароль входящей очереди,
- output_password — пароль исходящей очереди,
- broker — брокер очередей (например, RabbitMQ),
- input_max_count — максимальное количество для входящих сообщений в очереди.

Пример команды:

```
mdmctl update queue test --name "Синхронизация контуров" --broker RabbitMQ
```

Изменит очередь с именем "test". Имя изменится на «Синхронизация контуров» и брокер очередей будет заменен на RabbitMQ.

Удаление прослушиваемой очереди

Пример команды:

```
mdmctl delete queue --in SYNCH_QUEUE_IN --out SYNCH_QUEUE_OUT --input_host 127.0.0.1
```

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- in — имя входящей очереди
- out — имя исходящей очереди
- input_host — хост исходящей очереди. Этот параметр желательно указывать, чтобы не удалить лишнюю очередь случайно.

Константы

Установка значения константы

Команда:

```
mdmctl set constant log_folder "/../log/"
```

- log_folder – идентификатор константы,
- /../log/ – ее значение.

Дополнительный параметр (перед нимнеобходимо указывать двойное тире):

- name — задает имя для новой константы.

При выполнении команды создается или изменяется константа. Если команда указана без ключа name обновляется существующая константа.

Просмотр значений констант

Команда:

```
mdmctl show constant
```

выводит значения всех констант,

```
mdmctl show constant log_folder
```

выводит значение конкретной константы log_folder.

Системы-клиенты MDM

Создание системы (Originator'a)

Команда:

```
mdmctl add system "Имя_системы" system_code
```

где:

- Имя_системы — отображаемое имя системы,
- system_code — код системы.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- token — токен системы,
- main — информационный флаг, если 1 то система ассоциирована с редактором онтологий. По умолчанию 0.

- final - по умолчанию 0, если задать 1, то система, получив пакет по подписке дальше рассылать не будет, даже если есть подписчики на объект такого класса. Системы, заведенные для синхронизации, имеют final=1.

Пример команды:

```
mdmctl add system "Демо-клиент" demo --token democlient --main 1 --final 0
```

Будет создана система с именем "Демо-клиент", кодом demo токеном democlient, будет ассоциирована с редактором онтологий и признаком того, что от этой системы подписчики будут получать пакеты на обновление.

Предоставление доступа Originator'у к точке доступа

Команда:

```
mdmctl bind system code_system code_endpoint
```

где:

- code_system — код системы,
- code_endpoint — код точки доступа.

Пример команды:

```
mdmctl bind system test demo
```

свяжет систему test с точкой доступа demo.

Установка прав Originator'а на доступ к объектам класса

Команда:

```
mdmctl allow endpoint_code system_code entity_code read write confirm
```

где:

- endpoint_code — код точки доступа,
- system_code — код системы,
- entity_code — код класса,
- read — 1 если разрешено, 0, если запрещено,
- write — 1 если разрешено, 0, если запрещено ,
- confirm — 1 если разрешено, 0, если запрещено.

Параметры вводятся через пробел.

Пример команды:

```
mdmctl allow Demo demo "Событие" 1 1 0
```

Будут установлены права системы (Originator'а) на доступ к объектам класса «Событие» чтение, запись, без подтверждения.

Установка прав Originator'а на доступ к атрибутам объектов класса

Команда:

```
mdmctl allow attr endpoint_code system_code class entity_code read write confirm
```

где:

- endpoint_code — код точки доступа,
- system_code — код системы,
- class — код класса,
- entity_code — код атрибута,
- read — 1 если разрешено, 0, если запрещено,
- write — 1 если разрешено, 0, если запрещено ,
- confirm — 1 если разрешено, 0, если запрещено.

Параметры вводятся через пробел.

Пример команды:

```
mdmctl allow attr Demo demo "Событие" "связаноСКартой" 1 1 0
```

Будут установлены права системы (Originator'а) на доступ к атрибуту "связаноСКартой" объектов класса «Событие» чтение, запись, без подтверждения.

Просмотр списка Originator'ов

Команда:

```
mdmctl show system
```

просмотр всех свойств,

```
mdmctl show system demo
```

просмотр свойств только для системы demo.

Просмотр прав Originator'а на классы

Команда:

```
mdmctl show rights
```

просмотр всех заданных прав,

```
mdmctl show rights demo
```

просмотр заданных прав для точки доступа demo.

Просмотр прав Originator'а на атрибуты классов

Команда:

```
mdmctl show rights attr
```

просмотр всех заданных прав на атрибуты класса,

```
mdmctl show rights attr demo
```

просмотр заданных прав на атрибуты класса для точки доступа demo.

Обновление системы

Команда:

```
mdmctl update system system_code
```

где:

- system_code — код обновляемой системы.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- name — новое имя системы,
- token — токен системы,
- code — новый код системы,
- endpoint — код точки доступа,
- main — 1, если система ассоциирована с редактором онтологий, 0, если нет,
- final — 1, если система конечная(принимает пакеты обновлений от другой системы), 0, если нет.

Для обновления необходимо добавить хотя бы один дополнительный параметр со значением.

Пример команды:

```
mdmctl update system test --name "Тестовая система" --endpoint demo
```

Задаст для системы новое имя "Тестовая система" и свяжет ее с точкой доступа "demo"

Удаление системы (Originator'а)

Команда:

```
mdmctl delete system demo
```

Управление распределением данных между хранилищами

Привязка классов к хранилищам

Команда:

```
mdmctl bind class class_code storage_name endpoint_code
```

где:

- class_code - код класса. Можно использовать краткую форму записи, если он имеет префикс по умолчанию для данной точки доступа, и полную форму (в виде URI) в остальных случаях (пример: краткая форма: rdfs:label, полная форма: http://www.w3.org/2000/01/rdf-schema#label),
- storage_name - имя системы,
- endpoint_code - код точки доступа.

Пример команды:

```
mdmctl bind class "Событие" "Хранилище данных Demo Postgres" Demo
```

Просмотр списка классов, привязанных к хранилищам для точки доступа

Команда:

```
mdmctl show class
```

покажет все классы,

```
mdmctl show class "endpoint_code"
```

покажет только классы для точки доступа "endpoint_code".

Пример команды:

```
mdmctl show class demo
```

покажет классы для точки доступа "demo".

Отвязывание класса от хранилища

Команда:

```
mdmctl unbind class class_code "storage_name" endpoint_code
```

где:

- class_code - код класса,

- storage_name - имя системы,
- endpoint_code - код точки доступа.

Пример команды:

```
mdmctl unbind class "Событие" "Хранилище данных Demo Postgres" Demo
```

Отвяжет класс "Событие" от хранилища "Хранилище данных Demo Postgres" и точки доступа "Demo".

Управление хранением значений свойств в хранилищах

Создание мэппинга свойства

Команда:

```
mdmctl bind property attribute_name storage_name table_name field_name
```

где:

- attribute_name - идентификатор свойства в онтологической модели,
- storage_name - название хранилища,
- table_name - название таблицы базы данных,
- field_name - имя соответствующего поля в этой таблице.

Пример команды:

```
mdmctl bind property "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "Хранилище данных Demo Postgres" test type
```

Создаст мэппинг атрибута "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" в хранилище с именем "Хранилище данных Demo Postgres".

Для привязки столбца с идентификатором сущности используется следующая команда:

```
mdmctl bind property uri "storage_name" table_name uri
```

Просмотр мэппинга свойств

Команда:

```
mdmctl show property
```

просмотр всех свойств,

```
mdmctl show property "table_name"
```

просмотр свойств только для таблицы "table_name" .

Удаление свойства из мэппинга

Команда:

```
mdmctl unbind property "attribute_name" "storage_name" table_name
```

где:

- attribute_name - идентификатор свойства в онтологической модели,
- storage_name - название хранилища. Если хранилища для записи нет, ставим "",
- table_name - название таблицы базы данных.

Пример команды:

```
mdmctl unbind property "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" "Хранилище данных Demo Postgres" test
```

Удалит мэппинг для атрибута "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" в хранилище с именем Хранилище данных Demo Postgres"

Управление свойствами, определенными вне онтологии (стандартными свойствами)

Создание свойства в точке доступа

Команда:

```
mdmctl add property "property_uri" endpoint_code
```

где:

- property_uri - URI создаваемого свойства (например: owl:sameAs),
- endpoint_code - код точки доступа.

Дополнительные параметры (перед каждым из них необходимо указывать двойное тире):

- name — имя объекта с кодом языка через знак "@". Одновременно можно задавать несколько имен на разных языках,
- domain - область применения свойства,
- range - диапазон значений свойства,
- singular - равно true, если свойство может принимать единственное значение для каждого объекта,
- mandatory - равно true, если свойство должно обязательно иметь хотя бы одно значение для каждого объекта,
- multilang - равно true, если значения свойства могут быть заданы на разных языках,

- readonly - равно true, если значение свойства доступно только для просмотра и не может быть изменено.

Пример команды:

```
mdmctl add property owl:sameAs Demo --name "Эквивалентен"@ru --name "Equivalent"@en --domain owl:NamedIndividual --range owl:NamedIndividual --singular true --mandatory false --multilang true --readonly false
```

Создаст свойство owl:sameAs в точке доступа Demo с именами Эквивалентен и Equivalent на русском и английском языке с областью применения owl:NamedIndividual, диапазоном значений owl:NamedIndividual, свойство будет принимать единственное значение для каждого объекта, не обязательно для каждого объекта, свойство будет мультиязычное и может быть изменено.

Просмотр списка свойств для точки доступа

Команда:

```
mdmctl show field
```

выводит все свойства для всех точек доступа,

```
mdmctl show field endpoint_code
```

выводит свойства только для точки доступа endpoint_code.

Удаление свойства из точки доступа

Команда:

```
mdmctl delete property "property_uri" endpoint_code
```

где:

- property_uri - URI свойства (например: owl:sameAs),
- endpoint_code - код точки доступа.

Требования к квалификации персонала, осуществляющего сопровождение АрхиГраф.MDM

Операции по администрированию АрхиГраф.MDM должны выполняться системным администратором, имеющим опыт в администрировании Linux-систем, знакомым с принципами работы языков Java, PHP. Для настройки обмена между АрхиГраф.MDM с приложениями-клиентами (схема организации обмена описана в Руководстве пользователя) необходимо знание принципов и технологий брокера сообщений (Message Broker), корпоративной сервисной шины (Enterprise Service Bus).

Для поддержки модели данных АрхиГраф.MDM необходима квалификация аналитика/инженера по данным, а также знакомство с семантическими технологиями. Все необходимые сведения о них можно получить в нашем методическом пособии «Введение в онтологическое моделирование» (https://trinidata.ru/files/SemanticIntro.pdf), а также «Руководстве по созданию и поддержке семантической модели» (https://trinidata.ru/files/SemanticModelDesign.pdf).

Структура лог-файлов

Каждый лог-файл содержит лог за сутки. Названия файлов логов имеют вид:

- mdm_log_YYYYmmdd.log – логи входящих запросов.
- mdm_errorlog_YYYYmmdd.log – лог произошедших ошибок.

Столбцы файла лога запросов:

- дата и время запроса,
- идентификатор запросившей системы (если в запросе задан код системы и система с таким кодом есть),
- код запросившей системы (если найден в запросе),
- корневой тег запроса (например, GetObjectsGroup или DataModelRequest),
- ключевой атрибут запроса (Code в GetObjectsGroup и GetObject).

Столбцы файла с логом ошибок:

- дата и время запроса,
- текст ошибки,
- идентификатор запросившей системы (если задан код и система найдена),
- код запросившей системы (если задан в запросе),
- корневой тег запроса.
- ключевой атрибут запроса.

Те же логи могут сохраняться в базе PostgreSQL в таблице mdm_log. Оба лога (ошибок и осуществленных запросов) пишутся в одну таблицу и различаются значением поля type: 1 для ошибок и 2 для запросов. В отличие от текстового лога, в базу пишется так же полный текст запроса. Структура таблицы mdm_log:

- date - дата и время запроса,
- type — тип лога, 1 (для ошибки) или 2,
- comment — текст ошибки
- xml — поступивший запрос
- system — идентификатор запросившей системы (если система задана и найдена)
- system_code — код запросившей системы (если задан)
- request - корневой тег запроса
- code - ключевой атрибут запроса (Code в GetObjectsGroup и GetObject)

По вопросам, связанным с функционированием АрхиГраф.MDM, для решения возникающих проблем, можно обращаться в службу поддержки компании ТриниДата по телефону: +7 343 2 110 256.