DBRepo - Database Repository

Documentation for version v1.13.3

Martin Weise

Table of contents

1. Get Started

1.1 DBRepo

CI/CD Status CI/CD Coverage release v1.13.2 pypi v1.13.2 Image Pulls Artifact Hub dbrepo
license Apache License 2.0

Documentation for version: v1.13.3.

DBRepo is an open-source database repository that cover the data life cycle supporting data evolution, -citation and -versioning. It implements the query store of the RDA WGDC on precisely identifying arbitrary subsets of data.

1.1.1 Why use DBRepo?

- **Built-in search** makes your dataset searchable without extra effort: metadata is generated automatically for data in your databases.
- **Citable datasets** adopting the recommendations of the RDA-WGDC, arbitrary subsets can be precisely, persistently identified using data versioning of MariaDB and the DataCite schema for minting DOIs.
- **Powerful API for Data Scientists** with our strongly typed Python Library, Data Scientists can import, export and work with data from Jupyter Notebook or Python script, optionally using Pandas DataFrames.
- **Cloud Native** our lightweight Helm chart allows for installations on any cloud provider or private-cloud setting that has an underlying PV storage provider.

Installing DBRepo is very easy or give it a try online.

1.1.2 Who is using DBRepo?

- TU Wien (Austria)
- EGI (pan-European)
- Institut Teknologi Bandung (Indonesia)
- TU Darmstadt (Germany)
- TU Graz (Austria)
- Universität Hamburg (Germany)
- Universitas Gadjah Mada (Indonesia)
- Universiti Sains Malaysia (Malaysia)
- Universiti Teknikal Malaysia Melaka (Malaysia)
- University of the Philippines Diliman (Phillipines)

Stay up to date and subscribe to our mailing list for quarterly news on DBRepo. You can unsubscribe too.

1.1.3 How can I try DBRepo?

There's a hosted test environment maintained by DS-IFS where you can explore DBRepo using your existing account.

Demo Environment

1.2 Why use DBRepo

Digital repositories see themselves more frequently encountered with the problem of making databases accessible in their collection. Challenges revolve around organizing, searching and retrieving content stored within databases and constitute a major technical burden as their internal representation greatly differs from static documents most digital repositories are designed for.

1.2.1 Application Areas

We present a database repository system that allows researchers to ingest data into a central, versioned repository through common interfaces, provides efficient access to arbitrary subsets of data even when the underlying data store is evolving, allows reproducing of query results and supports findable-, accessible-, interoperable- and reusable data.

1.2.2 Features

Built-in search

DBRepo makes your dataset searchable without extra effort: most metadata is generated automatically for data in your databases. The fast and powerful OpenSearch database allows a fast retrieval of any information. Adding semantic mapping through a suggestion-feature, allows machines to properly understand the context of your data Learn more.

Citable datasets

Adopting the recommendations of the RDA-WGDC, arbitrary subsets can be precisely, persistently identified using system-versioned tables of MariaDB and the DataCite schema for minting DOIs. External systems i.e. metadata harvesters (OpenAIRE, Google Datasets) can access these datasets through OAI-PMH, JSON-LD and FAIR Signposting protocols Learn more.

Powerful API for Data Scientists

With our strongly typed Python Library, Data Scientists can import, export and work with data from Jupyter Notebook or Python script, optionally using Pandas DataFrames. For example: the AMQP API Client can collect continuous data from edge devices like sensors and store them asynchronous in DBRepo Learn more.

Cloud Native

Our lightweight Helm chart allows for installations on any cloud provider or private-cloud setting that has an underlying PV storage provider. DBRepo can be installed from the Artifact Hub repository. Databases are managed as MariaDB Galera Cluster with high degree of availability ensuring your data is always accessible Learn more.

1.2.3 Demo Site

We run a small demonstration instance so you can see the latest version of DBRepo in action. The demonstration instance is updated with new releases and should be considered ephemeral.

1.3 Help with DBRepo

If you need help getting started with DBRepo or with advanced usage, the following sources may be useful.

1.3.1 Usage Documentation

The User Guide documentation is the most complete guide on how to use DBRepo.

1.3.2 API Documentation

The API documentation present reference docs for all APIs.

Additional Help

Contact us via e-mail.

2. User Guide

2.1 Overview

Operational user-guide describing the graphical deposit, review and management of records.

- Quick start

Jump right into and use our pre-configured starter resources:

- Download the Jupyter Notebook for local development, or
- Use the Jupyter image `registry.datalab.tuwien.ac.at/dbrepo/starter-notebook:1.13.3` for your Jupyterhub infrastructure

2.2 Quickstart

Need to start using DBRepo ASAP? This page contains only a brief introduction. You can find in-depth information about these topics in the corresponding documentation pages on the left.

2.2.1 How to Login

The following instances have different authorization mechanisms:

- Open for anyone:
- https://test.dbrepo.tuwien.ac.at
- By affiliation:
- https://dbrepo.datalab.tuwien.ac.at for TU Wien staff

2.2.2 Create Database

UI

A database can be created by choosing a name (e.g. My Database), selecting an engine (e.g. MariaDB) and defining the visibility, this can be changed later.

Python

Python Compatibility

Ensure that you use the same Python library version as the target instance. For example: if you see `1.13.3` in the bottom left, you need to use the `1.13.3` Python library.

```
from dbrepo.RestClient import RestClient

client = RestClient("http://<hostname>", username="foo", password="bar")
database = client.create_database("My Database",
                                  "6cfb3b8e-1792-4e46-871a-f3d103527203",
                                  is_public=False,
                                  is_schema_public=False)
print(f"database id: {database.id}")
```

2.2.3 Import Data

UI

Create a table from a dataset by choosing a name (e.g. My Table), giving info on the CSV structure, uploading the CSV, selecting a primary key column and then importing the dataset.

Python

```
from dbrepo.RestClient import RestClient
from pandas import DataFrame

df = DataFrame({'some_col': 123})

client = RestClient("http://<hostname>", username="foo", password="bar")
table = client.create_table(<database_id>,
                            "My Table",
                            is_public=False,
                            is_schema_public=False,
```

```
                            dataframe=df)
print(f"table id: {table.id}")
```

2.2.4 Create View

UI

A view can be created by specifying a source table (e.g. My Table), select the columns that are included in the view and optionally filter by values (e.g. `street` = `Pave`).

Python

```
from dbrepo.RestClient import RestClient
from dbrepo.api.dto import QueryDefinition, FilterDefinition, FilterType

client = RestClient("http://<hostname>", username="foo", password="bar")
view = client.create_view(<database_id>,
                          "My View",
                          QueryDefinition("my_table",
                                          ["order", ...],
                                          filter=FilterDefinition(FilterType.WHERE, "street", "=", "Pave"),
                                          is_public=False,
                                          is_schema_public=False))
printf(f"view id: {view.id}")
```

2.2.5 Create Subset

UI

A subset can be created by specifying a source view (e.g. My View), select the columns that are included in the subset and optionally filter by values (e.g. `lot_shape` = `Reg`).

Python

```
from dbrepo.RestClient import RestClient
from dbrepo.api.dto import QueryDefinition, FilterDefinition, FilterType

client = RestClient("http://<hostname>", username="foo", password="bar")
subset = client.create_subset(<database_id>,
                              QueryDefinition("my_view",
                                              ["order", ...],
                                              filter=FilterDefinition(FilterType.WHERE, "lot_shape", "=", "Reg"),
                                              page=0,
                                              size=1000000))
printf(f"subset id: {subset.id}")
```

2.3 Account Management

2.3.1 Create Account

> **Note**
>
> The contents in this section do not apply to the instances that have affilation login mechanism (see Quickstart).

A user wants to create an account in DBRepo.

UI

To create a new account in DBRepo with the UI, provide your details in the Auth Service.

2.4 Database Management

2.4.1 Create Database

A user wants to create a database in DBRepo.

UI

A database can be created by choosing a name (e.g. My Database), selecting an engine (e.g. MariaDB) and defining the visibility, this can be changed later.

PYTHON

> **Python Compatibility**
>
> Ensure that you use the same Python library version as the target instance. For example: if you see `1.11.0` in the bottom left, you need to use the `1.11.0` Python library.

List all available containers with their database engine descriptions and obtain a container id.

```
from dbrepo.RestClient import RestClient

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
containers = client.get_containers()
print(containers)
```

Create a public database with the container id from the previous step.

```
from dbrepo.RestClient import RestClient

client = RestClient("http://<hostname>", username="foo", password="bar")
database = client.create_database("My Database",
                                  "6cfb3b8e-1792-4e46-871a-f3d103527203",
                                  is_public=False,
                                  is_schema_public=False)
print(f"database id: {database.id}")
```

2.4.2 Database Access

This short guide shows the decision flow when accessing data ("read") or importing data ("write").

- When accessing data to read it:

```
graph LR
A[Reader] --> B{Access?};
 B --> |No| C[\Error\];
B --> |Yes| D(((Allow)));
```

- When importing data to write it to the database, e.g. importing a dataset as table:

```
graph LR
A[Writer] --> B{Access?};
 B --> |No| C[\Error\];
    B --> |read| C;
B --> |write_own| D{Owner?}
    D --> |No| C;
D --> |Yes| E(((Allow)));
   B --> |write_all| E;
```

A user wants to give specific users read access to a private database.

UI

As a database owner, you can give specific users access to your database, only you can do this. Create access for another user, you can use the box to search and specify the level of access.

PYTHON

To give a user (with id `e9bf38a0-a254-4040-87e3-92e0f09e29c8` access to this database (e.g. read access), update their access using the HTTP API:

```
from dbrepo.RestClient import RestClient
from python.dbrepo.api.dto import AccessType

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
client.create_database_access(<database_id>,
                              type=AccessType.READ,
                              user_id="e9bf38a0-a254-4040-87e3-92e0f09e29c8")
```

In case the user already has access, use the method `update_database_access` or revoke `delete_database_access`.

2.4.3 Database Dashboard

A database dashboard is a managed collection of available data sources automatically provisioned by DBRepo. Every database dashboard always consists of exactly two areas:

- **Unmanaged**

  This area can be modified by the database owner and is not affected by provisioning. It is the area above the **Generated Dashboard** row.

- **Managed**

  This area is affected by provisioning, any changes to this area will be overridden and deleted without notice. Note that the dashboard links are also affected by provisioning.

Everytime the views of the database change (e.g. a new view is added, a view is deleted) then the database dashboard for this database is provisioned.

> **Note**
>
> Database dashboards are only available for non-hidden databases, see Update Visibility.

A user wants to create a dashboard of available data sources and add personal customization.

UI

As a database owner, you can disable database dashboards. They are enabled by default.

PYTHON

```
from dbrepo.RestClient import RestClient
from python.dbrepo.api.dto import AccessType

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
client.update_database_visibility(<database_id>,
                                  is_public=True,
                                  is_schema_public=False,
                                  is_dashboard_enabled=True)
```

2.4.4 Database Visibility

A user wants a datasource, e.g. a database, to be private and only give specific users access.

This can be applied to any datasource in DBRepo: tables, views, subsets and whole databases There are two attributes that define a datasource's visibility:

- **Transparency**

  Can hide the datasource in listings and search results or make it visible.

- **Insights**

  Can hide additional info of the datasource associated with schema and structure, e.g. can hide the SQL query used to create a view to not expose the database schema; or make that info visible.

A datasource therefore can have four possible visibility states:

- **Visible**

  When set to visible Transparency and visible Insights, everything is visible to the world.

  > **Note**
  >
  > Visible data and a PID with license to reuse (e.g. CC-BY 4.0) is equivalent to Open Data.

- **Data-only**

  When set to visible Transparency and hidden Insights, this affects data sources. Associated schema metadata is hidden.

- Database: removes the list of tables, views and subsets.
- Table: removes the list of columns.
- View: removes the SQL query.
- Subset: removes the SQL query.

- **Schema-only**

  When set to hidden Transparency and visible Insights, this affects data sources. Associated data information is hidden and the data preview is disabled for anyone without at least `read` access.

- **Hidden**

  When set to hidden Transparency and hidden Insights, this affects data sources, everything is hidden to anyone (even the existence). Only the owner and selected user accounts that have at least `read` access can see the datasource. No database dashboard is visible to anyone.

UI

PYTHON

To change the visibility of a database where you are the owner (this is the case for self-created databases), send a request to the HTTP API:

```
from dbrepo.RestClient import RestClient

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
client.update_database_visibility(<database_id>,
                                  is_public=True,
                                  is_schema_public=False,
                                  is_dashboard_enabled=True)
```

2.4.5 Delete Database

This section will be expanded in the future.

2.5 Data Management

2.5.1 Create Persistent Identifier

```
graph LR
A[Database] --> C(Metadata Service);
B[Subset] --> C;
D[Table] --> C;
E[View] --> C;
subgraph DataCite
F(Fabrica)
end
C <-.-> |DOI| F;
```

A persistent identifier (PID) stores metadata redundant in an external system that issues them (i.e. DataCite for DOIs). DBRepo supports PIDs in the form of URIs and can be optionally configured to use DOIs from DataCite.

A user wants to assign a PID to a data source, e.g. subset, owned or created by them.

UI

The subset can be persistently identified by creating a PID via the "Get PID" button in the toolbar of each data source. If you do not have a PID already from another system, you can create one. This is the default action.

First, provide information on the dataset creator(s), if possible, provide a PID on the person or organization. In some cases these will automatically be resolved (i.e. ORCID, ROR, DOI) and an attempt to load external metadata will be made. Using PIDs improves the accuracy of the PID record.

In case no PID is available, start by selecting the creator type (natural person or organization).

Fill out the given name, i.e. firstname(s) and family name, i.e. lastname(s). Both form the mandatory field name and should have the form `family name, given name`.

If available, provide the PID of your affiliation (e.g. `https://ror.org/04d836q62`), which increases accuracy of the PID record. If not, optionally provide an affiliation name.

The PID needs at least one title in the title field and optionally a title type from the selection and optionally a title language from the available list of languages.

Add additional title(s) where needed, a PID can have multiple titles.

The PID needs at least one description in the description field and optionally a description type from the selection and a description language.

Optionally provide a text description on the collection method and description of the dataset. This information greatly enhances (re-)usability of your data. You can remove those sections by clicking "Remove".

The dataset publisher in most cases is the organization that operates the DBRepo service.

Provide a publication year and optionally a publication month and publication day. In case of imported data, this date corresponds to the *original* publishing date.

Optionally reference other PIDs, and add a license from the list.

Optionally provide the main language for the PID record. This helps machines to understand the context of your data.

Optionally add funding information. When you are finished

PYTHON

```
from dbrepo.RestClient import RestClient
from python.dbrepo.api.dto import IdentifierType, CreateIdentifierFunder, CreateIdentifierDescription, \
```

```
    CreateIdentifierTitle, Identifier, CreateIdentifierCreator, CreateRelatedIdentifier, RelatedIdentifierType, \
    RelatedIdentifierRelation

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
identifier = client.create_identifier(<database_id>,
                                      type=IdentifierType.DATABASE,
                                      creators=[CreateIdentifierCreator(
                                          creator_name="Weise, Martin",
                                          name_identifier="https://orcid.org/0000-0003-4216-302X")],
                                      titles=[CreateIdentifierTitle(
                                          title="Danube river water measurements")],
                                      descriptions=[CreateIdentifierDescription(
                                          description="This dataset contains hourly measurements of the water \
                                           level in Vienna from 1983 to 2015")],
                                      funders=[CreateIdentifierFunder(
                                          funder_name="Austrian Science Fund",
                                          funder_identifier="https://doi.org/10.13039/100000001")],
                                      licenses=[Identifier(identifier="CC-BY-4.0")],
                                      publisher="TU Wien",
                                      publication_year=2024,
                                      related_identifiers=[CreateRelatedIdentifier(
                                          value="https://doi.org/10.5334/dsj-2022-004",
                                          type=RelatedIdentifierType.DOI,
                                          relation=RelatedIdentifierRelation.CITES)])
print(f"identifier id: {identifier.id}")
```

2.5.2 Create Subset

A subset is defined by a query that specifies which part of a dataset (=view) is included. By default, the DBRepo restricts certain operations (e.g. computing functions) to ensure i) long-term stability, ii) -compatibility and iii) cross-database vendor compatibility for future possible migration scenarios.

1.13.0

Supported MariaDB features as of 1.13.0

Starting version 1.13.0, the following subset-queries are supported:

- select queries
- `WHERE` conditions (optional)
- `GROUP BY` columns (optional)
- `ORDER BY` columns (optional)
- join queries
- `INNER JOIN` (default for MariaDB)
- `LEFT JOIN`
- `LEFT INNER JOIN`
- `RIGHT JOIN`
- `RIGHT INNER JOIN`
- `CROSS JOIN`

The join conditions are connected via conjuctions (i.e. `SELECT ... FROM tbl JOIN other_tbl ON c1 = c2 AND ...`).

Currently `HAVING` conditions, `LIMIT` (and by extension `OFFSET`) statements are not supported.

UI

A subset can be created by specifying a source view (e.g. My View), select the columns that are included in the subset and optionally filter by values (e.g. `lot_shape` = `Reg`).

PYTHON

```python
from dbrepo.RestClient import RestClient
from dbrepo.api.dto import QueryDefinition, FilterDefinition, FilterType

client = RestClient("http://<hostname>", username="foo", password="bar")
subset = client.create_subset(<database_id>,
                              QueryDefinition("my_view",
                                              ["order", ...],
                                              filter=FilterDefinition(FilterType.WHERE, "lot_shape", "=", "Reg"),
                                              page=0,
                                              size=1000000))
printf(f"subset id: {subset.id}")
```

Limitations

- The Python library currently does not support custom aliases when creating subsets with two columns with the same name. Instead, a suffix of `_a[n]` is added with `n` being an incremental counter starting at `0`.

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

2.5.3 Fetch Data

```
graph LR
B[Subset] --> C(Data Service);
D[Table] --> C;
E[View] --> C;
B --> |PID| G(Metadata Service);
D --> |PID| G;
E --> |PID| G;
subgraph Python
F[DataFrame];
end
C --> |get_subset_data| F;
C --> |get_table_data| F;
C --> |get_view_data| F;
C --> |REST API| F;
G --> |get_identifier_data| F;
G --> |REST API| F;
C --> |REST API| H[Application];
C --> |OAI-PMH API| H;
C --> |FAIR Signposting API| H;
C --> |JSON-LD API| H;
F <--> H;
```

A user wants to access and download data from a data source, e.g. a view.

UI

The UI can be used to fetch data. The data preview can be used to explore data by navigating through the fetched data with the arrows and selecting a bigger page size.

Export the complete data of the data source by clicking the "Download" button.

PYTHON

- Subset

```python
from dbrepo.RestClient import RestClient

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
df = client.get_subset_data(<database_id>,
                            subset_id='5cdd08b7-4c6d-483e-b857-0eeaa86c59b4',
                            size=1000000)
```

- Table

```python
from dbrepo.RestClient import RestClient

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
df = client.get_table_data(<database_id>,
                           table_id='c0e9dd95-f43a-4e2f-a5e8-755e058cf118',
                           size=1000000)
```

- View

```python
from dbrepo.RestClient import RestClient

client = RestClient(endpoint="http://<hostname>", username="foo",
                    password="bar")
df = client.get_view_data(<database_id>,
                          view_id='32567e6c-f9ce-4b08-b350-88a7e2a7e291',
                          size=1000000)
```

2.5.4 Import Dataset

```
graph LR
    A(UI)     --> |HTTP| D;
  B(Python) --> |HTTP| D(Data Service);
D          --> |JDBC| E[(Data Database)];
```

A user wants to import a static dataset (e.g. from a .csv file). In this action, a table will be created in the database.

Importing a dataset required at least `write-own` access, see Database Access. If you are the owner of the database, you are good to go by default.

UI

Click on "Create Table" in the database toolbar at the top. Then give the table a name, optional description (this can be added at a later point as well) and set the visibility settings for transparency and insights. In this example the dataset will be fully visible to the world.

In the next step, provide the dataset structure, the default will be sufficient for most cases.

Select the CSV dataset, it will upload the dataset automatically and analyse the contents to recommend the table structure including separator (e.g. `,`), newline terminator (e.g. `\n`), comments and lines to skip to infer the headers.

Next, confirm or correct the dataset schema that has been automatically recommended. For example, change the data type if it was incorrectly analysed. You need to select one or more columns to be the primary key that must contain a unique (combination of) values. Typically, this will be a column named `id` or similar.

The import settings in the import page already takes over the settings from the previous page. You need to click "Import Data". The table now contains the dataset.

PYTHON

> **Python Compatibility**
>
> Ensure that you use the same Python library version as the target instance. For example: if you see `1.9.2` in the bottom left, you need to use the `1.9.2` Python library.

You can import a dataset from a `pandas` DataFrame via our Python library.

- Table from Dataset

```python
from dbrepo.RestClient import RestClient
from pandas import DataFrame

df = DataFrame({'some_col': 123})

client = RestClient("http://<hostname>", username="foo", password="bar")
table = client.create_table(<database_id>,
                            "Cool Table",
                            is_public=True,
                            is_schema_public=True,
                            dataframe=df)
print(f"table id: {table.id}")
```

- Import Data into existing Table

```python
from dbrepo.RestClient import RestClient
from pandas import DataFrame

df = DataFrame({'some_col': 123})

client = RestClient("http://<hostname>", username="foo", password="bar")
client.import_table_data(<database_id>,
                         table_id='4ce60952-13d3-430f-a2ad-93e4759542a0',
                         dataframe=df)
```

2.5.5 Import Live Data

```
graph LR
A[Sensor] --> |AMQP| C(Broker Service);
B[Sensor] --> |MQTT| C;
C --> |HTTP| D(Data Service);
D --> |JDBC| E[(Data Database)];
```

DBRepo supports advanced data import through asynchronous data mechanisms. The Broker Service offers two protocols commonly used by Internet of Things (IoT) infrastructures: AMQP and the lightweight MQTT.

Find the advanced data import details for importing live data on the data source info page.

AMQP connection information

A user wants to import live data from e.g. sensor measurements into a table in DBRepo. The user needs to have at least `write-own` access to write data into a table owned by them.

PYTHON

- AMQP

```python
from dbrepo.AmqpClient import AmqpClient

client = AmqpClient(broker_host="<hostname>", broker_port=5672, username="foo",
                    password="bar")
client.publish("dbrepo.<database_id>.<table_id>", {'precipitation': 2.4})
```

- MQTTv5

```python
from dbrepo.MqttClient import MqttClient

client = MqttClient(broker_host="<hostname>", broker_port=1883, username="foo",
                    password="bar")
client.publish("dbrepo.<database_id>.<table_id>", {'precipitation': 2.4})
```

3. Maintainer Guide

3.1 Installation

Before you can use DBRepo on your own computer, you need to install it:

Install on Kubernetes

Production-ready deployment on Kubernetes (recommended).

→ Getting started

Install on Docker

Quick and simple Docker Compose deployment for local testing purposes.

→ Getting started

Install on VM

Use our pre-configured VM image for local testing purposes.

→ Getting started

Build from Sourcecode

Use our pre-configured VM image for local testing purposes.

→ Getting started

3.2 Configuration

This guide assumes you have already performed the quick Install on Docker. You can (re-)configure most components by executing:

```
make gen-secrets
```

Note that this script overwrites (previously generated) secrets, use with caution.

Identity Service

Add the admin password to the `.env` file:

```
echo "IDENTITY_SERVICE_ADMIN_PASSWORD=$(openssl rand -hex 16) >> .env"
```

Auth Service

We need to configure three components:

1. Environment
2. Client `dbrepo-client`
3. User federation `Identity Service`

First, add the admin password to the `.env` file:

```
echo "AUTH_SERVICE_ADMIN_PASSWORD=$(openssl rand -hex 16) >> .env"
```

First, access the Admin UI at http://localhost:8080 and change the client secret of the default `dbrepo-client` client:

Change the Client Secret

Next, log into the Auth Service with the default credentials `admin` and the value of `AUTH_SERVICE_ADMIN_PASSWORD` and select the "dbrepo" realm ①. In the sidebar, select the "User federation" ② and from the provider list, select the "Identity Service" provider ③.

Keycloak identitiy provider list

Second, modify the Bind DN ①. Change the **Bind credentials** to the desired password ② from the variable `IDENTITY_SERVICE_ADMIN_PASSWORD` in `.env`.

Keycloak identity provider settings

Apply

You need to remove all containers and volumes to apply the changes and then restart:

```
docker container stop $(docker container ls -aq -f "name=dbrepo-*")
docker container rm $(docker container ls -aq -f "name=dbrepo-*")
docker volume rm $(docker volume ls -q -f "name=dbrepo_*")
docker compose up -d
```

3.2.1 Next Steps

You should now be able to view the front end at https://localhost.

Please be warned that the default configuration is not intended for public deployments. It is only intended to have a running system within minutes to play around within the system and explore features. It is strongly advised to change the default `.env` environment variables.

Next, create a user account and then create a database to import a dataset.

3.3 Deployments

3.3.1 Install on Kubernetes

TL;DR

To install DBRepo in your existing cluster, download the sample `values.yaml` for your deployment and update the variables, especially `hostname`.

```
helm upgrade --install dbrepo \
  -n dbrepo \
  "oci://registry.datalab.tuwien.ac.at/dbrepo/helm/dbrepo" \
  --values ./values.yaml \
  --version "1.13.3" \
  --create-namespace \
  --cleanup-on-fail
```

Prerequisites

Info

- Kubernetes 1.30+ (tested on 1.31.5)
- PV provisioner support in the underlying infrastructure

RESOURCE QUOTA

By default, any service requests a minimum quota of cpu and memory by default. These values are based on the absolute minimum needed to start the service. From our experience your cluster needs to meet the following `ResourceQuota`:

```
requests.cpu: 12000m
requests.ephemeral-storage: 3072Mi
secrets: '50'
<storageClass>.storageclass.storage.k8s.io/persistentvolumeclaims: '20'
<storageClass>.storageclass.storage.k8s.io/requests.storage: 150Gi
persistentvolumeclaims: '20'
requests.memory: 14336Mi
pods: '40'
requests.storage: 150Gi
limits.memory: 20480Mi
configmaps: '20'
services: '40'
```

Uninstall

Remove DBRepo by stopping and removing all pods and persistent volume claims:

```
helm uninstall \
  dbrepo \
  -n dbrepo
kubectl -n dbrepo \
  delete \
  pvc \
  --all
```

Further configuration is highly recommended

See "Security Disclaimer" above, this quick & non-secure installation needs to be configured further to achieve basic security guarantees. Please visit the configuration page in the next step to complete the installation.

3.3.2 Install on Docker

> **Security Disclaimer**
>
> This quick default installation should **not be considered secure**. It is intended for local testing and demonstration and should not be used in public deployments or in production. It is a quick installation method and is intended for a quick look at DBRepo.

TL;DR

Install DBRepo in one line:

```
curl -sSL https://gitlab.phaidra.org/fair-data-austria-db-repository/fda-services/-/raw/release-1.13/install.sh | bash
```

Then start DBRepo and visit `https://localhost` in your browser:

```
docker compose up -d
```

Prerequisites

> **Info**
>
> - min. 8 vCPU cores
> - min. 20GB free RAM memory

Since DBRepo is intended to be a publicly available repository, an optional fixed/static IP-address with optional SSL/TLS certificate is recommended. Follow the Secure Installation guide.

Uninstall

Remove DBRepo by stopping and removing all containers and volumes:

```
docker container stop $(docker container ls -f "name=dbrepo-*" -aq)
docker container rm $(docker container ls -f "name=dbrepo-*" -aq)
docker volume rm $(docker volume ls -f "name=dbrepo_*" -q)
```

> **Further configuration is highly recommended**
>
> See "Security Disclaimer" above, this quick & non-secure installation needs to be configured further to achieve basic security guarantees. Please visit the configuration page in the next step to complete the installation.

3.3.3 Install on VM

Security Disclaimer

This quick default installation should **not be considered secure**. It is intended for local testing and demonstration and should not be used in public deployments or in production. It is a quick installation method and is intended for a quick look at DBRepo.

TL;DR

Download the live image and create a VM using your hypervisor. Below is an extended example on how to do this on Debian with libvirt.

Then start DBRepo and visit `http://<hostname_or_ip>` in your browser:

```
docker compose up -d
```

Example

In this extended example we setup the live image using `libvirt` and `virt-manager` via QEMU. First, install the dependencies:

Debian

```
apt install virt-manager
```

Next, check your system compatibility:

Debian

```
virt-host-validate qemu
```

The result should at least `PASS` for `/dev/kvm`.

Open the `virt-manager` application and choose "Network install":

```
virt-manager \
  --connect "qemu:///system" \
  --show-domain-creator
```

- Fill in the operating system install URL `https://www.ifs.tuwien.ac.at/infrastructures/dbrepo/1.12.0/debian-13-testing-live-dbrepo-amd64.iso`
- Choose the operating system `Debian 13` (or similar)
- Assign at least 4096 MiB memory and 4 CPUs
- Assign at least 25 GiB virtual disk storage

When the virtual machine is started, continue by starting DBRepo:

```
docker compose up -d
```

And then find out the hostname or IP address of the VM:

```
hostname -I | cut -d' ' -f1
```

This hostname or IP address is then used on your local machine (e.g. laptop) to access the DBRepo instance: `http://<hostname_or_ip>`

3.3.4 Build from Sourcecode

Security Disclaimer

This quick default installation should **not be considered secure**. It is intended for local testing and demonstration and should not be used in public deployments or in production. It is a quick installation method and is intended for a quick look at DBRepo.

TL;DR

Clone the source code repository and build the container images:

```
git clone git@gitlab.phaidra.org:fair-data-austria-db-repository/fda-services.git && \
git checkout release-1.10 && \
make build-java-lib build-images
```

Then start DBRepo and visit `http://localhost` in the VM browser:

```
docker compose up -d
```

Prerequisites

BUILDTIME

Info

- Docker 28.x

Utility tools to run the build commands:

Debian

```
apt install git make bash
```

RUNTIME

Info

- min. 8 vCPU cores
- min. 20GB free RAM memory

Build

Clone the source code repository:

```
git clone git@gitlab.phaidra.org:fair-data-austria-db-repository/fda-services.git && \
git checkout release-1.10
```

Build the Docker container images and auxiliary files needed for the Auth Service:

```
make build-java-lib build-images
```

Run

Then start DBRepo and visit `http://localhost` in the VM browser:

```
docker compose up -d
```

Shorthand Dev Build

You can build everything and (re-)start the necessary services between each build conveniently:

```
make start-dev
```

3.4 Management

3.4.1 Backup Data

Security Disclaimer

This quick default backup guide should not be considered secure. Please always encrypt your backups using `openssl` or `gpg` as recommended by MariaDB.

TL;DR

This page explains how to perform regular backups of MariaDB Galera (i.e. Data Database and Metadata Database) so they can be recovered at a later point in time.

Full Backup

We adhere to the official MariaDB 11.3.2 documentation in this section.

The following command creates a directory to store the backup and performs a full backup for node `0` (you need to do this for all nodes e.g. nodes `0`, `1` and `2`). Beware that not all paths are writable so we use one that is.

Data Database **Metadata Database**

```
kubectl -n $NAMESPACE exec pod/data-db-0 -c mariadb-galera -- \
    mkdir \
    /bitnami/mariadb/backup
```

Then create the backup using the values from `values.yaml`. The credentials are located at `datadb.galera.mariabackup.user` and `datadb.galera.mariabackup.password`.

```
kubectl -n $NAMESPACE exec pod/data-db-0 -c mariadb-galera -- \
    mariadb-backup \
    --backup \
    --galera-info \
    --target-dir=/bitnami/mariadb/backup \
    --user=backup \
    --password=backup
```

Compress the archive directory.

```
kubectl -n $NAMESPACE exec pod/data-db-0 -c mariadb-galera -- \
    tar czfv \
    /bitnami/mariadb/backup.tar.gz \
    /bitnami/mariadb/backup && \
    rm -rf \
        /bitnami/mariadb/backup
```

Move the backup from the nodes to your backup machine through e.g. copying it:

```
kubectl -n $NAMESPACE \
    cp \
    pod/data-db-0:/bitnami/mariadb/backup.tar.gz \
    backup-data-db-0.tar.gz && \
    rm -f \
        /bitnami/mariadb/backup.tar.gz
```

```
kubectl -n $NAMESPACE exec pod/metadata-db-0 -c mariadb-galera -- \
    mkdir \
    /bitnami/mariadb/backup
```

Then create the backup using the values from `values.yaml`. The credentials are located at `metadatadb.galera.mariabackup.user` and `metadatadb.galera.mariabackup.password`.

```
kubectl -n $NAMESPACE exec pod/metadata-db-0 -c mariadb-galera -- \
    mariadb-backup \
    --backup \
    --galera-info \
    --target-dir=/bitnami/mariadb/backup \
    --user=backup \
    --password=backup
```

Compress the archive directory.

```
kubectl -n $NAMESPACE exec pod/metadata-db-0 -c mariadb-galera -- \
    tar czfv \
    /bitnami/mariadb/backup.tar.gz \
    /bitnami/mariadb/backup && \
    rm -rf \
        /bitnami/mariadb/backup
```

Move the backup from the nodes to your backup machine through e.g. copying it:

```
kubectl -n $NAMESPACE \
    cp \
    pod/metadata-db-0:/bitnami/mariadb/backup.tar.gz \
    backup-metadata-db-0.tar.gz && \
    rm -f \
        /bitnami/mariadb/backup.tar.gz
```

Repeat this for nodes `1` and `2` (or more).

3.4.2 Restore Data

Please check the guide on how to perform a backup first.

TL;DR

This page explains how to restore from a backup of a MariaDB Galera (i.e. Data Database and Metadata Database).

Restore Data

There are multiple guides available on how to restore, we prioritize the Helm Chart documentation over the official MariaDB 11.3.2 documentation as it simplifies many steps.

ORDERLY SHUTDOWN

This section follows the official MariaDB 11.3.2 documentation adapted to the Bitnami Chart. This is the case when some (or all) nodes in the cluster fail to reach a quorum due to them switching to a non-primary state and refusing to serve queries to protect data integrity[1].

Follow the recommendation of the safe to bootstrap feature on each node N= `0`, `1` and `2` (or more). Look for `safe_to_bootstrap: 1`.

```
kubectl exec data-db-N -- cat \
    /bitnami/mariadb/data/grastate.dat
```

Use the daemon to read the InnoDB storage engine logs and report the last known transaction position

Alternatively, you can use the MySQL daemon `mysqld` to determine the sequence number:

```
kubectl exec data-db-N -- mysqld --wsrep-recover
```

Use the PVCs directly and read the sequence number from the volumes

Alternatively, if you cannot use the pods, you can try to find the most advanced node directly in the PVCs:

```
kubectl run -i --rm --tty volpod --overrides='
{
    "apiVersion": "v1",
    "kind": "Pod",
    "metadata": {
        "name": "volpod"
    },
    "spec": {
        "containers": [{
            "command": [
                "cat",
                "/mnt/data/grastate.dat"
            ],
            "image": "bitnami/minideb",
            "name": "mycontainer",
            "volumeMounts": [{
                "mountPath": "/mnt",
                "name": "galeradata"
            }]
        }],
        "restartPolicy": "Never",
        "volumes": [{
            "name": "galeradata",
            "persistentVolumeClaim": {
                "claimName": "data-data-db-N"
            }
        }]
    }
}' --image="bitnami/minideb"
```

If there exists a node with `safe_to_bootstrap: 1`, use this node `M` to bootstrap from. Otherwise, if **no** node is safe to bootstrap from, follow the next section of unexpected shutdown scenario.

Then tell the node `M` to form a new primary component by itself. This process is non-destructive and increases the chance of a fast incremental State Transfer (IST) for the other nodes.

```
helm upgrade \
    --install \
    my-release \
    "oci://registry.datalab.tuwien.ac.at/dbrepo/helm/dbrepo" \
    --namespace $NAMESPACE \
    --set datadb.rootUser.password=XXXX \
    --set datadb.galera.mariabackup.password=YYYY \
    --set datadb.galera.bootstrap.forceBootstrap=true \
    --set datadb.galera.bootstrap.bootstrapFromNode=M \
    --set datadb.galera.bootstrap.forceSafeToBootstrap=true \
    --set datadb.podManagementPolicy=Parallel
```

Then delete the failing pods (they will be created immediately again).

```
kubectl -n $NAMESPACE \
    delete \
    pods \
    -l app.kubernetes.io/name=datadb
```

Then remove the forced bootstrapping.

```
helm upgrade \
    --install \
    my-release \
    "oci://registry.datalab.tuwien.ac.at/dbrepo/helm/dbrepo" \
    --namespace $NAMESPACE \
    --set datadb.rootUser.password=XXXX \
    --set datadb.galera.mariabackup.password=YYYY \
    --set datadb.galera.bootstrap.forceBootstrap=true \
    --set datadb.galera.bootstrap.bootstrapFromNode=M \
    --set datadb.galera.bootstrap.forceSafeToBootstrap=false \
    --set datadb.podManagementPolicy=Parallel
```

CRASH SHUTDOWN

This section follows the official MariaDB 11.3.2 documentation. If you need to perform a disaster recovery, start by restoring the backups for each node.

Data Database **Metadata Database**

```
kubectl -n $NAMESPACE \
    cp \
    backup-data-db-0.tar.gz \
    data-db-0:/bitnami/mariadb/backup.tar.gz
```

Unpack the compressed archive.

```
kubectl -n $NAMESPACE exec pod/data-db-0 -c mariadb-galera -- \
    tar xzfv \
    /bitnami/mariadb/backup.tar.gz && \
    rm -f \
        /bitnami/mariadb/backup.tar.gz
```

Prepare the backup for restore.

```
kubectl -n $NAMESPACE exec pod/data-db-0 -c mariadb-galera -- \
    mariadb-backup \
    --prepare \
    --target-dir=/bitnami/mariadb/backup
```

And ultimately restore the backup.

```
kubectl -n $NAMESPACE exec pod/data-db-0 -c mariadb-galera -- \
    mariadb-backup \
    --move-back \
    --target-dir=/bitnami/mariadb/backup
```

```
kubectl -n $NAMESPACE \
    cp \
    backup-metadata-db-0.tar.gz \
    data-db-0:/bitnami/mariadb/backup.tar.gz
```

Unpack the compressed archive.

```
kubectl -n $NAMESPACE exec pod/metadata-db-0 -c mariadb-galera -- \
    tar xzfv \
    /bitnami/mariadb/backup.tar.gz && \
    rm -f \
        /bitnami/mariadb/backup.tar.gz
```

Prepare the backup for restore.

```
kubectl -n $NAMESPACE exec pod/metadata-db-0 -c mariadb-galera -- \
    mariadb-backup \
    --prepare \
    --target-dir=/bitnami/mariadb/backup
```

And ultimately restore the backup.

```
kubectl -n $NAMESPACE exec pod/metadata-db-0 -c mariadb-galera -- \
    mariadb-backup \
    --move-back \
    --target-dir=/bitnami/mariadb/backup
```

Repeat this for nodes `1` or `2` (or more).

TROUBLESHOOTING

If your restored cluster still fails to reach a quorum (e.g. due to nodes being out of sync and the `SST` mechanism fails to transmit the snapshots) you can try to perform a manual `SST`.

1. https://mariadb.com/docs/galera-cluster/readme/mariadb-galera-cluster-guide#two-nodes-disappear-from-the-cluster ↩

3.5 Troubleshooting

In case the deployment is unsuccessful, we have explanations on their origin and solutions to the most common errors:

Are you trying to mount a directory onto a file (or vice-versa)?

Origin: Docker Compose does not find all files referenced in the `volumes` section of your `docker-compose.yml` file.

Solution: Ensure all mounted files in the `volumes` section of your `docker-compose.yml` exist and have correct file permissions (`0644`) to be found in the filesystem. Note that paths containing directories may not work when using Windows instead of the supported Linux.

The Docker images have been updated but my deployment is not receiving the updates

Origin: Your local Docker image cache is not up-to-date and needs to fetch the remote changes.

Solution: Update your local Docker image cache by executing `docker compose pull`, it automatically downloads all Docker images that have updates. Then apply the new images with `docker compose up -d`.

Error response from daemon: Error starting userland proxy: listen tcp4 0.0.0.0:xyz: bind: address already in use

Origin: Your deployment machine (e.g. laptop, virtual machine) has the port `xyz` already assigned. Some service or application is already listening to this port.

Solution: This service or application needs to be stopped. You can find out the service or application via `sudo netstat -tulpn` (sudo is necessary for the process id) and then stop the service or application gracefully or force a stop via `kill -15 PID` (not recommended).

IllegalArgumentException values less than -1 bytes are not supported

Origin: Your deployment machine (e.g. laptop, virtual machine) appears to not have enough RAM assigned.

Solution: Assign more RAM to the deployment machine (e.g. add vRAM to the virtual machine).

HTTP access denied: user 'admin' - invalid credentials

Origin: The broker service cannot bind to the identity service due to wrong configuration.

Solution: This is very likely due to a wrong `auth_ldap.dn_lookup_bind.password` in `rabbitmq.conf`. The error indicates that LDAP check is not even attempted.

3.6 Updating the Application

This section will be expanded in the future.

4. API

4.1 Overview

The API guide describes entry points that can be used to interact with the repository:

Python API

This API is a convenience wrapper for data scientists and cover most of the REST API.

→ Getting started → Swagger UI

REST API

This API manages the entire system. Every component can be manged using this API.

→ Getting started

AMQP / MQTT API

This API can be used to ingest data from IoT sensors.

→ Getting started

OAI-PMH API

This API lets agents of metadata aggregators index datasets.

→ Getting started

JDBC API

This API is the main connection to the MariaDB databases.

Only available to system administrators.

The REST API manages all of DBRepo. This documentation is also available as Sphinx UI.

Ensure that you use the same Python library version as the target instance. For example: if you see `1.10.0` in the bottom left, you need to use the `1.10.0` Python library.

4.2 Python API

***class* dbrepo.RestClient.RestClient(endpoint: str = 'http://localhost', username: str = None, password: str = None, secure: bool = True)**

The RestClient class for communicating with the DBRepo REST API. All parameters can be set also via environment variables, e.g. set endpoint with REST_API_ENDPOINT, username with REST_API_USERNAME, etc. You can override the constructor parameters with the environment variables.

- **Parameters:**
- **endpoint** – The REST API endpoint. Optional. Default: http://gateway-service
- **username** – The REST API username. Optional.
- **password** – The REST API password. Optional.
- **secure** – When set to false, the requests library will not verify the authenticity of your TLS/SSL certificates (i.e. when using self-signed certificates). Default: True.

ANALYSE_DATATYPES(DATAFRAME: DATAFRAME, ENUM: BOOL = NONE, ENUM_TOL: INT = NONE) → DATATYPEANALYSIS

Import a csv dataset from a file and analyse it for the possible enums, line encoding and column data types.

- **Parameters:**
- **dataframe** – The dataframe.
- **enum** – If set to True, enumerations should be guessed, otherwise no guessing. Optional.
- **enum_tol** – The tolerance for guessing enumerations (ignored if enum=False). Optional.
- **Returns:** The determined data types, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **NotExistsError** – If the file was not found by the Analyse Service.
- **ResponseCodeError** – If something went wrong with the analysis.

ANALYSE_KEYS(DATAFRAME: DATAFRAME) → KEYANALYSIS

Import a csv dataset from a file and analyse it for the possible primary key.

- **Parameters: dataframe** – The dataframe.
- **Returns:** The determined ranking of the primary key candidates, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **NotExistsError** – If the file was not found by the Analyse Service.
- **ResponseCodeError** – If something went wrong with the analysis.

ANALYSE_TABLE_STATISTICS(DATABASE_ID: STR, TABLE_ID: STR) → TABLESTATISTICS

Analyses the numerical contents of a table in a database with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **Returns:** The table statistics, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **NotExistsError** – If the file was not found by the Analyse Service.
- **ServiceConnectionError** – If something went wrong with connection to the metadata service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the analysis.

CREATE_CONTAINER(NAME: STR, HOST: STR, IMAGE_ID: STR, PRIVILEGED_USERNAME: STR, PRIVILEGED_PASSWORD: STR, PORT: INT = NONE, UI_HOST: STR = NONE, UI_PORT: INT = NONE) → CONTAINER

Register a container instance executing a given container image. Note that this does not create a container, but only saves it in the metadata database to be used within DBRepo. The container still needs to be created through e.g. docker run image:tag -d.

- **Parameters:**
- **name** – The container name.
- **host** – The container hostname.
- **image_id** – The container image id.
- **privileged_username** – The container privileged user username.
- **privileged_password** – The container privileged user password.
- **port** – The container port bound to the host. Optional.
- **ui_host** – The container hostname displayed in the user interface. Optional. Default: value of host.
- **ui_port** – The container port displayed in the user interface. Optional. Default: default_port of image.
- **Returns:** The container, if successful.
- **Raises:**
- **MalformedError** – If the payload was rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **NameExistsError** – If a container with this name already exists.
- **ResponseCodeError** – If something went wrong with the retrieval.

CREATE_DATABASE(NAME: STR, CONTAINER_ID: STR, IS_PUBLIC: BOOL = TRUE, IS_SCHEMA_PUBLIC: BOOL = TRUE) → DATABASE

Create a databases in a container with given container id.

- **Parameters:**
- **name** – The name of the database.
- **container_id** – The container id.
- **is_public** – The visibility of the data. If set to True the data will be publicly visible. Optional. Default: True.
- **is_schema_public** – The visibility of the schema metadata. If set to True the schema metadata will be publicly visible. Optional. Default: True.
- **Returns:** The database, if successful.
- **Raises:**
- **MalformedError** – If the payload was rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **QueryStoreError** – If something went wrong with the query store.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the retrieval.

CREATE_DATABASE_ACCESS(DATABASE_ID: STR, USER_ID: STR, TYPE: ACCESSTYPE) → ACCESSTYPE

Create access to a database with given database id and user id.

- **Parameters:**
- **database_id** – The database id.
- **user_id** – The user id.
- **type** – The access type.
- **Returns:** The access type, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database or user does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the data service.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

CREATE_IDENTIFIER(DATABASE_ID: STR, TYPE: IDENTIFIERTYPE, TITLES: LIST[CREATEIDENTIFIERTITLE], PUBLISHER: STR, CREATORS: LIST[CREATEIDENTIFIERCREATOR], PUBLICATION_YEAR: INT, DESCRIPTIONS: LIST[CREATEIDENTIFIERDESCRIPTION] = NONE, FUNDERS: LIST[CREATEIDENTIFIERFUNDER] = NONE, LICENSES: LIST[LICENSE] = NONE, LANGUAGE: LANGUAGE = NONE, SUBSET_ID: STR = NONE, VIEW_ID: STR = NONE, TABLE_ID: STR = NONE, PUBLICATION_DAY: INT = NONE, PUBLICATION_MONTH: INT = NONE, RELATED_IDENTIFIERS: LIST[CREATERELATEDIDENTIFIER] = NONE) → IDENTIFIER

Create an identifier draft.

- **Parameters:**
- **database_id** – The database id of the created identifier.
- **type** – The type of the created identifier.
- **titles** – The titles of the created identifier.
- **publisher** – The publisher of the created identifier.
- **creators** – The creator(s) of the created identifier.
- **publication_year** – The publication year of the created identifier.
- **descriptions** – The description(s) of the created identifier. Optional.
- **funders** – The funders(s) of the created identifier. Optional.
- **licenses** – The license(s) of the created identifier. Optional.
- **language** – The language of the created identifier. Optional.
- **subset_id** – The subset id of the created identifier. Required when type=SUBSET, otherwise invalid. Optional.
- **view_id** – The view id of the created identifier. Required when type=VIEW, otherwise invalid. Optional.
- **table_id** – The table id of the created identifier. Required when type=TABLE, otherwise invalid. Optional.
- **publication_day** – The publication day of the created identifier. Optional.
- **publication_month** – The publication month of the created identifier. Optional.
- **related_identifiers** – The related identifier(s) of the created identifier. Optional.
- **Returns:** The identifier, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database, table/view/subset or user does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the creation of the identifier.

CREATE_SUBSET(DATABASE_ID: STR, QUERY: QUERYDEFINITION, PAGE: INT = 0, SIZE: INT = 10, TIMESTAMP: DATETIME = NONE) → DATAFRAME

Executes a SQL query in a database where the current user has at least read access with given database id. The result set can be paginated with setting page and size (both). Historic data can be queried by setting timestamp.

- **Parameters:**
- **database_id** – The database id.
- **query** – The query definition.
- **page** – The result pagination number. Optional. Default: 0.
- **size** – The result pagination size. Optional. Default: 10.
- **timestamp** – The timestamp at which the data validity is set. Optional. Default: .
- **Returns:** The result set, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database, table or user does not exist.
- **QueryStoreError** – The query store rejected the query.
- **FormatNotAvailable** – The subset query contains non-supported keywords.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

CREATE_TABLE(DATABASE_ID: STR, NAME: STR, IS_PUBLIC: BOOL, IS_SCHEMA_PUBLIC: BOOL, DATAFRAME: DATAFRAME, DESCRIPTION: STR = NONE, WITH_DATA: BOOL = TRUE) → TABLEBRIEF

Updates the database owner of a database with given database id.

- **Parameters:**
- **database_id** – The database id.
- **name** – The name of the created table.
- **is_public** – The visibility of the data. If set to True the data will be publicly visible.
- **is_schema_public** – The visibility of the schema metadata. If set to True the schema metadata will be publicly visible.
- **dataframe** – The pandas dataframe.
- **description** – The description of the created table. Optional.
- **with_data** – If set to True, the data will be included in the new table. Optional. Default: True.
- **Returns:** The table, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database does not exist.
- **NameExistsError** – If a table with this name already exists.
- **ServiceConnectionError** – If something went wrong with connection to the data service.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the creation.

CREATE_TABLE_DATA(DATABASE_ID: STR, TABLE_ID: STR, DATA: DICT) → NONE

Insert data into a table in a database with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **data** – The data dictionary to be inserted into the table with the form column=value of the table.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service (e.g. LOB could not be imported).
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the insert.

CREATE_VIEW(DATABASE_ID: STR, NAME: STR, QUERY: QUERYDEFINITION, IS_PUBLIC: BOOL, IS_SCHEMA_PUBLIC: BOOL) → VIEWBRIEF

Create a view in a database with given database id.

- **Parameters:**
- **database_id** – The database id.
- **name** – The name of the created view.
- **query** – The query definition of the view.
- **is_public** – The visibility of the data. If set to True the data will be publicly visible. Optional. Default: True.
- **is_schema_public** – The visibility of the schema metadata. If set to True the schema metadata will be publicly visible. Optional. Default: True.
- **Returns:** The created view, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database does not exist.
- **ExternalSystemError** – If the mapped view creation query is erroneous.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the retrieval.

DELETE_CONTAINER(CONTAINER_ID: STR) → NONE

Deletes a container with given id. Note that this does not delete the container, but deletes the entry in the metadata database. The container still needs to be removed, e.g. docker container stop hash and then docker container rm hash.

- **Parameters: container_id** – The container id.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **ResponseCodeError** – If something went wrong with the deletion.

DELETE_DATABASE_ACCESS(DATABASE_ID: STR, USER_ID: STR) → NONE

Deletes the access for a user to a database with given database id and user id.

- **Parameters:**
- **database_id** – The database id.
- **user_id** – The user id.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database or user does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the data service.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

DELETE_TABLE(DATABASE_ID: STR, TABLE_ID: STR) → NONE

Delete a table with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the data service.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the deletion.

DELETE_TABLE_DATA(DATABASE_ID: STR, TABLE_ID: STR, KEYS: DICT) → NONE

Delete data in a table in a database with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **keys** – The key dictionary matching the rows in the form column=value.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the deletion.

DELETE_VIEW(DATABASE_ID: STR, VIEW_ID: STR) → NONE

Deletes a view in a database with given database id and view id.

- **Parameters:**
- **database_id** – The database id.
- **view_id** – The view id.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **ExternalSystemError** – If the mapped view deletion query is erroneous.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the deletion.

GET_CONCEPTS() → LIST[CONCEPTBRIEF]

Get list of concepts known to the metadata database.

- **Returns:** List of concepts, if successful.

GET_CONTAINER(CONTAINER_ID: STR) → CONTAINER

Get a container with given id.

- **Returns:** List of containers, if successful.
- **Raises:**
- **NotExistsError** – If the container does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_CONTAINERS() → LIST[CONTAINERBRIEF]

Get all containers.

- **Returns:** List of containers, if successful.
- **Raises:** **ResponseCodeError** – If something went wrong with the retrieval.

GET_DATABASE(DATABASE_ID: STR) → DATABASE

Get a databases with given id.

- **Parameters:** **database_id** – The database id.
- **Returns:** The database, if successful.
- **Raises:**
- **NotExistsError** – If the container does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_DATABASE_ACCESS(DATABASE_ID: STR) → ACCESSTYPE

Get access of a view in a database with given database id and view id.

- **Parameters: database_id** – The database id.
- **Returns:** The access type, if successful.
- **Raises:**
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_DATABASES() → LIST[DATABASEBRIEF]

Get all databases.

- **Returns:** List of databases, if successful.
- **Raises: ResponseCodeError** – If something went wrong with the retrieval.

GET_DATABASES_COUNT() → INT

Count all databases.

- **Returns:** Count of databases if successful.
- **Raises: ResponseCodeError** – If something went wrong with the retrieval.

GET_IDENTIFIER(IDENTIFIER_ID: STR) → IDENTIFIER

Get the identifier by given id.

- **Parameters: identifier_id** – The identifier id.
- **Returns:** The identifier, if successful.
- **Raises:**
- **NotExistsError** – If the identifier does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval of the identifier.

GET_IDENTIFIER_DATA(IDENTIFIER_ID: STR, PAGE: INT = 0, SIZE: INT = 10000) → DATAFRAME

Get the identifier data by given id.

- **Parameters:**
- **identifier_id** – The identifier id.
- **page** – The result pagination number. Optional. Default: 0.
- **size** – The result pagination size. Optional. Default: 10000.
- **Returns:** The identifier, if successful.
- **Raises:**
- **NotExistsError** – If the identifier does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval of the identifier.

GET_IDENTIFIERS(DATABASE_ID: STR = NONE, SUBSET_ID: STR = NONE, VIEW_ID: STR = NONE, TABLE_ID: STR = NONE, TYPE: IDENTIFIERTYPE = NONE, STATUS: IDENTIFIERSTATUSTYPE = NONE) → LIST[IDENTIFIER]

Get list of identifiers, filter by the remaining optional arguments.

- **Parameters:**
- **database_id** – The database id. Optional.
- **subset_id** – The subset id. Optional. Requires database_id to be set.
- **view_id** – The view id. Optional. Requires database_id to be set.
- **table_id** – The table id. Optional. Requires database_id to be set.
- **type** – The identifier type. Optional.
- **status** – The identifier status. Optional.
- **Returns:** List of identifiers, if successful.
- **Raises:**
- **NotExistsError** – If the accept header is neither application/json nor application/ld+json.
- **FormatNotAvailable** – If the service could not represent the output.
- **ResponseCodeError** – If something went wrong with the retrieval of the identifiers.

GET_IMAGE(IMAGE_ID: STR) → IMAGE

Get container image.

- **Parameters: image_id** – The image id.
- **Returns:** The image, if successful.
- **Raises:**
- **NotExistsError** – If the image does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval of the image.

GET_IMAGES() → LIST[IMAGEBRIEF]

Get list of container images.

- **Returns:** List of images, if successful.

GET_LICENSES() → LIST[LICENSE]

Get list of licenses allowed.

- **Returns:** List of licenses, if successful.

GET_MESSAGES() → LIST[MESSAGE]

Get list of messages.

- **Returns:** List of messages, if successful.

GET_ONTOLOGIES() → LIST[ONTOLOGYBRIEF]

Get list of ontologies.

- **Returns:** List of ontologies, if successful.

GET_QUERIES(DATABASE_ID: STR) → LIST[QUERY]

Get queries from a database with given database id.

- **Parameters: database_id** – The database id.
- **Returns:** List of queries, if successful.
- **Raises:**
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database or user does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_SUBSET(DATABASE_ID: STR, SUBSET_ID: STR) → QUERY

Get query from a database with given database id and query id.

- **Parameters:**
- **database_id** – The database id.
- **subset_id** – The subset id.
- **Returns:** The query, if successful.
- **Raises:**
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database, query or user does not exist.
- **FormatNotAvailable** – If the service could not represent the output.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_SUBSET_DATA(DATABASE_ID: STR, SUBSET_ID: STR, PAGE: INT = 0, SIZE: INT = 10) → DATAFRAME

Re-executes a query in a database with given database id and query id.

- **Parameters:**
- **database_id** – The database id.
- **subset_id** – The subset id.
- **page** – The result pagination number. Optional. Default: 0.
- **size** – The result pagination size. Optional. Default: 10.
- **size** – The result pagination size. Optional. Default: 10.
- **Returns:** The subset data, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database, query or user does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_SUBSET_DATA_COUNT(DATABASE_ID: STR, SUBSET_ID: STR) → INT

Re-executes a query in a database with given database id and query id and only counts the results.

- **Parameters:**
- **database_id** – The database id.
- **subset_id** – The subset id.
- **Returns:** The result set, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database, query or user does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_TABLE(DATABASE_ID: STR, TABLE_ID: STR) → TABLE

Get a table with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **Returns:** List of tables, if successful.
- **Raises:**
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_TABLE_DATA(DATABASE_ID: STR, TABLE_ID: STR, PAGE: INT = 0, SIZE: INT = 10, TIMESTAMP: DATETIME = NONE) → DATAFRAME

Get data of a table in a database with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **page** – The result pagination number. Optional. Default: 0.
- **size** – The result pagination size. Optional. Default: 10.
- **timestamp** – The query execution time. Optional.
- **Returns:** The table data, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_TABLE_DATA_COUNT(DATABASE_ID: STR, TABLE_ID: STR, TIMESTAMP: DATETIME = NONE) → INT

Get data count of a table in a database with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **timestamp** – The query execution time. Optional.
- **Returns:** The result of the view query, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ExternalSystemError** – If the mapped view selection query is erroneous.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_TABLE_HISTORY(DATABASE_ID: STR, TABLE_ID: STR, SIZE: INT = 100) → []

Get the table history of insert/delete operations.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **size** – The number of operations. Optional. Default: 100.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_TABLES(DATABASE_ID: STR) → LIST[TABLEBRIEF]

Get all tables.

- **Parameters: database_id** – The database id.
- **Returns:** List of tables, if successful.
- **Raises:**
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_UNITS() → LIST[UNITBRIEF]

Get all units known to the metadata database.

- **Returns:** List of units, if successful.
- **Raises: ResponseCodeError** – If something went wrong with the retrieval.

GET_USER(USER_ID: STR) → USER

Get a user with given user id.

- **Returns:** The user, if successful.
- **Raises:**
- **ResponseCodeError** – If something went wrong with the retrieval.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the user does not exist.

GET_USERS() → LIST[USERBRIEF]

Get all users.

- **Returns:** List of users, if successful.
- **Raises:** **ResponseCodeError** – If something went wrong with the retrieval.

GET_VIEW(DATABASE_ID: STR, VIEW_ID: STR) → VIEW

Get a view of a database with given database id and view id.

- **Parameters:**
- **database_id** – The database id.
- **view_id** – The view id.
- **Returns:** The view, if successful.
- **Raises:**
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_VIEW_DATA(DATABASE_ID: STR, VIEW_ID: STR, PAGE: INT = 0, SIZE: INT = 10) → DATAFRAME

Get data of a view in a database with given database id and view id.

- **Parameters:**
- **database_id** – The database id.
- **view_id** – The view id.
- **page** – The result pagination number. Optional. Default: 0.
- **size** – The result pagination size. Optional. Default: 10.
- **Returns:** The view data, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the view does not exist.
- **ExternalSystemError** – If the mapped view selection query is erroneous.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_VIEW_DATA_COUNT(DATABASE_ID: STR, VIEW_ID: STR) → INT

Get data count of a view in a database with given database id and view id.

- **Parameters:**
- **database_id** – The database id.
- **view_id** – The view id.
- **Returns:** The result count of the view query, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the view does not exist.
- **ExternalSystemError** – If the mapped view selection query is erroneous.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the retrieval.

GET_VIEWS(DATABASE_ID: STR) → LIST[VIEWBRIEF]

Gets views of a database with given database id.

- **Parameters: database_id** – The database id.
- **Returns:** The list of views, if successful.
- **Raises:**
- **NotExistsError** – If the container does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

IMPORT_TABLE_DATA(DATABASE_ID: STR, TABLE_ID: STR, DATAFRAME: DATAFRAME) → NONE

Import a csv dataset from a file into a table in a database with given database id and table id. ATTENTION: the import is column-ordering sensitive! The csv dataset must have the same columns in the same order as the target table.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **dataframe** – The pandas dataframe.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service (e.g. LOB could not be imported).
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the insert.

PUBLISH_IDENTIFIER(IDENTIFIER_ID: STR) → IDENTIFIER

Publish an identifier with given id.

- **Parameters: identifier_id** – The identifier id.
- **Returns:** The identifier, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database, table/view/subset or user does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the creation of the identifier.

UPDATE_DATABASE_ACCESS(DATABASE_ID: STR, USER_ID: STR, TYPE: ACCESSTYPE) → ACCESSTYPE

Updates the access for a user to a database with given database id and user id.

- **Parameters:**
- **database_id** – The database id.
- **user_id** – The user id.
- **type** – The access type.
- **Returns:** The access type, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database or user does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the data service.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

UPDATE_DATABASE_OWNER(DATABASE_ID: STR, USER_ID: STR) → DATABASE

Updates the database owner of a database with given database id.

- **Parameters:**
- **database_id** – The database id.
- **user_id** – The user id of the new owner.
- **Returns:** The database, if successful.
- **Raises:**
- **MalformedError** – If the payload was rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the update.

UPDATE_DATABASE_SCHEMA(DATABASE_ID: STR) → DATABASEBRIEF

Updates the database table and view metadata of a database with given database id.

- **Parameters: database_id** – The database id.
- **Returns:** The updated database, if successful.
- **Raises:**
- **MalformedError** – If the payload was rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the data service.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the update.

UPDATE_DATABASE_VISIBILITY(DATABASE_ID: STR, IS_PUBLIC: BOOL, IS_SCHEMA_PUBLIC: BOOL, IS_DASHBOARD_ENABLED: BOOL) → DATABASE

Updates the database visibility of a database with given database id.

- **Parameters:**
- **database_id** – The database id.
- **is_public** – The visibility of the data. If set to True the data will be publicly visible.
- **is_schema_public** – The visibility of the schema metadata. If set to True the schema metadata will be publicly visible.
- **is_dashboard_enabled** – If set to True, the provisioned dashboard for this database is enabled.
- **Returns:** The database, if successful.
- **Raises:**
- **MalformedError** – If the payload was rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the update.

UPDATE_IDENTIFIER(IDENTIFIER_ID: STR, DATABASE_ID: STR, TYPE: IDENTIFIERTYPE, TITLES: LIST[SAVEIDENTIFIERTITLE], PUBLISHER: STR, CREATORS: LIST[SAVEIDENTIFIERCREATOR], PUBLICATION_YEAR: INT, DESCRIPTIONS: LIST[SAVEIDENTIFIERDESCRIPTION] = NONE, FUNDERS: LIST[SAVEIDENTIFIERFUNDER] = NONE, LICENSES: LIST[LICENSE] = NONE, LANGUAGE: LANGUAGE = NONE, SUBSET_ID: STR = NONE, VIEW_ID: STR = NONE, TABLE_ID: STR = NONE, PUBLICATION_DAY: INT = NONE, PUBLICATION_MONTH: INT = NONE, RELATED_IDENTIFIERS: LIST[SAVERELATEDIDENTIFIER] = NONE) → IDENTIFIER

Update an existing identifier and update the metadata attached to it.

- **Parameters:**
- **identifier_id** – The identifier id.
- **database_id** – The database id of the created identifier.
- **type** – The type of the created identifier.
- **titles** – The titles of the created identifier.
- **publisher** – The publisher of the created identifier.
- **creators** – The creator(s) of the created identifier.
- **publication_year** – The publication year of the created identifier.
- **descriptions** – The description(s) of the created identifier. Optional.
- **funders** – The funders(s) of the created identifier. Optional.
- **licenses** – The license(s) of the created identifier. Optional.
- **language** – The language of the created identifier. Optional.
- **subset_id** – The subset id of the created identifier. Required when type=SUBSET, otherwise invalid. Optional.
- **view_id** – The view id of the created identifier. Required when type=VIEW, otherwise invalid. Optional.
- **table_id** – The table id of the created identifier. Required when type=TABLE, otherwise invalid. Optional.
- **publication_day** – The publication day of the created identifier. Optional.
- **publication_month** – The publication month of the created identifier. Optional.
- **related_identifiers** – The related identifier(s) of the created identifier. Optional.
- **Returns:** The identifier, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database, table/view/subset or user does not exist.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the creation of the identifier.

UPDATE_SUBSET(DATABASE_ID: STR, SUBSET_ID: STR, PERSIST: BOOL) → QUERY

Save query or mark it for deletion (at a later time) in a database with given database id and query id.

- **Parameters:**
- **database_id** – The database id.
- **subset_id** – The subset id.
- **persist** – If set to True, the query will be saved and visible in the user interface, otherwise the query is marked for deletion in the future and not visible in the user interface.
- **Returns:** The query, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the database or user does not exist.
- **QueryStoreError** – The query store rejected the update.
- **ServiceError** – If something went wrong with obtaining the information in the data service.
- **ResponseCodeError** – If something went wrong with the retrieval.

UPDATE_TABLE_COLUMN(DATABASE_ID: STR, TABLE_ID: STR, COLUMN_ID: STR, CONCEPT_URI: STR = NONE, UNIT_URI: STR = NONE) → COLUMN

Update semantic information of a table column by given database id and table id and column id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **column_id** – The column id.
- **concept_uri** – The concept URI. Optional.
- **unit_uri** – The unit URI. Optional.
- **Returns:** The column, if successful.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the accept header is neither application/json nor application/ld+json.
- **ServiceConnectionError** – If something went wrong with connection to the search service.
- **ServiceError** – If something went wrong with obtaining the information in the search service.
- **ResponseCodeError** – If something went wrong with the retrieval of the identifiers.

UPDATE_TABLE_DATA(DATABASE_ID: STR, TABLE_ID: STR, DATA: DICT, KEYS: DICT) → NONE

Update data in a table in a database with given database id and table id.

- **Parameters:**
- **database_id** – The database id.
- **table_id** – The table id.
- **data** – The data dictionary to be updated into the table with the form column=value of the table.
- **keys** – The key dictionary matching the rows in the form column=value.
- **Raises:**
- **MalformedError** – If the payload is rejected by the service (e.g. LOB data could not be imported).
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the table does not exist.
- **ServiceError** – If something went wrong with obtaining the information in the metadata service.
- **ResponseCodeError** – If something went wrong with the update.

UPDATE_USER(USER_ID: STR, THEME: STR, LANGUAGE: STR, FIRSTNAME: STR = NONE, LASTNAME: STR = NONE, AFFILIATION: STR = NONE, ORCID: STR = NONE) → USERBRIEF

Updates a user with given user id.

- **Parameters:**
- **user_id** – The user id of the user that should be updated.
- **theme** – The user theme. One of “light”, “dark”, “light-contrast”, “dark-contrast”.
- **language** – The user language localization. One of “en”, “de”.
- **firstname** – The updated given name. Optional.
- **lastname** – The updated family name. Optional.
- **affiliation** – The updated affiliation identifier. Optional.
- **orcid** – The updated ORCID identifier. Optional.
- **Returns:** The user, if successful.
- **Raises:**
- **MalformedError** – If the payload was rejected by the service.
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the user does not exist.
- **ResponseCodeError** – If something went wrong with the update.

UPDATE_VIEW(DATABASE_ID: STR, VIEW_ID: STR, IS_PUBLIC: BOOL, IS_SCHEMA_PUBLIC: BOOL) → VIEWBRIEF

Get a view of a database with given database id and view id.

- **Parameters:**
- **database_id** – The database id.
- **view_id** – The view id.
- **is_public** – If set to True, the view data is publicly visible.
- **is_schema_public** – If set to True, the view schema is publicly visible.
- **Returns:** The view, if successful.
- **Raises:**
- **ForbiddenError** – If something went wrong with the authorization.
- **NotExistsError** – If the container does not exist.
- **ResponseCodeError** – If something went wrong with the retrieval.

WHOAMI() → STR | NONE

Print the username.

- **Returns:** The username, if set.

4.3 Python AMQP API

***class* dbrepo.AmqpClient.AmqpClient(broker_host: str = 'localhost', broker_port: int = 5672, broker_virtual_host: str = 'dbrepo', username: str = None, password: str = None)**

The AmqpClient class for communicating with the DBRepo AMQP API to import data. All parameters can be set also via environment variables, e.g. set endpoint with DBREPO_ENDPOINT. You can override the constructor parameters with the environment variables.

- **Parameters:**
- **broker_host** – The AMQP API host. Optional. Default: “localhost”.
- **broker_port** – The AMQP API port. Optional. Default: 5672,
- **broker_virtual_host** – The AMQP API virtual host. Optional. Default: “dbrepo”.
- **username** – The AMQP API username. Optional.
- **password** – The AMQP API password. Optional.

PUBLISH(ROUTING_KEY: STR, DATA=, EXCHANGE: STR = 'DBREPO') → NONE

Publishes data to a given exchange with the given routing key with a blocking connection.

- **Parameters:**
- **routing_key** – The routing key.
- **data** – The data.
- **exchange** – The exchange name. Default: “dbrepo”.

The REST API manages all of DBRepo. This documentation is also available as Swagger UI.

4.4 REST API

Version 1.13.3

The merged REST API of DBRepo for users, developers and data stewards to be accessed publicly. Have a look at the source code for non-public endpoints that are used between the services themselves.

4.4.1 Path Table

Method	Path	Description
GET	/api/v1/database/{databaseId}/view/{viewId}/data	Get view data
HEAD	/api/v1/database/{databaseId}/view/{viewId}/data	Get view data
GET	/api/v1/database/{databaseId}/table/{tableId}/data	Get table data
PUT	/api/v1/database/{databaseId}/table/{tableId}/data	Update tuple
POST	/api/v1/database/{databaseId}/table/{tableId}/data	Insert tuple
DELETE	/api/v1/database/{databaseId}/table/{tableId}/data	Delete tuple
HEAD	/api/v1/database/{databaseId}/table/{tableId}/data	Get table data
GET	/api/v1/database/{databaseId}/subset/{subsetId}/data	Get subset data
HEAD	/api/v1/database/{databaseId}/subset/{subsetId}/data	Get subset data
GET	/api/v1/database/{databaseId}/grant/{username}	Get grants
HEAD	/api/v1/database/{databaseId}/grant/{username}	Get grants
PUT	/api/v1/database/{databaseId}/subset/{queryId}	Persist subset
POST	/api/v1/upload	Uploads a multipart file
POST	/api/v1/database/{databaseId}/table/{tableId}/data/import	Import dataset
GET	/api/v1/database/{databaseId}/subset	Find subsets
POST	/api/v1/database/{databaseId}/subset	Create subset
GET	/api/v1/image/{imageId}/analyse/schema/{key}	Analyse schema
GET	/api/v1/database/{databaseId}/table/{tableId}/history	Get history
GET	/api/v1/database/{databaseId}/subset/{subsetId}	Find subset
GET	/api/v1/user/{username}	Get user
PUT	/api/v1/user/{username}	Update user
HEAD	/api/v1/user/{username}	Get user
GET	/api/v1/database	List databases
POST	/api/v1/database	Create database
HEAD	/api/v1/database	List databases
GET	/api/v1/database/{databaseId}/access/{username}	Find/Check access
PUT	/api/v1/database/{databaseId}/access/{username}	Modify access
POST	/api/v1/database/{databaseId}/access/{username}	Give access
DELETE	/api/v1/database/{databaseId}/access/{username}	Delete access
HEAD	/api/v1/database/{databaseId}/access/{username}	Find/Check access
PUT	/api/v1/message/{messageId}	Update message
DELETE	/api/v1/message/{messageId}	Delete message
GET	/api/v1/image/{imageId}	Find image
PUT	/api/v1/image/{imageId}	Update image

Method	Path	Description
DELETE	/api/v1/image/{imageId}	Delete image
GET	/api/v1/identifier/{identifierId}	Find identifier
PUT	/api/v1/identifier/{identifierId}	Save identifier
DELETE	/api/v1/identifier/{identifierId}	Delete identifier
PUT	/api/v1/identifier/{identifierId}/publish	Publish identifier
PUT	/api/v1/database/{databaseId}/visibility	Update database visibility
GET	/api/v1/database/{databaseId}/view/{viewId}	Get view
PUT	/api/v1/database/{databaseId}/view/{viewId}	Update view
DELETE	/api/v1/database/{databaseId}/view/{viewId}	Delete view
GET	/api/v1/database/{databaseId}/table/{tableId}	Find table
PUT	/api/v1/database/{databaseId}/table/{tableId}	Update table
DELETE	/api/v1/database/{databaseId}/table/{tableId}	Delete table
PUT	/api/v1/database/{databaseId}/table/{tableId}/statistic	Update statistics
PUT	/api/v1/database/{databaseId}/table/{tableId}/column/{columnId}	Update semantics
PUT	/api/v1/database/{databaseId}/owner	Update database owner
PUT	/api/v1/database/{databaseId}/metadata/view	Update database view schemas
PUT	/api/v1/database/{databaseId}/metadata/table	Update database table schemas
GET	/api/v1/database/{databaseId}/image	Get database preview image
PUT	/api/v1/database/{databaseId}/image	Update database preview image
GET	/api/v1/message	List messages
POST	/api/v1/message	Create message
GET	/api/v1/image	List images
POST	/api/v1/image	Create image
GET	/api/v1/identifier	List identifiers
POST	/api/v1/identifier	Create identifier
GET	/api/v1/database/{databaseId}/view	List views
POST	/api/v1/database/{databaseId}/view	Create view
GET	/api/v1/database/{databaseId}/table	List tables
POST	/api/v1/database/{databaseId}/table	Create table
GET	/api/v1/container	List containers
POST	/api/v1/container	Create container
GET	/api/v1/user	List users
GET	/api/v1/oai	Get record
GET	/api/v1/message/message/{messageId}	Find message

Method	Path	Description
GET	/api/v1/license	List licenses
GET	/api/v1/identifier/retrieve	Retrieve PID metadata
GET	/api/v1/database/{databaseId}	Find database
GET	/api/v1/container/{containerId}	Find container
DELETE	/api/v1/container/{containerId}	Delete container
GET	/api/v1/search	Performs a fuzzy search
POST	/api/v1/search/{field_type}	Performs a general search
GET	/api/v1/search/{field_type}/fields	Get searchable fields
GET	/api/v1/search/{index}	Gets the index

4.4.2 Reference Table

Name	Path	Description
basicAuth	#/components/securitySchemes/basicAuth	
bearerAuth	#/components/securitySchemes/bearerAuth	
DatabaseAccessDto	#/components/schemas/DatabaseAccessDto	
UserBriefDto	#/components/schemas/UserBriefDto	
TupleUpdateDto	#/components/schemas/TupleUpdateDto	
QueryPersistDto	#/components/schemas/QueryPersistDto	
CreatorBriefDto	#/components/schemas/CreatorBriefDto	
IdentifierBriefDto	#/components/schemas/IdentifierBriefDto	
IdentifierDescriptionDto	#/components/schemas/IdentifierDescriptionDto	
IdentifierTitleDto	#/components/schemas/IdentifierTitleDto	
QueryDto	#/components/schemas/QueryDto	
TupleDto	#/components/schemas/TupleDto	
ImportDto	#/components/schemas/ImportDto	
ConditionalDto	#/components/schemas/ConditionalDto	
FilterDto	#/components/schemas/FilterDto	
JoinDto	#/components/schemas/JoinDto	
OrderDto	#/components/schemas/OrderDto	
SubsetColumnDto	#/components/schemas/SubsetColumnDto	
SubsetDto	#/components/schemas/SubsetDto	
ColumnAnalysisResultDto	#/components/schemas/ColumnAnalysisResultDto	
SchemaAnalysisResultDto	#/components/schemas/SchemaAnalysisResultDto	
TableHistoryDto	#/components/schemas/TableHistoryDto	
TupleDeleteDto	#/components/schemas/TupleDeleteDto	
UserAttributesDto	#/components/schemas/UserAttributesDto	
UserDto	#/components/schemas/UserDto	
DatabaseBriefDto	#/components/schemas/DatabaseBriefDto	
UserUpdateDto	#/components/schemas/UserUpdateDto	
BannerMessageUpdateDto	#/components/schemas/BannerMessageUpdateDto	
BannerMessageBriefDto	#/components/schemas/BannerMessageBriefDto	
ImageChangeDto	#/components/schemas/ImageChangeDto	
DataTypeDto	#/components/schemas/DataTypeDto	
ImageDto	#/components/schemas/ImageDto	
OperatorDto	#/components/schemas/OperatorDto	
IdentifierSaveDto	#/components/schemas/IdentifierSaveDto	

Name	Path	Description
LicenseDto	#/components/schemas/LicenseDto	
SaveIdentifierCreatorDto	#/components/schemas/SaveIdentifierCreatorDto	
SaveIdentifierDescriptionDto	#/components/schemas/SaveIdentifierDescriptionDto	
SaveIdentifierFunderDto	#/components/schemas/SaveIdentifierFunderDto	
SaveIdentifierTitleDto	#/components/schemas/SaveIdentifierTitleDto	
SaveRelatedIdentifierDto	#/components/schemas/SaveRelatedIdentifierDto	
CreatorDto	#/components/schemas/CreatorDto	
IdentifierDto	#/components/schemas/IdentifierDto	
IdentifierFunderDto	#/components/schemas/IdentifierFunderDto	
LinksDto	#/components/schemas/LinksDto	
RelatedIdentifierDto	#/components/schemas/RelatedIdentifierDto	
DatabaseModifyVisibilityDto	#/components/schemas/DatabaseModifyVisibilityDto	
ViewUpdateDto	#/components/schemas/ViewUpdateDto	
ViewBriefDto	#/components/schemas/ViewBriefDto	
TableUpdateDto	#/components/schemas/TableUpdateDto	
TableBriefDto	#/components/schemas/TableBriefDto	
ColumnSemanticsUpdateDto	#/components/schemas/ColumnSemanticsUpdateDto	
ColumnDto	#/components/schemas/ColumnDto	
EnumDto	#/components/schemas/EnumDto	
SetDto	#/components/schemas/SetDto	
DatabaseTransferDto	#/components/schemas/DatabaseTransferDto	
DatabaseModifyImageDto	#/components/schemas/DatabaseModifyImageDto	
CreateAccessDto	#/components/schemas/CreateAccessDto	
BannerMessageCreateDto	#/components/schemas/BannerMessageCreateDto	
ImageCreateDto	#/components/schemas/ImageCreateDto	
CreateIdentifierCreatorDto	#/components/schemas/CreateIdentifierCreatorDto	
CreateIdentifierDescriptionDto	#/components/schemas/CreateIdentifierDescriptionDto	
CreateIdentifierDto	#/components/schemas/CreateIdentifierDto	
CreateIdentifierFunderDto	#/components/schemas/CreateIdentifierFunderDto	
CreateIdentifierTitleDto	#/components/schemas/CreateIdentifierTitleDto	
CreateRelatedIdentifierDto	#/components/schemas/CreateRelatedIdentifierDto	
CreateDatabaseDto	#/components/schemas/CreateDatabaseDto	
CreateViewDto	#/components/schemas/CreateViewDto	
CreateForeignKeyDto	#/components/schemas/CreateForeignKeyDto	

Name	Path	Description
CreateTableColumnDto	#/components/schemas/CreateTableColumnDto	
CreateTableConstraintsDto	#/components/schemas/CreateTableConstraintsDto	
CreateTableDto	#/components/schemas/CreateTableDto	
CreateContainerDto	#/components/schemas/CreateContainerDto	
ContainerDto	#/components/schemas/ContainerDto	
ApiErrorDto	#/components/schemas/ApiErrorDto	
OaiListRecordsParameters	#/components/schemas/OaiListRecordsParameters	
BannerMessageDto	#/components/schemas/BannerMessageDto	
ImageBriefDto	#/components/schemas/ImageBriefDto	
LdCreatorDto	#/components/schemas/LdCreatorDto	
LdDatasetDto	#/components/schemas/LdDatasetDto	
ViewColumnDto	#/components/schemas/ViewColumnDto	
ViewDto	#/components/schemas/ViewDto	
ColumnBriefDto	#/components/schemas/ColumnBriefDto	
ConstraintsDto	#/components/schemas/ConstraintsDto	
ForeignKeyBriefDto	#/components/schemas/ForeignKeyBriefDto	
ForeignKeyDto	#/components/schemas/ForeignKeyDto	
ForeignKeyReferenceDto	#/components/schemas/ForeignKeyReferenceDto	
PrimaryKeyDto	#/components/schemas/PrimaryKeyDto	
TableDto	#/components/schemas/TableDto	
UniqueDto	#/components/schemas/UniqueDto	
ContainerBriefDto	#/components/schemas/ContainerBriefDto	
ApiError	#/components/schemas/ApiError	
IndexDto	#/components/schemas/IndexDto	
IndexFieldDto	#/components/schemas/IndexFieldDto	
IndexFieldsDto	#/components/schemas/IndexFieldsDto	
SearchRequestDto	#/components/schemas/SearchRequestDto	

4.4.3 Path Details

[GET]/api/v1/database/{databaseId}/view/{viewId}/data

- Summary
  Get view data
- Description
  Gets data from a view of a database. For private databases, the user needs at least *READ* access to the associated database.

- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
page?: integer
```

```
size?: integer
```

```
timestamp?: string
```

HEADERS

```
Accept: string
```

RESPONSES

- 200 Retrieved view data

`application/json`

```
{
  "type": "string"
}
```

`text/csv`

- 400 Request pagination is malformed
- 403 Not allowed to retrieve view data
- 404 Failed to find view in metadata database
- 406 Failed to format data
- 409 View schema could not be mapped
- 503 Failed to establish connection with the metadata service

---

[HEAD]/api/v1/database/{databaseId}/view/{viewId}/data

- Summary
  Get view data
- Description
  Gets data from a view of a database. For private databases, the user needs at least *READ* access to the associated database.
- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
page?: integer
```

```
size?: integer
```

```
timestamp?: string
```

HEADERS

```
Accept: string
```

RESPONSES

- 200 Retrieved view data

`application/json`

```
{
  "type": "string"
}
```

`text/csv`

- 400 Request pagination is malformed
- 403 Not allowed to retrieve view data
- 404 Failed to find view in metadata database
- 406 Failed to format data
- 409 View schema could not be mapped
- 503 Failed to establish connection with the metadata service

---

[GET]/api/v1/database/{databaseId}/table/{tableId}/data

- Summary
  Get table data
- Description
  Gets data from a table with id. For a table in a private database, the user needs to have at least *READ* access to the associated database. Requests with HTTP method **GET** return the full dataset, requests with HTTP method **HEAD** only the number of tuples in the `X-Count` header.
- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
timestamp?: string
```

```
page?: integer
```

```
size?: integer
```

HEADERS

```
Accept: string
```

RESPONSES

- 200 Get table data

`application/json`

```
{
  "type": "string"
}
```

`text/csv`

- 400 Request pagination or table data select query is malformed
- 403 Not allowed to get table data
- 404 Failed to find table in metadata database
- 406 Failed to format data
- 503 Failed to establish connection with the metadata service

---

[PUT]/api/v1/database/{databaseId}/table/{tableId}/data

- Summary
  Update tuple
- Description
  Updates a data tuple into a table, then the table statistics are updated. The user needs to have at least *WRITE_OWN* access to the associated database. Requires role `insert-table-data`.
- Security
  basicAuth
  bearerAuth

REQUESTBODY

- application/json

```
{
  // The key-value data map
  data: {
  }
  // The map of conditions
  keys: {
  }
}
```

RESPONSES

- 202 Updated table data
- 400 Request pagination or table data select query is malformed
- 403 Update table data not allowed
- 404 Failed to find table in metadata database
- 503 Failed to establish connection with the metadata service

---

[POST]/api/v1/database/{databaseId}/table/{tableId}/data

- Summary
  Insert tuple
- Description
  Inserts a data tuple into a table, then the table statistics are updated. The user needs to have at least *WRITE_OWN* access to the associated database. Requires role `insert-table-data`.
- Security
  basicAuth
  bearerAuth

REQUESTBODY

- application/json

```
{
  // The key-value data map
  data: {
  }
}
```

RESPONSES

- 201 Created table data
- 400 Request pagination or table data select query is malformed
- 403 Create table data not allowed
- 404 Failed to find table in metadata database or blob in storage service
- 503 Failed to establish connection with the metadata service or storage service

---

[DELETE]/api/v1/database/{databaseId}/table/{tableId}/data

- Summary
  Delete tuple
- Description
  Deletes a data tuple into a table, then the table statistics are updated. The user needs to have at least *WRITE_OWN* access to the associated database. Requires role `delete-table-data`.
- Security
  basicAuth
  bearerAuth

REQUESTBODY

- application/json

```
{
  // The map of conditions
  keys: {
  }
}
```

RESPONSES

- 202 Deleted table data
- 400 Request pagination or table data select query is malformed
- 403 Delete table data not allowed
- 404 Failed to find table in metadata database
- 503 Failed to establish connection with the metadata service

---

[HEAD]/api/v1/database/{databaseId}/table/{tableId}/data

- Summary
  Get table data
- Description
  Gets data from a table with id. For a table in a private database, the user needs to have at least *READ* access to the associated database. Requests with HTTP method **GET** return the full dataset, requests with HTTP method **HEAD** only the number of tuples in the `X-Count` header.

- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
timestamp?: string
```

```
page?: integer
```

```
size?: integer
```

HEADERS

```
Accept: string
```

RESPONSES

- 200 Get table data

`application/json`

```
{
  "type": "string"
}
```

`text/csv`

- 400 Request pagination or table data select query is malformed
- 403 Not allowed to get table data
- 404 Failed to find table in metadata database
- 406 Failed to format data
- 503 Failed to establish connection with the metadata service

---

[GET]/api/v1/database/{databaseId}/subset/{subsetId}/data

- Summary
  Get subset data
- Description
  Gets data of subset with id. For private databases, the user needs at least *READ* grant to the associated database. Requests with HTTP method **GET** return the subset dataset. Requests with HTTP method **HEAD** only the number of rows in the subset dataset in the `X-Count` header, the subset id in the `X-Id` header and the `sha256`-result set hash in the `X-Result-Hash` header. Requests with HTTP method **GET** additionally return the result set in the response body.
- Security
  bearerAuth
  basicAuth

PARAMETERS(QUERY)

```
timestamp?: string
```

```
page?: integer
```

```
size?: integer
```

HEADERS

```
Accept: string
```

RESPONSES

- 200 Retrieved subset data

`application/json`

```
{
  "type": "string"
}
```

`text/csv`

- 400 Invalid pagination
- 403 Not allowed to retrieve subset data
- 404 Failed to find database in metadata database or query in query store of the data database
- 406 Failed to format data
- 422 Failed to re-execute query
- 503 Failed to communicate with database

[HEAD]/api/v1/database/{databaseId}/subset/{subsetId}/data

- Summary
  Get subset data
- Description
  Gets data of subset with id. For private databases, the user needs at least *READ* grant to the associated database. Requests with HTTP method **GET** return the subset dataset. Requests with HTTP method **HEAD** only the number of rows in the subset dataset in the `X-Count` header, the subset id in the `X-Id` header and the `sha256`-result set hash in the `X-Result-Hash` header. Requests with HTTP method **GET** additionally return the result set in the response body.
- Security
  bearerAuth
  basicAuth

PARAMETERS(QUERY)

```
timestamp?: string
```

```
page?: integer
```

```
size?: integer
```

HEADERS

```
Accept: string
```

RESPONSES

- 200 Retrieved subset data

`application/json`

```
{
  "type": "string"
}
```

`text/csv`

- 400 Invalid pagination
- 403 Not allowed to retrieve subset data
- 404 Failed to find database in metadata database or query in query store of the data database
- 406 Failed to format data
- 422 Failed to re-execute query
- 503 Failed to communicate with database

[GET]/api/v1/database/{databaseId}/grant/{username}

- Summary
  Get grants
- Description
  Get the grant permissions for a user of a given database.
- Security
  basicAuth
  bearerAuth

RESPONSES

- 200 Access found

`application/json`

```
{
  // The user name
  username: string
  user: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  type: enum[read, write_own, write_all]
}[]
```

- 401 Not authenticated
- 403 Not database owner or foreign user
- 404 Failed to find access
- 409 Grants malformed

[HEAD]/api/v1/database/{databaseId}/grant/{username}

- Summary
  Get grants
- Description
  Get the grant permissions for a user of a given database.
- Security
  basicAuth
  bearerAuth

RESPONSES

- 200 Access found

`application/json`

```
{
  // The user name
  username: string
  user: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  type: enum[read, write_own, write_all]
}[]
```

- 401 Not authenticated
- 403 Not database owner or foreign user
- 404 Failed to find access
- 409 Grants malformed

[PUT]/api/v1/database/{databaseId}/subset/{queryId}

- Summary
  Persist subset
- Description
  Persists a subset with id. Requires role `persist-query`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // If false, the query is marked for deletion at a later point in time
  persist: boolean
}
```

RESPONSES

- 202 Persisted subset

`application/json`

```
{
  // The query id
  id: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The timestamp when the query was executed
  execution: string
  // The mapped SQL query
  query: string
  // The query type
  type?: enum[query, view]
```

```
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The database id
  database_id: string
  // The normalized SQL query as executed
  query_normalized: string
  // The sha256-hash of the mapped query
  query_hash: string
  // The sha256-hash of the result
  result_hash?: string
  // The row count of the result
  result_number?: integer
  // If false, the query is marked for deletion at a later point in time
  is_persisted: boolean
}
```


- 400 Malformed select query
- 403 Not allowed to persist subset
- 404 Failed to find database in metadata database or query in query store of the data database
- 417 Failed to persist subset
- 503 Failed to communicate with database

[POST]/api/v1/upload

- Summary
  Uploads a multipart file
- Description
  Uploads a multipart file to the Storage Service. Requires role `upload-file`.
- Security
  basicAuth
  bearerAuth

REQUESTBODY

- application/json

```
{
  file: string
}
```

RESPONSES

- 201 Uploaded the file
- 204 File already present
- 503 Failed to establish connection with the storage service

---

[POST]/api/v1/database/{databaseId}/table/{tableId}/data/import

- Summary
  Import dataset
- Description
  Imports a dataset in a table. Then update the table statistics. The user needs to have at least *WRITE_OWN* access to the associated database when importing into a owned table. Otherwise *WRITE_ALL* access in needed. Requires role `insert-table-data`.
- Security
  basicAuth
  bearerAuth

HEADERS

```
Authorization: string
```

REQUESTBODY

- application/json

```
{
  // The key of the S3 binary object in the storage service
  location: string
  // If true, the first line contains the column names, otherwise it contains only data
  header: boolean
  // The column delimiter of the dataset
  separator: string
  // The quote symbol around values in the dataset
  quote?: string
  // The newline symbol
  line_termination?: string
}
```

RESPONSES

- 202 Imported dataset successfully
- 400 Dataset and/or query are malformed
- 403 Import table dataset not allowed
- 404 Failed to find table in metadata database
- 503 Failed to establish connection with the metadata service

---

[GET]/api/v1/database/{databaseId}/subset

- Summary
  Find subsets

- Description
  Finds subsets in the query store. When the database schema is marked as hidden, the user needs to be authorized, have at least read-access to the database. The result can be optionally filtered by setting `persisted`. When set to *true*, only persisted queries are returned, otherwise only non-persisted queries are returned.
- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
persisted?: boolean
```

RESPONSES

- 200 Found subsets

`application/json`

```
{
  // The query id
  id: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The timestamp when the query was executed
  execution: string
  // The mapped SQL query
  query: string
  // The query type
  type?: enum[query, view]
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The database id
  database_id: string
  // The normalized SQL query as executed
```

```
  query_normalized: string
  // The sha256-hash of the mapped query
  query_hash: string
  // The sha256-hash of the result
  result_hash?: string
  // The row count of the result
  result_number?: integer
  // If false, the query is marked for deletion at a later point in time
  is_persisted: boolean
}[]
```

- 403 Not allowed to find subsets
- 404 Failed to find database or user in metadata database or query in query store of the data database
- 503 Failed to communicate with database

[POST]/api/v1/database/{databaseId}/subset

- Summary
  Create subset
- Description
  Creates a subset in the query store of the data database. Can also be used without authentication if (and only if) the database is marked as public (i.e. when `is_public` = `is_schema_public` is set to `true`). Otherwise at least read access is required.
- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
timestamp?: string
```

```
page?: integer
```

```
size?: integer
```

REQUESTBODY

- application/json

```
{
  columns: {
    // The id of the column
    id: string
    // The column alias
    alias?: string
  }[]
  joins: {
    // The type of join
    type: enum[inner, left, right, cross]
    conditionals: {
      // The id of the column
      column_id: string
      // The id of the foreign column
      foreign_column_id: string
    }[]
    // The id of the data source
    datasource_id: string
  }[]
  filters: {
    // The filter type
    type: enum[where, or, and]
    // The filter value
    value?: string
    // The column id
    column_id: string
    // The operator id
    operator_id: string
  }[]
  orders: {
    // The sort direction
    direction?: enum[asc, desc]
    // The column id
    column_id: string
  }[]
```

```
  datasource_ids?: string[]
}
```

RESPONSES

- 201 Created subset

`application/json`

```
{
  "type": "string"
}
```

- 400 Malformed select query
- 403 Not allowed to find subset
- 404 Failed to find database in metadata database or query in query store of the data database
- 406 Failed to format data
- 417 Failed to insert query into query store of data database
- 422 Failed to execute query
- 501 Failed to execute query as it contains non-supported keywords
- 503 Failed to communicate with database

[GET]/api/v1/image/{imageId}/analyse/schema/{key}

- Summary
  Analyse schema
- Description
  Analyses a dataset stored at the Storage Service and attempts to map the datatypes, requires role `analyse-datatypes`.
- Security
  basicAuth

RESPONSES

- 200 Analysed schema successfully

`application/json`

```
{
  // The column delimiter of the dataset
  delimiter: string
  // The quote symbol around values in the dataset
  quote: string
  // The escape symbol
  escape: string
  // The comment symbol
  comment: string
  columns: {
    // The column name
    name: string
    // The column data type
    datatype: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit,
tinyint, bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    // The size determines the number of digits before the comma: x=size-d where size >= d
    size?: integer
    // The digits behind the comma
    d?: integer
    enums?: string[]
    sets?: string[]
    // If set to true, the column value can be null
    null_allowed?: boolean
    // The column is a candidate to be part of a composite primary key
    primary_key?: boolean
  }[]
  // The newline symbol
  newline_delimiter: string
  // The number of rows to skip
  skip_rows: integer
  // Detected the first line as header
```

```
  has_header: boolean
  // The format of the date
  date_format?: string
  // The format of the timestamp
  timestamp_format?: string
}
```

- 400 Schema is malformed or does not fit the image
- 404 Failed to find image or dataset
- 503 Failed to establish connection to metadata service

[GET]/api/v1/database/{databaseId}/table/{tableId}/history

- Summary
  Get history
- Description
  Gets the insert/delete operations history performed. For tables in private databases, the user needs to have at least *READ* access to the associated database.
- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
size?: integer
```

RESPONSES

- 200 Found table history

`application/json`

```
{
  // The timestamp
  timestamp: string
  // The event type
  event: enum[insert, delete]
  // The total number of rows affected by the event
  total: integer
}[]
```

- 400 Invalid pagination size request, must be > 0
- 403 Find table history not allowed
- 404 Failed to find table history in data database
- 503 Failed to establish connection with the metadata service

---

[GET]/api/v1/database/{databaseId}/subset/{subsetId}

- Summary
  Find subset
- Description
  Finds a subset in the data database. When the database schema is marked as hidden, the user needs to be authorized, have at least read-access to the database. Requests with HTTP header `Accept=application/json` return the metadata, requests with HTTP header `Accept=text/csv` return the data as downloadable file.
- Security
  basicAuth
  bearerAuth

PARAMETERS(QUERY)

```
timestamp?: string
```

RESPONSES

- 200 Found subset

`application/json`

```
{
  // The query id
  id: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The timestamp when the query was executed
  execution: string
  // The mapped SQL query
  query: string
  // The query type
  type?: enum[query, view]
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
```

```
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The database id
  database_id: string
  // The normalized SQL query as executed
  query_normalized: string
  // The sha256-hash of the mapped query
  query_hash: string
  // The sha256-hash of the result
  result_hash?: string
  // The row count of the result
  result_number?: integer
  // If false, the query is marked for deletion at a later point in time
  is_persisted: boolean
}
```


`text/csv`

- 400 Malformed select query
- 403 Not allowed to find subset
- 404 Failed to find database in metadata database or query in query store of the data database
- 406 Failed to find acceptable representation
- 503 Failed to communicate with database

[GET]/api/v1/user/{username}

- Summary
  Get user
- Description
  Gets own user information from the metadata database. Requires authentication. Foreign user information can only be obtained if additional role `find-foreign-user` is present. Finding information about internal users results in a 404 error.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Found user

`application/json`


```
{
  id?: string
  name?: string
  username: string
  password?: string
  attributes: {
    theme: string
    orcid?: string
    affiliation?: string
    language: string
  }
  qualified_name?: string
  given_name?: string
  family_name?: string
}
```


- 403 Find user is not permitted
- 404 User was not found

[PUT]/api/v1/user/{username}

- Summary
  Update user
- Description
  Updates user with given username. Requires role `modify-user-information`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  firstname?: string
  lastname?: string
  affiliation?: string
  orcid?: string
  theme: string
  language: string
}
```

RESPONSES

- 202 Modified user information

`application/json`

```
{
  id?: string
  name?: string
  username: string
  password?: string
  attributes: {
    theme: string
    orcid?: string
    affiliation?: string
    language: string
  }
  qualified_name?: string
  given_name?: string
  family_name?: string
}
```

- 400 Modify user query is malformed
- 403 Not allowed to modify user metadata
- 404 Failed to find database/user in metadata database
- 503 Failed to modify user at auth service

[HEAD]/api/v1/user/{username}

- Summary
  Get user
- Description
  Gets own user information from the metadata database. Requires authentication. Foreign user information can only be obtained if additional role `find-foreign-user` is present. Finding information about internal users results in a 404 error.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Found user

`application/json`


```
{
  id?: string
  name?: string
  username: string
  password?: string
  attributes: {
    theme: string
    orcid?: string
    affiliation?: string
    language: string
  }
  qualified_name?: string
  given_name?: string
  family_name?: string
}
```


- 403 Find user is not permitted
- 404 User was not found

[GET]/api/v1/database

- Summary
  List databases
- Description
  Lists all databases in the metadata database. Requests with HTTP method **GET** return the list of databases, requests with HTTP method **HEAD** only the number in the `X-Count` header.

PARAMETERS(QUERY)


```
internal_name?: string
```


RESPONSES

- 200 List of databases

`application/json`


```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
```

```
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}[]
```

[POST]/api/v1/database

- Summary
  Create database
- Description
  Creates a database in the container with id. Requires roles `create-database`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The name
  name: string
  // The container id
  container_id: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

RESPONSES

- 201 Created a new database

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
```

```
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}
```

- 400 Database create query is malformed or image is not supported
- 403 Database create permission is missing or grant permissions at broker service failed
- 404 Failed to fin container/user/database in metadata database
- 409 Query store could not be created
- 423 Database quota exceeded
- 502 Connection to search service failed
- 503 Failed to save in search service

[HEAD]/api/v1/database

- Summary
  List databases
- Description
  Lists all databases in the metadata database. Requests with HTTP method **GET** return the list of databases, requests with HTTP method **HEAD** only the number in the `X-Count` header.

PARAMETERS(QUERY)

```
internal_name?: string
```

RESPONSES

- 200 List of databases

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}[]
```

[GET]/api/v1/database/{databaseId}/access/{username}

- Summary
  Find/Check access
- Description
  Finds or checks access of a user with given username to a database with given id. Requests with HTTP method **GET** return the access object, requests with HTTP method **HEAD** only the status. When the user has at least *READ* access, the status 200 is returned, 403 otherwise. Requires role `check-database-access` or `check-foreign-database-access`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Found database access

`application/json`

```
{
  // The user name
  username: string
  user: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  type: enum[read, write_own, write_all]
}
```

- 403 No access to this database
- 404 Database not found

[PUT]/api/v1/database/{databaseId}/access/{username}

- Summary
  Modify access
- Description
  Modifies access of a user with given username to database with given id. Requires role `update-database-access`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The access type
  type: enum[read, write_own, write_all]
}
```

RESPONSES

- 202 Modified access
- 400 Modify access query or database connection is malformed
- 403 Modify access not permitted when no access is granted in the first place

- 404 Database or user not found
- 502 Access could not be updated due to connection error in the data service
- 503 Access could not be updated in the data service

[POST]/api/v1/database/{databaseId}/access/{username}

- Summary
  Give access
- Description
  Give a user with given username access to some database with given id. Requires role `create-database-access`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The access type
  type: enum[read, write_own, write_all]
}
```

RESPONSES

- 202 Granting access succeeded

`application/json`

```
{
  // The user name
  username: string
  user: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  type: enum[read, write_own, write_all]
}
```

- 400 Granting access query or database connection is malformed
- 403 Failed giving access
- 404 Database or user not found
- 502 Access could not be created due to connection error
- 503 Access could not be created in the data service

[DELETE]/api/v1/database/{databaseId}/access/{username}

- Summary
  Delete access
- Description
  Delete access of a user with given username to a database with id. Requires role `delete-database-access`.

- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Deleted access
- 400 Modify access query or database connection is malformed
- 403 Revoke of access not permitted as no access was found
- 404 User, database with access was not found
- 502 Access could not be created due to connection error
- 503 Access could not be revoked in the data service

[HEAD]/api/v1/database/{databaseId}/access/{username}

- Summary
  Find/Check access
- Description
  Finds or checks access of a user with given username to a database with given id. Requests with HTTP method **GET** return the access object, requests with HTTP method **HEAD** only the status. When the user has at least *READ* access, the status 200 is returned, 403 otherwise. Requires role `check-database-access` or `check-foreign-database-access`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Found database access

`application/json`

```
{
  // The user name
  username: string
  user: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  type: enum[read, write_own, write_all]
}
```

- 403 No access to this database
- 404 Database not found

[PUT]/api/v1/message/{messageId}

- Summary
  Update message
- Description
  Updates a message with id. Requires role `update-maintenance-message`.

- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
  display_start?: string
  display_end?: string
}
```

RESPONSES

- 202 Updated message

`application/json`

```
{
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
}
```

- 404 Could not find message

[DELETE]/api/v1/message/{messageId}

- Summary
  Delete message
- Description
  Deletes a message with id. Requires role `delete-maintenance-message`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Deleted message

`application/json`

- 404 Could not find message

[GET]/api/v1/image/{imageId}

- Summary
  Find image
- Description
  Finds a container image in the metadata database.

RESPONSES

- 200 Found image

`application/json`

```
{
  // The image id
  id: string
  // The image name
  name: string
  version: string
  operators: {
    // The operator id
    id: string
    // The machine-friendly name of the operator
    value: string
    // The documentation link
    documentation: string
    // The user-friendly name of the operator
    display_name: string
  }[]
  default: boolean
  data_types: {
    // The id of the data type
    id: string
    // The machine-friendly value of the data type
    value: string
    // The documentation link
    documentation: string
    // The human-friendly name of the data type
    display_name: string
    // The minimum size
    size_min?: integer
    // The maximum size
    size_max?: integer
    // The default size
    size_default?: integer
    // If true, the size parameter cannot be empty
    size_required?: boolean
    // The step increment
    size_step?: integer
    // The minimum d
    d_min?: integer
    // The maximum d
    d_max?: integer
    // The default d
    d_default?: integer
    // If true, the d parameter cannot be empty
    d_required?: boolean
    // The d increment
    d_step?: integer
    // The user-friendly description of the data format
    data_hint?: string
    // The user-friendly description of the data type
    type_hint?: string
    // frontend needs to quote this data type
    is_quoted: boolean
    // frontend can build this data type
    is_buildable: boolean
  }[]
}
```

- 404 Image could not be found

[PUT]/api/v1/image/{imageId}

- Summary
  Update image
- Description
  Updates container image in the metadata database. Requires role `modify-image`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The URL of the registry without protocol
  registry: string
  // The SQL dialect class name
  dialect: string
  // The default image port to access the database
  default_port: integer
```

```
  // The driver class name
  driver_class: string
  // The method used by JDBC
  jdbc_method: string
}
```

RESPONSES

- 202 Updated image successfully

`application/json`

```
{
  // The image id
  id: string
  // The image name
  name: string
  version: string
  operators: {
    // The operator id
    id: string
    // The machine-friendly name of the operator
    value: string
    // The documentation link
    documentation: string
    // The user-friendly name of the operator
    display_name: string
  }[]
  default: boolean
  data_types: {
    // The id of the data type
    id: string
    // The machine-friendly value of the data type
    value: string
    // The documentation link
    documentation: string
    // The human-friendly name of the data type
    display_name: string
    // The minimum size
    size_min?: integer
    // The maximum size
    size_max?: integer
    // The default size
    size_default?: integer
    // If true, the size parameter cannot be empty
    size_required?: boolean
    // The step increment
    size_step?: integer
    // The minimum d
    d_min?: integer
    // The maximum d
    d_max?: integer
    // The default d
    d_default?: integer
    // If true, the d parameter cannot be empty
    d_required?: boolean
    // The d increment
    d_step?: integer
    // The user-friendly description of the data format
    data_hint?: string
    // The user-friendly description of the data type
    type_hint?: string
    // frontend needs to quote this data type
    is_quoted: boolean
    // frontend can build this data type
    is_buildable: boolean
  }[]
}
```

- 404 Image could not be found

[DELETE]/api/v1/image/{imageId}

- Summary
  Delete image
- Description
  Deletes a container image in the metadata database. Requires role `delete-image`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Deleted image successfully
- 404 Image could not be found

[GET]/api/v1/identifier/{identifierId}

- Summary
  Find identifier
- Description
  Finds an identifier with id. The response format depends on the HTTP `Accept` header set on the request.

HEADERS

```
Accept: string
```

RESPONSES

- 200 Found identifier successfully

`application/json`

```
{
  id: string
  links: {
    self: string
    data?: string
    self_html: string
    dashboard_html?: string
  }
  type: enum[database, subset, table, view]
  titles: {
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  query: string
```

```
  execution?: string
  doi?: string
  publisher: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    name_identifier_scheme_uri?: string
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    affiliation_identifier_scheme_uri?: string
  }[]
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  query_normalized: string
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
  // query hash in sha512
  query_hash: string
  result_hash?: string
  result_number?: integer
  publication_day?: integer
  publication_month?: integer
  publication_year: integer
}
```

`application/ld+json`

```
{
  // The name
  name: string
  // The description
  description: string
  // The URL
  url: string
  identifier?: string[]
  license?: string
  creator: {
    // The name
    name: string
    // The unambiguous reference to a person's identifier
    sameAs?: string
    // The firstname
    givenName?: string
    // The lastname
    familyName?: string
    // The type
    @type: string
  }[]
  // The ciration URL
  citation: string
  hasPart: {
    // The name
```

```
    name: string
    // The description
    description: string
    // The URL
    url: string
    identifier?: string[]
    license?: string
    creator:#/components/schemas/LdCreatorDto[]
    // The ciration URL
    citation: string
    hasPart:#/components/schemas/LdDatasetDto[]
    // The temporal coverage
    temporalCoverage: string
    // The version
    version: string
    // The context schema URI
    @context: string
    // The type
    @type: string
  }[]
  // The temporal coverage
  temporalCoverage: string
  // The version
  version: string
  // The context schema URI
  @context: string
  // The type
  @type: string
}
```

`text/xml`

`text/bibliography`

`text/bibliography; style=apa`

`text/bibliography; style=ieee`

`text/bibliography; style=bibtex`

- 400 Identifier could not be exported, the requested style is not known
- 403 Not allowed to view identifier
- 404 Identifier could not be found
- 406 Failed to find acceptable representation
- 409 Exported resource was not found
- 410 Failed to retrieve from S3 endpoint
- 502 Connection to data service failed
- 503 Failed to find in data service

[PUT]/api/v1/identifier/{identifierId}

- Summary
  Save identifier
- Description
  Saves an identifier with id as a draft identifier. Identifiers can only be created for objects the user has at least *READ* access in the associated database (requires role `create-identifier`) or for any object in any database (requires role `create-foreign-identifier`).
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  id: string
  type: enum[database, subset, table, view]
  doi?: string
  titles: {
    id: string
    title: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  publisher: string
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    affiliation_identifier?: string
  }[]
  database_id: string
  query_id?: string
  view_id?: string
  table_id?: string
  publication_day?: integer
  publication_month?: integer
  publication_year: integer
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
}
```


RESPONSES

- 202 Saved identifier

`application/json`


```
{
  id: string
  links: {
    self: string
    data?: string
    self_html: string
    dashboard_html?: string
  }
  type: enum[database, subset, table, view]
  titles: {
```

```
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  query: string
  execution?: string
  doi?: string
  publisher: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    name_identifier_scheme_uri?: string
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    affiliation_identifier_scheme_uri?: string
  }[]
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  query_normalized: string
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
  // query hash in sha512
  query_hash: string
  result_hash?: string
  result_number?: integer
  publication_day?: integer
  publication_month?: integer
```

```
  publication_year: integer
}
```

- 400 Identifier form contains invalid request data
- 403 Insufficient access rights or authorities
- 404 Failed to find database, table or view
- 502 Connection to search service failed
- 503 Failed to save in search service

[DELETE]/api/v1/identifier/{identifierId}

- Summary
  Delete identifier
- Description
  Deletes an identifier with id. Requires role `delete-identifier`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Deleted identifier
- 403 Deleting identifier not permitted
- 404 Identifier or database could not be found
- 502 Connection to search service failed
- 503 Failed to delete in search service

[PUT]/api/v1/identifier/{identifierId}/publish

- Summary
  Publish identifier
- Description
  Publishes an identifier with id. A published identifier cannot be changed anymore. Requires role `publish-identifier`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Published identifier

`application/json`

```
{
  id: string
  links: {
    self: string
    data?: string
    self_html: string
    dashboard_html?: string
  }
  type: enum[database, subset, table, view]
  titles: {
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
```

```
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  query: string
  execution?: string
  doi?: string
  publisher: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    name_identifier_scheme_uri?: string
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    affiliation_identifier_scheme_uri?: string
  }[]
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  query_normalized: string
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
  // query hash in sha512
  query_hash: string
  result_hash?: string
  result_number?: integer
  publication_day?: integer
  publication_month?: integer
```

```
  publication_year: integer
}
```

- 400 Identifier form contains invalid request data
- 403 Insufficient access rights or authorities
- 404 Failed to find database, table or view
- 502 Connection to search service failed
- 503 Failed to save in search service

[PUT]/api/v1/database/{databaseId}/visibility

- Summary
  Update database visibility
- Description
  Updates the database with id on the visibility. Only the database owner can perform this operation. Requires role `modify-database-visibility`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the dashboard will be managed
  is_dashboard_enabled: boolean
}
```

RESPONSES

- 202 Visibility modified successfully

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
```

```
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}
```

- 400 The visibility payload is malformed
- 403 Visibility modification is not permitted
- 404 Failed to find database in metadata database
- 502 Connection to search service failed
- 503 Failed to save in search service

[GET]/api/v1/database/{databaseId}/view/{viewId}

- Summary
  Get view
- Description
  Gets a view with id in the metadata database.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Find view successfully

`application/json`

```
{
  // The id
  id: string
  // The user-friendly name
  name: string
  identifiers: {
    id: string
    links: {
      self: string
      data?: string
      self_html: string
```

```
      dashboard_html?: string
    }
    type: enum[database, subset, table, view]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    funders: {
      id: string
      funder_name: string
      funder_identifier?: string
      funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
      scheme_uri?: string
      award_number?: string
      award_title?: string
    }[]
    query: string
    execution?: string
    doi?: string
    publisher: string
    owner: {
      id?: string
      // Only contains lowercase characters
      username: string
      name?: string
      orcid?: string
      qualified_name?: string
      given_name?: string
      family_name?: string
    }
    language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    licenses: {
      // The id
      identifier: string
      // The link to the license such as an SPDX identifier
      uri: string
      // A user-friendly short abstract of key details of the license
      description?: string
    }[]
    creators: {
      id: string
      firstname?: string
      lastname?: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      name_identifier_scheme_uri?: string
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
      affiliation_identifier_scheme_uri?: string
    }[]
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    query_normalized: string
    related_identifiers: {
      id: string
      value: string
      type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
      relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
    }[]
    // query hash in sha512
    query_hash: string
    result_hash?: string
    result_number?: integer
    publication_day?: integer
```

```
    publication_month?: integer
    publication_year: integer
  }[]
  // The SQL statement used to create the view
  query: string
  owner:#/components/schemas/UserBriefDto
  columns: {
    // The id
    id: string
    // The user-friendly column name
    name: string
    // The column size, determines the number of digits before the comma as x=size-d where size >= d
    size?: integer
    // The column d
    d?: integer
    description?: string
    enums: {
      // The enum id
      id: string
      // The enum value
      value: string
    }[]
    sets: {
      // The set id
      id: string
      // The set value
      value: string
    }[]
    // The database id
    database_id: string
    // The ordinal position of the colum to order it
    ord: integer
    // The machine-friendly column name
    internal_name: string
    // The length of the index
    index_length?: integer
    // The length of the total data in the table (index + data)
    length?: integer
    // The column type name
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    is_null_allowed: boolean
  }[]
  // The created timestamp
  created: string
  // The database id
  database_id: string
  // The machine-friendly name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the view is default for the database
  initial_view?: boolean
  // The sha256-hash of the query
  query_hash: string
}
```


- 403 Find view is not permitted
- 404 Database, view or user could not be found

[PUT]/api/v1/database/{databaseId}/view/{viewId}

- Summary
  Update view
- Description
  Updates a view with id. This can only be performed by the view owner or database owner. Requires role `create-database-view`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json


```
{
  // The visibility; if true, The will be displayed publicly and is searchable
```

```
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

RESPONSES

- 202 Update view successfully

`*/*`

```
{
  // The id
  id: string
  // The user-friendly name
  name: string
  // The SQL statement used to create the view
  query: string
  // The database id
  database_id: string
  // The machine-friendly name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the view is default for the database
  initial_view?: boolean
  // The sha256-hash of the query
  query_hash: string
  // The owner username
  owned_by?: string
}
```

- 400 Update view query is malformed
- 403 Update not allowed
- 404 Database or View could not be found
- 502 Connection to search service failed
- 503 Failed to save in search service

[DELETE]/api/v1/database/{databaseId}/view/{viewId}

- Summary
  Delete view
- Description
  Deletes a view with id. Requires role `delete-database-view`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Delete view successfully
- 400 Delete view query is malformed
- 403 Deletion not allowed
- 404 Database, view or user could not be found
- 423 Delete view resulted in an invalid query statement
- 502 Connection to search service failed
- 503 Failed to save in search service

[GET]/api/v1/database/{databaseId}/table/{tableId}

- Summary
  Find table
- Description
  Finds a table with id. When a table is hidden (i.e. when `is_public` is `false`), then the user needs to have at least read access and the role `find-table`. When the `system` role is present, the endpoint responds with additional connection metadata in the header.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Find table successfully

`application/json`

```
{
  // The id
  id: string
  // The user-friendly name
  name: string
  // The comment
  description?: string
  // The alias
  alias?: string
  identifiers: {
    id: string
    links: {
      self: string
      data?: string
      self_html: string
      dashboard_html?: string
    }
    type: enum[database, subset, table, view]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    funders: {
      id: string
      funder_name: string
      funder_identifier?: string
      funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
      scheme_uri?: string
      award_number?: string
      award_title?: string
    }[]
    query: string
    execution?: string
    doi?: string
    publisher: string
    owner: {
      id?: string
      // Only contains lowercase characters
      username: string
      name?: string
      orcid?: string
      qualified_name?: string
      given_name?: string
      family_name?: string
    }
    language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
```

```
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    licenses: {
      // The id
      identifier: string
      // The link to the license such as an SPDX identifier
      uri: string
      // A user-friendly short abstract of key details of the license
      description?: string
    }[]
    creators: {
      id: string
      firstname?: string
      lastname?: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      name_identifier_scheme_uri?: string
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
      affiliation_identifier_scheme_uri?: string
    }[]
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    query_normalized: string
    related_identifiers: {
      id: string
      value: string
      type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
      relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
    }[]
    // query hash in sha512
    query_hash: string
    result_hash?: string
    result_number?: integer
    publication_day?: integer
    publication_month?: integer
    publication_year: integer
  }[]
  owner:#/components/schemas/UserBriefDto
  columns: {
    // The id
    id: string
    // The user-friendly column name
    name: string
    // The data source alias name
    alias?: string
    // The column size, determines the number of digits before the comma as x=size-d where size >= d
    size?: integer
    // The column d
    d?: integer
    // The statistically average numerical value
    mean?: number
    // The statistically most middle numerical value
    median?: number
    description?: string
    enums: {
      // The enum id
      id: string
      // The enum value
      value: string
    }[]
    sets: {
      // The set id
      id: string
      // The set value
      value: string
    }[]
    // The database id
    database_id: string
    // The table id
    table_id: string
    // The ordinal position of the colum to order it
    ord: integer
    // The machine-friendly column name
    internal_name: string
    // The length of the index
    index_length?: integer
    // The length of the total data in the table (index + data)
    length?: integer
    // The column type name
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    // The data length
    data_length?: integer
    // The maximum data length
```

```
    max_data_length?: integer
    // The number of rows
    num_rows?: integer
    // The statistically highest numerical value
    val_min?: number
    // The statistically lowest numerical value
    val_max?: number
    // The statistically determined standard deviation
    std_dev?: number
    // The concept URI
    concept_uri?: string
    // The unit URI
    unit_uri?: string
    is_null_allowed: boolean
  }[]
  constraints: {
    uniques: {
      // The unique key id
      id: string
      // The unique key name
      name: string
      table: {
        // The id
        id: string
        // The user-friendly table name
        name: string
        // The comment
        description?: string
        // The database id
        database_id: string
        // The machine-friendly table name
        internal_name: string
        // If true, The is using data versioning
        is_versioned: boolean
        // The visibility; if true, The will be displayed publicly and is searchable
        is_public: boolean
        // The insights; if true, The schema will be displayed publicly and is searchable
        is_schema_public: boolean
        // The owner username
        owned_by: string
      }
      columns: {
        // The column id
        id: string
        // The column name
        name: string
        // The data source alias name
        alias?: string
        // The database id
        database_id: string
        // The table id
        table_id: string
        // The machine-friendly internal name
        internal_name: string
        // The column type name
        type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit,
tinyint, bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
      }[]
    }[]
    checks?: string[]
    foreign_keys: {
      // The foreign key id
      id?: string
      // The foreign key name
      name: string
      references: {
        // The foreign key reference id
        id?: string
        column:#/components/schemas/ColumnBriefDto
        foreign_key: {
          // The foreign key id
          id?: string
        }
        referenced_column:#/components/schemas/ColumnBriefDto
      }[]
      table:#/components/schemas/TableBriefDto
      referenced_table:#/components/schemas/TableBriefDto
      // The integrity action when updating tuples
      on_update?: enum[restrict, cascade, set_null, no_action, set_default]
      // The integrity action when deleting tuples
      on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
    }[]
    primary_key: {
      // The primary key id
      id?: string
      table:#/components/schemas/TableBriefDto
      column:#/components/schemas/ColumnBriefDto
    }[]
  }
  // The created timestamp
  created: string
  // The database id
  database_id: string
  // The machine-friendly name
```

```
  internal_name: string
  // If true, The is using data versioning
  is_versioned: boolean
  // The queue name
  queue_name: string
  // The queue type
  queue_type?: string
  // The routing key
  routing_key: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // The statistical number of rows
  num_rows?: integer
  // The data length in bytes
  data_length?: integer
  // The maximum data length in bytes
  max_data_length?: integer
  // The average row length in bytes
  avg_row_length?: integer
}
```

- 403 Access to the database is forbidden
- 404 Table, database or container could not be found

[PUT]/api/v1/database/{databaseId}/table/{tableId}

- Summary
  Update table
- Description
  Updates a table in the database with id. Requires role `update-table` or `system`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The comment
  description?: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

RESPONSES

- 202 Updated the table

`application/json`

```
{
  // The id
  id: string
  // The user-friendly table name
  name: string
  // The comment
  description?: string
  // The database id
  database_id: string
  // The machine-friendly table name
  internal_name: string
  // If true, The is using data versioning
  is_versioned: boolean
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // The owner username
```

```
  owned_by: string
}
```

- 400 Update table visibility payload is malformed
- 403 Update table visibility not permitted
- 404 Table could not be found
- 502 Connection to search service failed
- 503 Failed to save in search service

[DELETE]/api/v1/database/{databaseId}/table/{tableId}

- Summary
  Delete table
- Description
  Deletes a table with id. Only the owner of a table can perform this action (requires role `delete-table`) or anyone can delete a table (requires role `delete-foreign-table`).
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Delete table successfully
- 400 Delete table query resulted in an invalid query statement
- 403 Access to the database is forbidden
- 404 Table, database or container could not be found
- 502 Connection to search service failed
- 503 Failed to save in search service

[PUT]/api/v1/database/{databaseId}/table/{tableId}/statistic

- Summary
  Update statistics
- Description
  Updates basic statistical properties (min, max, mean, median, std.dev) for numerical columns in a table with id. This action can only be performed by the table owner. Requires role `update-table-statistic`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Updated table statistics successfully
- 400 Failed to map column statistic to known columns
- 403 Not the owner
- 404 Failed to find database/table in metadata database
- 502 Connection to search service failed
- 503 Failed to save in search service

[PUT]/api/v1/database/{databaseId}/table/{tableId}/column/{columnId}

- Summary
  Update semantics
- Description
  Updates column semantics of a table column with id. Only the table owner with at least *READ* access to the associated database can update the column semantics (requires role `modify-table-column-semantics`) or foreign table columns if role `modify-foreign-table-column-semantics`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The description
  description?: string
  // The URI of the concept
  concept_uri?: string
  // The URI of the unit
  unit_uri?: string
}
```

RESPONSES

- 202 Updated column semantics successfully

`application/json`

```
{
  // The id
  id: string
  // The user-friendly column name
  name: string
  // The data source alias name
  alias?: string
  // The column size, determines the number of digits before the comma as x=size-d where size >= d
  size?: integer
  // The column d
  d?: integer
  // The statistically average numerical value
  mean?: number
  // The statistically most middle numerical value
  median?: number
  description?: string
  enums: {
    // The enum id
    id: string
    // The enum value
    value: string
  }[]
  sets: {
    // The set id
    id: string
    // The set value
    value: string
  }[]
  // The database id
  database_id: string
  // The table id
  table_id: string
  // The ordinal position of the colum to order it
  ord: integer
  // The machine-friendly column name
  internal_name: string
  // The length of the index
  index_length?: integer
  // The length of the total data in the table (index + data)
  length?: integer
  // The column type name
  type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  // The data length
  data_length?: integer
  // The maximum data length
  max_data_length?: integer
```

```
  // The number of rows
  num_rows?: integer
  // The statistically highest numerical value
  val_min?: number
  // The statistically lowest numerical value
  val_max?: number
  // The statistically determined standard deviation
  std_dev?: number
  // The concept URI
  concept_uri?: string
  // The unit URI
  unit_uri?: string
  is_null_allowed: boolean
}
```

- 400 Update semantic concept query is malformed or update unit of measurement query is malformed
- 403 Access to the database is forbidden
- 404 Failed to find user/table/database/ontology in metadata database
- 502 Connection to search service failed
- 503 Failed to save in search service

[PUT]/api/v1/database/{databaseId}/owner

- Summary
  Update database owner
- Description
  Updates the database with id on the owner. Only the database owner can perform this operation. Requires role `modify-database-owner`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The username of the new owner
  username: string
}
```

RESPONSES

- 202 Transfer of ownership was successful

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
```

```
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}
```

- 400 Owner payload is malformed
- 403 Transfer of ownership is not permitted
- 404 Database could not be found
- 502 Connection to search service failed
- 503 Failed to save in search service

[PUT]/api/v1/database/{databaseId}/metadata/view

- Summary
  Update database view schemas
- Description
  Updates the database with id with generated metadata from view that are not yet known to the database. Only the database owner can perform this operation. Requires role `find-database`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Refreshed database views metadata

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}
```


- 403 Refresh view metadata is not permitted
- 404 Failed to find database in metadata database
- 502 Connection to search service failed
- 503 Failed to save in search service

[PUT]/api/v1/database/{databaseId}/metadata/table

- Summary
  Update database table schemas

- Description
  Updates the database with id with generated metadata from tables that are not yet known to the database. Only the database owner can perform this operation. Requires role `find-database`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Refreshed database tables metadata

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
```

```
  preview_image?: string
}
```

- 400 Failed to parse payload at search service
- 403 Not allowed to refresh table metadata
- 404 Failed to find database in metadata database
- 502 Connection to search service failed
- 503 Failed to save in search service

[GET]/api/v1/database/{databaseId}/image

- Summary
  Get database preview image
- Description
  Gets the database with id on the preview image.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 View of image was successful

`*/*`

```
{
  "type": "string",
  "format": "byte"
}
```

- 404 Database or user could not be found

[PUT]/api/v1/database/{databaseId}/image

- Summary
  Update database preview image
- Description
  Updates the database with id on the preview image. Only the database owner can perform this operation. Requires role `modify-database-image`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The key of the S3 binary object in the storage service
  key?: string
}
```

RESPONSES

- 202 Modify of image was successful

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}
```

- 403 Modify of image is not permitted
- 404 Database could not be found
- 410 File was not found in the Storage Service
- 502 Connection to search service failed
- 503 Failed to save in search service

[GET]/api/v1/message

- Summary
  List messages
- Description
  Lists messages known to the metadata database. Messages can be filtered be filtered with the optional `active` parameter. If set to *true*, only active messages (that is, messages whose end time has not been reached) will be returned. Otherwise only inactive messages are returned. If not set, active and inactive messages are returned.

PARAMETERS(QUERY)

```
active?: boolean
```

RESPONSES

- 200 List messages

`application/json`

```
{
  id: string
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
  display_start?: string
  display_end?: string
}[]
```

[POST]/api/v1/message

- Summary
  Create message
- Description
  Creates a message in the metadata database. Requires role `create-maintenance-message`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
  display_start?: string
  display_end?: string
}
```

RESPONSES

- 201 Created message

`application/json`

```
{
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
}
```

[GET]/api/v1/image

- Summary
  List images
- Description
  Lists all container images known to the metadata database.

RESPONSES

- 200 List images

`application/json`

```
{
  // The image id
  id: string
  // The image name
  name: string
  // The image tag
  version: string
  // Marks the image as default
  default: boolean
}[]
```

[POST]/api/v1/image

- Summary
  Create image
- Description
  Creates a container image in the metadata database. Requires role `create-image`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The URL of the registry without protocol
  registry: string
  // The image name
  name: string
  version: string
  // The SQL dialect class name
  dialect: string
  // The default image port to access the database
  default_port: integer
  // The driver class name
  driver_class: string
  // The method used by JDBC
  jdbc_method: string
}
```

RESPONSES

- 201 Created image

`application/json`

```
{
  // The image id
  id: string
  // The image name
  name: string
  version: string
  operators: {
    // The operator id
    id: string
    // The machine-friendly name of the operator
    value: string
```

```
    // The documentation link
    documentation: string
    // The user-friendly name of the operator
    display_name: string
  }[]
  default: boolean
  data_types: {
    // The id of the data type
    id: string
    // The machine-friendly value of the data type
    value: string
    // The documentation link
    documentation: string
    // The human-friendly name of the data type
    display_name: string
    // The minimum size
    size_min?: integer
    // The maximum size
    size_max?: integer
    // The default size
    size_default?: integer
    // If true, the size parameter cannot be empty
    size_required?: boolean
    // The step increment
    size_step?: integer
    // The minimum d
    d_min?: integer
    // The maximum d
    d_max?: integer
    // The default d
    d_default?: integer
    // If true, the d parameter cannot be empty
    d_required?: boolean
    // The d increment
    d_step?: integer
    // The user-friendly description of the data format
    data_hint?: string
    // The user-friendly description of the data type
    type_hint?: string
    // frontend needs to quote this data type
    is_quoted: boolean
    // frontend can build this data type
    is_buildable: boolean
  }[]
}
```

- 400 Image specification is invalid
- 409 Image already exists

[GET]/api/v1/identifier

- Summary
  List identifiers
- Description
  Lists all identifiers known to the metadata database

PARAMETERS(QUERY)

```
type?: enum[database, subset, table, view]
```

```
status?: enum[draft, published]
```

```
dbid?: string
```

```
qid?: string
```

```
vid?: string
```

```
tid?: string
```

HEADERS

```
Accept: string
```

RESPONSES

- 200 Found identifiers successfully

`application/json`

```
{
  id: string
  type: enum[database, subset, table, view]
  creators: {
    id: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
  }[]
  titles: {
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  doi?: string
  publisher: string
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  publication_year: integer
  // The owner username
  owned_by: string
}[][]
```

`application/ld+json`

```
{
  // The name
  name: string
  // The description
  description: string
  // The URL
  url: string
  identifier?: string[]
  license?: string
  creator: {
    // The name
    name: string
    // The unambiguous reference to a person's identifier
    sameAs?: string
    // The firstname
    givenName?: string
    // The lastname
    familyName?: string
    // The type
    @type: string
  }[]
  // The ciration URL
  citation: string
  hasPart:#/components/schemas/LdDatasetDto[]
  // The temporal coverage
  temporalCoverage: string
  // The version
  version: string
  // The context schema URI
  @context: string
  // The type
  @type: string
}[]
```

[POST]/api/v1/identifier

- Summary
  Create identifier
- Description
  Create an identifier with id to create a draft identifier. Identifiers can only be created for objects the user has at least *READ* access in the associated database (requires role `create-identifier`) or for any object in any database (requires role `create-foreign-identifier`).
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The identifier type
  type: enum[database, subset, table, view]
  // The doi persistent identifier with optional https://doi.org/ prefix
  doi?: string
  titles: {
    // The title
    title: string
    // The language
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    // The type
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    // The description value
    description: string
    // The language
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    // The type
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    // The funder name
    funder_name: string
    // The identifier that identifies the funder unambiguously
    funder_identifier?: string
    // The funder type, when the `funder_identifier` is a DOI, the `funder_identifier_type` field must be `Crossref Funder ID`
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    // The scheme URI of the `funder_identifier`
    scheme_uri?: string
    // The award number
    award_number?: string
    // The award title
    award_title?: string
  }[]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  // The publisher
  publisher: string
  // The language
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  creators: {
    // The given name
    firstname?: string
    // The family name
    lastname?: string
    // The affiliation
    affiliation?: string
    // The full name
    creator_name: string
```

```
    // The name type
    name_type?: enum[Personal, Organizational]
    // The persistent identifier that identifies the creator unambiguously
    name_identifier?: string
    // The persistent identifier that identifies the affiliation unambiguously
    affiliation_identifier?: string
  }[]
  // The id
  database_id: string
  // The subset id, is only set when type=`subset`
  query_id?: string
  // The view id, is only set when type=`view`
  view_id?: string
  // The table id, is only set when type=`table`
  table_id?: string
  // The day of publication
  publication_day?: integer
  // The month of publication
  publication_month?: integer
  // The year of publication
  publication_year: integer
  related_identifiers: {
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
}
```

RESPONSES

- 201 Drafted identifier

`application/json`

```
{
  id: string
  links: {
    self: string
    data?: string
    self_html: string
    dashboard_html?: string
  }
  type: enum[database, subset, table, view]
  titles: {
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  query: string
  execution?: string
  doi?: string
  publisher: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
```

```
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    name_identifier_scheme_uri?: string
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    affiliation_identifier_scheme_uri?: string
  }[]
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  query_normalized: string
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
  // query hash in sha512
  query_hash: string
  result_hash?: string
  result_number?: integer
  publication_day?: integer
  publication_month?: integer
  publication_year: integer
}
```

- 400 Identifier form contains invalid request data
- 403 Insufficient access rights or authorities
- 404 Failed to find database, table or view
- 502 Connection to search service failed
- 503 Failed to save in search service

[GET]/api/v1/database/{databaseId}/view

- Summary
  List views
- Description
  Lists views known to the metadata database.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Find views successfully

`application/json`

```
{
  // The id
```

```
  id: string
  // The user-friendly name
  name: string
  // The SQL statement used to create the view
  query: string
  // The database id
  database_id: string
  // The machine-friendly name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the view is default for the database
  initial_view?: boolean
  // The sha256-hash of the query
  query_hash: string
  // The owner username
  owned_by?: string
}[]
```

- 404 Database or user could not be found

[POST]/api/v1/database/{databaseId}/view

- Summary
  Create view
- Description
  Creates a view. This can only be performed by the database owner. Requires role `create-database-view`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The name
  name: string
  query: {
    columns: {
      // The id of the column
      id: string
      // The column alias
      alias?: string
    }[]
    joins: {
      // The type of join
      type: enum[inner, left, right, cross]
      conditionals: {
        // The id of the column
        column_id: string
        // The id of the foreign column
        foreign_column_id: string
      }[]
      // The id of the data source
      datasource_id: string
    }[]
    filters: {
      // The filter type
      type: enum[where, or, and]
      // The filter value
      value?: string
      // The column id
      column_id: string
      // The operator id
      operator_id: string
    }[]
    orders: {
      // The sort direction
      direction?: enum[asc, desc]
      // The column id
      column_id: string
    }[]
    datasource_ids?: string[]
  }
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
```

```
  is_schema_public: boolean
}
```

RESPONSES

- 201 Create view successfully

`application/json`

```
{
  // The id
  id: string
  // The user-friendly name
  name: string
  // The SQL statement used to create the view
  query: string
  // The database id
  database_id: string
  // The machine-friendly name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the view is default for the database
  initial_view?: boolean
  // The sha256-hash of the query
  query_hash: string
  // The owner username
  owned_by?: string
}
```

- 400 Create view query is malformed
- 403 Credentials missing
- 404 Failed to find database/user in metadata database.
- 409 View exists with name
- 423 Create view resulted in an invalid query statement
- 502 Connection to search service failed
- 503 Failed to save in search service

[GET]/api/v1/database/{databaseId}/table

- Summary
  List tables
- Description
  Lists all tables known to the metadata database. When a database has a hidden schema (i.e. when `is_schema_public` is `false`), then the user needs to have at least read access and the role `list-tables`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 List tables

`application/json`

```
{
  // The id
  id: string
  // The user-friendly table name
  name: string
  // The comment
  description?: string
  // The database id
  database_id: string
  // The machine-friendly table name
  internal_name: string
```

```
  // If true, The is using data versioning
  is_versioned: boolean
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // The owner username
  owned_by: string
}[]
```

- 403 List tables not permitted
- 404 Database could not be found

[POST]/api/v1/database/{databaseId}/table

- Summary
  Create table
- Description
  Creates a table in the database with id. Requires role `create-table`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The table name
  name: string
  // The table comment
  description?: string
  columns: {
    // The column name
    name: string
    // The data type
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    // The size determines the number of digits before the comma: x=size-d where size >= d
    size?: integer
    // The digits behind the comma
    d?: integer
    // The column comment
    description?: string
    enums?: string[]
    sets?: string[]
    // The index length
    index_length?: integer
    // If set to true, the column value can be null
    null_allowed: boolean
    // The column concept
    concept_uri?: string
    // The column unit
    unit_uri?: string
  }[]
  constraints: {
    uniques?: string[][]
    checks?: string[]
    foreign_keys: {
      columns?: string[]
      // The name of the foreign table
      referenced_table: string
      referenced_columns?: string[]
      // The integrity action when updating tuples
      on_update?: enum[restrict, cascade, set_null, no_action, set_default]
      // The integrity action when deleting tuples
      on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
    }[]
    primary_key?: string[]
  }
  // The table visibility; if true, the table will be displayed publicly and is searchable
  is_public: boolean
  // The table insights; if true, the table schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

RESPONSES

- 201 Created a new table

`application/json`

```
{
  // The id
  id: string
  // The user-friendly table name
  name: string
  // The comment
  description?: string
  // The database id
  database_id: string
  // The machine-friendly table name
  internal_name: string
  // If true, The is using data versioning
  is_versioned: boolean
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // The owner username
  owned_by: string
}
```

- 400 Create table query is malformed
- 403 Create table not permitted
- 404 Database, container or user could not be found
- 409 Create table conflicts with existing table name
- 502 Connection to search service failed
- 503 Failed to save in search service

[GET]/api/v1/container

- Summary
  List containers
- Description
  List all containers in the metadata database.

PARAMETERS(QUERY)

```
limit?: integer
```

RESPONSES

- 200 List containers

`application/json`

```
{
  // The container id
  id: string
  // The container hash
  hash: string
  // The user-friendly container name
  name: string
  image: {
    // The image id
    id: string
    // The image name
    name: string
    // The image tag
    version: string
    // Marks the image as default
    default: boolean
  }
  // The number of databases the container is capable to hold simultaneously, if null the container has no limit
  quota: integer
  // The number of databases currently in the container
```

```
  count: integer
  // The machine-friendly container name
  internal_name: string
}[]
```

[POST]/api/v1/container

- Summary
  Create container
- Description
  Creates a container in the metadata database. Requires role `create-container`.
- Security
  bearerAuth
  basicAuth

REQUESTBODY

- application/json

```
{
  // The user-friendly container name
  name: string
  // The container hostname
  host: string
  // The container port
  port: integer
  quota: integer
  // The image id used for the container database engine
  image_id: string
  // The username of the privileged user
  privileged_username: string
  // The password of the privileged user
  privileged_password: string
}
```

RESPONSES

- 201 Created a new container

`application/json`

```
{
  // The container id
  id: string
  // The user-friendly container name
  name: string
  image: {
    // The image id
    id: string
    // The image name
    name: string
    version: string
    operators: {
      // The operator id
      id: string
      // The machine-friendly name of the operator
      value: string
      // The documentation link
      documentation: string
      // The user-friendly name of the operator
      display_name: string
    }[]
    default: boolean
    data_types: {
      // The id of the data type
      id: string
      // The machine-friendly value of the data type
      value: string
      // The documentation link
      documentation: string
      // The human-friendly name of the data type
      display_name: string
      // The minimum size
      size_min?: integer
      // The maximum size
      size_max?: integer
      // The default size
      size_default?: integer
```

```
      // If true, the size parameter cannot be empty
      size_required?: boolean
      // The step increment
      size_step?: integer
      // The minimum d
      d_min?: integer
      // The maximum d
      d_max?: integer
      // The default d
      d_default?: integer
      // If true, the d parameter cannot be empty
      d_required?: boolean
      // The d increment
      d_step?: integer
      // The user-friendly description of the data format
      data_hint?: string
      // The user-friendly description of the data type
      type_hint?: string
      // frontend needs to quote this data type
      is_quoted: boolean
      // frontend can build this data type
      is_buildable: boolean
    }[]
  }
  // The number of databases the container is capable to hold simultaneously, if null the container has no limit
  quota: integer
  // The number of databases currently in the container
  count: integer
  // The machine-friendly container name
  internal_name: string
}
```

- 400 Container payload malformed
- 403 Create container not permitted, need authority `create-container`
- 404 Container image or user could not be found
- 409 Container name already exists

[GET]/api/v1/user

- Summary
  List users
- Description
  Lists users known to the metadata database. Internal users are omitted from the result list. If the optional query parameter `username` is present, the result list can be filtered by matching this exact username.

PARAMETERS(QUERY)

```
username?: string
```

RESPONSES

- 200 List users

`application/json`

```
{
  id?: string
  // Only contains lowercase characters
  username: string
  name?: string
  orcid?: string
  qualified_name?: string
  given_name?: string
  family_name?: string
}[]
```

- 403 Listing not allowed

`application/json`

```
{
  status: enum[100 CONTINUE, 101 SWITCHING_PROTOCOLS, 102 PROCESSING, 103 EARLY_HINTS, 103 CHECKPOINT, 200 OK, 201 CREATED, 202 ACCEPTED, 203
NON_AUTHORITATIVE_INFORMATION, 204 NO_CONTENT, 205 RESET_CONTENT, 206 PARTIAL_CONTENT, 207 MULTI_STATUS, 208 ALREADY_REPORTED, 226 IM_USED, 300
```

```
MULTIPLE_CHOICES, 301 MOVED_PERMANENTLY, 302 FOUND, 302 MOVED_TEMPORARILY, 303 SEE_OTHER, 304 NOT_MODIFIED, 305 USE_PROXY, 307 TEMPORARY_REDIRECT, 308
PERMANENT_REDIRECT, 400 BAD_REQUEST, 401 UNAUTHORIZED, 402 PAYMENT_REQUIRED, 403 FORBIDDEN, 404 NOT_FOUND, 405 METHOD_NOT_ALLOWED, 406 NOT_ACCEPTABLE, 407
PROXY_AUTHENTICATION_REQUIRED, 408 REQUEST_TIMEOUT, 409 CONFLICT, 410 GONE, 411 LENGTH_REQUIRED, 412 PRECONDITION_FAILED, 413 PAYLOAD_TOO_LARGE, 413
REQUEST_ENTITY_TOO_LARGE, 414 URI_TOO_LONG, 414 REQUEST_URI_TOO_LONG, 415 UNSUPPORTED_MEDIA_TYPE, 416 REQUESTED_RANGE_NOT_SATISFIABLE, 417
EXPECTATION_FAILED, 418 I_AM_A_TEAPOT, 419 INSUFFICIENT_SPACE_ON_RESOURCE, 420 METHOD_FAILURE, 421 DESTINATION_LOCKED, 422 UNPROCESSABLE_ENTITY, 423 LOCKED,
424 FAILED_DEPENDENCY, 425 TOO_EARLY, 426 UPGRADE_REQUIRED, 428 PRECONDITION_REQUIRED, 429 TOO_MANY_REQUESTS, 431 REQUEST_HEADER_FIELDS_TOO_LARGE, 451
UNAVAILABLE_FOR_LEGAL_REASONS, 500 INTERNAL_SERVER_ERROR, 501 NOT_IMPLEMENTED, 502 BAD_GATEWAY, 503 SERVICE_UNAVAILABLE, 504 GATEWAY_TIMEOUT, 505
HTTP_VERSION_NOT_SUPPORTED, 506 VARIANT_ALSO_NEGOTIATES, 507 INSUFFICIENT_STORAGE, 508 LOOP_DETECTED, 509 BANDWIDTH_LIMIT_EXCEEDED, 510 NOT_EXTENDED, 511
NETWORK_AUTHENTICATION_REQUIRED]
  // The error message in English
  message: string
  // The error code used for internationalization of the error messages
  code: string
}
```

[GET]/api/v1/oai

- Summary
  Get record

PARAMETERS(QUERY)

```
{
  "name": "verb",
  "in": "query"
}
```

```
parameters: {
  metadataPrefix?: string
  from?: string
  until?: string
  set?: string
  resumptionToken?: string
  fromDate?: string
  untilDate?: string
  parametersString?: string
}
```

RESPONSES

- 200 List containers

`text/xml`

```
{
  "type": "string"
}
```

[GET]/api/v1/message/message/{messageId}

- Summary
  Find message
- Description
  Finds a message with id in the metadata database.

RESPONSES

- 200 Get messages

`application/json`

```
{
  id: string
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
  display_start?: string
```

```
  display_end?: string
}
```

- 404 Could not find message

[GET]/api/v1/license

- Summary
  List licenses
- Description
  Lists licenses known to the metadata database.

RESPONSES

- 200 List of licenses

`application/json`

```
{
  // The id
  identifier: string
  // The link to the license such as an SPDX identifier
  uri: string
  // A user-friendly short abstract of key details of the license
  description?: string
}[]
```

[GET]/api/v1/identifier/retrieve

- Summary
  Retrieve PID metadata
- Description
  Retrieves Persistent Identifier (PID) metadata from external endpoints. Requires authentication. Supported PIDs are: ORCID, ROR, DOI.

PARAMETERS(QUERY)

```
url: string
```

RESPONSES

- 200 Retrieved metadata from identifier

`application/json`

```
{
  id: string
  links: {
    self: string
    data?: string
    self_html: string
    dashboard_html?: string
  }
  type: enum[database, subset, table, view]
  titles: {
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
```

```
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  query: string
  execution?: string
  doi?: string
  publisher: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    name_identifier_scheme_uri?: string
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    affiliation_identifier_scheme_uri?: string
  }[]
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  query_normalized: string
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
  // query hash in sha512
  query_hash: string
  result_hash?: string
  result_number?: integer
  publication_day?: integer
  publication_month?: integer
  publication_year: integer
}
```

- 404 Failed to find metadata for identifier

[GET]/api/v1/database/{databaseId}

- Summary
  Find database

- Description
  Finds a database with id.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 200 Database found successfully

`application/json`

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
```

```
  preview_image?: string
}
```

- 403 Not allowed to view database
- 404 Database could not be found

[GET]/api/v1/container/{containerId}

- Summary
  Find container
- Description
  Finds a container in the metadata database.

RESPONSES

- 200 Found container

`application/json`

```
{
  // The container id
  id: string
  // The user-friendly container name
  name: string
  image: {
    // The image id
    id: string
    // The image name
    name: string
    version: string
    operators: {
      // The operator id
      id: string
      // The machine-friendly name of the operator
      value: string
      // The documentation link
      documentation: string
      // The user-friendly name of the operator
      display_name: string
    }[]
    default: boolean
    data_types: {
      // The id of the data type
      id: string
      // The machine-friendly value of the data type
      value: string
      // The documentation link
      documentation: string
      // The human-friendly name of the data type
      display_name: string
      // The minimum size
      size_min?: integer
      // The maximum size
      size_max?: integer
      // The default size
      size_default?: integer
      // If true, the size parameter cannot be empty
      size_required?: boolean
      // The step increment
      size_step?: integer
      // The minimum d
      d_min?: integer
      // The maximum d
      d_max?: integer
      // The default d
      d_default?: integer
      // If true, the d parameter cannot be empty
      d_required?: boolean
      // The d increment
      d_step?: integer
      // The user-friendly description of the data format
      data_hint?: string
      // The user-friendly description of the data type
      type_hint?: string
      // frontend needs to quote this data type
      is_quoted: boolean
      // frontend can build this data type
      is_buildable: boolean
    }[]
  }
```

```
  // The number of databases the container is capable to hold simultaneously, if null the container has no limit
  quota: integer
  // The number of databases currently in the container
  count: integer
  // The machine-friendly container name
  internal_name: string
}
```

- 404 Container image could not be found

[DELETE]/api/v1/container/{containerId}

- Summary
  Delete container
- Description
  Deletes a container in the metadata database. Requires role `delete-container`.
- Security
  bearerAuth
  basicAuth

RESPONSES

- 202 Deleted container
- 403 Create container not permitted, need authority `delete-container`
- 404 Container not found

[GET]/api/v1/search

- Summary
  Performs a fuzzy search
- Description
  Performs a fuzzy search

PARAMETERS(QUERY)

```
q: string
```

RESPONSES

- 200 OK, contains the elements formatted as an array of JSON arrays

`application/json`

```
[]
```

- 415 Wrong accept type

[POST]/api/v1/search/{field_type}

- Summary
  Performs a general search
- Description
  Performs a general search

PARAMETERS(QUERY)

```
t1?: integer
```

```
t2?: integer
```

PARAMETERS(BODY)

```
body: {
  field_value_pairs: {
  }
  search_term: string
}
```

RESPONSES

- 200 OK, contains the elements formatted as an array of JSON arrays

`application/json`

```
{
  results: {
  }[]
  // Same as the requested type
  type?: enum[database, table, view, column, user, identifier, concept, unit]
}
```

[GET]/api/v1/search/{field_type}/fields

- Summary
  Get searchable fields

RESPONSES

- 200 List of fields

`application/json`

```
{
  results: {
    attr_friendly_name: string
    attr_name: string
    // OpenSearch data types.
    type: string
  }[]
}
```

- 404 Invalid type.

[GET]/api/v1/search/{index}

- Summary
  Gets the index
- Description
  Gets the index

PARAMETERS(BODY)

```
body: {
  field_value_pairs: {
  }
  search_term?: string
  t1?: integer
  t2?: integer
}
```

RESPONSES

- 200 OK, contains the elements formatted as an array of JSON arrays

`application/json`


```
{
  "properties": {
    "results": {
      "items": {
        "type": "object"
      },
      "type": "array"
    },
    "type": {
      "description": "Same as the requested type",
      "enum": [
        "database",
        "table",
        "view",
        "column",
        "user",
        "identifier",
        "concept",
        "unit"
      ],
      "type": "string"
    }
  },
  "required": [
    "results",
    "type"
  ]
}
```


4.4.4 References

#/components/securitySchemes/basicAuth


```
{
  "in": "header",
  "scheme": "basic",
  "type": "http"
}
```


#/components/securitySchemes/bearerAuth


```
{
  "bearerFormat": "JWT",
  "in": "header",
  "scheme": "bearer",
  "type": "http"
}
```


#/components/schemas/DatabaseAccessDto

```
{
  // The user name
  username: string
  user: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  type: enum[read, write_own, write_all]
}
```

#/components/schemas/UserBriefDto

```
{
  id?: string
  // Only contains lowercase characters
  username: string
  name?: string
  orcid?: string
  qualified_name?: string
  given_name?: string
```

```
  family_name?: string
}
```

#/components/schemas/TupleUpdateDto

```
{
  // The key-value data map
  data: {
  }
  // The map of conditions
  keys: {
  }
}
```

#/components/schemas/QueryPersistDto

```
{
  // If false, the query is marked for deletion at a later point in time
  persist: boolean
}
```

#/components/schemas/CreatorBriefDto

```
{
  id: string
  affiliation?: string
  creator_name: string
  name_type?: enum[Personal, Organizational]
  name_identifier?: string
  name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
  affiliation_identifier?: string
  affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
}
```

#/components/schemas/IdentifierBriefDto

```
{
  id: string
  type: enum[database, subset, table, view]
  creators: {
    id: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
  }[]
  titles: {
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  doi?: string
  publisher: string
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  publication_year: integer
  // The owner username
  owned_by: string
}
```

#/components/schemas/IdentifierDescriptionDto

```
{
  id: string
  description?: string
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
}
```

#/components/schemas/IdentifierTitleDto

```
{
  id: string
  title?: string
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
}
```

#/components/schemas/QueryDto

```
{
  // The query id
  id: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The timestamp when the query was executed
  execution: string
  // The mapped SQL query
  query: string
  // The query type
  type?: enum[query, view]
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
```

```
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The database id
  database_id: string
  // The normalized SQL query as executed
  query_normalized: string
  // The sha256-hash of the mapped query
  query_hash: string
  // The sha256-hash of the result
  result_hash?: string
  // The row count of the result
  result_number?: integer
  // If false, the query is marked for deletion at a later point in time
  is_persisted: boolean
}
```

#/components/schemas/TupleDto

```
{
  // The key-value data map
  data: {
  }
}
```

#/components/schemas/ImportDto

```
{
  // The key of the S3 binary object in the storage service
  location: string
  // If true, the first line contains the column names, otherwise it contains only data
  header: boolean
  // The column delimiter of the dataset
  separator: string
  // The quote symbol around values in the dataset
  quote?: string
  // The newline symbol
  line_termination?: string
}
```

#/components/schemas/ConditionalDto

```
{
  // The id of the column
  column_id: string
  // The id of the foreign column
  foreign_column_id: string
}
```

#/components/schemas/FilterDto

```
{
  // The filter type
  type: enum[where, or, and]
  // The filter value
  value?: string
  // The column id
  column_id: string
  // The operator id
  operator_id: string
}
```

#/components/schemas/JoinDto

```
{
  // The type of join
  type: enum[inner, left, right, cross]
  conditionals: {
    // The id of the column
    column_id: string
    // The id of the foreign column
    foreign_column_id: string
  }[]
  // The id of the data source
  datasource_id: string
}
```

#/components/schemas/OrderDto

```
{
  // The sort direction
  direction?: enum[asc, desc]
  // The column id
  column_id: string
}
```

#/components/schemas/SubsetColumnDto

```
{
  // The id of the column
  id: string
  // The column alias
  alias?: string
}
```

#/components/schemas/SubsetDto

```
{
  columns: {
    // The id of the column
    id: string
    // The column alias
    alias?: string
  }[]
  joins: {
    // The type of join
    type: enum[inner, left, right, cross]
    conditionals: {
      // The id of the column
      column_id: string
      // The id of the foreign column
      foreign_column_id: string
    }[]
    // The id of the data source
    datasource_id: string
  }[]
  filters: {
    // The filter type
    type: enum[where, or, and]
    // The filter value
    value?: string
    // The column id
    column_id: string
    // The operator id
    operator_id: string
  }[]
  orders: {
    // The sort direction
    direction?: enum[asc, desc]
    // The column id
    column_id: string
  }[]
  datasource_ids?: string[]
}
```

#/components/schemas/ColumnAnalysisResultDto

```
{
  // The column name
  name: string
  // The column data type
  datatype: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit,
tinyint, bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  // The size determines the number of digits before the comma: x=size-d where size >= d
  size?: integer
  // The digits behind the comma
  d?: integer
  enums?: string[]
  sets?: string[]
  // If set to true, the column value can be null
  null_allowed?: boolean
  // The column is a candidate to be part of a composite primary key
  primary_key?: boolean
}
```

#/components/schemas/SchemaAnalysisResultDto

```
{
  // The column delimiter of the dataset
  delimiter: string
  // The quote symbol around values in the dataset
  quote: string
  // The escape symbol
  escape: string
  // The comment symbol
  comment: string
  columns: {
    // The column name
    name: string
    // The column data type
    datatype: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit,
tinyint, bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    // The size determines the number of digits before the comma: x=size-d where size >= d
    size?: integer
    // The digits behind the comma
    d?: integer
    enums?: string[]
    sets?: string[]
    // If set to true, the column value can be null
    null_allowed?: boolean
    // The column is a candidate to be part of a composite primary key
    primary_key?: boolean
  }[]
  // The newline symbol
  newline_delimiter: string
  // The number of rows to skip
  skip_rows: integer
  // Detected the first line as header
  has_header: boolean
  // The format of the date
  date_format?: string
  // The format of the timestamp
  timestamp_format?: string
}
```

#/components/schemas/TableHistoryDto

```
{
  // The timestamp
  timestamp: string
  // The event type
  event: enum[insert, delete]
  // The total number of rows affected by the event
  total: integer
}
```

#/components/schemas/TupleDeleteDto

```
{
  // The map of conditions
  keys: {
  }
}
```

#/components/schemas/UserAttributesDto

```
{
  theme: string
  orcid?: string
  affiliation?: string
  language: string
}
```

#/components/schemas/UserDto

```
{
  id?: string
  name?: string
  username: string
  password?: string
  attributes: {
    theme: string
    orcid?: string
    affiliation?: string
    language: string
  }
  qualified_name?: string
  given_name?: string
```

```
  family_name?: string
}
```

#/components/schemas/DatabaseBriefDto

```
{
  id: string
  // The name
  name: string
  // The comment
  description?: string
  identifiers: {
    id: string
    type: enum[database, subset, table, view]
    creators: {
      id: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    }[]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    doi?: string
    publisher: string
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    publication_year: integer
    // The owner username
    owned_by: string
  }[]
  // The machine-friendly database name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  contact: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  // The owner username
  owned_by: string
  // The preview image
  preview_image?: string
}
```

#/components/schemas/UserUpdateDto

```
{
  firstname?: string
  lastname?: string
  affiliation?: string
  orcid?: string
  theme: string
  language: string
}
```

#/components/schemas/BannerMessageUpdateDto

```
{
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
  display_start?: string
  display_end?: string
}
```

#/components/schemas/BannerMessageBriefDto

```
{
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
}
```

#/components/schemas/ImageChangeDto

```
{
  // The URL of the registry without protocol
  registry: string
  // The SQL dialect class name
  dialect: string
  // The default image port to access the database
  default_port: integer
  // The driver class name
  driver_class: string
  // The method used by JDBC
  jdbc_method: string
}
```

#/components/schemas/DataTypeDto

```
{
  // The id of the data type
  id: string
  // The machine-friendly value of the data type
  value: string
  // The documentation link
  documentation: string
  // The human-friendly name of the data type
  display_name: string
  // The minimum size
  size_min?: integer
  // The maximum size
  size_max?: integer
  // The default size
  size_default?: integer
  // If true, the size parameter cannot be empty
  size_required?: boolean
  // The step increment
  size_step?: integer
  // The minimum d
  d_min?: integer
  // The maximum d
  d_max?: integer
  // The default d
  d_default?: integer
  // If true, the d parameter cannot be empty
  d_required?: boolean
  // The d increment
  d_step?: integer
  // The user-friendly description of the data format
  data_hint?: string
  // The user-friendly description of the data type
  type_hint?: string
  // frontend needs to quote this data type
  is_quoted: boolean
  // frontend can build this data type
  is_buildable: boolean
}
```

#/components/schemas/ImageDto

```
{
  // The image id
```

```
  id: string
  // The image name
  name: string
  version: string
  operators: {
    // The operator id
    id: string
    // The machine-friendly name of the operator
    value: string
    // The documentation link
    documentation: string
    // The user-friendly name of the operator
    display_name: string
  }[]
  default: boolean
  data_types: {
    // The id of the data type
    id: string
    // The machine-friendly value of the data type
    value: string
    // The documentation link
    documentation: string
    // The human-friendly name of the data type
    display_name: string
    // The minimum size
    size_min?: integer
    // The maximum size
    size_max?: integer
    // The default size
    size_default?: integer
    // If true, the size parameter cannot be empty
    size_required?: boolean
    // The step increment
    size_step?: integer
    // The minimum d
    d_min?: integer
    // The maximum d
    d_max?: integer
    // The default d
    d_default?: integer
    // If true, the d parameter cannot be empty
    d_required?: boolean
    // The d increment
    d_step?: integer
    // The user-friendly description of the data format
    data_hint?: string
    // The user-friendly description of the data type
    type_hint?: string
    // frontend needs to quote this data type
    is_quoted: boolean
    // frontend can build this data type
    is_buildable: boolean
  }[]
}
```

#/components/schemas/OperatorDto

```
{
  // The operator id
  id: string
  // The machine-friendly name of the operator
  value: string
  // The documentation link
  documentation: string
  // The user-friendly name of the operator
  display_name: string
}
```

#/components/schemas/IdentifierSaveDto

```
{
  id: string
  type: enum[database, subset, table, view]
  doi?: string
  titles: {
    id: string
    title: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
```

```
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  publisher: string
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    affiliation_identifier?: string
  }[]
  database_id: string
  query_id?: string
  view_id?: string
  table_id?: string
  publication_day?: integer
  publication_month?: integer
  publication_year: integer
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
}
```

#/components/schemas/LicenseDto

```
{
  // The id
  identifier: string
  // The link to the license such as an SPDX identifier
  uri: string
  // A user-friendly short abstract of key details of the license
  description?: string
}
```

#/components/schemas/SaveIdentifierCreatorDto

```
{
  id: string
  firstname?: string
  lastname?: string
  affiliation?: string
  creator_name: string
  name_type?: enum[Personal, Organizational]
  name_identifier?: string
  affiliation_identifier?: string
}
```

#/components/schemas/SaveIdentifierDescriptionDto

```
{
  id: string
```

```
  description: string
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
}
```

#/components/schemas/SaveIdentifierFunderDto

```
{
  id: string
  funder_name: string
  funder_identifier?: string
  funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
  scheme_uri?: string
  award_number?: string
  award_title?: string
}
```

#/components/schemas/SaveIdentifierTitleDto

```
{
  id: string
  title: string
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
}
```

#/components/schemas/SaveRelatedIdentifierDto

```
{
  id: string
  value: string
  type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
  relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
}
```

#/components/schemas/CreatorDto

```
{
  id: string
  firstname?: string
  lastname?: string
  affiliation?: string
  creator_name: string
  name_type?: enum[Personal, Organizational]
  name_identifier?: string
  name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
  name_identifier_scheme_uri?: string
  affiliation_identifier?: string
  affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
  affiliation_identifier_scheme_uri?: string
}
```

#/components/schemas/IdentifierDto

```
{
  id: string
  links: {
    self: string
    data?: string
    self_html: string
    dashboard_html?: string
  }
  type: enum[database, subset, table, view]
  titles: {
    id: string
    title?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
```

```
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    id: string
    description?: string
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
    id: string
    funder_name: string
    funder_identifier?: string
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    scheme_uri?: string
    award_number?: string
    award_title?: string
  }[]
  query: string
  execution?: string
  doi?: string
  publisher: string
  owner: {
    id?: string
    // Only contains lowercase characters
    username: string
    name?: string
    orcid?: string
    qualified_name?: string
    given_name?: string
    family_name?: string
  }
  language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  creators: {
    id: string
    firstname?: string
    lastname?: string
    affiliation?: string
    creator_name: string
    name_type?: enum[Personal, Organizational]
    name_identifier?: string
    name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
    name_identifier_scheme_uri?: string
    affiliation_identifier?: string
    affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
    affiliation_identifier_scheme_uri?: string
  }[]
  status: enum[draft, published]
  created?: string
  database_id: string
  query_id?: string
  table_id?: string
  view_id?: string
  query_normalized: string
  related_identifiers: {
    id: string
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
  // query hash in sha512
  query_hash: string
  result_hash?: string
  result_number?: integer
  publication_day?: integer
  publication_month?: integer
  publication_year: integer
}
```

#/components/schemas/IdentifierFunderDto

```
{
  id: string
  funder_name: string
  funder_identifier?: string
  funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
  scheme_uri?: string
  award_number?: string
  award_title?: string
}
```

#/components/schemas/LinksDto

```
{
  self: string
  data?: string
  self_html: string
  dashboard_html?: string
}
```

#/components/schemas/RelatedIdentifierDto

```
{
  id: string
  value: string
  type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
  relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
}
```

#/components/schemas/DatabaseModifyVisibilityDto

```
{
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the dashboard will be managed
  is_dashboard_enabled: boolean
}
```

#/components/schemas/ViewUpdateDto

```
{
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

#/components/schemas/ViewBriefDto

```
{
  // The id
  id: string
  // The user-friendly name
  name: string
  // The SQL statement used to create the view
  query: string
  // The database id
  database_id: string
  // The machine-friendly name
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the view is default for the database
  initial_view?: boolean
  // The sha256-hash of the query
  query_hash: string
  // The owner username
  owned_by?: string
}
```

#/components/schemas/TableUpdateDto

```
{
  // The comment
  description?: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

#/components/schemas/TableBriefDto

```
{
  // The id
  id: string
  // The user-friendly table name
  name: string
  // The comment
  description?: string
  // The database id
  database_id: string
  // The machine-friendly table name
  internal_name: string
  // If true, The is using data versioning
  is_versioned: boolean
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // The owner username
  owned_by: string
}
```

#/components/schemas/ColumnSemanticsUpdateDto

```
{
  // The description
  description?: string
  // The URI of the concept
  concept_uri?: string
  // The URI of the unit
  unit_uri?: string
}
```

#/components/schemas/ColumnDto

```
{
  // The id
  id: string
  // The user-friendly column name
  name: string
  // The data source alias name
  alias?: string
  // The column size, determines the number of digits before the comma as x=size-d where size >= d
  size?: integer
  // The column d
  d?: integer
  // The statistically average numerical value
  mean?: number
  // The statistically most middle numerical value
  median?: number
  description?: string
  enums: {
    // The enum id
    id: string
    // The enum value
    value: string
  }[]
  sets: {
    // The set id
    id: string
    // The set value
    value: string
  }[]
  // The database id
  database_id: string
  // The table id
  table_id: string
  // The ordinal position of the colum to order it
  ord: integer
  // The machine-friendly column name
  internal_name: string
  // The length of the index
```

```
  index_length?: integer
  // The length of the total data in the table (index + data)
  length?: integer
  // The column type name
  type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  // The data length
  data_length?: integer
  // The maximum data length
  max_data_length?: integer
  // The number of rows
  num_rows?: integer
  // The statistically highest numerical value
  val_min?: number
  // The statistically lowest numerical value
  val_max?: number
  // The statistically determined standard deviation
  std_dev?: number
  // The concept URI
  concept_uri?: string
  // The unit URI
  unit_uri?: string
  is_null_allowed: boolean
}
```

#/components/schemas/EnumDto

```
{
  // The enum id
  id: string
  // The enum value
  value: string
}
```

#/components/schemas/SetDto

```
{
  // The set id
  id: string
  // The set value
  value: string
}
```

#/components/schemas/DatabaseTransferDto

```
{
  // The username of the new owner
  username: string
}
```

#/components/schemas/DatabaseModifyImageDto

```
{
  // The key of the S3 binary object in the storage service
  key?: string
}
```

#/components/schemas/CreateAccessDto

```
{
  // The access type
  type: enum[read, write_own, write_all]
}
```

#/components/schemas/BannerMessageCreateDto

```
{
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
  display_start?: string
  display_end?: string
}
```

#/components/schemas/ImageCreateDto

```
{
  // The URL of the registry without protocol
  registry: string
  // The image name
  name: string
  version: string
  // The SQL dialect class name
  dialect: string
  // The default image port to access the database
  default_port: integer
  // The driver class name
  driver_class: string
  // The method used by JDBC
  jdbc_method: string
}
```

#/components/schemas/CreateIdentifierCreatorDto

```
{
  // The given name
  firstname?: string
  // The family name
  lastname?: string
  // The affiliation
  affiliation?: string
  // The full name
  creator_name: string
  // The name type
  name_type?: enum[Personal, Organizational]
  // The persistent identifier that identifies the creator unambiguously
  name_identifier?: string
  // The persistent identifier that identifies the affiliation unambiguously
  affiliation_identifier?: string
}
```

#/components/schemas/CreateIdentifierDescriptionDto

```
{
  // The description value
  description: string
  // The language
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  // The type
  type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
}
```

#/components/schemas/CreateIdentifierDto

```
{
  // The identifier type
  type: enum[database, subset, table, view]
  // The doi persistent identifier with optional https://doi.org/ prefix
  doi?: string
  titles: {
    // The title
    title: string
    // The language
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    // The type
    type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
  }[]
  descriptions: {
    // The description value
    description: string
    // The language
    language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    // The type
    type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
  }[]
  funders: {
```

```
    // The funder name
    funder_name: string
    // The identifier that identifies the funder unambiguously
    funder_identifier?: string
    // The funder type, when the `funder_identifier` is a DOI, the `funder_identifier_type` field must be `Crossref Funder ID`
    funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
    // The scheme URI of the `funder_identifier`
    scheme_uri?: string
    // The award number
    award_number?: string
    // The award title
    award_title?: string
  }[]
  licenses: {
    // The id
    identifier: string
    // The link to the license such as an SPDX identifier
    uri: string
    // A user-friendly short abstract of key details of the license
    description?: string
  }[]
  // The publisher
  publisher: string
  // The language
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  creators: {
    // The given name
    firstname?: string
    // The family name
    lastname?: string
    // The affiliation
    affiliation?: string
    // The full name
    creator_name: string
    // The name type
    name_type?: enum[Personal, Organizational]
    // The persistent identifier that identifies the creator unambiguously
    name_identifier?: string
    // The persistent identifier that identifies the affiliation unambiguously
    affiliation_identifier?: string
  }[]
  // The id
  database_id: string
  // The subset id, is only set when type=`subset`
  query_id?: string
  // The view id, is only set when type=`view`
  view_id?: string
  // The table id, is only set when type=`table`
  table_id?: string
  // The day of publication
  publication_day?: integer
  // The month of publication
  publication_month?: integer
  // The year of publication
  publication_year: integer
  related_identifiers: {
    value: string
    type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
    relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
  }[]
}
```

#/components/schemas/CreateIdentifierFunderDto

```
{
  // The funder name
  funder_name: string
  // The identifier that identifies the funder unambiguously
  funder_identifier?: string
  // The funder type, when the `funder_identifier` is a DOI, the `funder_identifier_type` field must be `Crossref Funder ID`
  funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
  // The scheme URI of the `funder_identifier`
  scheme_uri?: string
  // The award number
  award_number?: string
  // The award title
  award_title?: string
}
```

#/components/schemas/CreateIdentifierTitleDto

```
{
  // The title
  title: string
  // The language
  language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw, co,
cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia, ie,
iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv, nd,
ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw, ss,
sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
  // The type
  type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
}
```

#/components/schemas/CreateRelatedIdentifierDto

```
{
  value: string
  type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
  relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
}
```

#/components/schemas/CreateDatabaseDto

```
{
  // The name
  name: string
  // The container id
  container_id: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

#/components/schemas/CreateViewDto

```
{
  // The name
  name: string
  query: {
    columns: {
      // The id of the column
      id: string
      // The column alias
      alias?: string
    }[]
    joins: {
      // The type of join
      type: enum[inner, left, right, cross]
      conditionals: {
        // The id of the column
        column_id: string
        // The id of the foreign column
        foreign_column_id: string
      }[]
      // The id of the data source
      datasource_id: string
    }[]
    filters: {
      // The filter type
      type: enum[where, or, and]
      // The filter value
      value?: string
      // The column id
      column_id: string
      // The operator id
      operator_id: string
    }[]
    orders: {
      // The sort direction
      direction?: enum[asc, desc]
      // The column id
      column_id: string
    }[]
    datasource_ids?: string[]
  }
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

#/components/schemas/CreateForeignKeyDto

```
{
  columns?: string[]
  // The name of the foreign table
  referenced_table: string
  referenced_columns?: string[]
  // The integrity action when updating tuples
  on_update?: enum[restrict, cascade, set_null, no_action, set_default]
  // The integrity action when deleting tuples
  on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
}
```

#/components/schemas/CreateTableColumnDto

```
{
  // The column name
  name: string
  // The data type
  type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  // The size determines the number of digits before the comma: x=size-d where size >= d
  size?: integer
  // The digits behind the comma
  d?: integer
  // The column comment
  description?: string
  enums?: string[]
  sets?: string[]
  // The index length
  index_length?: integer
  // If set to true, the column value can be null
  null_allowed: boolean
  // The column concept
  concept_uri?: string
  // The column unit
  unit_uri?: string
}
```

#/components/schemas/CreateTableConstraintsDto

```
{
  uniques?: string[][]
  checks?: string[]
  foreign_keys: {
    columns?: string[]
    // The name of the foreign table
    referenced_table: string
    referenced_columns?: string[]
    // The integrity action when updating tuples
    on_update?: enum[restrict, cascade, set_null, no_action, set_default]
    // The integrity action when deleting tuples
    on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
  }[]
  primary_key?: string[]
}
```

#/components/schemas/CreateTableDto

```
{
  // The table name
  name: string
  // The table comment
  description?: string
  columns: {
    // The column name
    name: string
    // The data type
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    // The size determines the number of digits before the comma: x=size-d where size >= d
    size?: integer
    // The digits behind the comma
    d?: integer
    // The column comment
    description?: string
    enums?: string[]
    sets?: string[]
    // The index length
    index_length?: integer
    // If set to true, the column value can be null
    null_allowed: boolean
    // The column concept
    concept_uri?: string
```

```
    // The column unit
    unit_uri?: string
  }[]
  constraints: {
    uniques?: string[][]
    checks?: string[]
    foreign_keys: {
      columns?: string[]
      // The name of the foreign table
      referenced_table: string
      referenced_columns?: string[]
      // The integrity action when updating tuples
      on_update?: enum[restrict, cascade, set_null, no_action, set_default]
      // The integrity action when deleting tuples
      on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
    }[]
    primary_key?: string[]
  }
  // The table visibility; if true, the table will be displayed publicly and is searchable
  is_public: boolean
  // The table insights; if true, the table schema will be displayed publicly and is searchable
  is_schema_public: boolean
}
```

#/components/schemas/CreateContainerDto

```
{
  // The user-friendly container name
  name: string
  // The container hostname
  host: string
  // The container port
  port: integer
  quota: integer
  // The image id used for the container database engine
  image_id: string
  // The username of the privileged user
  privileged_username: string
  // The password of the privileged user
  privileged_password: string
}
```

#/components/schemas/ContainerDto

```
{
  // The container id
  id: string
  // The user-friendly container name
  name: string
  image: {
    // The image id
    id: string
    // The image name
    name: string
    version: string
    operators: {
      // The operator id
      id: string
      // The machine-friendly name of the operator
      value: string
      // The documentation link
      documentation: string
      // The user-friendly name of the operator
      display_name: string
    }[]
    default: boolean
    data_types: {
      // The id of the data type
      id: string
      // The machine-friendly value of the data type
      value: string
      // The documentation link
      documentation: string
      // The human-friendly name of the data type
      display_name: string
      // The minimum size
      size_min?: integer
      // The maximum size
      size_max?: integer
      // The default size
      size_default?: integer
      // If true, the size parameter cannot be empty
      size_required?: boolean
      // The step increment
      size_step?: integer
      // The minimum d
      d_min?: integer
      // The maximum d
```

```
      d_max?: integer
      // The default d
      d_default?: integer
      // If true, the d parameter cannot be empty
      d_required?: boolean
      // The d increment
      d_step?: integer
      // The user-friendly description of the data format
      data_hint?: string
      // The user-friendly description of the data type
      type_hint?: string
      // frontend needs to quote this data type
      is_quoted: boolean
      // frontend can build this data type
      is_buildable: boolean
    }[]
  }
  // The number of databases the container is capable to hold simultaneously, if null the container has no limit
  quota: integer
  // The number of databases currently in the container
  count: integer
  // The machine-friendly container name
  internal_name: string
}
```

#/components/schemas/ApiErrorDto

```
{
  status: enum[100 CONTINUE, 101 SWITCHING_PROTOCOLS, 102 PROCESSING, 103 EARLY_HINTS, 103 CHECKPOINT, 200 OK, 201 CREATED, 202 ACCEPTED, 203
NON_AUTHORITATIVE_INFORMATION, 204 NO_CONTENT, 205 RESET_CONTENT, 206 PARTIAL_CONTENT, 207 MULTI_STATUS, 208 ALREADY_REPORTED, 226 IM_USED, 300
MULTIPLE_CHOICES, 301 MOVED_PERMANENTLY, 302 FOUND, 302 MOVED_TEMPORARILY, 303 SEE_OTHER, 304 NOT_MODIFIED, 305 USE_PROXY, 307 TEMPORARY_REDIRECT, 308
PERMANENT_REDIRECT, 400 BAD_REQUEST, 401 UNAUTHORIZED, 402 PAYMENT_REQUIRED, 403 FORBIDDEN, 404 NOT_FOUND, 405 METHOD_NOT_ALLOWED, 406 NOT_ACCEPTABLE, 407
PROXY_AUTHENTICATION_REQUIRED, 408 REQUEST_TIMEOUT, 409 CONFLICT, 410 GONE, 411 LENGTH_REQUIRED, 412 PRECONDITION_FAILED, 413 PAYLOAD_TOO_LARGE, 413
REQUEST_ENTITY_TOO_LARGE, 414 URI_TOO_LONG, 414 REQUEST_URI_TOO_LONG, 415 UNSUPPORTED_MEDIA_TYPE, 416 REQUESTED_RANGE_NOT_SATISFIABLE, 417
EXPECTATION_FAILED, 418 I_AM_A_TEAPOT, 419 INSUFFICIENT_SPACE_ON_RESOURCE, 420 METHOD_FAILURE, 421 DESTINATION_LOCKED, 422 UNPROCESSABLE_ENTITY, 423 LOCKED,
424 FAILED_DEPENDENCY, 425 TOO_EARLY, 426 UPGRADE_REQUIRED, 428 PRECONDITION_REQUIRED, 429 TOO_MANY_REQUESTS, 431 REQUEST_HEADER_FIELDS_TOO_LARGE, 451
UNAVAILABLE_FOR_LEGAL_REASONS, 500 INTERNAL_SERVER_ERROR, 501 NOT_IMPLEMENTED, 502 BAD_GATEWAY, 503 SERVICE_UNAVAILABLE, 504 GATEWAY_TIMEOUT, 505
HTTP_VERSION_NOT_SUPPORTED, 506 VARIANT_ALSO_NEGOTIATES, 507 INSUFFICIENT_STORAGE, 508 LOOP_DETECTED, 509 BANDWIDTH_LIMIT_EXCEEDED, 510 NOT_EXTENDED, 511
NETWORK_AUTHENTICATION_REQUIRED]
  // The error message in English
  message: string
  // The error code used for internationalization of the error messages
  code: string
}
```

#/components/schemas/OaiListRecordsParameters

```
{
  metadataPrefix?: string
  from?: string
  until?: string
  set?: string
  resumptionToken?: string
  fromDate?: string
  untilDate?: string
  parametersString?: string
}
```

#/components/schemas/BannerMessageDto

```
{
  id: string
  type: enum[error, warning, info]
  message: string
  link?: string
  link_text?: string
  display_start?: string
  display_end?: string
}
```

#/components/schemas/ImageBriefDto

```
{
  // The image id
  id: string
  // The image name
  name: string
  // The image tag
  version: string
  // Marks the image as default
```

```
  default: boolean
}
```

#/components/schemas/LdCreatorDto

```
{
  // The name
  name: string
  // The unambiguous reference to a person's identifier
  sameAs?: string
  // The firstname
  givenName?: string
  // The lastname
  familyName?: string
  // The type
  @type: string
}
```

#/components/schemas/LdDatasetDto

```
{
  // The name
  name: string
  // The description
  description: string
  // The URL
  url: string
  identifier?: string[]
  license?: string
  creator: {
    // The name
    name: string
    // The unambiguous reference to a person's identifier
    sameAs?: string
    // The firstname
    givenName?: string
    // The lastname
    familyName?: string
    // The type
    @type: string
  }[]
  // The ciration URL
  citation: string
  hasPart: {
    // The name
    name: string
    // The description
    description: string
    // The URL
    url: string
    identifier?: string[]
    license?: string
    creator:#/components/schemas/LdCreatorDto[]
    // The ciration URL
    citation: string
    hasPart:#/components/schemas/LdDatasetDto[]
    // The temporal coverage
    temporalCoverage: string
    // The version
    version: string
    // The context schema URI
    @context: string
    // The type
    @type: string
  }[]
  // The temporal coverage
  temporalCoverage: string
  // The version
  version: string
  // The context schema URI
  @context: string
  // The type
  @type: string
}
```

#/components/schemas/ViewColumnDto

```
{
  // The id
  id: string
  // The user-friendly column name
  name: string
  // The column size, determines the number of digits before the comma as x=size-d where size >= d
  size?: integer
  // The column d
```

```
  d?: integer
  description?: string
  enums: {
    // The enum id
    id: string
    // The enum value
    value: string
  }[]
  sets: {
    // The set id
    id: string
    // The set value
    value: string
  }[]
  // The database id
  database_id: string
  // The ordinal position of the colum to order it
  ord: integer
  // The machine-friendly column name
  internal_name: string
  // The length of the index
  index_length?: integer
  // The length of the total data in the table (index + data)
  length?: integer
  // The column type name
  type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  is_null_allowed: boolean
}
```

#/components/schemas/ViewDto

```
{
  // The id
  id: string
  // The user-friendly name
  name: string
  identifiers: {
    id: string
    links: {
      self: string
      data?: string
      self_html: string
      dashboard_html?: string
    }
    type: enum[database, subset, table, view]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    funders: {
      id: string
      funder_name: string
      funder_identifier?: string
      funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
      scheme_uri?: string
      award_number?: string
      award_title?: string
    }[]
    query: string
    execution?: string
    doi?: string
    publisher: string
    owner: {
      id?: string
      // Only contains lowercase characters
      username: string
      name?: string
      orcid?: string
      qualified_name?: string
      given_name?: string
      family_name?: string
    }
    language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
```

```
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    licenses: {
      // The id
      identifier: string
      // The link to the license such as an SPDX identifier
      uri: string
      // A user-friendly short abstract of key details of the license
      description?: string
    }[]
    creators: {
      id: string
      firstname?: string
      lastname?: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      name_identifier_scheme_uri?: string
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
      affiliation_identifier_scheme_uri?: string
    }[]
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    query_normalized: string
    related_identifiers: {
      id: string
      value: string
      type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
      relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
    }[]
    // query hash in sha512
    query_hash: string
    result_hash?: string
    result_number?: integer
    publication_day?: integer
    publication_month?: integer
    publication_year: integer
  }[]
  // The SQL statement used to create the view
  query: string
  owner:#/components/schemas/UserBriefDto
  columns: {
    // The id
    id: string
    // The user-friendly column name
    name: string
    // The column size, determines the number of digits before the comma as x=size-d where size >= d
    size?: integer
    // The column d
    d?: integer
    description?: string
    enums: {
      // The enum id
      id: string
      // The enum value
      value: string
    }[]
    sets: {
      // The set id
      id: string
      // The set value
      value: string
    }[]
    // The database id
    database_id: string
    // The ordinal position of the colum to order it
    ord: integer
    // The machine-friendly column name
    internal_name: string
    // The length of the index
    index_length?: integer
    // The length of the total data in the table (index + data)
    length?: integer
    // The column type name
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    is_null_allowed: boolean
  }[]
  // The created timestamp
  created: string
  // The database id
  database_id: string
  // The machine-friendly name
```

```
  internal_name: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // If true, the view is default for the database
  initial_view?: boolean
  // The sha256-hash of the query
  query_hash: string
}
```

#/components/schemas/ColumnBriefDto

```
{
  // The column id
  id: string
  // The column name
  name: string
  // The data source alias name
  alias?: string
  // The database id
  database_id: string
  // The table id
  table_id: string
  // The machine-friendly internal name
  internal_name: string
  // The column type name
  type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
}
```

#/components/schemas/ConstraintsDto

```
{
  uniques: {
    // The unique key id
    id: string
    // The unique key name
    name: string
    table: {
      // The id
      id: string
      // The user-friendly table name
      name: string
      // The comment
      description?: string
      // The database id
      database_id: string
      // The machine-friendly table name
      internal_name: string
      // If true, The is using data versioning
      is_versioned: boolean
      // The visibility; if true, The will be displayed publicly and is searchable
      is_public: boolean
      // The insights; if true, The schema will be displayed publicly and is searchable
      is_schema_public: boolean
      // The owner username
      owned_by: string
    }
    columns: {
      // The column id
      id: string
      // The column name
      name: string
      // The data source alias name
      alias?: string
      // The database id
      database_id: string
      // The table id
      table_id: string
      // The machine-friendly internal name
      internal_name: string
      // The column type name
      type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit,
tinyint, bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    }[]
  }[]
  checks?: string[]
  foreign_keys: {
    // The foreign key id
    id?: string
    // The foreign key name
    name: string
    references: {
      // The foreign key reference id
      id?: string
      column:#/components/schemas/ColumnBriefDto
      foreign_key: {
```

```
        // The foreign key id
        id?: string
      }
      referenced_column:#/components/schemas/ColumnBriefDto
    }[]
    table:#/components/schemas/TableBriefDto
    referenced_table:#/components/schemas/TableBriefDto
    // The integrity action when updating tuples
    on_update?: enum[restrict, cascade, set_null, no_action, set_default]
    // The integrity action when deleting tuples
    on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
  }[]
  primary_key: {
    // The primary key id
    id?: string
    table:#/components/schemas/TableBriefDto
    column:#/components/schemas/ColumnBriefDto
  }[]
}
```

#/components/schemas/ForeignKeyBriefDto

```
{
  // The foreign key id
  id?: string
}
```

#/components/schemas/ForeignKeyDto

```
{
  // The foreign key id
  id?: string
  // The foreign key name
  name: string
  references: {
    // The foreign key reference id
    id?: string
    column: {
      // The column id
      id: string
      // The column name
      name: string
      // The data source alias name
      alias?: string
      // The database id
      database_id: string
      // The table id
      table_id: string
      // The machine-friendly internal name
      internal_name: string
      // The column type name
      type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit,
tinyint, bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    }
    foreign_key: {
      // The foreign key id
      id?: string
    }
    referenced_column:#/components/schemas/ColumnBriefDto
  }[]
  table: {
    // The id
    id: string
    // The user-friendly table name
    name: string
    // The comment
    description?: string
    // The database id
    database_id: string
    // The machine-friendly table name
    internal_name: string
    // If true, The is using data versioning
    is_versioned: boolean
    // The visibility; if true, The will be displayed publicly and is searchable
    is_public: boolean
    // The insights; if true, The schema will be displayed publicly and is searchable
    is_schema_public: boolean
    // The owner username
    owned_by: string
  }
  referenced_table:#/components/schemas/TableBriefDto
  // The integrity action when updating tuples
  on_update?: enum[restrict, cascade, set_null, no_action, set_default]
  // The integrity action when deleting tuples
  on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
}
```

#/components/schemas/ForeignKeyReferenceDto

```
{
  // The foreign key reference id
  id?: string
  column: {
    // The column id
    id: string
    // The column name
    name: string
    // The data source alias name
    alias?: string
    // The database id
    database_id: string
    // The table id
    table_id: string
    // The machine-friendly internal name
    internal_name: string
    // The column type name
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  }
  foreign_key: {
    // The foreign key id
    id?: string
  }
  referenced_column:#/components/schemas/ColumnBriefDto
}
```

#/components/schemas/PrimaryKeyDto

```
{
  // The primary key id
  id?: string
  table: {
    // The id
    id: string
    // The user-friendly table name
    name: string
    // The comment
    description?: string
    // The database id
    database_id: string
    // The machine-friendly table name
    internal_name: string
    // If true, The is using data versioning
    is_versioned: boolean
    // The visibility; if true, The will be displayed publicly and is searchable
    is_public: boolean
    // The insights; if true, The schema will be displayed publicly and is searchable
    is_schema_public: boolean
    // The owner username
    owned_by: string
  }
  column: {
    // The column id
    id: string
    // The column name
    name: string
    // The data source alias name
    alias?: string
    // The database id
    database_id: string
    // The table id
    table_id: string
    // The machine-friendly internal name
    internal_name: string
    // The column type name
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  }
}
```

#/components/schemas/TableDto

```
{
  // The id
  id: string
  // The user-friendly name
  name: string
  // The comment
  description?: string
  // The alias
  alias?: string
  identifiers: {
    id: string
    links: {
```

```
      self: string
      data?: string
      self_html: string
      dashboard_html?: string
    }
    type: enum[database, subset, table, view]
    titles: {
      id: string
      title?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[AlternativeTitle, Subtitle, TranslatedTitle, Other]
    }[]
    descriptions: {
      id: string
      description?: string
      language?: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
      type?: enum[Abstract, Methods, SeriesInformation, TableOfContents, TechnicalInfo, Other]
    }[]
    funders: {
      id: string
      funder_name: string
      funder_identifier?: string
      funder_identifier_type?: enum[Crossref Funder ID, ROR, GND, ISNI, Other]
      scheme_uri?: string
      award_number?: string
      award_title?: string
    }[]
    query: string
    execution?: string
    doi?: string
    publisher: string
    owner: {
      id?: string
      // Only contains lowercase characters
      username: string
      name?: string
      orcid?: string
      qualified_name?: string
      given_name?: string
      family_name?: string
    }
    language: enum[ab, aa, af, ak, sq, am, ar, an, hy, as, av, ae, ay, az, bm, ba, eu, be, bn, bh, bi, bs, br, bg, my, ca, km, ch, ce, ny, zh, cu, cv, kw,
co, cr, hr, cs, da, dv, nl, dz, en, eo, et, ee, fo, fj, fi, fr, ff, gd, gl, lg, ka, de, ki, el, kl, gn, gu, ht, ha, he, hz, hi, ho, hu, is, io, ig, id, ia,
ie, iu, ik, ga, it, ja, jv, kn, kr, ks, kk, rw, kv, kg, ko, kj, ku, ky, lo, la, lv, lb, li, ln, lt, lu, mk, mg, ms, ml, mt, gv, mi, mr, mh, ro, mn, na, nv,
nd, ng, ne, se, no, nb, nn, ii, oc, oj, or, om, os, pi, pa, ps, fa, pl, pt, qu, rm, rn, ru, sm, sg, sa, sc, sr, sn, sd, si, sk, sl, so, st, nr, es, su, sw,
ss, sv, tl, ty, tg, ta, tt, te, th, bo, ti, to, ts, tn, tr, tk, tw, ug, uk, ur, uz, ve, vi, vo, wa, cy, fy, wo, xh, yi, yo, za, zu]
    licenses: {
      // The id
      identifier: string
      // The link to the license such as an SPDX identifier
      uri: string
      // A user-friendly short abstract of key details of the license
      description?: string
    }[]
    creators: {
      id: string
      firstname?: string
      lastname?: string
      affiliation?: string
      creator_name: string
      name_type?: enum[Personal, Organizational]
      name_identifier?: string
      name_identifier_scheme?: enum[ORCID, ROR, ISNI, GRID]
      name_identifier_scheme_uri?: string
      affiliation_identifier?: string
      affiliation_identifier_scheme?: enum[ROR, GRID, ISNI]
      affiliation_identifier_scheme_uri?: string
    }[]
    status: enum[draft, published]
    created?: string
    database_id: string
    query_id?: string
    table_id?: string
    view_id?: string
    query_normalized: string
    related_identifiers: {
      id: string
      value: string
      type: enum[DOI, URL, URN, ARK, arXiv, bibcode, EAN13, EISSN, Handle, IGSN, ISBN, ISTC, LISSN, LSID, PMID, PURL, UPC, w3id]
      relation: enum[IsCitedBy, Cites, IsSupplementTo, IsSupplementedBy, IsContinuedBy, Continues, IsDescribedBy, Describes, HasMetadata, IsMetadataFor,
HasVersion, IsVersionOf, IsNewVersionOf, IsPreviousVersionOf, IsPartOf, HasPart, IsPublishedIn, IsReferencedBy, References, IsDocumentedBy, Documents,
IsCompiledBy, Compiles, IsVariantFormOf, IsOriginalFormOf, IsIdenticalTo, IsReviewedBy, Reviews, IsDerivedFrom, IsSourceOf, IsRequiredBy, Requires,
IsObsoletedBy, Obsoletes]
    }[]
    // query hash in sha512
    query_hash: string
```

```
    result_hash?: string
    result_number?: integer
    publication_day?: integer
    publication_month?: integer
    publication_year: integer
  }[]
  owner:#/components/schemas/UserBriefDto
  columns: {
    // The id
    id: string
    // The user-friendly column name
    name: string
    // The data source alias name
    alias?: string
    // The column size, determines the number of digits before the comma as x=size-d where size >= d
    size?: integer
    // The column d
    d?: integer
    // The statistically average numerical value
    mean?: number
    // The statistically most middle numerical value
    median?: number
    description?: string
    enums: {
      // The enum id
      id: string
      // The enum value
      value: string
    }[]
    sets: {
      // The set id
      id: string
      // The set value
      value: string
    }[]
    // The database id
    database_id: string
    // The table id
    table_id: string
    // The ordinal position of the colum to order it
    ord: integer
    // The machine-friendly column name
    internal_name: string
    // The length of the index
    index_length?: integer
    // The length of the total data in the table (index + data)
    length?: integer
    // The column type name
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
    // The data length
    data_length?: integer
    // The maximum data length
    max_data_length?: integer
    // The number of rows
    num_rows?: integer
    // The statistically highest numerical value
    val_min?: number
    // The statistically lowest numerical value
    val_max?: number
    // The statistically determined standard deviation
    std_dev?: number
    // The concept URI
    concept_uri?: string
    // The unit URI
    unit_uri?: string
    is_null_allowed: boolean
  }[]
  constraints: {
    uniques: {
      // The unique key id
      id: string
      // The unique key name
      name: string
      table: {
        // The id
        id: string
        // The user-friendly table name
        name: string
        // The comment
        description?: string
        // The database id
        database_id: string
        // The machine-friendly table name
        internal_name: string
        // If true, The is using data versioning
        is_versioned: boolean
        // The visibility; if true, The will be displayed publicly and is searchable
        is_public: boolean
        // The insights; if true, The schema will be displayed publicly and is searchable
        is_schema_public: boolean
        // The owner username
        owned_by: string
      }
```

```
      columns: {
        // The column id
        id: string
        // The column name
        name: string
        // The data source alias name
        alias?: string
        // The database id
        database_id: string
        // The table id
        table_id: string
        // The machine-friendly internal name
        internal_name: string
        // The column type name
        type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit,
tinyint, bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
      }[]
    }[]
    checks?: string[]
    foreign_keys: {
      // The foreign key id
      id?: string
      // The foreign key name
      name: string
      references: {
        // The foreign key reference id
        id?: string
        column:#/components/schemas/ColumnBriefDto
        foreign_key: {
          // The foreign key id
          id?: string
        }
        referenced_column:#/components/schemas/ColumnBriefDto
      }[]
      table:#/components/schemas/TableBriefDto
      referenced_table:#/components/schemas/TableBriefDto
      // The integrity action when updating tuples
      on_update?: enum[restrict, cascade, set_null, no_action, set_default]
      // The integrity action when deleting tuples
      on_delete?: enum[restrict, cascade, set_null, no_action, set_default]
    }[]
    primary_key: {
      // The primary key id
      id?: string
      table:#/components/schemas/TableBriefDto
      column:#/components/schemas/ColumnBriefDto
    }[]
  }
  // The created timestamp
  created: string
  // The database id
  database_id: string
  // The machine-friendly name
  internal_name: string
  // If true, The is using data versioning
  is_versioned: boolean
  // The queue name
  queue_name: string
  // The queue type
  queue_type?: string
  // The routing key
  routing_key: string
  // The visibility; if true, The will be displayed publicly and is searchable
  is_public: boolean
  // The insights; if true, The schema will be displayed publicly and is searchable
  is_schema_public: boolean
  // The statistical number of rows
  num_rows?: integer
  // The data length in bytes
  data_length?: integer
  // The maximum data length in bytes
  max_data_length?: integer
  // The average row length in bytes
  avg_row_length?: integer
}
```

#/components/schemas/UniqueDto

```
{
  // The unique key id
  id: string
  // The unique key name
  name: string
  table: {
    // The id
    id: string
    // The user-friendly table name
    name: string
    // The comment
    description?: string
    // The database id
```

```
    database_id: string
    // The machine-friendly table name
    internal_name: string
    // If true, The is using data versioning
    is_versioned: boolean
    // The visibility; if true, The will be displayed publicly and is searchable
    is_public: boolean
    // The insights; if true, The schema will be displayed publicly and is searchable
    is_schema_public: boolean
    // The owner username
    owned_by: string
  }
  columns: {
    // The column id
    id: string
    // The column name
    name: string
    // The data source alias name
    alias?: string
    // The database id
    database_id: string
    // The table id
    table_id: string
    // The machine-friendly internal name
    internal_name: string
    // The column type name
    type: enum[char, varchar, binary, varbinary, tinyblob, tinytext, text, blob, mediumtext, mediumblob, longtext, longblob, enum, set, serial, bit, tinyint,
bool, smallint, mediumint, int, bigint, float, double, decimal, date, datetime, timestamp, time, year]
  }[]
}
```

#/components/schemas/ContainerBriefDto

```
{
  // The container id
  id: string
  // The container hash
  hash: string
  // The user-friendly container name
  name: string
  image: {
    // The image id
    id: string
    // The image name
    name: string
    // The image tag
    version: string
    // Marks the image as default
    default: boolean
  }
  // The number of databases the container is capable to hold simultaneously, if null the container has no limit
  quota: integer
  // The number of databases currently in the container
  count: integer
  // The machine-friendly container name
  internal_name: string
}
```

#/components/schemas/ApiError

```
{
  code?: string
  message?: string
  status?: string
}
```

#/components/schemas/IndexDto

```
{
  "properties": {
    "results": {
      "items": {
        "type": "object"
      },
      "type": "array"
    },
    "type": {
      "description": "Same as the requested type",
      "enum": [
        "database",
        "table",
        "view",
        "column",
        "user",
        "identifier",
```

```
        "concept",
        "unit"
      ],
      "type": "string"
    }
  },
  "required": [
    "results",
    "type"
  ]
}
```


#/components/schemas/IndexFieldDto


```
{
  attr_friendly_name: string
  attr_name: string
  // OpenSearch data types.
  type: string
}
```


#/components/schemas/IndexFieldsDto


```
{
  results: {
    attr_friendly_name: string
    attr_name: string
    // OpenSearch data types.
    type: string
  }[]
}
```


#/components/schemas/SearchRequestDto


```
{
  field_value_pairs: {
  }
  search_term: string
}
```

4.5 AMQP / MQTT API

This section will be expanded in the future.

4.6 OAI-PMH API

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) standard provides and endpoint to expose the metadata records in DBRepo in an XML format defined by version 2.0 of the OAI-PMH protocol.

Reference:

- Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)
- OpenAIRE Guidelines for Data Archive Managers ("OpenAIRE Provide")

Related:

- Metadata Service

Harvester

We are OpenAIRE Provide 2.0-compliant:

OpenAIRE Provide Logo

4.7 JSON-LD API

JSON-LD is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. JSON-LD is an ideal data format for programming environments, REST Web services, and unstructured databases such as Apache CouchDB and MongoDB.

Reference:

- JSON for Linking Data (JSON-LD)

Related:

- User Interface

4.8 FAIR Signposting API

Signposting specifies the use of typed web links to interlink resources of a Digital Object and it specifies which link relation types to use to link to which resources. All of this is done in a standards-based manner, with link types drawn from the IANA link relation registry. FAIR Signposting is a detailed implementation guideline that aims for uniformity in the provision of these typed links across systems.

FAIR signposting uses a standards-based REST approach and is a concrete Implementation Guideline for Signposting aimed at uniformity across systems: Typed web links available via landing page, content resources, metadata resources; Typed links are a small subset of IANA-registered relation types defined in formal specifications.

Reference:

- Web Linking (RFC 8288)
- IANA Link Relations
- FAIR Signposting

Related:

- User Interface

5. Use Cases

5.1 Overview

Big Sensor Data

Data that is of high volume or -variety.

→ Air Quality, → Industry 4.0 Power Data, → Health Data Data, → Hazard Data,

Machine Learning Data

Quality data sources for algorithmic systems.

→ Music-ML Data

Lab Research Data

Desk research data sets.

→ XPS Data, → PLS Data, → Survey Data,

Start by re-using some of the use-cases from our public pilots repository.

5.2 Air Quality Data

5.2.1 tl;dr

Dataset **Dashboard**

5.2.2 Description

This digital record contains historical air pollution and air quality data from approximately 20 air monitoring stations in Vienna, spanning the years from 1980 to 2021. The data was provided by the Umweltbundesamt and is stored in its original form in this record. This record forms the basis of an analysis carried out in a bachelor's thesis at the TU Wien.

Grafana Dashboard

Figure 1: Grafana dashboard visualizing the dataset.

The analysis was carried out in a Jupyter Notebook hosted by our IT-department JupyterHub as part of TU Wien's virtual research environment.

Jupyter Notebook

Figure 2: Jupyter Notebook accessing data on DBRepo using the Python Library.

5.2.3 Solution

One of the first use-cases of importing external data into DBRepo which was provided as .csv flat file. We developed a database schema and a web scraper that scrapes live air quality data from the public map of the environment agency of Austria.

5.2.4 DBRepo Features

- nport complex dataset
- ystem versioning
- ubset exploration
- ggregated views
- recise & PID of queries tables
- xternal data access for analysis

5.2.5 Acknowledgement

This work was part of a cooperation with the Umweltbundesamt.

5.3 COVID-19 Data

5.3.1 tl;dr

Dataset Archive

5.3.2 Description

This dataset contains the daily COVID-19 data provided publicly by Johns Hopkins University.

5.3.3 Solution

We imported their daily snapshots provided as 1145 versioned .csv files from their Git repository archive and imported them daily into DBRepo as system-versioned data that can be queried. During the time of this project the COVID-19 pandemic was still ongoing and therefore daily snapshots demanded a correct import script to be maintained.

5.3.4 DBRepo Features

- ata pipeline from Git repository
- ystem versioning
- ubset exploration

5.4 Hazard Data

5.4.1 tl;dr

Dataset **Archive**

5.4.2 Description

This data base contains concentrations of hazardous substances and other water quality parameters in different environmental compartments:

- river water (water and suspended sediments)
- ground water
- waste water (treated and untreated) and sewage sludge
- storm water runoff from combined and separate sewer systems
- atmospheric deposition
- soil

Data from many different data sources were collected, checked and combined and meta data were harmonized to allow for a combined data evaluation.

5.4.3 Solution

We imported the database from the archive repository, converted the PostgreSQL language code to MariaDB:

- `"current_user"()` to `current_user()`
- `SEQUENCE CACHE 1` to `NOCACHE` because of MariaDB Galera distribution not being able to cache sequences
- `(0)::double precision` to `0.0`
- `character varying` to `text` because MariaDB needs sizes, `text` data type can have arbitrary size
- `!~~` to `NOT LIKE`
- `ANY ARRAY` to `<array> OR <array2>` because of different MariaDB syntax

The complex nature of the views (i.e. `SELECT-FROM-SELECT` statements) required a logical overhaul of how DBRepo obtains view metadata. As a consequence, we removed the SQL parser that tried to parse the metadata up to a depth of 10 to a simple query to the `information_schema` that is maintained by the database engine.

5.4.4 DBRepo Features

- ✅ omplex database schema
- ✅ omplex views
- ✅ ystem versioning
- ✅ ubset exploration

5.4.5 Acknowledgement

This work was part of a cooperation with the Institute of Water Quality and Resource Management.

5.5 Health Data

5.5.1 tl;dr

5.5.2 Description

This dataset contains anonymous behavioural and ocular recordings from healthy participants during performance of a decision-making task between action movements in which we studied the influence of social facilitation and social pressure. The original dataset is available on Kaggle.

5.5.3 Solution

Importing the original dataset from Kaggle into DBRepo allows for exploratory analysis and increased value for secondary use, e.g. with our `pandas`-compatible Python Library.

5.5.4 DBRepo Features

- ystem versioning
- ubset exploration
- arge dataset (3 tables, around 6GB)
- xternal data access for analysis

5.6 Industry 4.0 Power Data

5.6.1 tl;dr

Dataset

5.6.2 Description

The Pilotfabrik of TU Wien is monitored for energy-efficiency and productivity of machinery. In principle, certain conditions/ parameters are observed such as: electric rate of energy transfer, transmission of cooling liquid, transmission of compressed air, acceleration, forces at work and temperatures to research on preventive/predictive maintenance, quality of products and ultimately process efficiency and -productivity.

Figure 1: Total power usage of machine floor TU Pilotfabrik, image from Hacksteiner (2016).

5.6.3 Solution

We connected our Broker Service with the MQTT broker of the Pilotfabrik using a self-written connector service, bridging the two different protocols. The tuples are ingested into DBRepo at a rate of about 10/s.

5.6.4 DBRepo Features

- High-throughput real-time data import (MQTT)
- Private database
- Public embargoed data view

5.6.5 Acknowledgement

This work was part of a cooperation with the Institute of Production Engineering and Photonic Technologies.

5.7 Survey Data

5.7.1 tl;dr

Dataset Notebook

5.7.2 Description

As part of a literature study, the research unit of data science has collected data on 47 Trusted Research Environments (TREs) who enable analysis of confidential data under strict security assertions who protect the data with technical, organizational and legal measures from (accidentally) being leaked outside the facility. The literature study shows that 47 TREs worldwide provide access to confidential data of which two-thirds provide data themselves (n=32, 68%), predominantly via secure remote access (n=46, 98%).

5.7.3 Solution

We designed a database schema that allows collection of the data with correct primary key and foreign-key relationships. Three defined views allow for a simpler exploration of the study data. The analysis of the data was performed in TU Wien's virtual research environment using JupyterHub as well as the chart

Jupyter Notebook

Figure 1: Jupyter Notebook accessing data on DBRepo using the Python Library.

5.7.4 DBRepo Features

- ystem versioning
- ubset exploration
- ggregated views
- recise & PID of queries tables
- xternal data access for analysis

5.7.5 Acknowledgement

This work was part of a cooperation with the Research Unit of Data Science.

5.8 Lute Data

5.8.1 tl;dr

Mirroring of internal database for users not authorized in the partner university.

5.8.2 Description

The main aim of the E-LAUTE project is to create a novel form of music edition: an "open knowledge platform" in which musicology, music practice, music informatics and literary studies intertwine and transform the "classic" edition into a space of interdisciplinary and discipline-specific work.

In order to create a comprehensive, complete modern scholarly edition, the E-LAUTE project synchronises high technology informatics in the fields of encoding, linking, recognition (OMR) and automatic transcription with manual music transcription and musical performance practice. We consider recordings of lute music a conceptual component of the edition.

5.8.3 Solution

We set up a mirror script that copies the internal database for users not authorized in the partner university to a local, downstream database where they can access the database for analysis.

5.8.4 DBRepo Features

- ata preservation of historic data
- ubset exploration
- xternal visualization of the database
- ata mirroring

5.8.5 Acknowledgement

This work was part of a cooperation with the University of Vienna in the context of E-LAUTE.

5.9 Music-ML Data

5.9.1 tl;dr

Dataset Archive

5.9.2 Description

We use a dataset collected by Aljanaki et al., consisting of 400 MP3 music files, each having a playtime of one minute and labeled with one of four genres: rock, pop, classical and electronic, each genre contains 100 files, the genre will be used as label for the ML model. Then by generating MFCC vectors and training a SVM, the ML-model can classify emotions of the provided .mp3 files with and accuracy of 76.25%.

Figure 1: Accuracy of predictions matrix in Jupyter Notebook.

5.9.3 Solution

DBRepo is used as relational data storage of the raw- and aggregated features, prediction results and the splits of the training- and test data. For each of the 400 .mp3 files, 40 MFCC feature vectors are generated. This data is stored in aggregated form in the `aggregated_features` table.

5.9.4 DBRepo Features

- atabase as storage for machine learning data
- ystem versioning
- ubset exploration
- recise & PID of database tables
- xternal data access for analysis

5.10 Theater Data

5.10.1 tl;dr

5.10.2 Description

This dataset is a donation from Prof. Dr. Bernhard Jahn in support of DBRepo.

5.10.3 Solution

We host a mirror from the public data provided by UHH.

5.10.4 DBRepo Features

- ystem versioning
- ubset exploration
- xternal data access for analysis

5.10.5 Acknowledgement

This work was part of a cooperation with the University of Hamburg.

5.11 Transportation Data

5.11.1 tl;dr

Dataset Dashboard

5.11.2 Description

The Subway Transportation Data-Dataset is a comprehensive and dynamic collection of data that captures the intricate details of the city's public transportation system. This dataset encompasses a wide array of information, including bus and tram schedules, subway routes, ticketing details, and real-time updates on vehicle locations.

5.11.3 Solution

We wrote an algorithm that parses open data (available) information from Wiener Linien, Vienna's public transportation agency directly and feeds it, after some cleaning, into DBRepo on a 5-minute interval.

Subway Transportation Data Dashboard
Figure 1: Dashboard visualizing the live data of the current interruptions.

5.11.4 DBRepo Features

- ynamic data (live data)
- ystem versioning
- ubset exploration
- xternal visualization of the database
- ix of managed and unmanaged content for dashboards

5.12 XPS Data

5.12.1 tl;dr

Dataset Notebook

5.12.2 Description

X-ray Photoelectron Spectroscopy (XPS) is one of the most used methods in material sciences. Irradiation of solid materials with X-ray radiation kicks out electrons from atoms that are near the atomic nucleus. With XPS data being highly reproducible once machine parameters are known and understood, the demand for creating a comprehensive database connecting material properties to compositions via XPS spectra becomes evident.

5.12.3 Solution

We read XPS data from the VAMAS-encoded format and inserted it into a database schema that captures the VAMAS-schema. It can then be read using the Python Library that executes a database query in SQL to obtain only the experiment data (c.f. subset page).

Jupyter Notebook

Figure 1: Jupyter Notebook accessing data on DBRepo using the Python Library.

Using the DataFrame representation of the Python Library and the `plotly` library, we can visualize the ordinate values directly in the Jupyter Notebook.

Three charts displaying surface analysis data of C, O and Su

Figure 2: Plot of ordinate values encoded within the experiment block.

5.12.4 DBRepo Features

- ata preservation of VAMAS-encoded XPS data
- ubset exploration
- xternal visualization of the database
- eplication of experiments using only open-source software

5.12.5 Acknowledgement

This work was part of a cooperation with the Institute of Applied Physics.

6. Development

6.1 Overview

Development of DBRepo.

6.1.1 Architecture Diagram

image

Figure 1: Microservice architecture of DBRepo.

6.1.2 Support

Branch	Initial Release	End of Life
1.13.x	2025-11-23	2025-12-31
1.12.x	2025-11-09	2026-11-23
1.11.x	2025-09-26	2025-11-23
1.10.x	2025-06-27	2025-09-27
1.9.x	2025-05-30	2025-08-30
1.8.x	2025-04-04	2025-07-04
1.7.x	2025-03-07	2025-06-07
1.6.x	2025-01-07	2025-04-07
1.5.x	2024-11-07	2025-01-07
1.4.x	2024-01-19	2025-04-19

6.2 Databases

6.2.1 Cache Database

Debug Information

Image: `docker.io/bitnamilegacy/valkey:8.1.3`

- Ports: 6379/tcp

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/cache-db 6379:6379
```

Overview

The Cache Database contains cached metadata and credentials that automatically expire after a set time.

Configuration

By default, all items stored have a time-to-live (TTL) of `60` seconds. This can be changed by setting `CACHE_TTL` in the Metadata Service, Data Service and Consumer Service according to the Valkey documentation.

Disable Caching

Disable caching in a service by setting the environment variable `CACHE_TTL=-1`.

Limitations

(none)

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.2.2 Data Database

Debug Information

Image: `docker.io/bitnamilegacy/mariadb:11.3.2`

- Ports: 3306/tcp
- JDBC: `jdbc://mariadb:<hostname>:3306`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/data-db 3306:3306
```

Overview

The Data Database contains the research data. In the default configuration, only one database of this type is deployed. Any number of MariaDB ata databases can be integrated into DBRepo, even non-empty databases. The database needs to be registered in the Metadata Database to be visible in the User Interface and usable from e.g. the Python Library.

Data

The procedures require the in parameter of the `hash_table` stored procedure to have the same collation as the `information_schema.columns` table. We observed this unexpected behavior for the MariaDB Galera chart powered by Bitnami and had to set extra flags.

Backup & Restore

Please refer to our detailed documentation to perform a full backup and restore.

Limitations

(none)

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.2.3 Metadata Database

Debug Information

Image: `docker.io/bitnamilegacy/mariadb:11.3.2`

- Ports: 3306/tcp
- JDBC: `jdbc://mariadb:<hostname>:3306`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/metadata-db 3306:3306
```

Overview

The metadata database is the single, central source of truth within DBRepo and holds all metadata information for interaction between the services and displaying information in the UI.

On the first start, the schema is generated by the file `/docker-entrypoint-initdb.d/setup-schema.sql` within the container. You can add custom files that should be executed on the first start by placing them into the `/docker-entrypoint-initdb.d/` folder as well. For example:

```
services:
  dbrepo-metadata-db:
    ...
    volumes:
      - /path/to/setup-schema.sql:/docker-entrypoint-initdb.d/1_setup-schema.sql
      - /path/to/setup-data.sql:/docker-entrypoint-initdb.d/2_setup-data.sql
    ...
```

Image

1.4.4

We recommend to use the MariaDB image directly instead of our own maintained image which just copied the `setup-schema.sql` into the container. This can be done more transparently through volume mounts.

Alphabetic Filename Sorting

Beware that the init script provided by Bitnami executes files in alphabetic order! For example: the file `setup-schema.sql` is executed **after** the file `setup-data.sql`! Thefore a sorting prefix 1-9 is recommended!

Backup & Restore

Please refer to our detailed documentation to perform a full backup and restore.

Limitations

(none)

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

6.2.4 Metric Database

tl;dr

> **Debug Information**
>
> Image: `bitnami/prometheus:2.54.1`
>
> - Ports: 8080/tcp
>
> To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine on port `8080`:
>
> ```
> kubectl [-n namespace] port-forward svc/data-db 8080:80
> ```

Overview

The Metric Database is responsible for saving time-series data for the Dashboard Service.

Metrics

Auth Service

See Keycloak documentation.

Broker Service

See RabbitMQ documentation.

Databases

See MariaDB Galera documentation.

Data Service

Metric	Description
`dbrepo_message_receive`	Received AMQP message from Broker Service
`dbrepo_subset_create`	Create subset
`dbrepo_subset_data`	Retrieved subset data
`dbrepo_subset_find`	Find subset
`dbrepo_subset_list`	Find subsets
`dbrepo_subset_persist`	Persist subset
`dbrepo_table_data_create`	Create table data
`dbrepo_table_data_delete`	Delete table data
`dbrepo_table_data_export`	Export table data
`dbrepo_table_data_history`	Find table history
`dbrepo_table_data_import`	Import dataset
`dbrepo_table_data_list`	Retrieve table data
`dbrepo_table_data_update`	Update table data
`dbrepo_view_data`	Retrieve view data
`dbrepo_view_schema_list`	Find view schemas

Metadata Service

Metric	Description
`dbrepo_database_count`	The total number of managed research databases
`dbrepo_view_count`	The total number of available view data sources
`dbrepo_subset_count`	The total number of available subset data sources
`dbrepo_table_count`	The total number of available table data sources
`dbrepo_volume_sum`	The total volume of available research data
`dbrepo_user_refresh_token`	Refresh user token
`dbrepo_identifier_save`	Save identifier
`dbrepo_oai_record_get`	Get the record
`dbrepo_access_give`	Give access to some database
`dbrepo_ontologies_find`	Find one ontology
`dbrepo_database_findall`	List databases
`dbrepo_tables_refresh`	Refresh database tables metadata
`dbrepo_license_findall`	Get all licenses
`dbrepo_user_modify`	Modify user information
`dbrepo_container_findall`	Find all containers
`dbrepo_maintenance_delete`	Delete maintenance message
`dbrepo_maintenance_update`	Update maintenance message
`dbrepo_ontologies_create`	Register a new ontology
`dbrepo_identifier_delete`	Delete some identifier
`dbrepo_oai_identify`	Identify the repository
`dbrepo_database_create`	Create database
`dbrepo_oai_metadataformats_list`	List the metadata formats
`dbrepo_user_password_modify`	Modify user password
`dbrepo_semantic_concepts_findall`	List semantic concepts
`dbrepo_identifier_retrieve`	Retrieve metadata from identifier
`dbrepo_identifier_list`	Find all identifiers
`dbrepo_views_findall`	Find all views
`dbrepo_identifier_create`	Draft identifier
`dbrepo_oai_identifiers_list`	List the identifiers
`dbrepo_image_findall`	Find all images
`dbrepo_database_visibility`	Update database visibility
`dbrepo_container_create`	Create container
`dbrepo_views_refresh`	Refresh database views metadata
`dbrepo_database_find`	Find some database

Metric	Description
`dbrepo_access_get`	Check access to some database
`dbrepo_identifier_find`	Find some identifier
`dbrepo_maintenance_create`	Create maintenance message
`dbrepo_container_delete`	Delete some container
`dbrepo_ontologies_delete`	Delete an ontology
`dbrepo_ontologies_findall`	List all ontologies
`dbrepo_user_token`	Obtain user token
`dbrepo_view_find`	Find one view
`dbrepo_user_create`	Create user
`dbrepo_ontologies_update`	Update an ontology
`dbrepo_maintenance_findall`	Find maintenance messages
`dbrepo_users_list`	Find all users
`dbrepo_image_find`	Find some image
`dbrepo_user_find`	Get a user info
`dbrepo_image_delete`	Delete some image
`dbrepo_identifier_publish`	Publish identifier
`dbrepo_image_update`	Update some image
`dbrepo_view_create`	Create a view
`dbrepo_semantic_units_findall`	List semantic units
`dbrepo_image_create`	Create image
`dbrepo_database_image`	Update database image
`dbrepo_view_delete`	Delete one view
`dbrepo_database_transfer`	Update database owner
`dbrepo_maintenance_find`	Find one maintenance message
`dbrepo_access_modify`	Modify access to some database
`dbrepo_ontologies_entities_find`	Find entities
`dbrepo_access_delete`	Revoke access to some database
`dbrepo_container_find`	Find some container

Search Service

Metric	Description
`dbrepo_search_index_list`	Time needed to list search index
`dbrepo_search_type_list`	Time needed to list search types
`dbrepo_search_fuzzy`	Time needed to search fuzzy
`dbrepo_search_type`	Time needed to search by type
`dbrepo_search_update_database`	Time needed to update a database in the search database
`dbrepo_search_delete_database`	Time needed to delete a database in the search database

Limitations

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.2.5 Search Database

Debug Information

Image: `docker.io/bitnamilegacy/opensearch:2.10.0`

- Ports: 9200/tcp

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/search-db 9200:9200
```

Overview

The Search Database contains duplicate metadata from the Metadata Database for fast and efficient search.

Configuration

This section will be expanded in the future.

Limitations

(none)

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

- By default, the security plugin is not used in the Docker installation. This allows anyone to read/write and is considered not secure.

6.3 Services

6.3.1 Auth Service

tl;dr

Debug Information

Image: `quay.io/keycloak/keycloak:26.4.0`

- Ports: 8080/tcp
- UI: `http://<hostname>:8080/`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/auth-service 8080:80
```

Overview

By default, users are created using the User Interface and the sign-up page in the User Interface. This creates a new user in Keycloak. The user identity is then managed by the Auth Service. Only a very small subset of immutable properties (id, username) is mirrored in the Metadata Database for faster access.

Identities

1.4.5

Identities are managed via LDAP through the Identity Service. Users can register themselves through the Auth service or via the Auth Service admin user (`admin:admin` by default). The recommended workflow is:

1. Login to the Auth Service as **Admin** and in the dbrepo realm navigate to **Users**
2. Click the **Add user** button and fill out the Username field and assign the group `researchers` by clicking the **Join Groups** and selecting it. Click **Join** and **Create**.
3. Click the **Credentials** tab above and **Set password**. In the popup window assign a secure password to the user and set **Temporary** to `Off`.

The REST API supports three kinds of authentication:

- OAuth2.0 Bearer Authentication
- Basic Authentication
- Internal Authentication: limited to a local system user that is only used between services (i.e. *never* publicly)

Auth

Figure 1: Authentication Mechanisms in DBRepo

Groups

The authorization scheme follows a group-based access control (GBAC). Users are organized in three distinct (non-overlapping) groups:

1. Researchers (*default*)
2. Developers
3. Data Stewards

Based on the membership in one of these groups, the user is assigned a set of roles that authorize specific actions. By default, all users are assigned to the `researchers` group.

Roles

We organize the roles into default- and escalated composite roles. There are three composite roles, one for each group. Each of the composite role has a set of other associated composite roles. The roles in Keycloak are mapped to authorities in the services and can be used to give fine-grained permissions to users. There is one role for one specific action in the services. For example: the `create-database` role authorizes a user to create a database.

Limitations

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

1. Keycloak should be configured to use TLS certificates, follow the official documentation.

6.3.2 Broker Service

tl;dr

Debug Information

Image: `bitnami/rabbitmq:3.13.7`

- Ports: 5672/tcp, 15672/tcp, 15692/tcp
- AMQP: `amqp://<hostname>:5672`
- Prometheus: `http://<hostname>:15692/metrics`
- Management: `http://<hostname>:15672`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/broker-service 15672:15672
```

Overview

It holds exchanges and topics responsible for holding AMQP messages for later consumption. We use RabbitMQ in the implementation. By default, the endpoint listens to the insecure port `5672` for incoming AMQP tuples and insecure port `15672` for the management UI.

Supported Protocols

- AMQP (v0.9.1, v1.0), see RabbitMQ docs.
- MQTT (v3.1, v3.1.1, v5), see RabbitMQ docs.

Authentication

The default configuration allows any user in the `cn=system,ou=users,dc=dbrepo,dc=at` from the Identity Service to access the Broker Service as user with `administrator` role, i.e. the `cn=admin,dc=dbrepo,dc=at` user that is created by default.

The Broker Service allows two ways of authentication for AMQP tuples:

1. LDAP
2. Plain (RabbitMQ's internal authentication)

Architecture

The queue architecture of the Broker Service is very simple. There is only one durable, topic exchange `dbrepo` and one quorum queue `dbrepo`, connected with a binding of `dbrepo.#` which routes all tuples with routing key prefix `dbrepo.` to this queue.

Data ingest

The consumer takes care of writing it to the correct table in the Data Service.

Data ingest

Limitations

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

For a secure deployment it is necessary to configure the Broker Service as follows:

1. Once you change the admin password of the Identity Service, you need to change it in the `rabbitmq.conf` as well: `auth_ldap.dn_lookup_bind.password=newpassword`.
2. Enable TLS and mount your previously generated certificate and RSA public key pair (PEM-encoded) to `/app/cert.pem` and `/app/pubkey.pem`. Note that these are *not* used for TLS encryption, but only for authentication of users. It is not recommended to use "real" TLS certificates, self-signed certificates with *sufficient keylength* are best-practice. Mount your TLS certificate authority file into `/etc/rabbitmq/cacert.crt` and your TLS certificate / private key pair into `/etc/tls/tls.crt` and `/etc/tls/tls.key`.

6.3.3 Data Service

tl;dr

Debug Information

Image: `registry.datalab.tuwien.ac.at/dbrepo/data-service:1.13.3`

- Info: `http://<container_ip>:8080/actuator/info`
- Health: `http://<container_ip>:8080/actuator/health`
- Readiness: `http://<container_ip>:8080/actuator/health/readiness`
- Liveness: `http://<container_ip>:8080/actuator/health/liveness`
- Prometheus: `http://<container_ip>:8080/actuator/prometheus`
- Swagger UI: `http://<container_ip>:8080/swagger-ui/index.html`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/data-service 8080:80
```

Overview

The Data Service is responsible for inserting AMQP tuples from the Broker Service into the Data DB via Spring AMQP. To increase the number of consumers, scale the Data Service up.

Data Analytics

The Data Service uses Apache Spark, a open-source data analytics engine to load data from/into the Data Database with a wide range of open-source connectors. Retrieving data from a subset internally generates a view with the 64-character hash of the query. This view is not automatically deleted currently.

The Data Service also uses DuckDB, a open-source column-oriented database for large-data analytics. It is used as in-memory data type analytics database to suggest data types from a (big) dataset using the larger-than-memory feature.

Caching

The Data Service uses Caffeine, a caching solution that is used to temporarily cache the connection details from the Metadata Service such that they don't have to be queried everytime e.g. a sensor measurement is inserted. By default, this information is stored for 60 minutes. System administrators can disable this behavior by setting `CREDENTIAL_CACHE_TIMEOUT=0` (cache is deleted after 0 seconds).

Storage

The Data Service also is capable to upload files to the S3 backend. The default limit of `Tomcat` in Spring Boot is configured to be `2GB`. You can provide your own limit with setting `MAX_UPLOAD_SIZE`.

By default, the Data Service removes datasets older than 24 hours on a regular basis every 60 minutes. You can set the `MAX_AGE` (in seconds) and `S3_STALE_CRON` to fit your use-case. You can disable this feature by setting `S3_STALE_CRON` to `-`, this may lead to storage issues as no space will be available inevitably. Note that Spring Boot uses its own flavor of cron syntax.

Limitations

- Views in DBRepo can only have 63-character length (it is assumed only internal views have the maximum length of 64 characters).

- Local mode of embedded processing of Apache Spark directly in the service using a `local[2]` configuration.

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.3.4 Dashboard Service

tl;dr

Debug Information

Image: `registry.datalab.tuwien.ac.at/dbrepo/dashboard-service:1.10.0`

- Health: `http://<container_ip>:8080/api/dashboard/health`
- Prometheus: `http://<container_ip>:8080/metrics`
- Swagger UI: `http://<container_ip>:8080/swagger-ui/`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/dashboard-service 8080:80
```

Overview

tbd

Example

More info on how to customize database dashboards can be found in the User Guide.

Limitations

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.3.5 Gateway Service

tl;dr

Debug Information

Image: `docker.io/nginx:1.27.3-alpine3.20-slim`

- Ports: 80/tcp

Overview

Provides a single point of access to the *application programming interface* (API) and configures a standard NGINX reverse proxy for load balancing. This component is optional if you already have a load balancer or reverse proxy running.

Settings

SSL/TLS SECURITY

To setup SSL/TLS encryption, mount your TLS certificate and TLS private key into the container directly into the `/etc/nginx/` directory.

docker-compose.yml

```
services:
  ...
  dbrepo-gateway-service:
    image: docker.io/nginx:1.25-alpine-slim
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./fullchain.pem:/etc/nginx/fullchain.pem
      - ./privkey.pem:/etc/nginx/privkey.pem
  ...
```

If your TLS private key as a password, you need to specify it in the `dbrepo.conf` file.

CONNECTION TIMEOUTS

The reverse proxy has a defined timeout of 90 seconds on all requests (these are also enforced in the User Interface using the `axios` module). For large databases these timeouts may need to be increased, e.g. the timeout for creating subsets is by default already increased to 600 seconds.

USER INTERFACE

To serve the User Interface under different port than `80`, change the port mapping in the `docker-compose.yml` to e.g. port `8000`:

docker-compose.yml

```
services:
  ...
  dbrepo-gateway-service:
    image: docker.io/nginx:1.27.0-alpine3.19-slim
    ports:
      - "8000:80"
  ...
```

Monitoring (Optional)

By default the Gateway Service is not monitored. You need to add the following to the `docker-compose.yml` file.

docker-compose.yml

```
services:
  ...
```

```
  dbrepo-gateway-service-sidecar:
    restart: "no"
    container_name: dbrepo-gateway-service-sidecar
    hostname: dbrepo-gateway-service-sidecar
    image: docker.io/nginx/nginx-prometheus-exporter:1.3.0
    command:
      - "-nginx.scrape-uri=http://gateway-service/basic_status"
    ports:
      - "9113:9113"
    depends_on:
      dbrepo-gateway-service:
        condition: service_started
    logging:
      driver: json-file
```

Then, uncomment the scrape config from the `prometheus.yml` file.

prometheus.yml

```
scrape_configs:
  ...
  - job_name: 'gateway scrape'
    metrics_path: '/metrics'
    static_configs:
      - targets: ['dbrepo-gateway-service-sidecar:9113']
```

Limitations

(none relevant to DBRepo)

> **Do you miss functionality? Do these limitations affect you?**
>
> We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

1. Enable TLS encryption by downloading the `dbrepo.conf` and editing the *server* block to include your TLS certificate (with trust chain) `fullchain.pem` and TLS private key `privkey.pem` (PEM-encoded).

```
server {
  listen 443 ssl;
  server_name _;
  ssl_certificate     /etc/nginx/fullchain.pem;
  ssl_certificate_key /etc/nginx/privkey.pem;
  ...
}
```

6.3.6 Identity Service

tl;dr

> **Debug Information**
>
> Image: `docker.io/bitnamilegacy/openldap:2.6.8`
>
> - Ports: 1389/tcp, 1636/tcp
>
> To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:
>
> ```
> kubectl [-n namespace] port-forward svc/identity-service 1389:389
> ```

Overview

This optional service holds the user identities which we simply call identities in the following. It is integrated into the Auth Service through an LDAP federation, allowing any identity to authenticate through the Auth Service. The LDAP protocol is not used for authentication.

The Identity Service can be optionally replaced with your existing LDAP solution. Your LDAP solution should store users using the RFC 2798 `InetOrgPerson` schema which is standard to most LDAP solutions.

Identities

Any identity is identified by its `entryUUID` by default in the Auth Service. Note that Keycloak (the software running the Auth Service) may assign a different UUID to a user. DBRepo **always** uses the UUID provided through the Identity Service.

The field `uid` is the username and is used for bind/unbind operations. The fields `cn` and `sn` are ignored by the Auth Service and can be empty `""`.

Limitations

- Limited support for scaling in Kubernetes, see the guide of the chart developers.
- Currently no support for LDAP in the Data Database.
- Currently no support for LDAP in the Search Database.

> **Do you miss functionality? Do these limitations affect you?**
>
> We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

1. By default, no ingress is enabled. If you need ingress on LTP Password and phpLDAPadmin, configure the ingress to use your TLS secret `tls-cert-secret` containing the `tls.crt` and `tls.key`, e.g.:

values.yaml

```yaml
identityservice:
  ltb-passwd:
    ingress:
      enabled: true
      tls:
        - secretName: tls-cert-secret
          hosts:
            - example.com
```

```
  phpldapadmin:
    ingress:
      enabled: true
      tls:
        - secretName: tls-cert-secret
          hosts:
            - example.com
```

6.3.7 Metadata Service

tl;dr

> **Debug Information**
>
> Image: `registry.datalab.tuwien.ac.at/dbrepo/metadata-service:1.11.0`
>
> - Info: `http://<container_ip>:8080/actuator/info`
> - Health: `http://<container_ip>:8080/actuator/health`
> - Readiness: `http://<container_ip>:8080/actuator/health/readiness`
> - Liveness: `http://<container_ip>:8080/actuator/health/liveness`
> - Prometheus: `http://<container_ip>:8080/actuator/prometheus`
> - Swagger UI: `http://<container_ip>:8080/swagger-ui/index.html`
>
> To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:
>
> ```
> kubectl [-n namespace] port-forward svc/metadata-service 8080:80
> ```

Overview

The metadata service manages metadata of identities, the Broker Service (i.e. obtaining queue types), semantic concepts (i.e. ontologies) and relational metadata (databases, tables, queries, views) and identifiers.

Generation

DBRepo generates metadata for managed tables automatically by querying MariaDB's internal structures (e.g. `information_schema`).

> **Managed Tables**
>
> DBRepo only manages system-versioned tables, other tables are not supported. These other tables are ignored by DBRepo and thus can co-exist in the same database. If you want a non-system-versioned table `my_table` to be managed by DBRepo, make it system-versioned:
>
> ```
> ALTER TABLE `my_table` ADD SYSTEM VERSIONING;
> ```
>
> Then, refresh the managed table index by navigating to your database > Settings > Schema > Refresh. This action can only be performed by the database owner.

Identifiers

The service is responsible for creating and resolving a *persistent identifier* (PID) attached to a database, subset, table or view to obtain the metadata attached to it and allow reproduction of the exact same result.

The service generates internal PIDs, essentially representing internal URIs in the DataCite Metadata Schema 4.4. This can be enhanced with activating the external DataCite Fabrica system to generate DOIs, this is disabled by default.

To activate DOI minting, pass your DataCite Fabrica credentials in the environment variables:

docker-compose.yml

```
services:
  dbrepo-metadata-service:
    image: registry.datalab.tuwien.ac.at/dbrepo/metadata-service:1.11.0
```

```
    environment:
      spring_profiles_active: doi
      DATACITE_URL: https://api.datacite.org
      DATACITE_PREFIX: 10.12345
      DATACITE_USERNAME: username
      DATACITE_PASSWORD: password
  ...
```

Semantics

The service provides metadata to the table columns in the Metadata Database from registered ontologies like Wikidata `wd:`, Ontology of Units of Measurement `om2:`, Friend of a Friend `foaf:`, the `prov:` namespace, etc.

Limitations

- No support for other databases than MariaDB because of system-versioning capabilities missing in other database engines.

> **Do you miss functionality? Do these limitations affect you?**
>
> We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.3.8 Search Service

tl;dr

Debug Information

Image: `registry.datalab.tuwien.ac.at/dbrepo/search-service:1.11.0`

- Health: `http://<container_ip>:8080/api/search/health`
- Prometheus: `http://<container_ip>:8080/metrics`
- Swagger UI: `http://<container_ip>:8080/swagger-ui/`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/search-service 8080:80
```

Overview

This service communicates between the Search Database and the User Interface to allow structured search of databases, tables, columns, users, identifiers, views, semantic concepts & units of measurements used in databases.

Built-in search

Index

There is only one index `database` that holds all the metadata information which is mirrored from the Metadata Database.

Mirroring statistical properties in Metadata Database and Search Database

Faceted Browsing

This service enables the frontend to search the `database` index with eight different *types* of desired results (database, table, column, view, identifier, user, concept, unit) and their *facets*.

For example, the User Interface allows for the search of databases that contain a certain semantic concept (provided as URI, e.g. temperature http://www.wikidata.org/entity/Q11466) and unit of measurement (provided as URI, e.g. degree Celsius http://www.ontology-of-units-of-measure.org/resource/om-2/degreeCelsius).

Limitations

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.3.9 Storage Service

tl;dr

Debug Information

Image: `docker.io/chrislusf/seaweedfs:3.71.0`

- Ports: 8888/tcp, 9000/tcp
- Prometheus: `http://<hostname>:9091/metrics`
- Filer UI: `http://<hostname>:8888`
- Cluster UI: `http://<hostname>:9333`

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/storage-service-s3 9000:8333
```

Overview

We use SeaweedFS as a high-performance, S3 compatible object store for easy, cloud-ready deployments that by default support replication and monitoring. No graphical user interface is provided out-of-the-box, administrators can access the S3 storage via S3-compatible clients e.g. AWS CLI (see below).

The default configuration creates admin credentials `seaweedfsadmin:seaweedfsadmin`. By default, one bucket `dbrepo` is created that holds uploads temporarily. It is recommended to delete the contents regularly.

The S3 endpoint of the Storage Service is available on port `9000`.

FILER UI

The storage service comes with a simple UI that can be used to explore the uploaded files, rename them and delete them. Please note that the Filer UI is not intended for production and should be turned off for security purposes.

Filer UI with a list of uploaded files in the bucket dbrepo

Limitations

- No support for multiple regions.

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

1. For public deployments, change the default credentials.

6.4 UI

6.4.1 Repository

tl;dr

Debug Information

Image: `registry.datalab.tuwien.ac.at/dbrepo/ui:1.11.0`

- Ports: 3000/tcp

To directly access in Kubernetes (for e.g. debugging), forward the svc port to your local machine:

```
kubectl [-n namespace] port-forward svc/ui 3000:80
```

The User Interface is configured in the `runtimeConfig` section of the `nuxt.config.ts` file during build time. For the runtime, you need to override those values through environment variables or by mounting a `.env` file. As a small example, you can configure the logo ① below. Make sure you mount the logo as image as well, in this example we want to mount a custom logo `my_logo.png` into the container and specify the name.

Architecture of the UI microservice

Docker Compose Kubernetes

Text values like the title ② can be configured as well via the Nuxt runtime configuration through single environment variables or `.env` files.

.env

```
NUXT_PUBLIC_TITLE="My overriden title"
NUXT_PUBLIC_LOGO="https://mydomain/my_logo.png"
NUXT_PUBLIC_ICON="https://mydomain/my_favicon.ico"
...
```

To work, you need to serve the `my_logo.png` and `my_favicon.ico` from a separate webserver. Note that simply copying the files into the Nuxt `public/` directory will not work as the content length is calculated only during build time. The development team #19263 does not plan to fix this.

Text values like the title ② can be configured as well via the Nuxt runtime configuration through adding the logo file as `ConfigMap`.

```
kubectl [-n namespace] create configmap gateway-service-config \
  --from-file=logo.png
```

Then you need to mount the configmap into the Gateway Service under `/etc/nginx/assets/assets`.

dbrepo-ui-custom.yaml

```
gatewayservice:
  extraVolumes:
    - name: config-map
      configMap:
        name: gateway-service-config
  extraVolumeMounts:
    - name: config-map
      mountPath: /etc/nginx/assets/assets
```

All files mounted that way are accessible through svc/gateway-service:80 (or ingress if you enabled it) with prefix `/assets`, e.g. `https://<hostname>/assets/logo.png`. Therefore, set the logo path:

values.yaml

```
ui:
  public:
    api:
      logo: "https://<hostname>/assets/logo.png"
...
```

ARCHITECTURE

The server-client architecture of the User Interface is shown below, it is supposed to help debug the User Interface on development.

Architecture of the UI microservice

- Runtime: Node.js 22 LTS
- Builder: Vite
- Server: Nuxt.js 3+
- Components: Vue.js 3+
- Frontend: Vuetify 3+
- State: Pinia

CUSTOMIZATION

The UI supports adding pages in Markdown format to describe the terms of use, policies and the repository itself. Inject your content through the environment variables:

- `NUXT_PUBLIC_ABOUT_CONTENT` (`/about`)
- `NUXT_PUBLIC_POLICIES_CONTENT` (`/policies`)
- `NUXT_PUBLIC_TERMS_CONTENT` (`/terms`)

If any of these environment variables are not empty, they will be displayed on the navigation and the content will be rendered.

Limitations

- When developing locally, the `axios` module does not parse custom headers (such as `X-Count`, `X-Headers`) and/or blocks CORS requests wrongfully.

Do you miss functionality? Do these limitations affect you?

We strongly encourage you to help us implement it as we are welcoming contributors to open-source software and get in contact with us, we happily answer requests for collaboration with attached CV and your programming experience!

Security

(none)

6.5 Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

6.5.1 v1.13.3 - 2025-12-??

Changes

- Disable Dashboard UI by default in #584.

Fixes

- Fix metadata retrieval from ROR.org when parsing non-English organizations in #578.
- Fix a bug where the cached database is not removed when the table visibility changes in #585.

6.5.2 v1.13.2 - 2025-12-10

Changes

- Map all JSON and CSV in Spark directly instead of the Data Service in #577.

6.5.3 v1.13.1 - 2025-12-03

Features

- Generate secrets on first install in #575.

6.5.4 v1.13.0 - 2025-11-23

Fixes

- Reduce dependencies for Java-based services to resolve a bug where Spring runs into problems with verifying OAuth tokens.
- Fixed the storage of the `nuxt-oidc-auth` module in the UI to share the session secrets in the Cache Database.

Features

- Support `JOIN` queries when generating views and subsets in #574.

6.5.5 v1.12.1 - 2025-11-14

Changes

- Changed the Data Service to provide hash metadata (`md5` , `sha1` , `sha256`) and on duplication do not attempt to upload the file again in #573.

6.5.6 v1.12.0 - 2025-11-10

Fixes

- Fixed a silent connection issue causing the consumer in the Consumer Service to die silently + having no chance of recovery by configuring the `ConnectionFactory` correctly.
- Fixed a bug where sometimes DuckDB has issues with loading the extensions. Instead now load them from an offline directory in #568.

Features

- Added an integration test to the CI/CD pipeline that ensures the Broker Service is configured properly (e.g. MQTT is enabled).
- Re-hash subset results and expose them via `X-Result-Hash` header to enable verification in #569.

Changes

- Refactored the user information out of the Metadata Database into the Auth Service (i.e. Auth DB) to get rid of consistency problems related to SSO-based identity providers.
- Bumped Keycloak from version `26.2.4` to `26.4.4` to include default-values for profile attributes.
- Changed the Bitnami catalog to the legacy repository `docker.io/bitnamilegacy` where possible instead of our local registry which was meant only as temporary fix.

6.5.7 v1.11.0 - 2025-10-13

Changes

- Changed not-anymore-existing `docker.io/bitnami` based images to the mirrored images at our local registry `registry.datalab.tuwien.ac.at/bitnami`.

6.5.8 v1.11.0 - 2025-09-25

Fixes

- Fixed the schema column add feature in #561.

Changes

- Bumped Java version to 21 in #562.

6.5.9 v1.10.5 - 2025-09-05

Changes

- Added a free-text field to optionally add data description comments to each column in #564.
- Updated the version of all Java-based services from 17 (EOL) to 21 and updated the version of all Node-based services from 20 (EOL) to 24 in #562.

6.5.10 v1.10.4 - 2025-08-04

CHANGES

- Added the language next to descriptions and titles in #559.
- Added the verbs `ListRecords` and `ListSets` to comply with OpenAIRE Provide and make the records findable using the OAI-PMH harvesting protocol.

6.5.11 v1.10.3 - 2025-07-28

FIXES

- Fixed a wrong configuration of the Auth Service in the Helm Chart, where new pods reset the realm settings, leading to a deletion of e.g. the SSO settings in #550.

6.5.12 v1.10.2 - 2025-07-12

FIXES

- Fixed a bug where saving a PID multiple times created duplicate titles, creators, descriptions, funders, etc. in #545.

6.5.13 v1.10.1 - 2025-07-10

FIXES

- Fixed a bug where PID presentation in the UI did not select the newest PID.
- Fixed a bug where a user could get a different id from the Identity Service when using a different authentication provider (i.e. SAML)

CHANGES

- Changed the mechanism to obtain the subset columns to use DuckDB and now create views anymore.
- Changed the internal user identifier from using the user id (i.e. `java.util.UUID`) to the username (i.e. `java.lang.String`) for all interfaces including the query store due to continued instability from SSO-integrations relying on the `EntryUUID` in the Identity Service.

6.5.14 v1.10.0 - 2025-07-05

REMOVALS

- Removed the Analyse Service in favor of DuckDB in the Data Service covering the same functionality.

FIXES

- Fixed a bug where saving a PID multiple times broke the entity linking within Hibernate in #545.

6.5.15 v1.9.3 - 2025-06-06

Features

- Added a Jupyter Starter notebook for data scientists that is pre-configured to use DBRepo in #529.

Changes

- Gave the documentation website a major overhaul with detailed user guide, maintainer guide, animated step-by-step manuals and flowcharts for better overview in #514.

Fixes

- Fixed a bug where identifiers could not be created because of missing `ordinal_position` field for creators.
- Fixed multiple bugs where access grants were mapped wrongly.
- Fixed a bug where subsets could not be created because the wrong data source (table) was used in the data service.
- Fixed a bug in the UI where grants were not displayed once created/update/revoked.

6.5.16 v1.9.2 - 2025-06-05

FEATURES

- Added an endpoint to check for database grants and display them in the UI (Database > Settings) for the owner in #536.

6.5.17 v1.9.1 - 2025-05-31

CHANGES

- Release a new patch since version 1.9.0 was accidentally released on PyPI.

6.5.18 v1.9.0 - 2025-05-30

FIXES

- Fixed a bug where another PID was registered when using the DOI profile in the Metadata Service.
- Fixed a bug where titles, descriptions, creators, etc. were not sorted to the user-specified ordering in #531.
- Fixed a design issue where the `get_identifier_data` method in the Python library only fetched the first 10.000 rows and could not paginate in #527.
- Fixed a bug where Spark did not map the column headers correct when importing a dataset in #518.

CHANGES

- Improved S3-related mechanisms to de-duplicate uploaded datasets and remove them on successful import, various structured logging improvements in #528.

6.5.19 v1.8.2 - 2025-05-15

FIXES

- Fixed a bug in the UI where the resource status was displayed as identifier when a draft identifier has been created.

FEATURES

- Added structured logging through the `fluentd` protocol via the lightweight fluentbit in a separate Logging Service. in #524.
- Added a separate database for the Dashboard Service for high-availability deployment of Grafana in #526.

CHANGES

- Improved internal packaging mechanism that is compatible with multiarch deployments in #523.

6.5.20 v1.8.1 - 2025-04-13

CHANGES

- Improved default mask for PIDs in #521.
- Added a visual filter for displaying starred/unstarred/all subsets in the UI.
- Specified image platform as `linux/amd64` in `docker-compose.yaml` deployment to enable host platform (e.g. ARM) to emulate it.
- Specified resource limits in the `docker-compose.yaml` deployment in #517.

FIXES

- Fixed a bug in the UI that displays the "Create View" button only when the user has at least read access.

REMOVALS

- Removed the stale objects scheduler from the Data Service and pushed it to next release.

6.5.21 v1.8.0 - 2025-04-04

FEATURES

- Refactored internal Java-based testing data that improves test consistency in #510.
- Added automated dashboard generation for all public databases where each view has an overview of its data and schema in #460.

CHANGES

- Removed OpenSearch security plugin from the Docker test deployment and changed to the `bitnami/opensearch` image of the same version in #515.

FIXES

- Fixed a bug where validation of missing `Principal` object in Java services caused a 400 error instead of a 401 error in #512.

6.5.22 v1.7.3 - 2025-03-17

FIXES

- Fixed a wrong configuration where assets were not considered in the Kubernetes deployment.
- Fixed a wrong configuration in the Docker deployment where the OIDC provider did not consider other URLs than `http://localhost`.

6.5.23 v1.7.2 - 2025-03-13

FIXES

- Fixed a wrong configuration of `caffeine` in the Data Service that did not find views/subsets after table creation within the cache period of 60 seconds in #506.

6.5.24 v1.7.1 - 2025-03-06

FEATURES

- Added support to download `pandas` DataFrame by PID in #503.
- Added the possibility to create and fill a table from a `pandas` DataFrame (or optionally just create the schema) in #496.

FIXES

- Fixed a bug where quick interaction with the UI caused the user to trigger the brute-force login detection in #501.

6.5.25 v1.7.0 - 2025-03-03

⚠ Contains Breaking Changes

This release updates the Metadata Database schema which is incompatible to v1.6.3! Follow the steps:

1. Make a backup of the database with `mariadb-dump`.
2. Apply the schema changes script: `schema.sql`:

```
mariadb -h 127.0.0.1 -p3306 -u root --password=<password> -D dbrepo < schema.sql
```

3. Install the dependencies from the `requirements.txt` file or use your local environment:

```
pip install dbrepo==1.6.5rc15
```

4. Run the data migration script `data.py`:

```
python data.py > data.sql
```

It generates the SQL statements used for migrating to the new schema.

5. Run the generated `data.sql` script:

```
mariadb -h 127.0.0.1 -p3306 -u root --password=<password> -D dbrepo < data.sql
```

FEATURES

- Implemented a basic brute-force security defense strategy in the Auth Service that increments the wait time on wrong logins in #494.
- Implemented a password policy in #495.

CHANGES

- Replaced sequential numerical ids with non-guessable random ids in the Metadata Database in #491.
- Changed the interface for executing query in subsets/views in #493.

REMOVALS

- Removed the Upload Service in favor of an internal stable upload endpoint in the Data Service in #492.

6.5.26 v1.6.5 - 2025-02-18

FIXES

- Fixed a bug where listing the views in the Python library did not work.
- Fixed a wrong MariaDB configuration where the `innodb_buffer_pool_size` variable was not configured to 70% of the available memory in the Helm chart.

6.5.27 v1.6.4 - 2025-02-14

FIXES

- Fixed a bug where the users were not synced with the Metadata Database in #489.

6.5.28 v1.6.3 - 2025-02-05

CHANGES

- Refactored the UI to support OIDC and added an event listener to the Auth Service that syncs users on creation to the Metadata DB in #488.

6.5.29 v1.6.2 - 2025-01-24

CHANGES

- Added interface tests for the Python library in Gitlab CI/CD pipeline in #486.

FIXES

- Fixed a bug where no pagination was possible in #487.

6.5.30 v1.6.1 - 2025-01-21

CHANGES

- Added privacy feature for hidden databases (and optionally tables, views, subsets) that hides them completely from e.g. the search in #482.

FIXES

- Added init container that adds the admin user to the Metadata Database in #480.

6.5.31 v1.6.0 - 2025-01-07

FEATURES

- Added possibility to modify table description and privacy mode that hides metadata of databases, tables, subsets and views in #472.
- Added Auth Service init container to sync internal system ("admin") user into Metadata Database in #470

CHANGES

- Drop MariaDB Galera chart version from `14.0.12` to `13.2.7` because of stability issues with the newer versions on the OpenShift Kubernetes cluster.
- Bumped Grafana chart version from `10.1.1` to `11.4.2`.
- Bumped NGINX chart version from `18.2.6` to `18.3.1`.
- Bumped SeaweedFS chart version from `1.0.2` to `4.2.1` (created a pull request #31192).
- Improve RESTfulness of HTTP API to e.g. not include the database image due to size in #463.

FIXES

- Fixed a wrong JPA entity configuration in Hibernate that in some cases causes the `SERIAL` increment to not trigger in MariaDB Galera. Removed the deprecated `@GenericGenerator` annotation on JPA entities.
- Fixed a bug where the dataset separator was being ignored for imports in #478.

6.5.32 v1.5.3 - 2024-12-13

FIXES

- Fixed a bug where subsets containing sub-queries are not able to retrieve data in #476.

6.5.33 v1.5.2 - 2024-12-03

CHANGES

- Adapt Helm chart to support `runAsNonRoot` throughout and specify `resource` presets for the highly-constrained OpenShift Kubernetes environment in #467.
- Require authentication for uploading files in #466.

FIXES

- Fixed a validation problem failing to validate UUIDs in #471.
- Fixed the `dist.tar.gz` file not being found in the CI/CD pipeline on `release-` branches in #465.

6.5.34 v1.5.1 - 2024-11-09

FIXES

- Bug where the data volume could not be calculated when the data length column in the Metadata Database is `null` in #462.
- Bug where the schema could not be created manually in #461.

6.5.35 v1.5.0 - 2024-11-06

> ⚠ **Contains Breaking Changes**
>
> This release updates the Metadata Database schema which is incompatible to v1.4.6! Use the migration script `schema_1.4.5-to-1.5.0.sql` to apply the changes manually.

FEATURES

- Added `SERIAL` data type to create incrementing key in #454

CHANGES

- Remove the Data Database Sidecar and replace it with Apache Spark in #458.
- Allow anonymous users to create subsets for public databases in #449.
- Show file upload progress in #448.
- Change the Docker image of the Auth Service to Bitnami-maintained similar to Kubernetes deployment with accompanying Auth Database change to PostgreSQL in #455

FIXES

- Preventing the semicolon `;` to be used in UI and fixed cryptic subset execution error messages in #456.
- Multiple UI errors in #453.
- Fixed install script.sh in #444
- No hardcoded data type metadata in UI but instead added it hardcoded (associated with `image_id`) Metadata Database.

6.5.36 v1.4.6 - 2024-10-11

> ⚠ **Contains Breaking Changes**
>
> This release updates the Metadata Database schema which is incompatible to v1.4.5!

FEATURES

- Added Dashboard Service and monitoring in default setup.

CHANGES

- Show the progress of dataset uploads in the UI in #448
- Anonymous users are allowed to create (non-persistent) subsets in #449
- Removed logic that maps `True`, `False` and `null`

FIXES

- Import of datasets stabilized in the UI in #442
- Install script in #444

6.6 Contributing

We welcome contributions to DBRepo!

6.6.1 Dependencies

Local development depends on the following packages for Debian 12:

```
apt install maven openjdk-17-jdk make nodejs npm
```

Required tools for local development:

- Docker Engine (`docker --version` must be at least `24.x`)
- Minikube (`minikube version` must be at least `1.32.x`)

6.6.2 Getting Started

Build the Docker images from source and run them:

```
make start-dev
```

6.6.3 Testing

We practice test-driven development and require contributors to test their code with at least 80% code coverage.

The Java-based services have the coverage reports generated by `JaCoCo` in the `report/site/` subdirectory, the Python-based services have the coverage reports generated by `coverage` in the `.coverage` SQLite3 database and `coverage.txt` log file respectively.

6.6.4 Code Versioning

Branching Strategy

Branching strategy from the master-dev-feature branches and release branches

CI/CD

We get compute resources in-kind from dataLAB to run our pipeline:

Gitlab runner configuration in the cluster

For each job in the CI/CD pipeline, a pod with three containers is started:

1. `build` the main build container, you can *freely* specify any image with `image: xyz` as base
2. `helper` the default helper container.
3. `svc-0` the Docker-in-Docker sidecar (rootless executed as user `rootless` / `1000`) exposing the Docker socket to the `build` container under `

Note. Only Docker-in-Docker (dind) is allowed as service in the pipeline currently. For each job, a dind-sidecar container `svc-0` is started that exposes the Docker socket at `/var/run/dind/docker.sock` in the `build` container you can freely configure how you want. We are aware this is not optimal as it exposes *root* privileges in the cluster.

6.6.5 Documentation

For consistency reasons across the documentation, the resolution needs to be 1280x800 (16:10 ratio)

7. Publications

How to Cite DBRepo

Please cite the following paper:

Weise, M., Staudinger, M., Michlits, C., Gergely, E., Stytsenko, K., Ganguly, R., & Rauber, A. (2022). DBRepo: a Semantic Digital Repository for Relational Databases. *International Journal of Digital Curation*, 17(1), 11. DOI: 10.2218/ijdc.v17i1.825

[BibTeX] [RIS] [RDF] [EndNote]

7.1 Logos

DBRepo logo in various formats:

- PNG: bigger (smaller)
- SVG: bigger (smaller)

7.2 Refereed

2024

- Weise, M., & Rauber, A. (2024). DBRepo: A Data Repository System for Research Data in Databases, in *Proceedings of the 2024 IEEE International Conference on Big Data*, pp. 322–331, December 15-18th (Washington DC, USA). DOI: 10.1109/bigdata62323.2024.10825401
  [PDF] [Slides]
- Altug, B., Weise, M., & Rauber, A. (2024). Generating Semantic Context for Data Interoperability in Relational Databases using BGE M3-Embeddings, at *ML in PL Conference*. [Poster]. DOI: 10.34726/7280

2023

- Weise, M., Miksa, T., Grantner, T., Taha, J., Moser, M., Tsepelakis, S., Sanchez-Solis, B. & Rauber, A. (2023). Repository Infrastructure Supporting Virtual Research Environments. [Poster]. *International Data Week*, October 23-26th (Salzburg, Austria). DOI: 10.34726/5143
  [PDF] [Poster]
- Weise, M., Miksa, T., Grantner, T., Taha, J., Moser, M., Tsepelakis, S., Sanchez-Solis, B. & Rauber, A. (2023). Research Data Infrastructure Landscape at TU Wien. [Poster]. *Austrian-Slowenian High Performance Computing Meeting*.
  [PDF] [Poster]
- Weise, M., Grantner, T., Taha, J., Staudinger, M., Gergely, E., Stytsenko, K., Ganguly, R. & Rauber, A. (2023). DBRepo: A Repository for Databases supporting Data Versioning, Schema Semantics and FAIR Principles, at *Open Repositories Conference*, June 12-15th. (Stellenbosch, South Africa). June 12-15th, 2023. DOI: 10.5281/zenodo.8075496
  [PDF]
- Weise, M., Knees, P., Hofmann, A., Ahmedaja, A., Beitane, A. & Rauber, A. (2023). Connecting Ethnomusicology Data Collections Using Distributed Repositories and Linked Data Technology. *Proceedings of the 3rd Conference of the ICTM National Committee for Portugal*, (Aveiro, Portugal). DOI: 10.34726/4323

2022

- Ekaputra, F. E., Weise, M., Flicker, K., Salleh, M. R., Rahman, N. A., Miksa, T., & Rauber, A. (2022). Towards A Data Repository for Educational Factories. *Proceedings of the 8th International Conference on Data and Software Engineering*, pp. 149-154. DOI: 10.1109/ICoDSE56892.2022.9971958

- Weise, M., Staudinger, M., Michlits, C., Gergely, E., Stytsenko, K., Ganguly, R., & Rauber, A. (2022). DBRepo: a Semantic Digital Repository for Relational Databases. *International Journal of Digital Curation*, 17(1), 11. DOI: 10.2218/ijdc.v17i1.825

2021

- Weise, M., Michlits, C., Staudinger, M., Gergely, E., Stytsenko, K., Ganguly, R. and Rauber A., 2021. FDA-DBRepo: A Data Preservation Repository Supporting FAIR Principles, Data Versioning and Reproducible Queries. *Proceedings of the 17th International Conference on Digital Preservation*, Beijing, China, p.34. DOI: 10.17605/OSF.IO/B7NX5

7.3 Other

- Weise, M. (2023). A Repository and Compute Environment for Sensitive Data. FAIRness for Closed Data, at *EMBL Bioimaging and the European Open Science Cloud*, (Heidelberg, Germany). April, 19-20th, 2023. DOI: 10.34726/3946
- Staudinger, M. (2022). DBRepo: A Repository to Save Research Databases. [Online]. URL: https://www.tuwien.at/en/tu-wien/news/news-articles/news/dbrepo accessed 2022-04-12.
- Gergely, E. (2021). Better Support for Research: Current Cooperation Projects. [Online]. URL: https://zid.univie.ac.at/it-news/artikel/news/cluster-forschungsdaten/ accessed 2022-04-12.

8. Contact

8.1 Team

8.1.1 Strategy & Partnerships

Ao.univ.Prof. Dr. Andreas Rauber
Technische Universität Wien
Research Unit Data Science
Favoritenstraße 9-11
A-1040 Vienna, Austria

8.1.2 Technical Lead

Projektass. Dipl.-Ing. Martin Weise
Technische Universität Wien
Research Unit Data Science
Favoritenstraße 9-11
A-1040 Vienna, Austria

8.2 Contributors (alphabetically)

- Ganguly, Raman
- Gergely, Eva
- Güçlü, Gökay
- Grantner, Tobias
- Karnbach, Geoffrey
- Lukic, Nikola
- Mahler, Lukas
- Michlits, Cornelia
- Rauber, Andreas
- Spannring, Max
- Staudinger, Moritz
- Stytsenko, Kirill
- Taha, Josef
- Tsepelakis, Sotirios
- Weise, Martin

Interested in contributing? Send us an e-mail!