
Metadata Stores: The TROMPA perspective

Alastair Porter (MTG-UPF)
LinkedMusic Project Meeting
21 October 2023

TROMPA

Towards Richer Online Music Public domain Archives

<https://trompamusic.eu/>

EU Funded project from 2018-2021

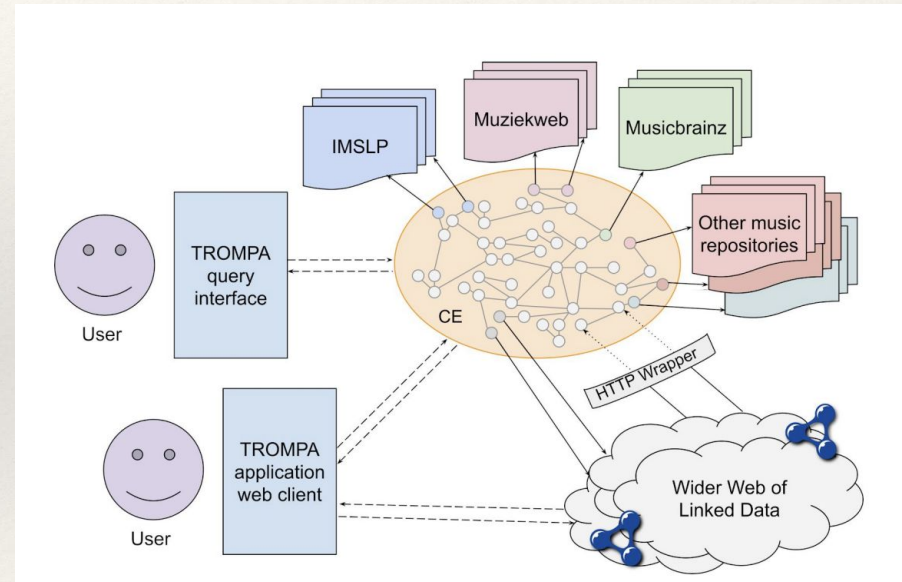
Consortium of 4 universities and 5 companies

Some of the things we made

The “Contributor Environment”

Main interface to the data published by the project

The idea was that each project would interact with



Contributor Environment

GraphQL interface

Playground - https://api.tromp... x +

← → ↻ https://api.trompamusic.eu

MusicComposition Person multiple ItemList MusicComposition MediaObject SoftwareApplication

PRETTIFY HISTORY https://api.trompamusic.eu/ COPY CURL

```
1 query {
2   Person(identifier: "349c6350-3945-4d36-ae88-e1b763f3d432") {
3     identifier
4     source
5     title
6     birthDate {
7       formatted
8     }
9     birthPlace {
10      ... on Place {
11        identifier
12        name
13        source
14      }
15    }
16    exactMatch {
17      ... on Person {
18        source
19      }
20    }
21  }
```

Hit the Play Button to get a response here

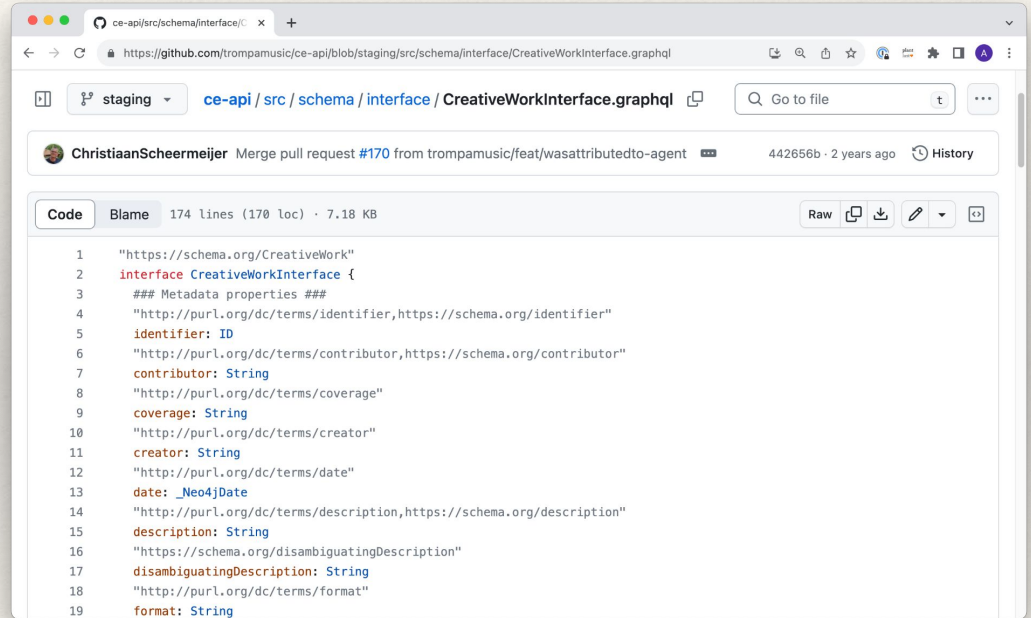
DOCS
SCHEMA

4

Contributor Environment

Data structure is based on Linked Data schemas (mostly schema.org with other things added as we needed)

Custom interface to output json-ld based on the schema definition



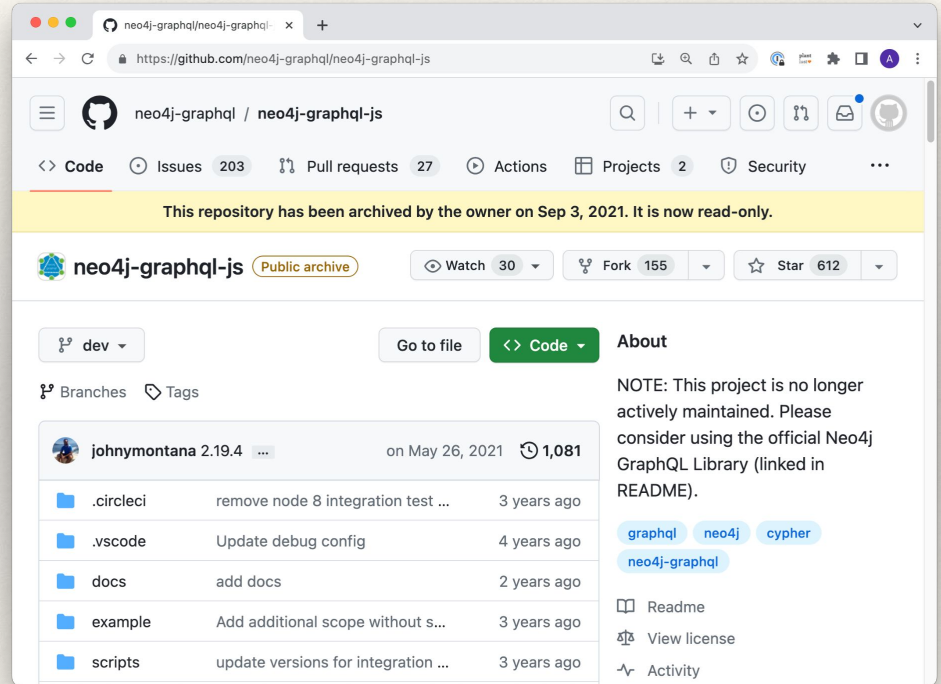
The screenshot shows a web browser displaying a GitHub repository page for the file `ce-api/src/schema/interface/CreativeWorkInterface.graphql`. The page shows a merge pull request by ChristianaScheermeijer. The code content is as follows:

```
1 "https://schema.org/CreativeWork"
2 interface CreativeWorkInterface {
3   ### Metadata properties ###
4   "http://purl.org/dc/terms/identifier,https://schema.org/identifier"
5   identifier: ID
6   "http://purl.org/dc/terms/contributor,https://schema.org/contributor"
7   contributor: String
8   "http://purl.org/dc/terms/coverage"
9   coverage: String
10  "http://purl.org/dc/terms/creator"
11  creator: String
12  "http://purl.org/dc/terms/date"
13  date: _Neo4jDate
14  "http://purl.org/dc/terms/description,https://schema.org/description"
15  description: String
16  "https://schema.org/disambiguatingDescription"
17  disambiguatingDescription: String
18  "http://purl.org/dc/terms/format"
19  format: String
```

<https://github.com/trompamusic/ce-api>

Contributor Environment

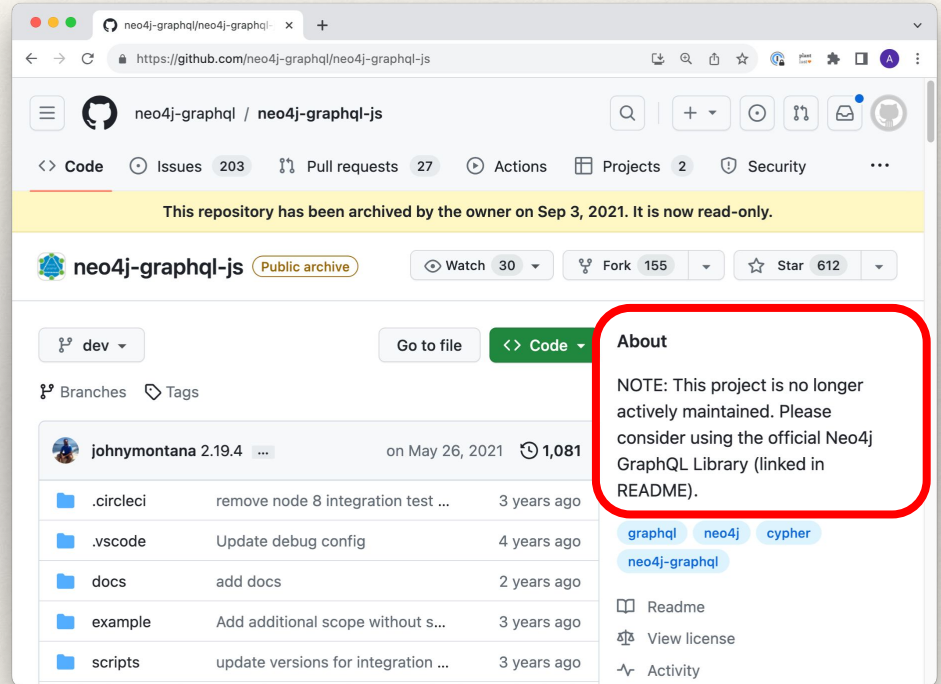
Uses Neo4J Graph Database
ORM to automatically turn a
GraphQL schema into a
database structure and map
API queries to the database



Contributor Environment

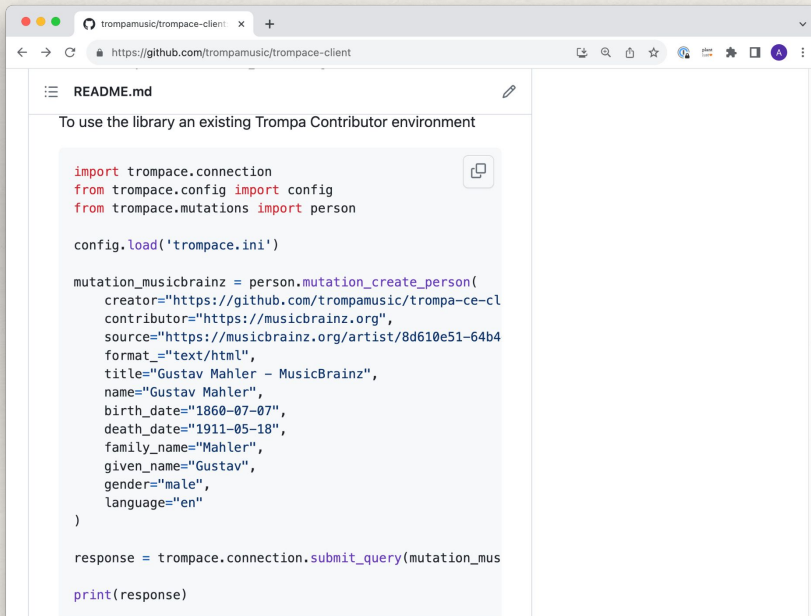
Uses Neo4J Graph Database
ORM to automatically turn a
GraphQL schema into a
database structure and map
API queries to the database

... deprecated half way
through the project



Clients to read/write data

Python client library to interact with CE (using GraphQL)



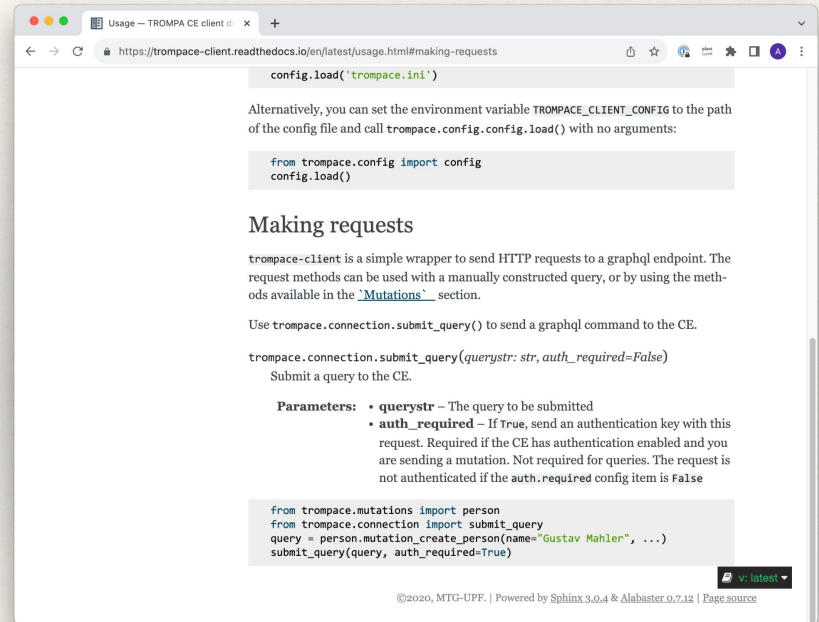
The screenshot shows the GitHub README for the trompace-client library. The title is "README.md". The main heading is "To use the library an existing Trompa Contributor environment". Below this, there is a code block containing Python code for setting up the client and making a request to create a person in MusicBrainz.

```
import trompace.connection
from trompace.config import config
from trompace.mutations import person

config.load('trompace.ini')

mutation_musicbrainz = person.mutation_create_person(
    creator="https://github.com/trompamusic/trompa-ce-cl
    contributor="https://musicbrainz.org",
    source="https://musicbrainz.org/artist/8d610e51-64b4
    format_='text/html",
    title="Gustav Mahler - MusicBrainz",
    name="Gustav Mahler",
    birth_date="1860-07-07",
    death_date="1911-05-18",
    family_name="Mahler",
    given_name="Gustav",
    gender="male",
    language="en"
)

response = trompace.connection.submit_query(mutation_mus
print(response)
```



The screenshot shows the documentation page for the trompace-client library. The title is "Usage - TROMPA CE client". The page contains instructions on how to use the library, including how to load the configuration file and how to make requests to the CE.

```
config.load('trompace.ini')
```

Alternatively, you can set the environment variable `TROMPACE_CLIENT_CONFIG` to the path of the config file and call `trompace.config.config.load()` with no arguments:

```
from trompace.config import config
config.load()
```

Making requests

`trompace-client` is a simple wrapper to send HTTP requests to a graphql endpoint. The request methods can be used with a manually constructed query, or by using the methods available in the [Mutations](#) section.

Use `trompace.connection.submit_query()` to send a graphql command to the CE.

```
trompace.connection.submit_query(querystr: str, auth_required=False)
```

Submit a query to the CE.

Parameters:

- `querystr` – The query to be submitted
- `auth_required` – If True, send an authentication key with this request. Required if the CE has authentication enabled and you are sending a mutation. Not required for queries. The request is not authenticated if the `auth_required` config item is False

```
from trompace.mutations import person
from trompace.connection import submit_query
query = person.mutation_create_person(name="Gustav Mahler", ...)
submit_query(query, auth_required=True)
```

©2020, MTG-UPF. | Powered by Sphinx 3.0.4 & Alabaster 0.7.12 | Page source

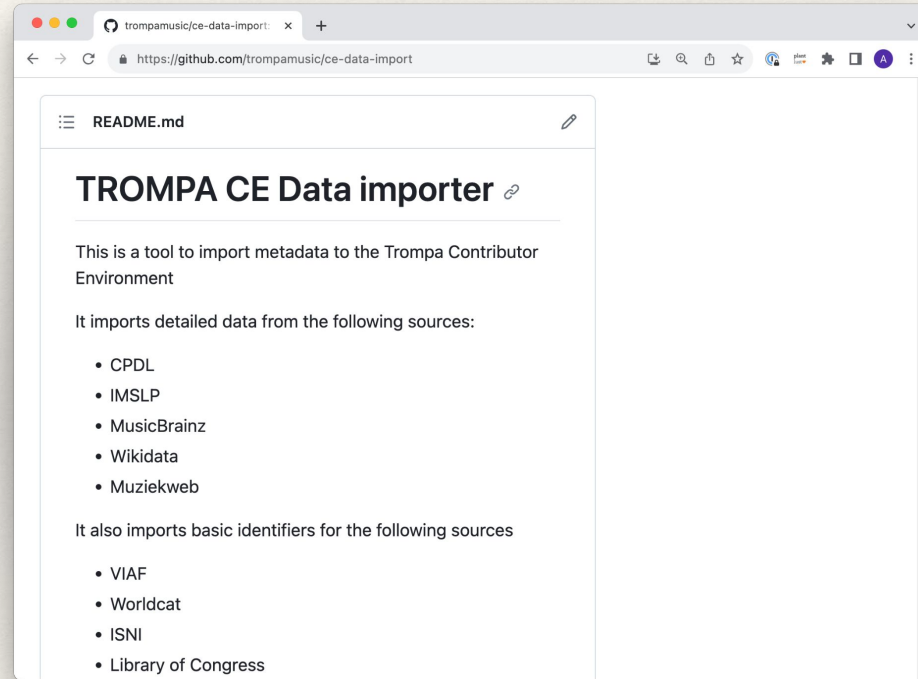
8 <https://github.com/trompamusic/trompace-client>

Populating CE with content

The CE was not designed to be a data repository or a source of truth

Rather it would contain references to other repositories on the internet and help to join them together

We populated it with sample data from a number of repositories



The Bad...

Constrained by deliverables

The proposal outlined our deliverables ahead of time

In some cases, the deliverables in the proposal were written by the grant writer and not the partner working on the task

Work package number	5		Lead beneficiary				VD		
Work package title	TROMPA Contributor Environment								
Participant number	1	2	3	4	5	6	7	8	9
Short name of participant	UPF	TUD	GOLD	MDW	VD	PN	VL	RCO	CDR
Person months per participant	21	6	8	8	14	6	8	3	2
Start month	1				End month			34	

Objectives

This work package lays the fundament for the pilots developed in WP6. It delivers an environment for mid-level integration of components that will be further exploited in WP6 pilots. In order to do so, the data produced in WP3 (musical repertoire, automatic descriptions and generated audio) and annotations delivered through WP4 need to be made accessible and usable in reusable components, meeting common standards (Objective [O3] of the project). The expected outcomes and success criteria are the following ones:

- Online TROMPA data infrastructure for musical component creation;
- a digital score edition component;
- a library for embeddable descriptions of music data coming from supported repositories;
- a list of piece and performance assessment mechanisms adapted for specific music profiles (e.g. singing) and repertoire (e.g. early music), relevant and valid for each configuration; and
- an annotation tool including a set of technical components that integrates crowd planning strategies and allow crowdsourcing input

Description of work

Task 5.1: Data infrastructure [VD, UPF, TUD, VL, PN, CDR]

This task will be devoted to develop a sustainable online data infrastructure available online that turns musical data processing results from WP3 (audiovisual material, scores, metadata from the selected repertoire together with automatic descriptions, alignments, synthesised material and processed audio files) and WP4 (annotations and interpretations by the crowd) into reusable components. Common API access conventions will be implemented. For storage, we will rely on existing online repositories developed by partners and Associated Partners as much as possible.

Task 5.2: Digital score edition [GOLD, RCO, MDW, VD, UPF, TUD]

This task will focus on the integration of data and technologies (WP3–WP4) for the collaborative edition of digital scores. In this functionality, users will be able to access to digital scores that can be annotated and linked. The will ultimately lead to detailed linking and annotation across and between score documents and audio recordings. This task will be aligned with the Music Encoding Initiative (MEI). A digital score edition component, including a test-set fully marked up, and the publication of this test-set as Linked Data, will be generated.

Task 5.3: Multimodal integration of music data [VD, MDW, GOLD, UPF, TUD, PN, RCO, CDR]

This task focuses on the creation, combination and integration of musical data in different modalities, such that their combination can be used in multimodal multi platform end user experiences that will be used as a generic starting point for the pilots defined in WP6.

Task 5.4: Music performance assessment mechanisms [MDW, GOLD, UPF, TUD, VL, PN]

This task aims at investigating ways to formalise expert (musicologists and educators) and crowd (music enthusiasts) knowledge on various aspects of the multiple performances and the scores linked to them (T3.5 and T5.3) in assessing their individual qualities. Strategies will be defined to identify valid representations of overall (per performance or piece) and detailed (section-wise, T3.6) ratings of individual aspects of performance quality (such as intonation and voice quality in case of singing or technical brilliance

What is the goal of a task?

Is the focus on the process, the tools, or the result?

In some cases, partners used tasks as a motivation to learn a new tool/technology. By the time we thought that it might not be a great fit, they were invested in its development

If the functionality of a database or other resource is important, choose boring technology: <https://boringtechnology.club/>

Linked Data: Building a schema

TROMPA was multi-faceted; no one schema was able to capture all the data that we ended up with

The initial data model started with schema.org but we then determined that it didn't do everything that we needed to do

Upfront all-singing-all-dancing schema based on *assumed* requirements

We didn't use entire parts of the schema that were developed up front. This resulted in maintenance work for things that were never used

Key data types such as the annotations data model and alignment data model were "bolted on" and not part of the same schema

We swapped between dcelements and dcterms as we determined what was needed. This took time to go back and update the whole schema.

What are you going to make? What resources do those things need? Base your data source/model on those requirements

Keep your schema simple: fulfill your initial requirements. Add to it as you need

Data repositories

We built a repository of data and populated it with existing sources

There was no data or relationships in the CE which didn't already exist in other databases

If we created original content in the CE, what was our goal? Would we expect other people to start using our database as a source of truth?

Could we have just used an existing data repository?

Linked data: Synchronisation


Why are you collecting everything if it's already in all of the other public sources? Will you go 2-way with the other sources? How will you keep data up-to-date and synchronized?

Environment creep: we were stuck with the CE as a neo4j database, but then found that we wanted to added rdf/semantic web concepts and this required us to "bolt on" additional functionality


Linked data: Availability

Don't rely on external services to be up all the time, especially if you want to perform sparql queries to them on demand


← Post

 **Alastair Porter**
@alastairporter

Reviewing a paper on the semantic web. They have a demo. Business as usual for linked data...



This site can't be reached

 took too long to respond.

Try:

- Checking the connection
- [Checking the proxy and the firewall](#)

ERR_CONNECTION_TIMED_OUT

4:16 PM · Jul 22, 2018

Takeaways

Don't make infrastructure choices without knowing what everyone in the project needs

Don't invent something all-singing-all-dancing, start with what you need first

Use tools that have a chance of being supported in the long term (or mid-term!)