POLY//ATICA

РУКОВОДСТВО АДМИНИСТРАТОРА МОДУЛЯ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ АНАЛИТИЧЕСКОЙ ПЛАТФОРМЫ POLYMATICA

115280,, МОСКВА, УЛ. АВТОЗАВОДСКАЯ, Д. 23, СТР.931, К.1
T +7 (495) 748-84-84 E INFO@POLYMATICA.RU

ОГЛАВЛЕНИЕ

1 ИСПОЛЬЗУЕМЫЕ ОПРЕДЕЛЕНИЯ

Модуль – модуль Polymatica ML

Раздел Исследование данных – компонент Исследование данных (Data Discovery – DD)

Исследование – это процесс графического и статистического анализа с целью поиска взаимосвязей атрибутов, закономерностей, тенденций и аномалий в данных.

Раздел Построение модели – компонент Построение модели (Model Designer – MD)

Проект MD – это сценарий моделирования, настраиваемый пользователем для решения конкретной задачи.

Сценарий моделирования– это комбинация узлов моделирования, настраиваемая пользователем в зависимости от решаемой задачи.

Узел – отдельная операция над данными

Раздел Управление моделями – компонент Управление моделями (Model Manager – MM)

Репозиторий – хранилище моделей и версий модели

Проект MM –логическое пространство, объединяющее несколько моделей по конкретной бизнес-задаче.

Модель – метаинформация, описывающая программное решение, позволяющее произвести трансформацию или анализ переменных, которые она ожидает на вход (Входные Переменные Модели) и вернуть результаты (Выходные Переменные Модели).

Версия модели – итеративная реализация модели.

ETL function – пользовательские функции

Instance Configs – настройки запуска публикации

Шаблон согласования – BPMN-процесс, в соответствии с которым будет выполняться согласование модели.

2 ТРЕБОВАНИЯ К АППАРАТНОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

2.1 Требования к аппаратному обеспечению

Приложение Модуль Polymatica ML предполагает свою работу на выделенном сервере приложений, аппаратные средства которого удовлетворяют следующим требованиям:

- 4x CPU Core;
- 64 Gb RAM;
- 128 Gb HDD.

2.2 Требования к программному обеспечению

Для обеспечения работоспособности Модуля Polymatica ML, программное обеспечение сервера должно соответствовать следующим требованиям:

- Актуальный дистрибутив Linux (Linux 3.10 и старше);
- Установленный и запущенный docker;
- Установленные python3, openssl, gettext/gettext-base, apache2-utils/httpd-tools;
- Доступ в интернет.

3 АРХИТЕКТУРА

Модуль представляет собой тонкий клиент (thin client) с веб-интерфейсом.

В основу серверной части (Backend) Модуля заложена микросервисная архитектура. На Рисунок 1 приведена целевая функциональная архитектура Модуля.

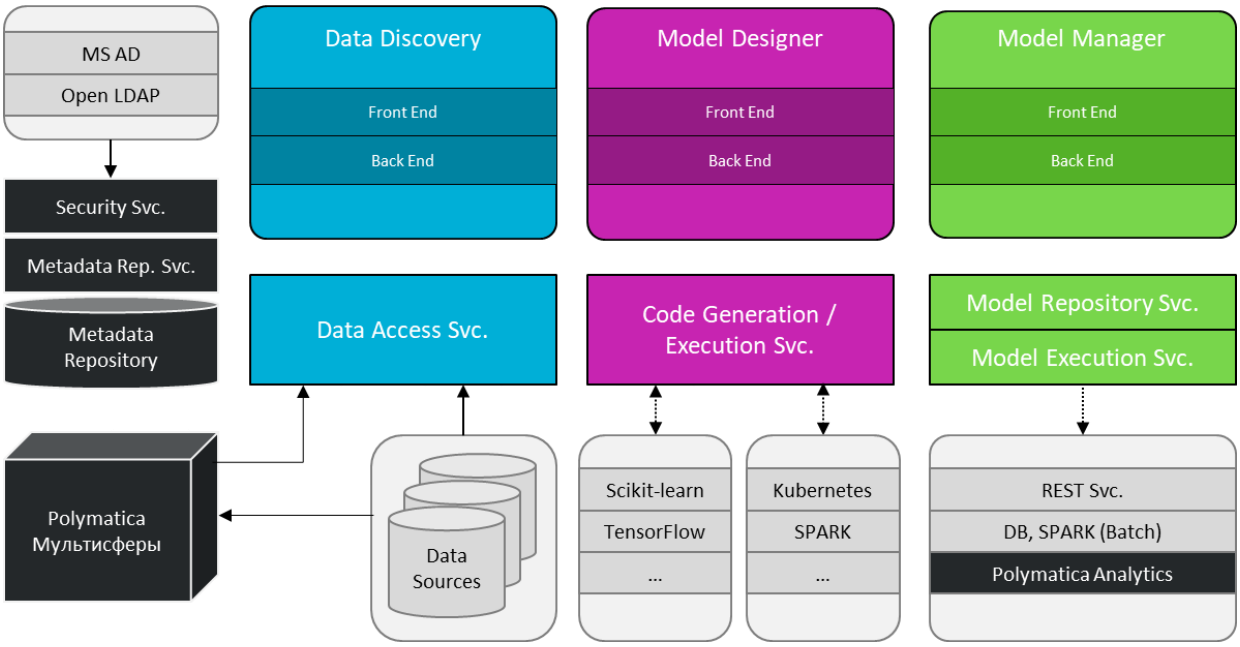


Рисунок 1 - Функциональная архитектура Модуля

Основными компонентами архитектуры являются:

- Интерфейс пользователя (Frontend / Backend). Выделяются следующие подкомпоненты пользовательского интерфейса:
  - Исследование данных (Data Discovery) - пользовательский интерфейс для выбора исследуемых данных и их последующего интерактивного анализа. В текущей версии Модуля обеспечивается поддержка следующих операций:
    - выбор источника данных и формирования выборки;
    - визуализация выборки или ее части;
    - профилирование данных и оценка выборки по основным статистическим критериям;
    - отбор и преобразование признаков;
    - тестирование статистических гипотез о виде распределения непрерывных и дискретных наблюдений.
  - Построение моделей машинного обучения (Model Designer) - пользовательский интерфейс, позволяющий пользователю в графическом режиме в виде последовательности шагов (pipeline) построить модели машинного обучения, провести оценки точности моделей и сохранить модели в репозитории для дальнейшего использования. В текущей версии Модуля обеспечивается поддержка следующих функций:
    - отбор и преобразование признаков.

        - разделение выборки на обучающую, валидационную и проверочную;
        - построение моделей машинного обучения различными методами;
        - автоподбор гиперпараметров модели;
        - кросс-валидация;
        - тестирование моделей;
        - интерпретация моделей;
        - скоринг и оценка качества моделей - расчет, табличная и графическая визуализация различных характеристик модели;
        - выбор лучшей модели по заданному критерию;
        - возможность переобучения моделей при появлении новых данных;
        - сохранение и регистрация моделей в репозитории.
    - Управление жизненным циклом моделей (Model Manager) - интерфейс пользователя для управления репозиторием моделей. Позволяет публиковать модели для применения в пакетном режиме или режиме сервиса, а также настраивать процессы согласования моделей. В текущей версии Модуля обеспечивается поддержка следующих операций:
        - ведение единого репозитория и версионности моделей;
        - публикация моделей для применения в пакетном режиме - формирование скрипта для применения модели на данных в БД;
        - публикация моделей в качестве сервиса;
        - публикация моделей из репозитория в среду Polymatica;
        - сравнение моделей и выбор лучшей;
        - снятие моделей с публикации;
        - переобучение/дообучение моделей из репозитория на новых данных.
- Сервис управления метаданными (Metadata Rep. Svc.) - компонент отвечает за хранение и управление внутренними метаданными модуля, такими как параметры подключения к источникам данных, проекты по построению моделей, параметры для публикации моделей и т.д. Остальные компоненты решения используют сервис управления метаданными для сохранения, чтения, изменения внутренних метаданных.
- Сервис управления безопасностью (Security Svc.) - компонент отвечает за управление правами доступа, интеграцию с каталогами пользователей (Open LDAP, MS ActiveDirectory), аутентификацию и авторизацию пользователей.
- Сервис по доступу к источникам данных (Data Access Svc.) - данный компонент отвечает за доступ к различным источникам и получение из них данных.
- Сервис по генерации и запуску вычислений (Code Generation and Execution Svc.) - данный компонент предназначен для перевода команд в выполняемый код на языке Python. В решении предполагается механизм регистрации различных библиотек Python (scikit-learn и др.), содержащих алгоритмы машинного обучения. При работе с данными в компоненте Data Discovery, а также при создании проекта и построении моделей в компоненте Model Designer - выполняется генерация кода Python в соответствии с заданными настройками.
- Сервис по управлению и публикации моделей (Model Repository Svc.) – компонент содержит процедуры, необходимые для управления жизненным циклом моделей - ведение реестра моделей, процессы согласования, процессы публикации.
- Сервис(ы) исполнения моделей (Model Execution Svc.) — это сервис(ы) исполнения моделей, т.е. каждая опубликованная модель – это исполняемый сервис.

4 ИНСТАЛЛЯЦИЯ МОДУЛЯ

4.1 Установка необходимого ПО

Для работы системы требуется установить СУБД PostgreSQL а также распределенную систему хранения SeaweedFS. Установку рекомендуется выполнять на один или несколько выделенных серверов.

Установка СУБД PostgreSQL выполняется согласно инструкции для соответствующего дистрибутива, расположенной в сети интернет по адресу https://www.postgresql.org/download/.

Последовательность установки SeaweedFS представлена и доступна на ресурсе по адресу https://github.com/chrislusf/seaweedfs/wiki/Getting-Started.

После установки указанного программного обеспечения, для работы системы требуется запустить сервисы ***master, filer, s3, volume***, предварительно сконфигурировав SeaweedFS.

4.2 Конфигурация SeaweedFS

Настроить роль для **S3 API** с полными правами, создав файл ***/etc/seaweedfs/s3-config.json*** со следующим содержимым:

```
{
  "identities": [
   {
      "name": "anonymous",
      "actions": [
        "Read",
        "List"
      ]
    },
    {
      "name": "admin",
      "credentials": [
        {
          "accessKey": "qwerty",
          "secretKey": "qwerty"
        }
      ],
      "actions": [
        "Admin",
        "Read",
        "List",
        "Tagging",
        "Write"
      ]
    }
  ]
}
```

Данную конфигурацию следует указывать при запуске сервиса S3 через параметр ***config***:

```
-config=/etc/seaweedfs/s3-config.json
```

Также необходимо создать необходимые бакеты хранения S3, выполнив на сервере SeaweedFS команды:

```
echo "s3.bucket.create -name mm" | weed shell
echo "s3.bucket.create -name md" | weed shell
echo "s3.bucket.create -name cs" | weed shell
echo "s3.bucket.create -name cnf" | weed shell
```

4.3 Конфигурация Polymatica ML

Необходимо распаковать дистрибутив в целевую папку.

Перед установкой следует указать основные параметры развертывания в файле ***charts/polymatica-ml/values.yaml*** в следующих разделах:

Раздел global

RegistryUrl - адрес для доступа к docker registry в виде <ip-адрес или DNS-имя сервера polymatica ml>:5000

DomainName - ip-адрес или DNS-имя для доступа к серверу

GatewayUrl - адрес для доступа к публикуемым сервисам в виде http://<ip-адрес или DNS-имя сервера polymatica ml>/

MetadataServerUrl - адрес для доступа к сервису метаданных в виде http://<ip-адрес или DNS-имя сервера polymatica ml>

Подраздел postgres

host - адрес или имя хоста PostgreSQL

port - порт PostgreSQL

user - пользователь для подключения к БД PostgreSQL

password - пароль для подключения к БД PostgreSQL

databaseMeta - имя БД сервиса метаданных в PostgreSQL

databaseDD - имя БД сервиса datadiscovery в PostgreSQL

databaseMD - имя БД сервиса modeldesigner в PostgreSQL

Подраздел s3:

url - адрес сервиса S3 API в развернутой SeawedFS в виде http://<ip-адрес или DNS-имя сервера seaweedfs>:<порт>

Раздел modelmanager

Подраздел postgres

user - пользователь для подключения к БД PostgreSQL

password - пароль для подключения к БД PostgreSQL

databaseMM - имя БД сервиса modelmanager в PostgreSQL

Подраздел smtp

username - пользователь для подключения к SMTP-серверу

password - пароль для подключения к SMTP-серверу

from - адрес отправителя

port - порт SMTP-сервера

server - ip-адрес или DNS-имя SMTP-сервера

Подраздел camunda

jdbcUrl - строка подлючения к БД camunda в виде jdbc:postgresql://<ip-адрес или DNS-имя сервера PostgreSQL>:<порт>/cs_dev1?sslmode=disable

jdbcUser - пользователь для подключения к БД PostgreSQL

jdbcPassword - пароль для подключения к БД PostgreSQL

keycloakUrl - адрес для доступа к сервису keycloak в виде http://<ip-адрес или DNS-имя сервера polymatica ml>

Раздел keycloak

Подраздел postgres

user - пользователь для подключения к БД PostgreSQL

password - пароль для подключения к БД PostgreSQL

databaseName - имя БД сервиса keycloak в PostgreSQL

4.4 Требования к серверу

- Актуальный дистрибутив Linux (Linux 3.10 и старше)
- Установленный и запущенный docker
- Доступ в интернет

4.5 Установка Polymatica ML

Установку следует выполнять под непривилегированным пользователем с правами **sudo**.

Пользователь должен быть включен в группу docker.

Для установки запустить ***install.sh*** из каталога с дистрибутивом:

```
DOMAINNAME=<ip-адрес или DNS-имя сервера polymatica-ml>
./install.sh
```

В переменной DOMAINNAME указывается сетевой адрес или DNS-имя сервера, на котором выполняется установка..

5 ИНТЕРФЕЙС

5.1 Авторизация

При открытии Модуля открывается экран авторизации в системе (Рисунок 2).

Рисунок 2 - Окно авторизации

Для входа в систему необходимо ввести Логин и пароль учетной записи.

При корректной авторизации в системе в зависимости от выданных пользователю прав откроется один из компонентов Модуля.

5.2 Основное меню

Основным элементом интерфейса является панель меню, расположенная в верхней части экрана (Рисунок 3). Она неизменна для всех компонентов Модуля.

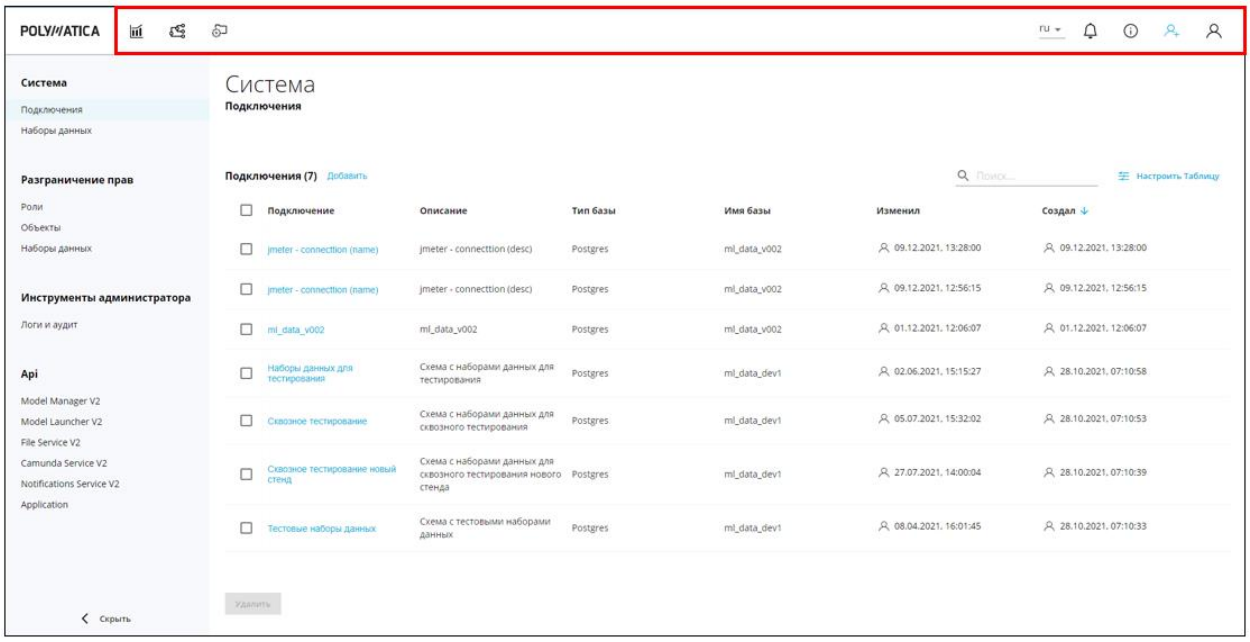


Рисунок 3 – Интерфейс Модуля. Меню

В левой части панели меню расположен объекты, которые позволяют:

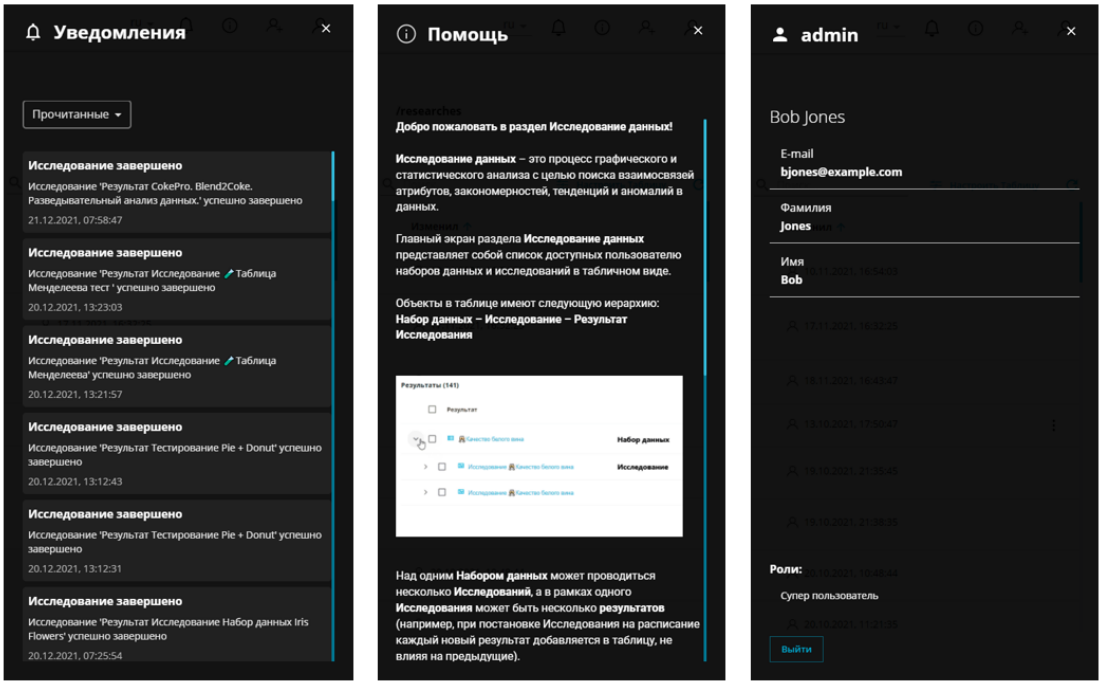
- Выбор иконки  открывает компонент Исследование данных (Data Discovery, DD).
- Выбор иконки  открывает компонент Построение модели (Model Designer, MD).
- Выбор иконки  открывает компонент Управление моделями (Model Manager, MM).

Отображение этих объектов на панели меню зависит от роли и прав пользователя.

В правой части меню расположены объекты, которые позволяют:

- Для выбора языка отображения интерфейса предусмотрен список  с доступными языками.
- Выбор иконки  открывает боковую панель с уведомлениями о завершении расчета результатов Исследований в DD (Рисунок 4 а).
- Выбор иконки  открывает боковую панель с интерактивной справкой (Рисунок 4 б).
- Выбор иконки  открывает раздел Администрирования (подробнее в Руководстве Администратора).

- Выбор иконки открывает боковую панель с информацией об учетной записи пользователя и кнопкой выхода из Модуля (Рисунок 4 в).



а) б) в)

Рисунок 4 – Боковая панель Уведомления (а), боковая панель Помощь (б), боковая панель Пользователь (в)

5.3 Раздел Администрирование

Главный экран раздела **Администрирование** открывается при выборе иконки в правом верхнем меню.

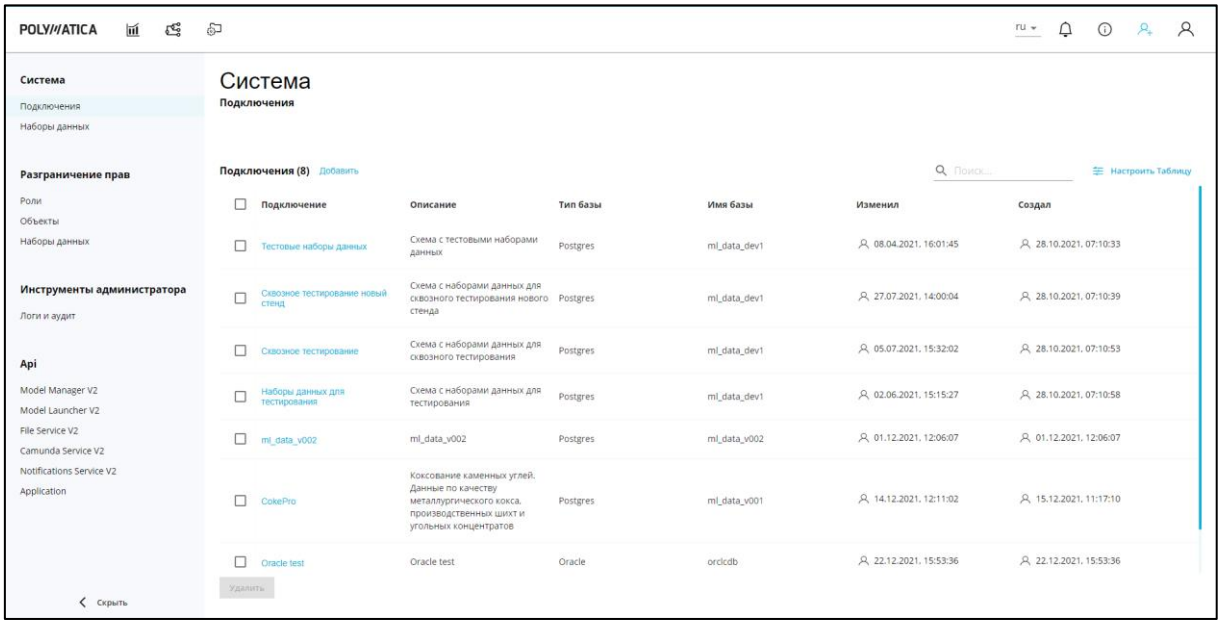
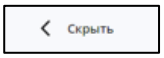


Рисунок 5 – Интерфейс раздела Администрирование

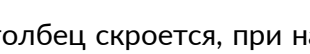
Основным элементом навигации в разделе **Администрирование** является боковая панель. Данная боковая панель содержит следующие подразделы:

- Система:
  - Подключения - данный раздел позволяет создать подключения к внешним источникам данных. На экране в табличном виде должны отображаться список существующих подключений.
  - Наборы данных – данный раздел позволяет настроить подключение к наборам данных, задать пользовательские имена атрибутам и тип, а также создать новые расчетные атрибуты и фильтры.
- Разграничение прав:
  - Роли – данный раздел содержит список ролей и пользователей, которым эти роли заданы.
  - Объекты – данный раздел содержит список объектов и пользователей, которым выдано разрешение на доступ к этим объектам.
  - Наборы данных – данный раздел содержит список наборов данных и пользователей, которым выдано разрешение на доступ к этим наборам данных.
- Инструменты администратора
  - Логи и аудит – данный раздел предназначен для отслеживания состояния компонентов модуля.
- API
  - Model Manager V2.
  - Model Launcher V2.
  - File Service V2.
  - Camunda Service V2.
  - Application.

Для удобства пользования панель можно скрыть при помощи кнопки 

, находящейся в нижней части.

Таблицы в разделе **Управление моделями** имеют гибкие настройки отображения. Так, пользователь может:

- Изменить ширину любого столбца (для этого необходимо перетащить границу его заголовка до нужной ширины).
- Сортировать таблицу (для этого необходимо выбрать иконку рядом с заголовком сортируемого столбца).
- Скрывать/отображать столбцы и изменять их порядок в окне **Вид таблицы** (для открытия окна необходимо выбрать 

в правом верхнем углу таблицы; при выборе иконки  столбец скроется, при наведении на иконку  активируется возможность перемещения столбца).
- Сбросить внесенные изменения также в окне **Вид таблицы** (для этого выбрать кнопку **Сбросить**).

Для быстрого поиска объекта в таблице предусмотрено поле 

в правой верхней части таблицы.

6 АДМИНИСТРИРОВАНИЕ МОДУЛЯ

6.1 Подключения и наборы данных

Для расчета Пользователем исследования данных, построения сценария моделирования и публикации моделей необходимо подключить набор данных к системе.

6.1.1 Подключение к внешнему источнику

Для подключения набора данных в первую очередь необходимо настроить Подключение к внешнему источнику. Для этого необходимо:

- Перейти в раздел **Подключения** (Рисунок 6) боковой панели и в верхней части таблицы со списком существующих подключений выбрать **Добавить.**
- В открывшемся окне **Подключение** задать следующие параметры:
    - Тип базы – выбирается из выпадающего списка при помощи кнопки в правой части поля ввода. Для выбора доступны два типа: Postgres и Oracle.
    - Наименование подключения – ввод с клавиатуры,
    - Описание – ввод с клавиатуры,
    - Хост,
    - Имя базы,
    - Порт,
    - Имя Пользователя,
    - Пароль,
    - cache_timeout.
- Внизу списка параметров также нужно выбрать дополнительные опции, установив метку напротив требуемых пунктов.
    - allow_ctas – разрешить инструкцию CREATE TABLE AS SELECT (распараллеленная операция, которая создает новую таблицу на основе выходных данных инструкции SELECT),
    - async_query_execution – выполнение асинхронного запроса,
    - allow_create_table – разрешить создание таблицы,
    - allow_create_view – разрешить создание представления,
    - allow_dml – разрешить DML (DML – это язык манипулирования данными, который используется для хранения, изменения, извлечения, удаления и обновления данных в базе данных).
- После задания всех параметров выбрать **Проверить подключение**. Если все параметры указаны верно и источник доступен, появится сообщение об успешном соединении. Это значит, что подключение создано и можно перейти к следующему шагу – регистрации Набора данных

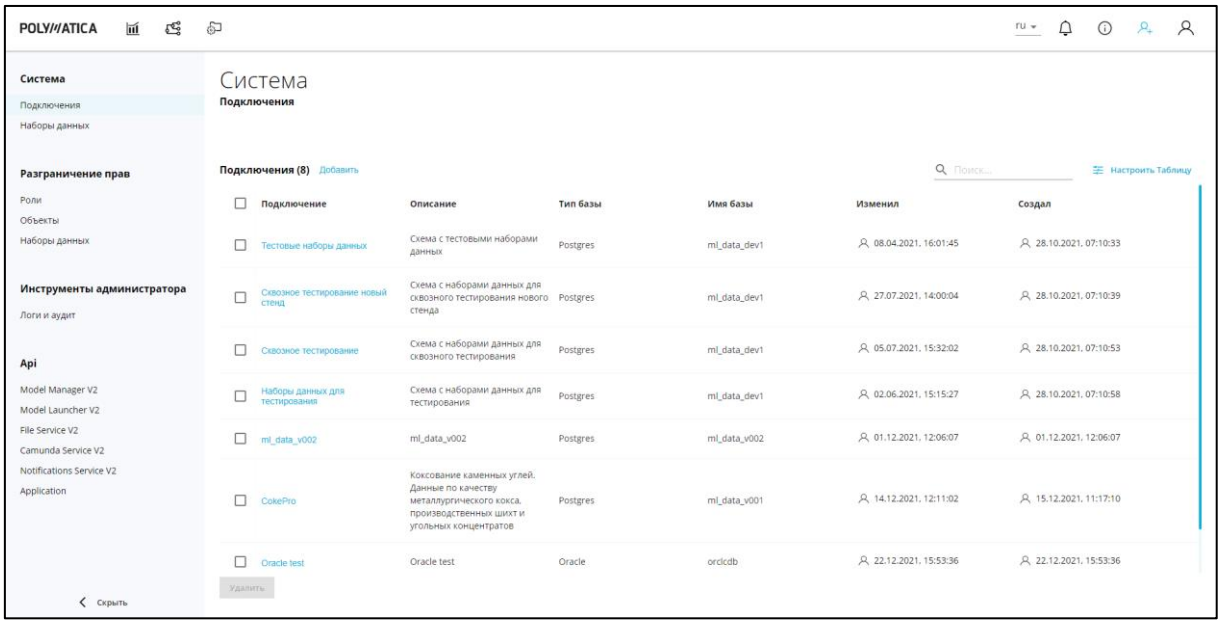


Рисунок 6 – Интерфейс раздела Подключения

6.1.2 Регистрация набора данных

Для регистрации Набора данных необходимо:

- Перейти в раздел **Наборы данных** боковой панели (Рисунок 7) и в верхней части таблицы со списком существующих подключений выбрать **Добавить,**
- В открывшемся окне **Регистрация набора данных** задать следующие параметры:
  - Название,
  - Описание,
  - Подключение – выбирается из выпадающего списка (кнопка в правой части поля ввода),
  - Таблица – выбирается из выпадающего списка при помощи кнопки в правой части поля ввода, после заполнения поля Подключение. Для каждого Подключения свой перечень таблиц.
- После заполнения всех полей нажать на кнопку **«Сохранить»**. Кнопка **«Отменить»** позволяет выйти из окна настройки, не внося изменений, аналогично нажатию на стандартный элемент управления в правом верхнем углу окна **Регистрация набора данных** (элемент).

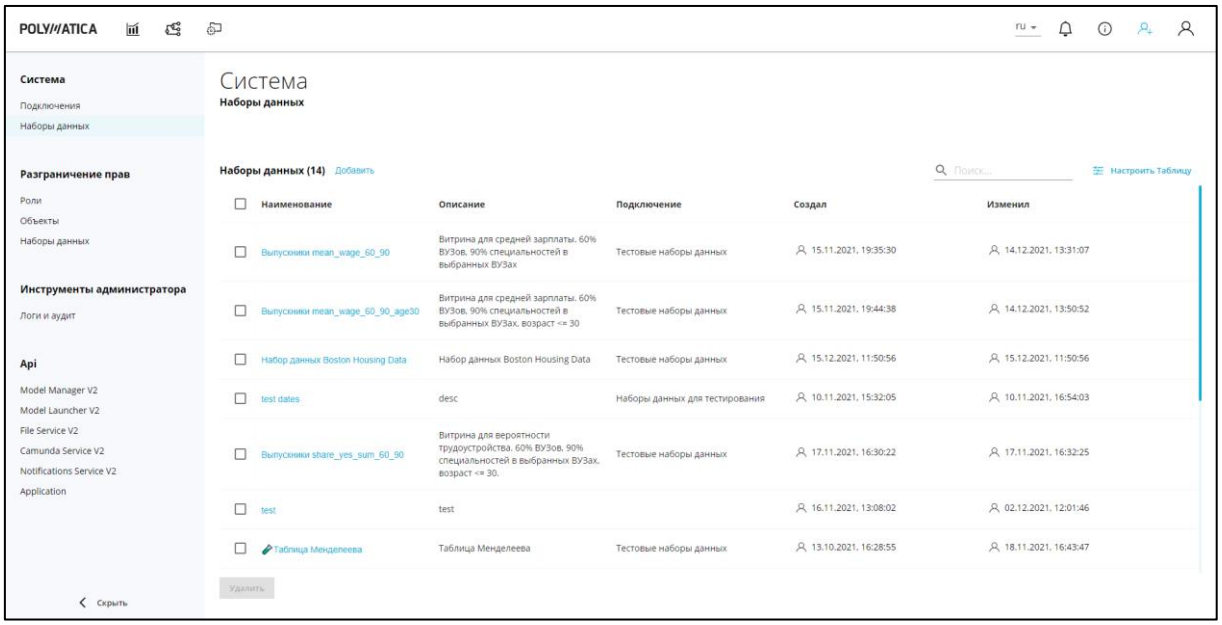


Рисунок 7 – Интерфейс раздела Наборы данных

После регистрации **Набора данных** можно ознакомиться с ним. Для этого нужно выбрать интересующий набор данных из списка. В открывшемся окне **Сведения о наборе данных** (Рисунок 8) отображаются следующие вкладки:

- Вкладка **Информация** содержит сведения о наборе данных – имя таблицы в источнике, пользовательское имя таблицы (в системе) и описание,
- Вкладка **Атрибуты** позволяет задать пользовательское имя атрибута,
- Вкладка **Расчетные атрибуты** позволяет создать новые атрибуты,
- Вкладка **Фильтр** позволяет отфильтровать набор данных. Отфильтрованные (не соответствующие критериям фильтрации) данные не будут использоваться далее при работе.
- Вкладка **Пример Данных** позволяет ознакомиться с данными.

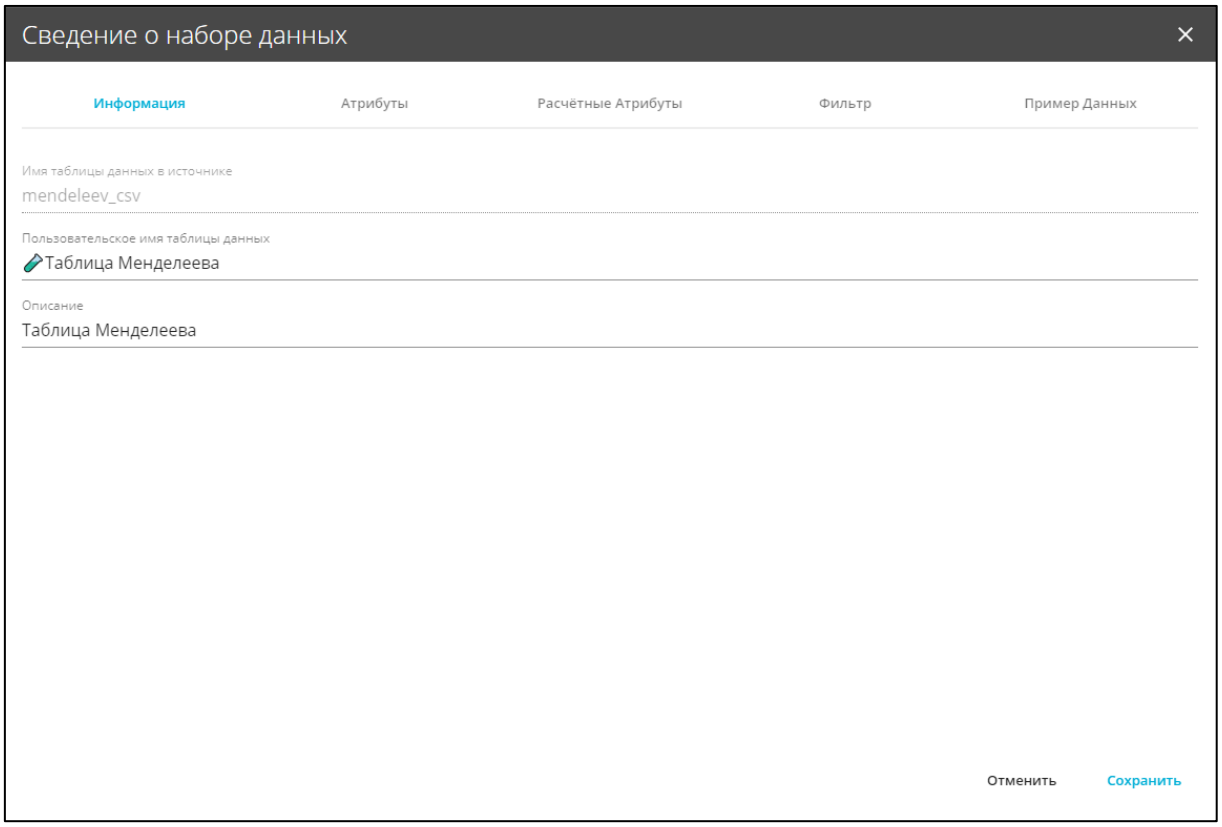


Рисунок 8 – Окно Сведение о наборе данных

6.1.3 Фильтрация набора данных

Для задания критериев фильтрации во вкладке **Фильтр** необходимо выбрать иконку рядом с полем для ввода формулы. В результате откроется окно **Формула** (Рисунок 9).

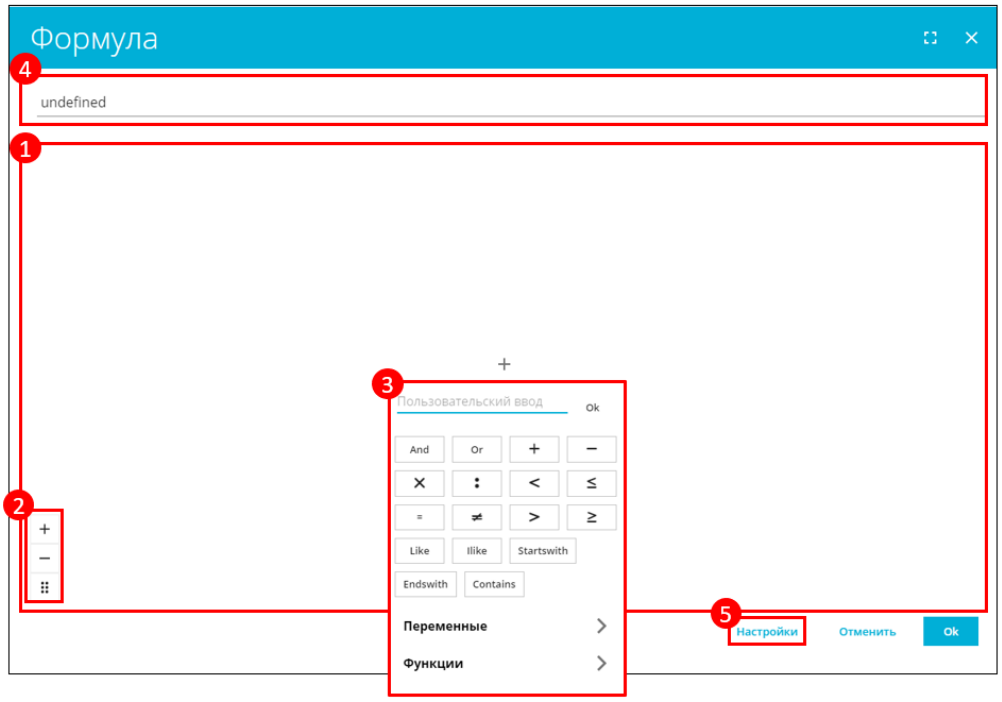


Рисунок 9 – Окно Формула

Основными элементами окна **Формула** являются:

1. Рабочее поле, в центре которого расположена иконка +
2. Панель с настройками рабочего поля, где предусмотрены кнопки масштабирования
3. Вкладка, которая открывается при выборе иконки + в центре рабочего поля и включает в себя:
    - Поле пользовательского ввода – позволяет ввести числовые (в качестве десятичного разделитель используется запятая) и строковые значения (для ввода строкового значения необходимо обернуть его в кавычки)
    - Переменные из набора данных
    - Функции, представленные в Таблица 1
4. Строка, в которой дублируется линейное текстовое представление вводимой формулы
5. Меню с настройками отображения формулы в рабочем поле

Таблица 1 – Набор функций узла «Фильтр»

Функция	Описание
And	Логическая операция «И»
Or	Логическая операция «ИЛИ»
+	Операция сложения
-	Операция вычитания
×	Операция умножения
:	Операция деления
<	Меньше
≤	Меньше или равно
>	Больше
≥	Больше или равно
=	Равно
≠	Не равно
Like	Проверяет, удовлетворяет ли символьная строка заданному образцу, который может содержать поисковые символы. Учитывает регистр
illike	Проверяет, удовлетворяет ли символьная строка заданному образцу, который может содержать поисковые символы.
Startswith	Проверяет, есть ли в начале одной текстовой строки другая текстовая строка

Функция	Описание
Endswith	Проверяет, есть ли в конце одной текстовой строки другая текстовая строка
Contains	Проверяет, есть ли совпадения с отдельными словами или фразами
create_time(Number, Number, Number)	Создает переменную времени из последовательно введенных часов, минут и секунд
concat(String)	Объединяет в единую строковую переменную из нескольких строковых переменных
char_lenght(String)	Возвращает длину строковой переменной
random()	Возвращает случайное число
ln(Number)	Натуральный логарифм
exp(Number)	Экспонента
power(Number, Number)	Возведение в степень
sqrt(Number)	Квадратный корень
abs(Number)	Абсолютное значение
ceil(Number)	Возвращает наименьшее целое число, которое больше или равно текущему значению
floor(Number)	Возвращает наибольшее целое число, которое меньше или равно текущему значению
extract_from_datetime(String, Datetime)	Получает указанную часть (день, месяц, год, часы, минуты и т. д.) из значения даты и времени
extract_from_time(String, Time)	Получает указанную часть (часы, минуты, секунды) из значения времени
extract_from_date(String, Date)	Получает указанную часть (день, месяц, год) из значения даты
current_date()	Возвращает текущую дату
current_time()	Возвращает текущее время
current_timestamp()	Возвращает текущие дату и время
now()	Возвращает текущие дату и время

Функция	Описание
create_datetime()	Создает переменную даты времени из последовательно введенных даты месяца года часов минут и секунд
create_date ()	Создает переменную даты из последовательно введенных даты месяца и года
not(Boolean)	Задает противоположное условие
between(Boolean)	Проверяет, входит ли значение в определённый диапазон
In(Any, Any)	Используются для сравнения проверяемого значения поля с заданным списком
not_in(Any, Any)	Используются для сравнения проверяемого значения поля с заданным списком
coalesce(Any)	Возвращает первое значение из списка

6.1.4 Создание новых расчетных атрибутов

Для создания нового **расчетного атрибута** в Наборе данных необходимо:

- Во вкладка **Расчетные атрибуты** (Рисунок 10) выбрать иконку +
- Задать следующие параметры:
  - Пользовательское имя,
  - Тип данных,
  - Формулу (подробнее раздел 6.4).
- Сохранить изменения, выбрав иконку .

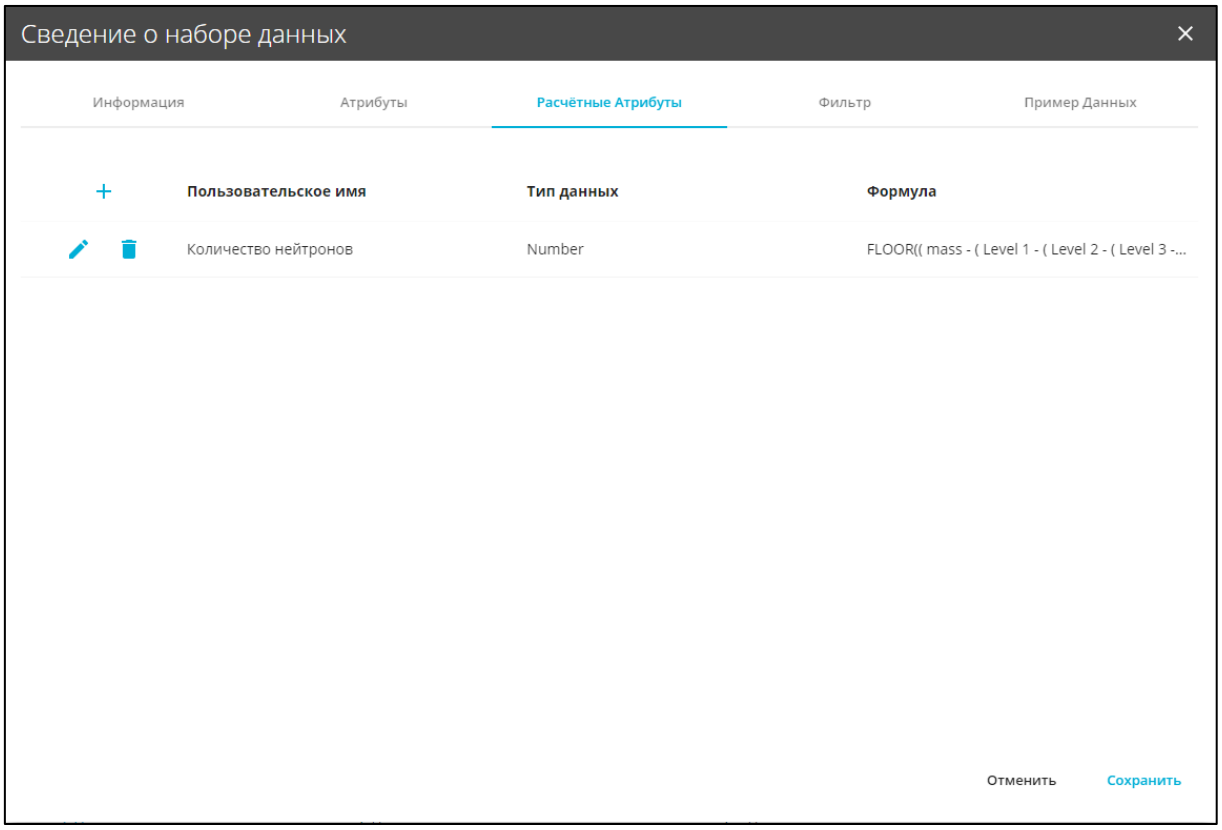


Рисунок 10 – Вкладка Расчётные атрибуты

6.2 Разграничение прав

Для каждого пользователя необходимо указать роль, в соответствии с которой предоставляются или ограничиваются права данного пользователя для взаимодействия с Модулем.

В Модуле реализовано разграничение прав доступа:

- На уровне функций системы;
- На уровне объектов системы;
- На уровне данных.

6.2.1 Назначение ролей

В зависимости от выданных пользователю прав в интерфейсе отображаются соответствующие компоненты. Так

- **Аналитик** имеет доступ к функционалу компонентов Исследование данных (DD) и Построение моделей (MD).
- **Менеджер моделей** имеет доступ ко всему функционалу компонента Управление моделями (MM).
- **Пользователь моделей** имеет доступ к разделу Опубликованные модели MM.
- **Согласующий** имеет доступ к разделу Согласование MM.
- **Суперпользователь** имеет полный доступ к функционалу Модуля.
- **Системный администратор** имеет доступ только к разделу Администрирование.

Для задания роли пользователю необходимо:

- Перейти в раздел **Роли** бокового меню (Рисунок 11);

- Выбрать интересующую роль;
- В открывшемся окне **Управление ролью** нажать на кнопку **Добавить** и рядом с интересующими пользователями/группами пользователей выбрать флажок;
- После выбора необходимых пользователей сохранить изменения, выбрав кнопку **Добавить.**

Чтобы удалить пользователя/группу пользователей в окне **Управление ролью** необходимо отметить требуемые строки в поле слева и нажать кнопку **«Удалить»**.

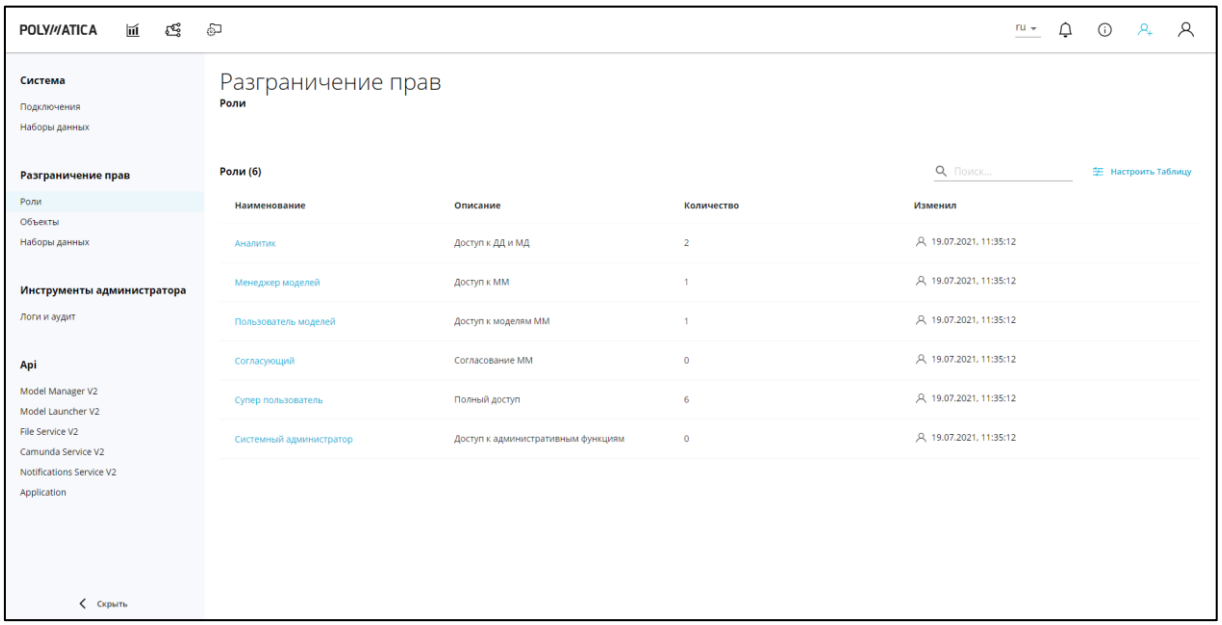


Рисунок 11 – Раздел Роли

6.2.2 Доступ к объектам

Раздел **Объекты** бокового меню позволяет назначить права доступа к проектам разделов Построение моделей (MD) и Управление моделями (MM).

На экране раздела в табличном виде будут отображены наименования объектов, их описание, тип, дата создания и изменения записи (Рисунок 12).

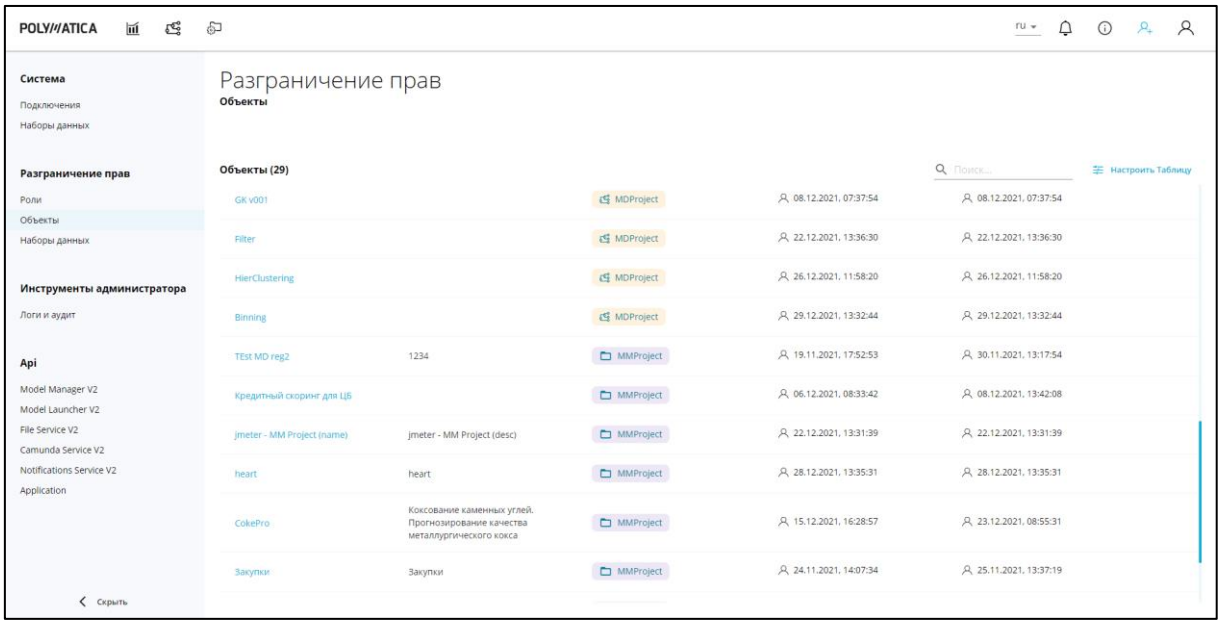


Рисунок 12 – Раздел Объекты

Для разрешения доступа к объекту необходимо:

- щелкнуть по наименованию объекта и перейти в окно **Управление объектами**.
- В открывшемся окне выбрать кнопку **Добавить.**
- Далее отметить требуемых пользователей/группы пользователей выбрав флажок рядом с наименованием и нажать на кнопку **Добавить.**
- В окне **Управление объектами** назначить пользователям/группам права на **Чтение**, **Обновление** и(или) **Удаление** объекта, отметив меткой требуемые права и нажать на кнопку **«Ок»**.

Для удаления права доступа к объекту необходимо:

- щелкнуть по наименованию объекта и перейти в окно **Управление объектами**.
- В окне **Управление объектами** пользователям/группам убрать права на **Чтение**, **Обновление** и(или) **Удаление** объекта, убрав флажок напротив, и нажать на кнопку **«Ок»**.
- Для удаления всех прав необходимо нажать на значок около выбранного пользователя/группы.

Если у Пользователя нет никаких прав на объект, то он не увидит его в соответствующих разделах.

Владельцу объекта (тому, кто его создал) автоматически назначаются полные права доступа.

6.2.3 Доступ к наборам данных

Раздел **Наборы данных** бокового меню позволяет назначить права доступа к наборам данных (Рисунок 13).

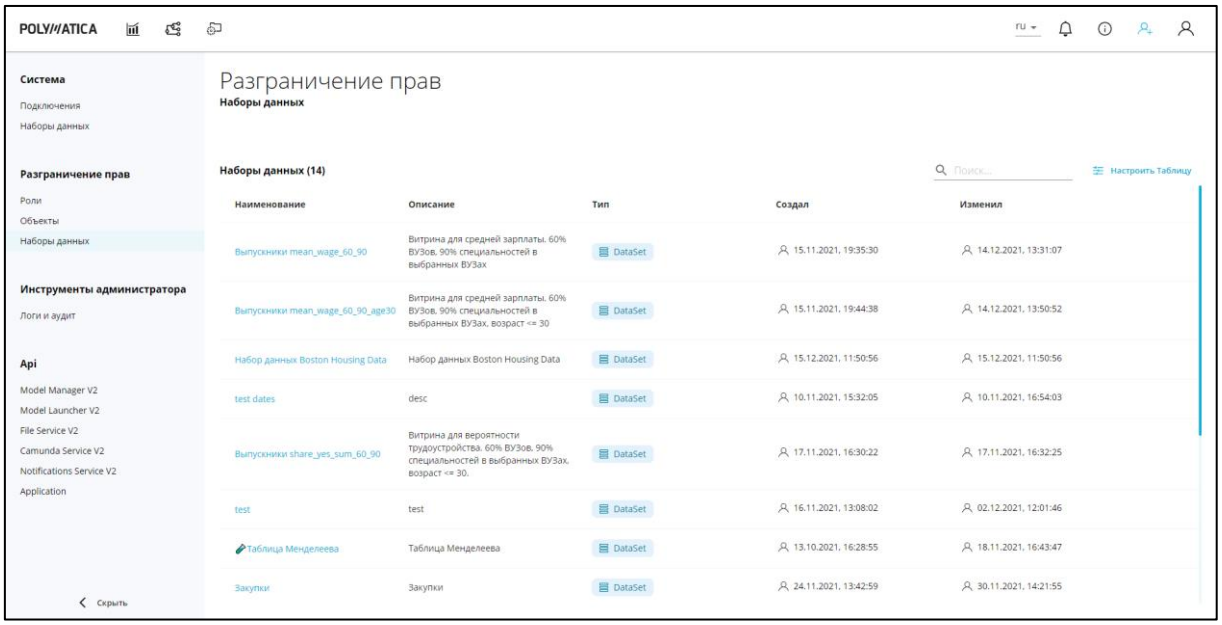


Рисунок 13 – Раздел Наборы данных Разграничение прав

Для разрешения доступа к набору данных необходимо:

- щелкнуть по наименованию набора данных и перейти в окно **Управление объектами**.
- В открывшемся окне выбрать кнопку **Добавить.**
- Далее отметить требуемых пользователей/группы пользователей выбрав флажок рядом с наименованием и нажать на кнопку **Добавить.**
- В окне **Управление объектами** назначить пользователям/группам права на **Чтение**, **Обновление** и(или) **Удаление** объекта, отметив меткой требуемые права и нажать на кнопку **«Ок»**.

Для удаления права доступа к объекту необходимо:

- щелкнуть по наименованию объекта и перейти в окно **Управление объектами**.
- В окне **Управление объектами** пользователям/группам убрать права на **Чтение**, **Обновление** и(или) **Удаление** объекта, убрав флажок напротив, и нажать на кнопку **«Ок»**.
- Для удаления всех прав необходимо нажать на значок около выбранного пользователя/группы.

Если у Пользователя нет никаких прав на набор данных, то он не увидит его в разделе Исследование данных (DD) и не сможет строить по нему модели в разделе Построение моделей (MD).

6.3 Работа с API

В интерфейс сервисов можно попасть при переходе в соответствующий сервис раздела **API** бокового меню. **API** предоставляет возможность тестировать функции, которые используются фронтендом для работы. Описание каждой функции указано в сваггере (Рисунок 14).

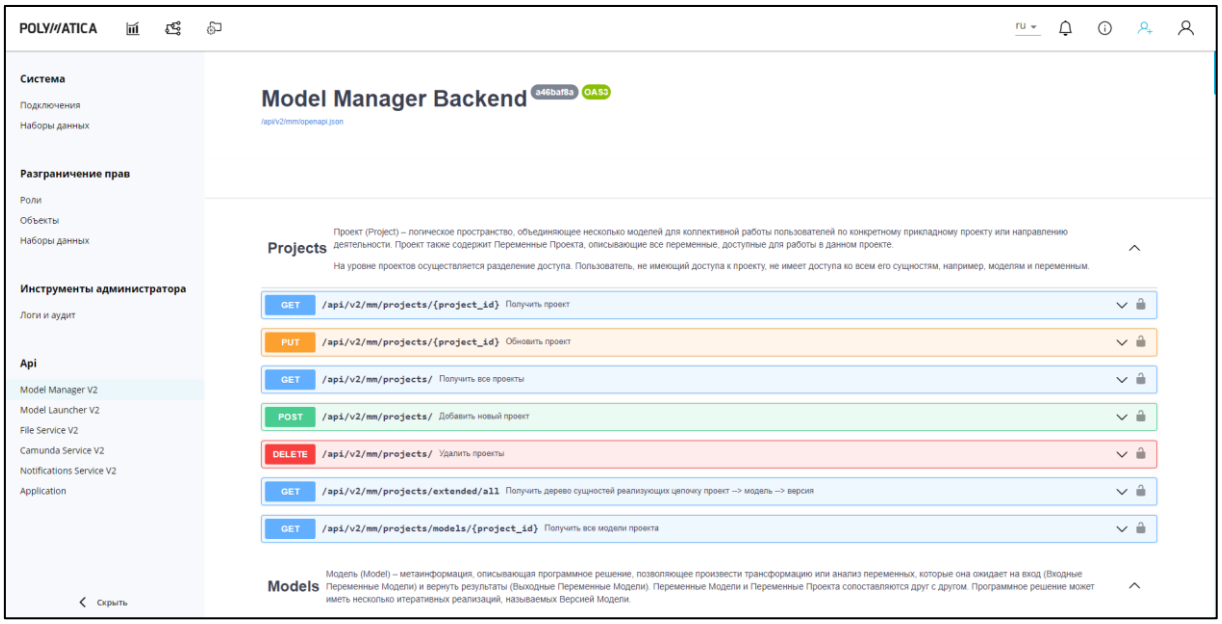


Рисунок 14 – Раздел API.Model Manager V2

6.3.1 Model Manager V2

API Model Manager V2 включает в себя HTTP запросы GET, PUT, POST, DELETE и PATCH связанные с:

- **Projects**

  Проект (Project) – логическое пространство, объединяющее несколько моделей для коллективной работы пользователей по конкретному прикладному проекту или направлению деятельности. Проект также содержит Переменные Проекта, описывающие все переменные, доступные для работы в данном проекте.

  На уровне проектов осуществляется разделение доступа. Пользователь, не имеющий доступа к проекту, не имеет доступа ко всем его сущностям, например моделям и переменным.

- **Models**

  Модель (Model) – метаинформация, описывающая программное решение, позволяющее произвести трансформацию или анализ переменных, которые она ожидает на вход (Входные Переменные Модели) и вернуть результаты (Выходные Переменные Модели). Переменные Модели и Переменные Проекта сопоставляются друг с другом. Программное решение может иметь несколько итеративных реализаций, называемых Версией Модели.

- **ModelVersions**

  Версия Модели (Model Version) - реализация заданной Модели в виде программ архива с произвольным кодом. Содержит обязательную функцию скоринга и несколько опциональных функций. При загрузке в Model Manager архив трансформируется в готовый к запуску код или docker-контейнер. Для запуска конкретного Источника Данных, собранный код параметров запуска при Конфигурации Экземпляра Модели.

- **Libraries**

  Библиотека (Library) – библиотека Python, которую использует версия модели.

- **InstanceConfigs**

  Конфигурация Экземпляра Модели (Instance Config) - описание параметров сборки и запуска Версии Модели. Например, в виде сервиса или пакетного обработчика, с использованием определенных аппаратных ресурсов (количество ядер процессора, объем оперативной памяти), с определенным приоритетом, подключением определенных ETL-функций и метрик. После сборки Версия Модели называется Экземпляром (Instance).

- **Metrics**

  Метрика (Metric) – метод расчета характеристики модели (Feature).

- **Approvements**

  Согласование (Approvement) – утверждение параметров другими пользователями. Например, версия модели изначально имеет статус «черновик», позволяющий редактировать ее параметры. После назначения параметров версию можно отправить на согласование другим пользователям. В результате согласования параметры версии модели становятся окончательными. Если в согласовании будет отказано, параметры можно будет изменить снова. Аналогично процесс согласования реализован для конфигураций экземпляров и ETL-функций.

- **ETL Functions**

  ETL-функция (ETL function) – произвольный пользовательский код, который можно добавить при запуске модели для предварительной или пост-обработки. Созданная ETL-функция добавляется к модели с помощью Конфигурации Экземпляра Модели (Instance Config).

- **Features**

  Функции – Характеристика версии модели (Feature) показывает, насколько хорошо модель работает. Характеристики рассчитываются с помощью метрик.

- **Variables**

  Переменные – Входные данные модели представляют собой набор переменных (Variables типа input), хранящихся в таблице реляционной базы данных. Результат работы модели - выходные переменные (Variables типа output).

- **VariablesMappings** – сопоставление переменных проекта и переменных модели.
- **File Operations** – подтверждение загрузки данных для объекта.
- **Mappings** – сопоставление переменных
- **Stats** – статистика для графиков
- **Status** – Статус

6.3.2 Model Launcher V2

API Model Launcher V2 включает в себя HTTP запросы GET, PUT, POST, DELETE и PATCH связанные с:

- **RunningInstances**

  Экземпляр Модели (Instance) – Python-приложение, которое было собрано Model Builder на основе исходного кода из архива с версией модели. Вид приложения

определяется в Конфигурации Экземпляра модели. Запущенный экземпляр модели называется Running Instance.

- **BuildJobs** – сборка модели
- **Status** – Статус.

6.3.3 File Service V2

API File service V2 включает в себя HTTP запросы GET, PUT, POST, DELETE и PATCH связанные с:

- **ModelVersions**

  Версия Модели (Model Version) - реализация заданной Модели в виде программ архива с произвольным кодом. Содержит обязательную функцию скоринга и несколько опциональных функций. При загрузке в Model Manager архив трансформируется в готовый к запуску код или docker-контейнер. Для запуска конкретного Источника Данных, собранный код параметров запуска при Конфигурации Экземпляра Модели. Хранятся в S3 как model.zip.

- **ETL functions**

  ETL-функция (ETL function) – произвольный пользовательский код, который можно добавить при запуске модели для предварительной или пост-обработки.

- **RenderredInstances**

  Экземпляры моделей (экземпляры), собранные в виде скрипт для запуска в сторонних средах (не через Model Launcher). Хранятся в S3 как rendered_model.zip.

- **Dependency Cache**

  Списки Python-зависимостей экземпляров моделей (instances) в формате poetry.lock. Идентифицируются по checksum файла pyproject.toml. Кэширование позволяет ускорить сборку.

- **Status** – Статус.

6.3.4 Camunda Service V2

API Camunda Service V2 включает в себя HTTP запросы GET, PUT, POST, DELETE и PATCH связанные с:

- **Camunda**

  Согласование (Approvement) – утверждение параметров другими пользователями. Например, версия модели изначально имеет статус «черновик», позволяющий редактировать ее параметры. После назначения параметров версию можно отправить на согласование другим пользователям. В результате согласования параметры версии модели становятся окончательными. Если в согласовании будет отказано, параметры можно будет изменить снова. Аналогично процесс согласования реализован для конфигураций экземпляров и ETL-функций.

  Процесс согласования задается шаблоном Camunda.

- **CamundaNotifier** – Уведомление участников процесса согласования по электронной почте. При запросе к эндпойнту Camunda Service находит адресатов,

заполняет шаблон письма, затем передает его в Notifications Service для непосредственной отправки.

- **Status** – Статус.

6.3.5 Notifications Service

API Notifications Service V2 включает в себя HTTP запросы GET, PUT, POST, DELETE и PATCH связанные с:

- **email**

  Отправка электронных писем. Письма формируются Camunda Service для оповещения участников процесса согласования.

- **Status** – Статус

6.3.6 Application

API Applications включает в себя HTTP запросы GET, PUT, POST, DELETE и PATCH связанные с:

- **metadata** – метаданные.
- **envmgr** – полученные данные из файлов настройки.
- **dd** – компонент Исследование данных (DD).
- **md** – компонент Построение модели (MD).
- **notifications** – уведомления.

6.4 Создание шаблонов согласования

Создание шаблонов реализуется инструментами **Camunda Modeler** (Рисунок 15), в подпространстве BPMN Diagram (Camunda Platform/Cloud).

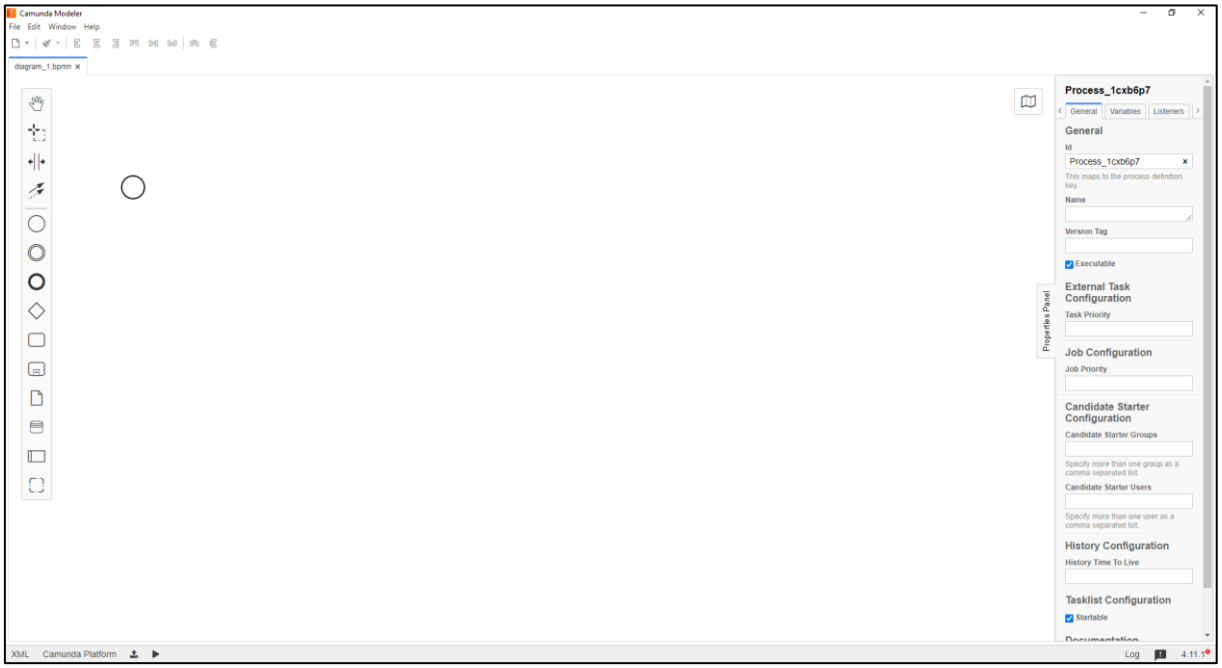


Рисунок 15 – Интерфейс Camunda Modeler

Поскольку с процессом согласования обычно работают больше, чем один человек, необходимо создавать блок **Pool/Participant** (Рисунок 16).

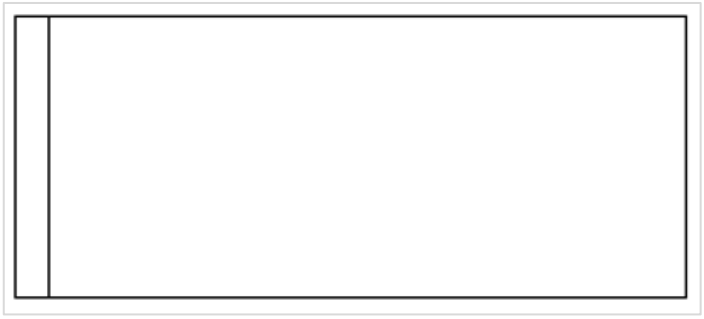


Рисунок 16 – Блок Pool/Participant

Предварительно необходимо удалить с диаграммы все созданные автоматически элементы.

Внутри диаграммы необходимо создать блоки, разделяющие роли (Рисунок 17, Рисунок 18).

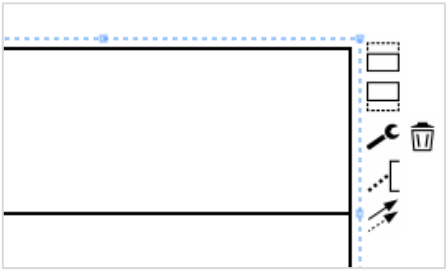


Рисунок 17 – Кнопка создания блока

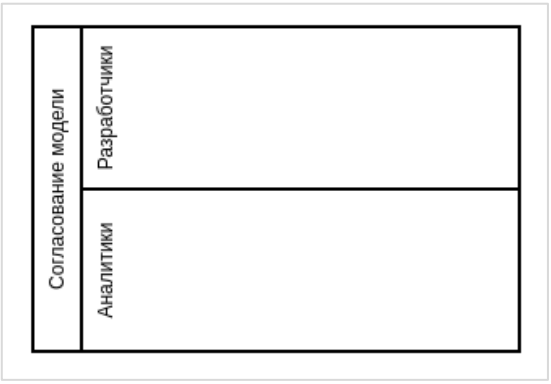


Рисунок 18 – Блоки, разделяющие роли

Необходимо заполнить поля процесса внутри **Participant block (**Рисунок 19**)**:

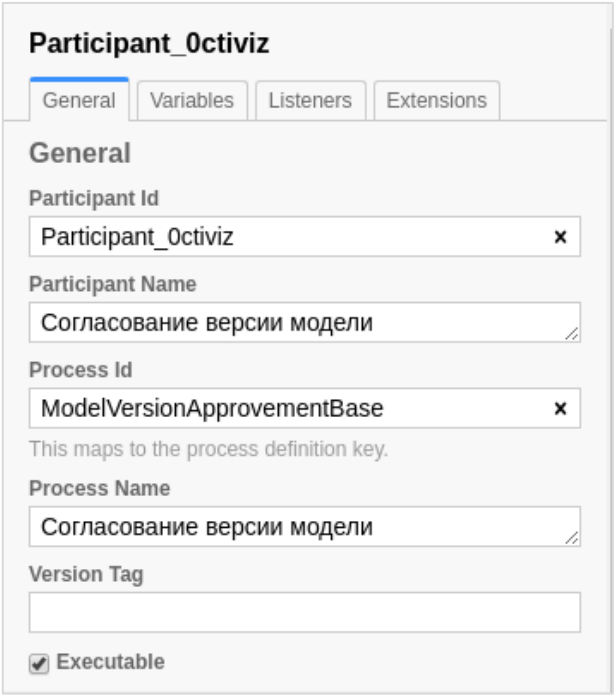


Рисунок 19 – Настройки Participant block

Процесс будет запускаться по Process Id, который по совместительству будет ключом этого Process Defenition. На рабочем процессе **всегда** должна стоять метка Executable.

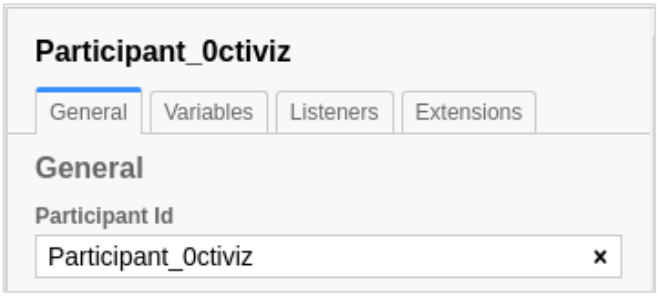


Рисунок 20 – Настройка Participant block

Далее необходимо выбрать вкладку **Extensions**, в которую будет записано значение процесса согласование, то есть к какому типа согласования данный процесс относится.

Всего существует три типа процессов согласования:

- **model_version**,
- **etl_function**,
- **instance_config**.

Тип процесса должен быть обязательно записан в Properties с именем **process_type**.

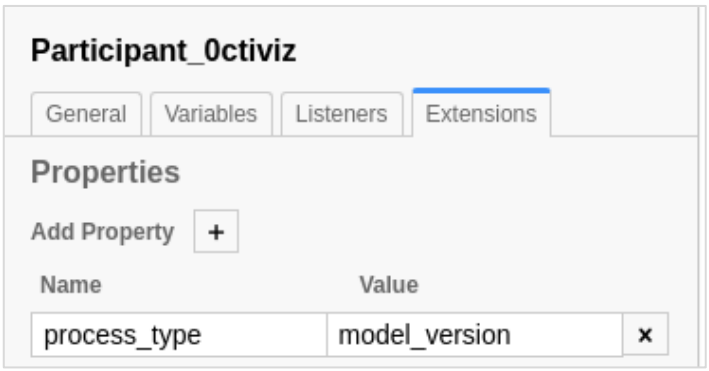


Рисунок 21 – Настройка Participant block

Далее необходимо задать стартовую точку процесса, именуемую как StartEvent (Рисунок 22).

Рисунок 22 – Стартовая точка StartEvent

После, в зависимости от задуманного процесса согласования, необходимо создать схему в BPMN нотации.

Все диаграммы обязательно должны заканчиваться объектом SignalEndEvent (Рисунок 23).

Рисунок 23 – Объект SignalEndEvent

На текущий момент количество вариантов согласования равно двум – элемент «согласован»/«не согласован» (Рисунок 24):

Рисунок 24 – Элемент согласован/не согласован

Элементы типа «не согласован» должны иметь заполненные поля с Global Signal Name (Рисунок 25).

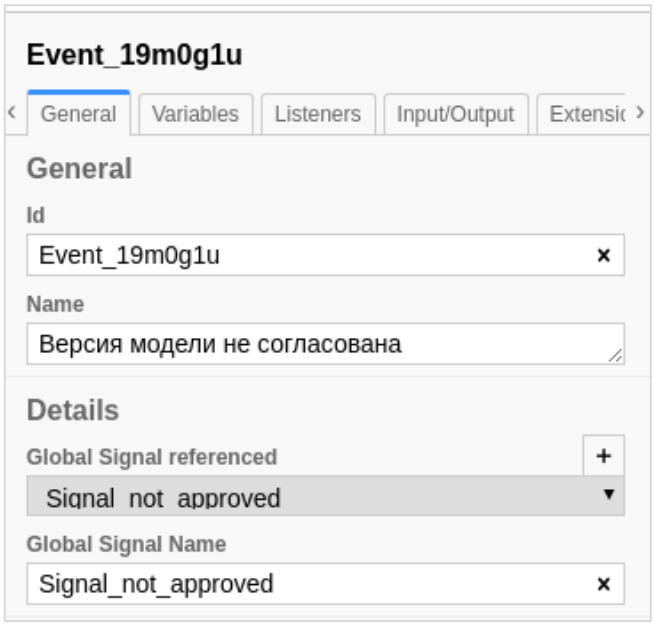


Рисунок 25 – Заполнение полей элемента типа «не согласован»

Элементы типа «согласован» заполняется так, как показано на рисунке (Рисунок 26):

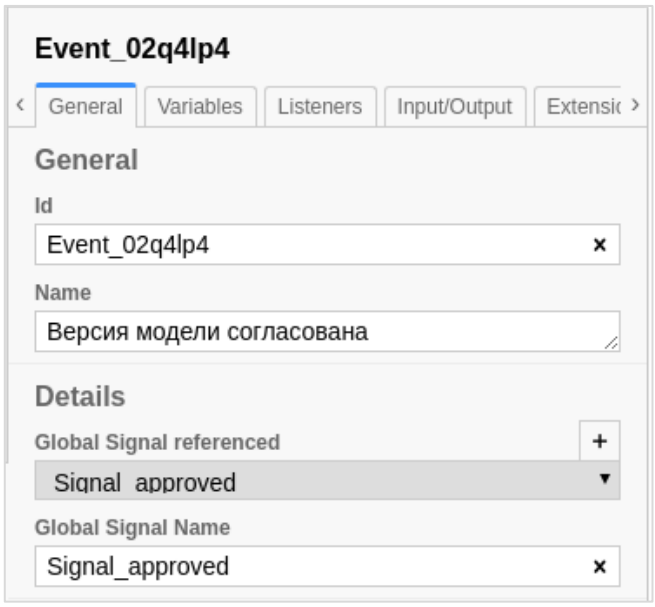


Рисунок 26 – Заполнение полей элемента типа «согласован»

Никаких других имен сигналов быть **НЕ ДОЛЖНО!**

Каждый конечный элемент должен иметь заполненные переменные маппинга (Рисунок 27).

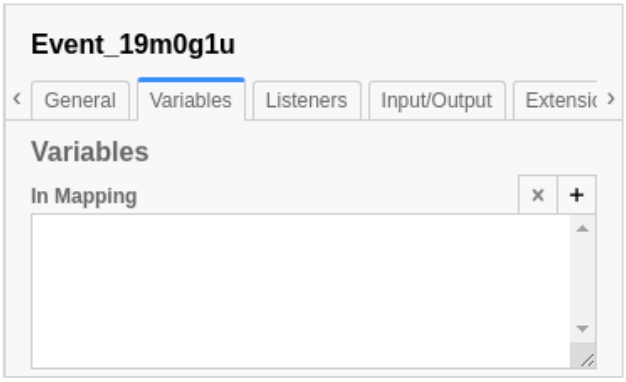


Рисунок 27 – Поле для заполнения маппинга

Переменных может быть любое количество, однако существует три обязательных поля. Два поля для каждого элемента одинаковы. Это переменные: **instance_id** , **instance_type**. Также существует переменная **status**.

Пример заполнения **instance_id** (Рисунок 28)

instance_id определяется как id элемента, которого мы согласуем. В данном случае это процесс согласования версии модели.

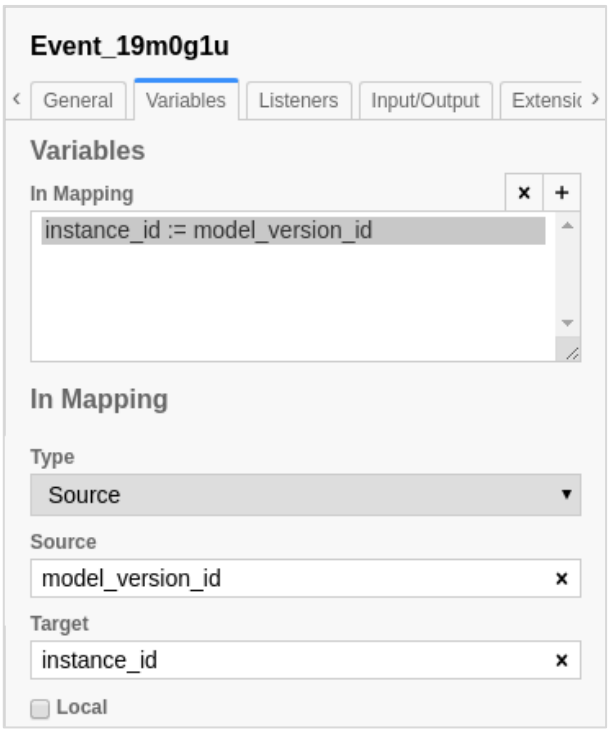


Рисунок 28 – Пример заполнения instance_id

Поле **Type** должно быть описано как **Source**, потому что создаётся связь с переменной, которая уже существует.

Пример заполнения **instance_type** (Рисунок 29)

instance_type указывает на тип сущности согласования.

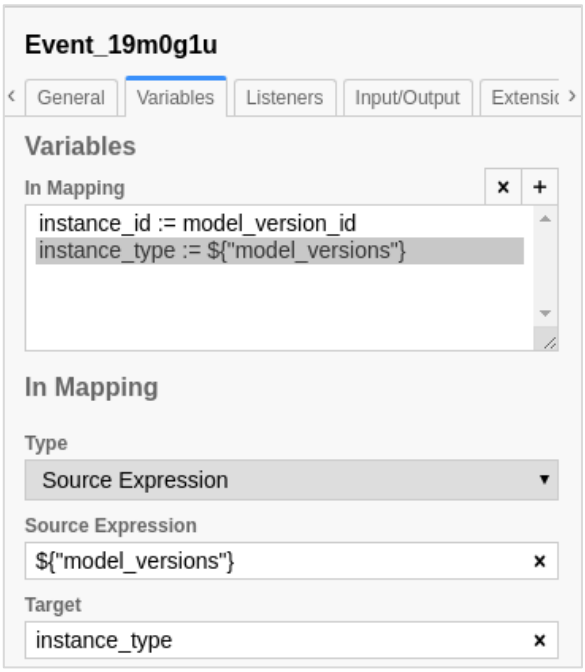


Рисунок 29 – Пример заполнения instance_type

Поле **Type** выбрано как **Source Expression**, потому что в переменную записывается новое значение (прим. **Camunda Expressions**).

Пример заполнения **status** (Рисунок 30)

У элемента сигнала типа «не согласовано» данная переменная имеет значение «-1», соответственно у типа «согласовано» значение – «1».

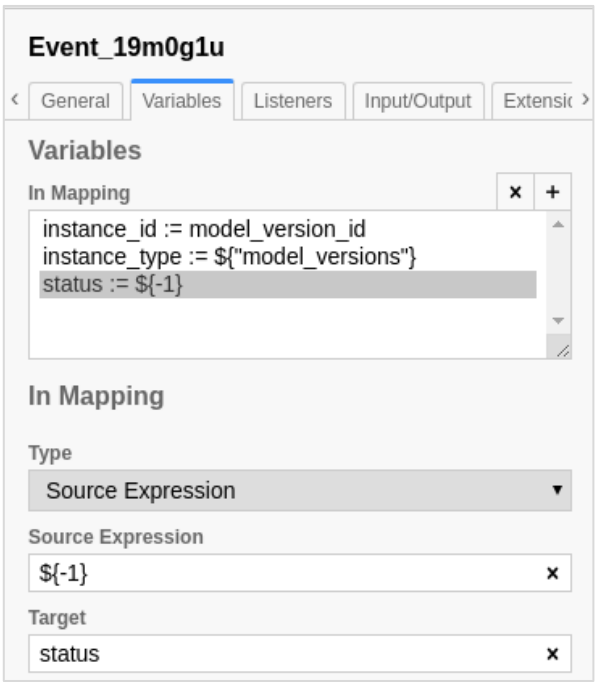


Рисунок 30 – Пример заполнения status

Выбран тип **Source Expression**, потому что создается новая маппинг-переменная.

6.4.1 Пример создания процесса по согласованию версии модели

Предварительно нужно подготовить основные элементы диаграммы, описанные выше (Рисунок 31).

Примечание. Все действия, описанные ниже, можно изменить на свое усмотрение. Так же как и состав схемы и количество participant. Тем не менее, на диаграмме должны присутствовать: элемент начала и сигналы завершения.

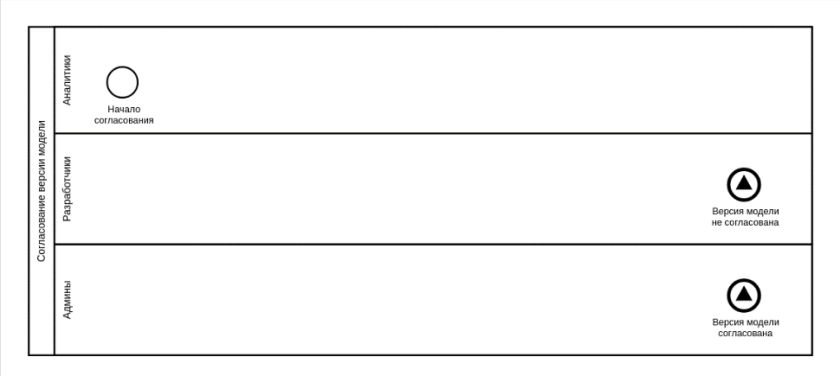


Рисунок 31 – Блок Pool/Participant

Шаблон будет согласовываться тремя группами: Аналитики, Разработчики, Администраторы.

Аналитики и разработчики будут согласовывать параллельно, после, исходя из их результатов, будет выноситься решение о передаче возможности согласования Администраторами.

За пользовательские действия отвечает элемент **User Task**.

Рисунок 32 – Элемент User Task

Для начала необходимо выбрать назначенных на задачу или же кандидатов. Указывать можно любое количество (Рисунок 33). Значения пишутся слитно через запятую.

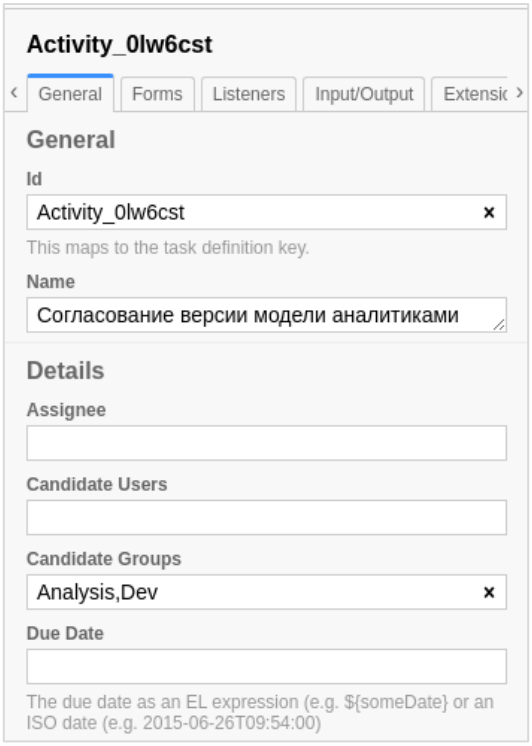


Рисунок 33 – Назначение кандидатов

Далее необходимо заполнить дополнительные поля формы (дополнительные, потому что помимо них в задаче будут основные, в зависимости от выбранного типа согласования) (Рисунок 34).

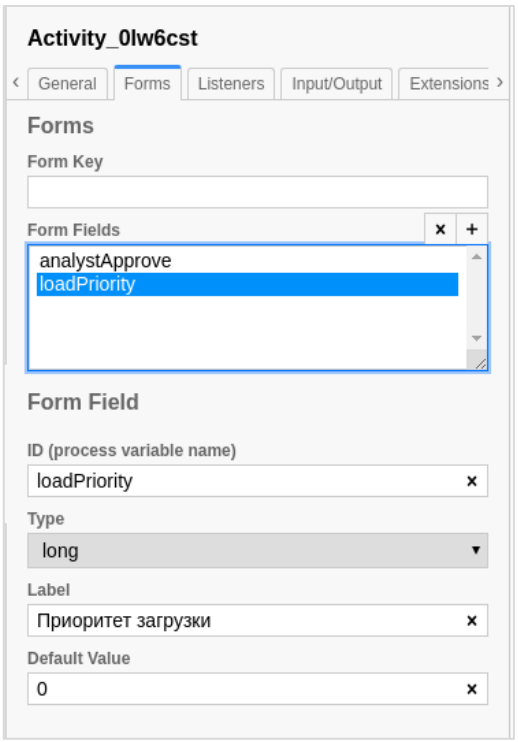


Рисунок 34 – Заполнение дополнительных полей формы

Поле **Form Key** можно не заполнять. Если его заполнить, ничего не изменится.

На данный момент доступные типы: **string**, **long**, **boolean**, **date**, **enum**.

Имя, написанное в ID, после выполнение данной задачи можно будет использовать как переменную внутри процесса.

Во избежание ошибок рекомендуется всегда заполнять Default Value в зависимости от выбранного типа поля.

Parallel Gateway используется для слияния и расщепления процесса выполнения, все единицы выполнения ожидают в слиянии такое же количество элементов, как и при расщеплении (Рисунок 35).

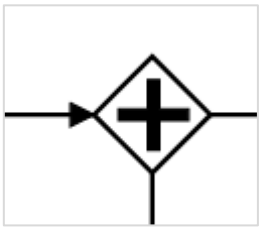


Рисунок 35 – Элемент Parallel Gateway

Через Exclusive Gate можно совершать сравнение с условиями (Рисунок 36).

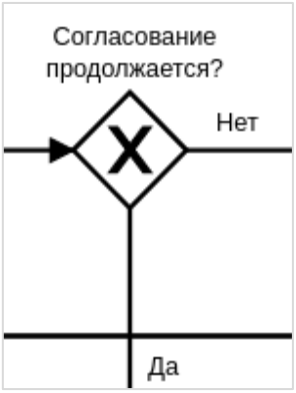


Рисунок 36 – Элемент Exclusive Gate

Синтаксис expressions

К примеру, модель не будет согласована, если аналитик или разработчик не дадут свое согласие.

```
${(!developerApprove||!analystApprove)}
```

Далее процесс согласования переносится на администраторов (Рисунок 37).

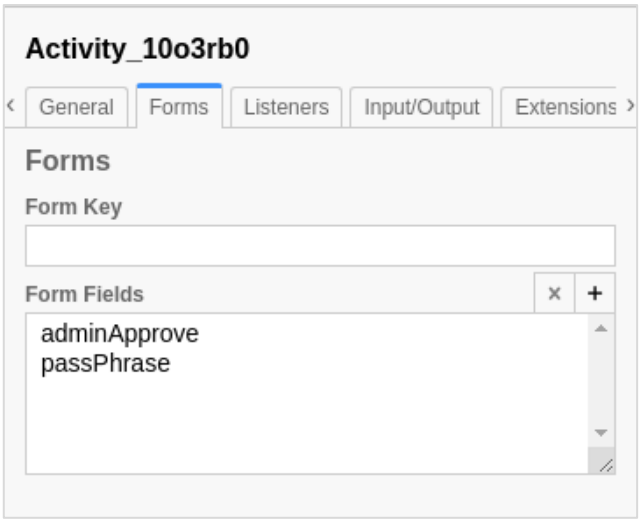


Рисунок 37 – Перенос процесса согласования на администраторов

Им нужно согласовать версию модели и ввести кодовую фразу.

После чего будет определено, согласована версия модели или нет (Рисунок 38).

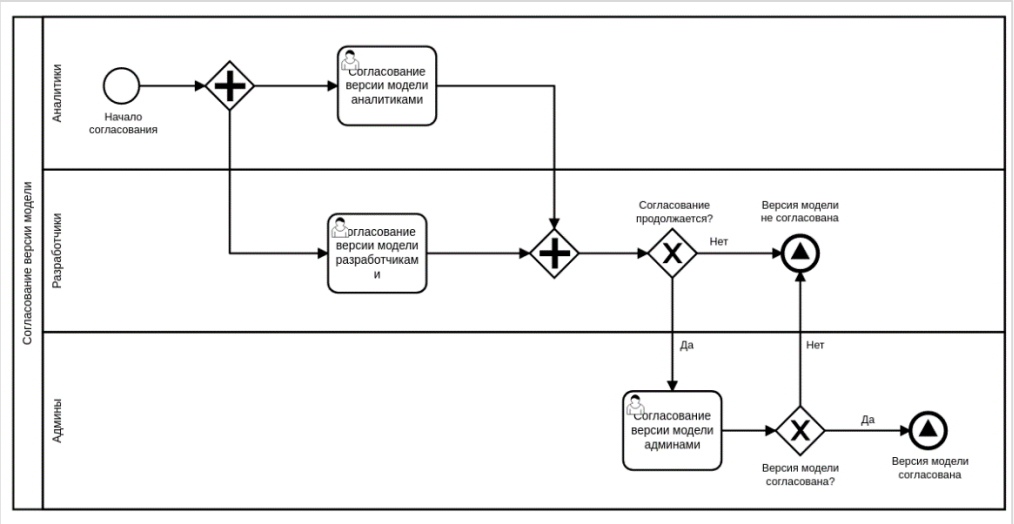


Рисунок 38 – Итоговый шаблон согласования

6.5 Добавление новых узлов в MD

Для добавления пользовательских узлов в модуль Model Designer необходимо:

- создать и добавить к файлам приложения код исполнения узла и словарь пользовательских узлов;
- создать метаданные узла при помощи API приложения.

6.5.1 Добавление пользовательских файлов с кодом исполнения узла

Пользователь должен создать файл с классом узла, наследуемым от базового класса MDNode, переопределив 3 абстрактных метода:

- check_status - метод, определяющий поведение узла при проверке статусов узлов pipeline;
- update_metadata - метод, определяющий поведение узла при обновлении метаданных узлов pipeline;
- run - метод, определяющий поведение узла при его запуске;

Пример кода исполнения узла см. в **Приложении 1**.

Кроме того, нужно добавить файл с со словарем кастомных узлов custom_node_map.py (по аналогии с файлом /app/app/workers/md/md_nodes/nodes_map.py), в котором установить связь между node_type_id (см. п. 2) метаданных узла и пользовательским классом исполнения.

Файл custom_node_map.py необходимо разместить в /app/app/workers/md/md_nodes, расположение файлов с кастомными узлами устанавливается пользователем в custom_node_map.py

6.5.2 Добавление метаданных узла

Добавление метаданных узла происходит по REST запросам к приложению:

- для создания пользовательского узла необходимо отправить POST запрос по /api/v2/md/node_types/ со схемой метаданных нового узла (пример схемы - в

swagger приложения). Важно указать в атрибуте node_type_id схемы то значение, которое было указано в custom_node_map.py для связи метаданных с кодом исполнения.

- для удаления пользовательского узла необходимо отправить DELETE запрос по /api/v2/md/node_types/?node_type_id={node_type_id}.
- для просмотра списка всех добавленных пользовательских узлов необходимо отправить GET запрос по /api/v1/md/node_types

ПРИЛОЖЕНИЕ 1. ПРИМЕР КОДА ИСПОЛНЕНИЯ УЗЛА (УЗЕЛ ТРАНСФОРМАЦИЯ)

```
import json
from pandas import DataFrame
from typing import List

from app.schemas.formula import Formula, ValueType, ComputedColumn
from app.core.formula.pandas.helpers import apply_computed_columns
from app.workers.md.md_nodes.md_node import MDNode, ModelResult
from app.workers.md.md_nodes.node_utils import copy_variables
from app import schemas, models

from sqlalchemy.orm import Session
from sqlalchemy import and_

class Transform(MDNode):

   def run(
       self,
       db: Session,
       node: models.md.TNode,
       df: DataFrame,
       vars_in: List[schemas.md.MDVariable],
       vars_out: List[schemas.md.MDVariable],
       params
   ):
       transform_param = params['TRANSFORM_TABLE']['value']
       computed_columns = [
           ComputedColumn(
               guid=column['ID'],
               name=column['Attribute'],
               logical_type="Numeric" if column['Level']['id'] ==
'Interval' \
                   else "String",
               formula=Formula.parse_obj(column['Formula'])
           )
```

```
            for column in transform_param['rows']
        ]
        nom_vars = {
            var.var_name: ValueType.String
            for var  in filter(
                lambda v: v.var_level in ["Nominal", "Binary"],
                vars_in)
        }
        int_vars = {
            var.var_name: ValueType.Number
            for var  in filter(
                lambda v: v.var_level == "Interval",
                vars_in)
        }
        transformed_df = apply_computed_columns(
            columns={**nom_vars,**int_vars},
            df=df,
            computed_columns=computed_columns
        )
        sample_table = self.create_result_table_from_dataframe(
            df=transformed_df,
            table_name='Пример данных',
            filter_rows=100
        )
        transform_stats = {
            "tables": [sample_table]
        }
        return transformed_df, ModelResult(
            df = transformed_df,
            stats = transform_stats,
            mm_code = "",
            pickle_bytes = None
        )

    def check_status(
        self,
        vars_out
```

```
    ):
        pass

    def update_metadata(
        self,
        vars_out,
        diff_out,
        diff_in,
        node_out=None
    ):
        copy_variables (self.db, self.node_orm, vars_out)

def create_vars_for_transform( db: Session,
    transform_table_param: models.md.TParameterValue,
    node_orm: models.md.TNode,
):
    vars_in = db.query(models.md.TVariable).filter(
        and_(
            models.md.TVariable.node_guid ==
transform_table_param.node_guid,
            models.md.TVariable.var_type == 'IN'
        )
    ).all()
    # recreating basic variables of node
    copy_variables(db, node_orm, vars_in)
    value_orm: models.md.TParameterValue =
db.query(models.md.TParameterValue).filter(
        and_(
            models.md.TParameterValue.node_guid ==
node_orm.node_guid,
            models.md.TParameterValue.parameter_id ==
'TRANSFORM_TABLE'
        )
    ).first()
    vars_dict = json.loads(value_orm.value_json)
    for var_out in vars_dict['value']['rows']:
        transform_var_out: models.md.TVariable = models.md.TVariable(
```

```
            node_guid=node_orm.node_guid,
            pipeline_guid=node_orm.pipeline_guid,
            var_type='OUT',
            var_guid=var_out['ID'],
            var_name=var_out['Attribute'],
            var_level=var_out['Level']['id'],
            var_role=var_out['Roles']['id'],
        )
        db.add(transform_var_out)
    db.commit()
```