Hands-on

Ecole Polytech'Nice Sophia

Franck MICHEL

Your mission: integrate data from 2 data sources using SPARQL

- Figure out a use case to integrate data from:
  - **1 CSV file** of your choice

    Translate the CSV file into RDF using CSVW.

    Example: find concerts of a given artist in a region

    Possible source: https://www.data.gouv.fr/fr/datasets/?format=csv&tag=musique
  - **1 Web API** of your choice

    Create a SPARQL micro-service using an appropriate vocabulary.

    Example: find facts about the same artists from MusicBrainz/Spotify/Deezer/SoundCloud
- Execute a query (e.g. in Corese) that involves both data sources
  - Load the RDF data translated from CSV into Corese
  - Query the SPARQL μ-service with a SERVICE clause

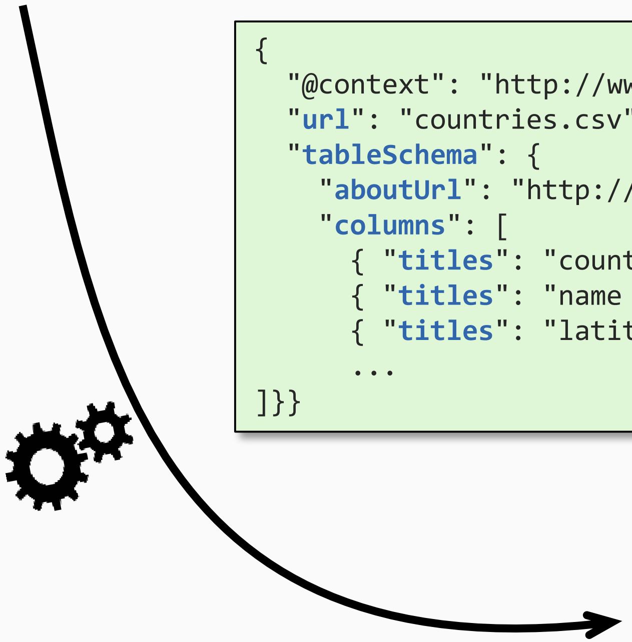
CSV on the Web

CSVW: CSV on the Web

https://www.w3.org/TR/tabular-data-primer/
https://www.w3.org/2013/csvw/

countries.csv

```
"country","country group","name (en)","name (fr)","latitude",   "longitude"
"at",     "eu",           "Austria",  "Autriche", "47.6965545", "13.34598005"
"be",     "eu",           "Belgium",  "Belgique", "50.501045",  "4.47667405"
"bg",     "eu",           "Bulgaria", "Bulgarie", "42.72567375","25.4823218"
```



JSON-LD

```
{
  "@context": "http://www.w3.org/ns/csvw",
  "url": "countries.csv",
  "tableSchema": {
    "aboutUrl": "http://example.org/country/{country}",
    "columns": [
      { "titles": "country group", "propertyUrl": "http://example.org/vocab/countryGroup" },
      { "titles": "name (en)", "propertyUrl": "schema:name", "lang": "en" },
      { "titles": "latitude", "datatype": "number", "propertyUrl": "schema:latitude" },
      ...
]}}
```

CSVW transformation written in JSON-LD

```
<http://example.org/country/at>
  <http://example.org/vocab/countryGroup> "eu";
  <http://schema.org/name>       "Austria"@en, "Autriche"@fr;
  <http://schema.org/latitude>   47.6965545;
  <http://schema.org/longitude> 13.34598005.
```

CSVW: CSV on the Web

countries.csv

```
"country","country group","name (en)","name (fr)","latitude",   "longitude"
"at",     "eu",           "Austria",  "Autriche", "47.6965545", "13.34598005"
"be",     "eu",           "Belgium",  "Belgique", "50.501045",  "4.47667405"
"bg",     "eu",           "Bulgaria", "Bulgarie", "42.72567375","25.4823218"
```

metadata.json

```
{
  "@context": "http://www.w3.org/ns/csvw",
  "url": "countries.csv",

  "tableSchema": {
    "primaryKey": "code",
    "aboutUrl": "http://example.org/country/{code}",

    "columns": [{
      "titles": "country",
      "name": "code",
      "required": true,
      "suppressOutput": true
    },{
      "titles": "country group",
      "name": "country_group",
      "propertyUrl": "http://example.org/vocab/countryGroup"
    },{
      "titles": "name (en)",
      "propertyUrl": "schema:name",
      "lang": "en"
    },{
      "titles": "name (fr)",
      "propertyUrl": "schema:name",
      "lang": "fr"
    },{
      "titles": "latitude",
      "datatype": "number",
      "propertyUrl": "schema:latitude"
    },{
      "titles": "longitude",
      "datatype": "number",
      "propertyUrl": "schema:longitude"
    },{
      "virtual": true,
      "propertyUrl": "rdf:type",
      "valueUrl": "schema:Country"
}]}}
```

CSVW processors: https://csvw.org/tools.html

https://github.com/Swirrl/csv2rdf/

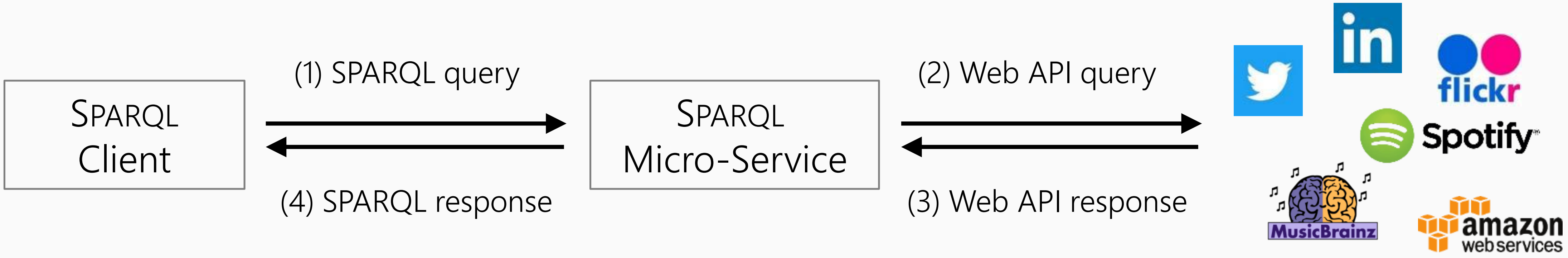
run.sh

```
./csv2rdf \
    --mode minimal \
    --user-metadata metadata.json \
    --output-file countries.ttl
```

SPARQL micro-services

The SPARQL Micro-Service Architecture

Lightweight method to **query a Web API with SPARQL,** and assign dereferenceable URIs to Web API resources



SPARQL Client

(1) SPARQL query

(4) SPARQL response

SPARQL Micro-Service

(2) Web API query

(3) Web API response

https://github.com/frmichel/sparql-micro-service

A SPARQL µ-service is a CONFIGURABLE SPARQL endpoint whose ARGUMENTS delineate the graph being queried.

Utiliser ca :

```
Endpoint: http://example.org/flickr/getPhotosByTag?tag=bridge

SELECT * WHERE {
    ?photo a schema:Photograph;
      schema:name       ?title;
      schema:contentUrl ?img.
}
```

Arguments passed as HTTP parameters

```
Endpoint: http://example.org/flickr/getPhotosByTag_sd

SELECT * WHERE {
    ?photo a schema:Photograph;
      schema:keywords   "bridge";
      schema:name       ?title;
      schema:contentUrl ?img.
}
```

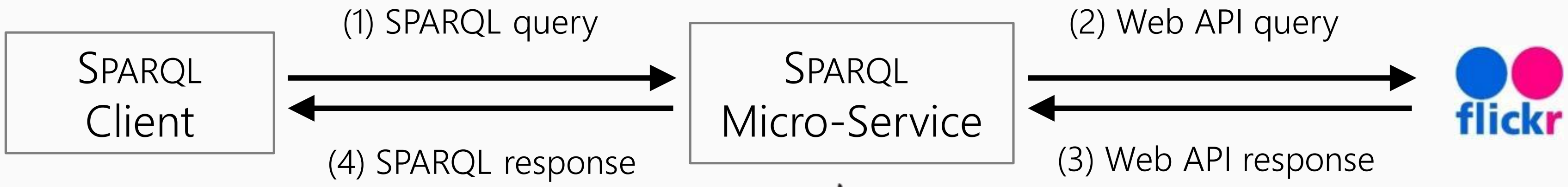
Arguments passed in the graph pattern

Into the details - Arguments passed **as HTTP parameters**

```
SELECT * WHERE {
  ?photo a schema:Photograph;
     schema:name        ?title;
     schema:contentUrl ?img.
}
```

Endpoint: http://example.org/flickr/getPhotosByTag?tag=bridge

https://api.flickr.com/services/rest/?
method=flickr.photos.search&
format=json&per_page=100&tags=bridge&…



(1) SPARQL query

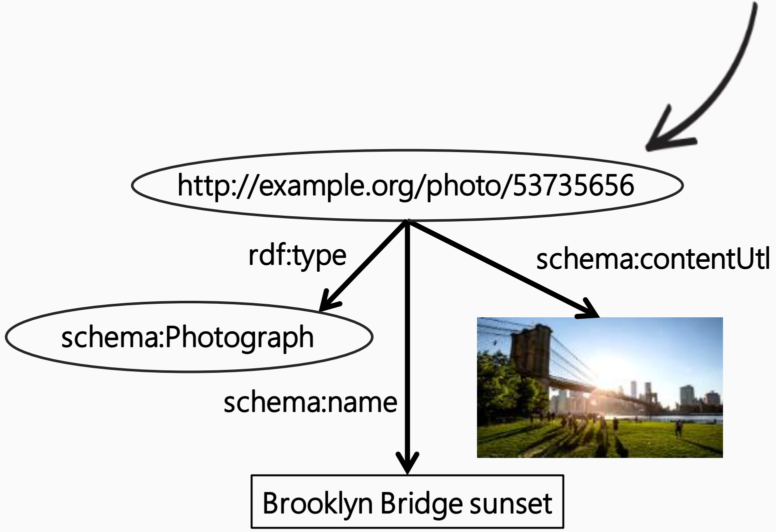
SPARQL Client

SPARQL Micro-Service

(4) SPARQL response

(2) Web API query

(3) Web API response



http://example.org/photo/53735656

rdf:type

schema:contentUtl

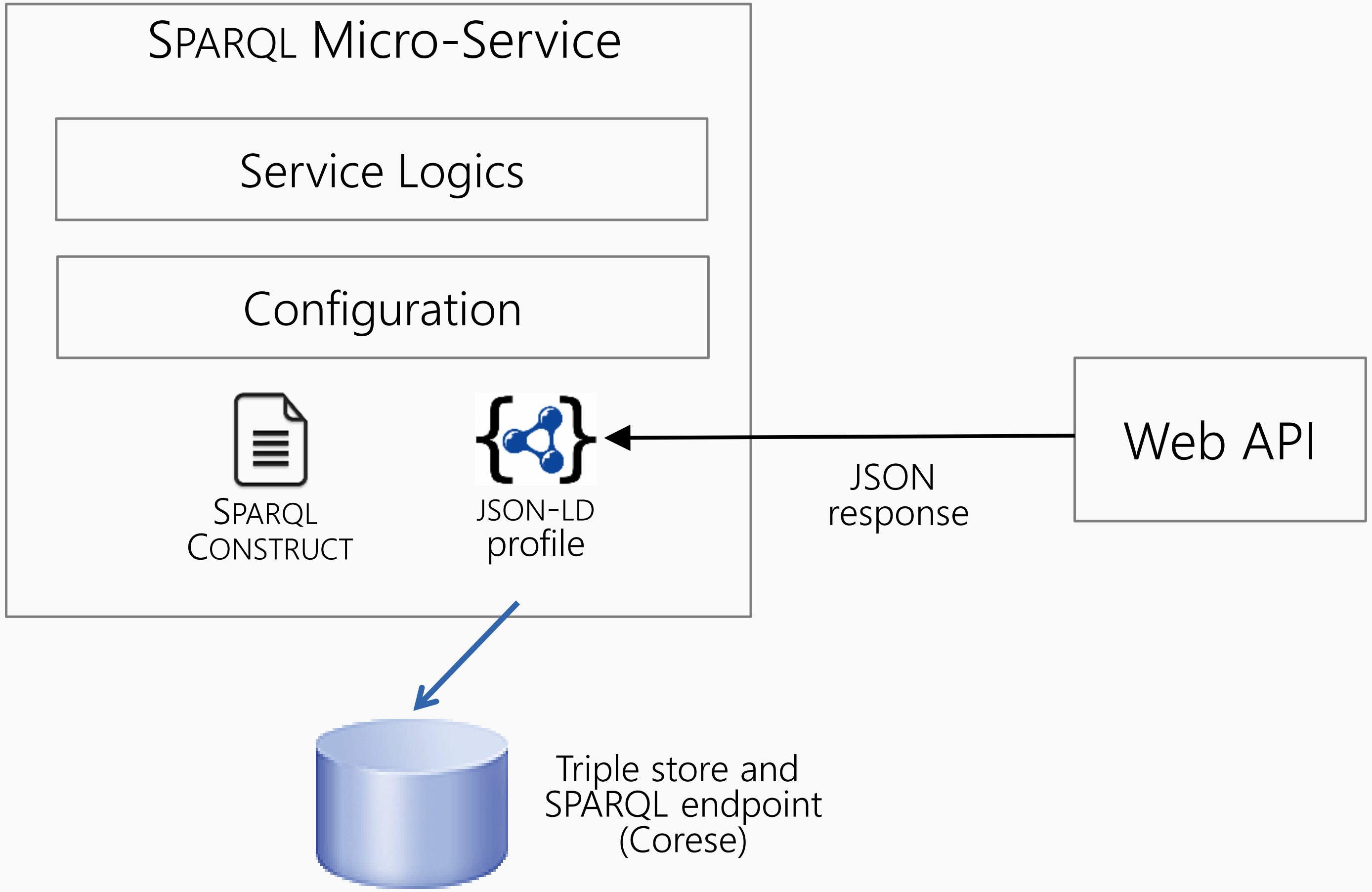
schema:Photograph

schema:name

Brooklyn Bridge sunset

```
{ "photos": {
    "page": 1, "pages": 1, "total": "189", …
    "photo": [
      { "id": "53735656",
        "title": "Brooklyn Bridge sunset", … },
      …
```

Translating the Web API response to RDF



First translation with a JSON-LD context

API response

```
{
  "id": "53735656",
  "title": "Brooklyn Bridge sunset",
}
```

profile.jsonld

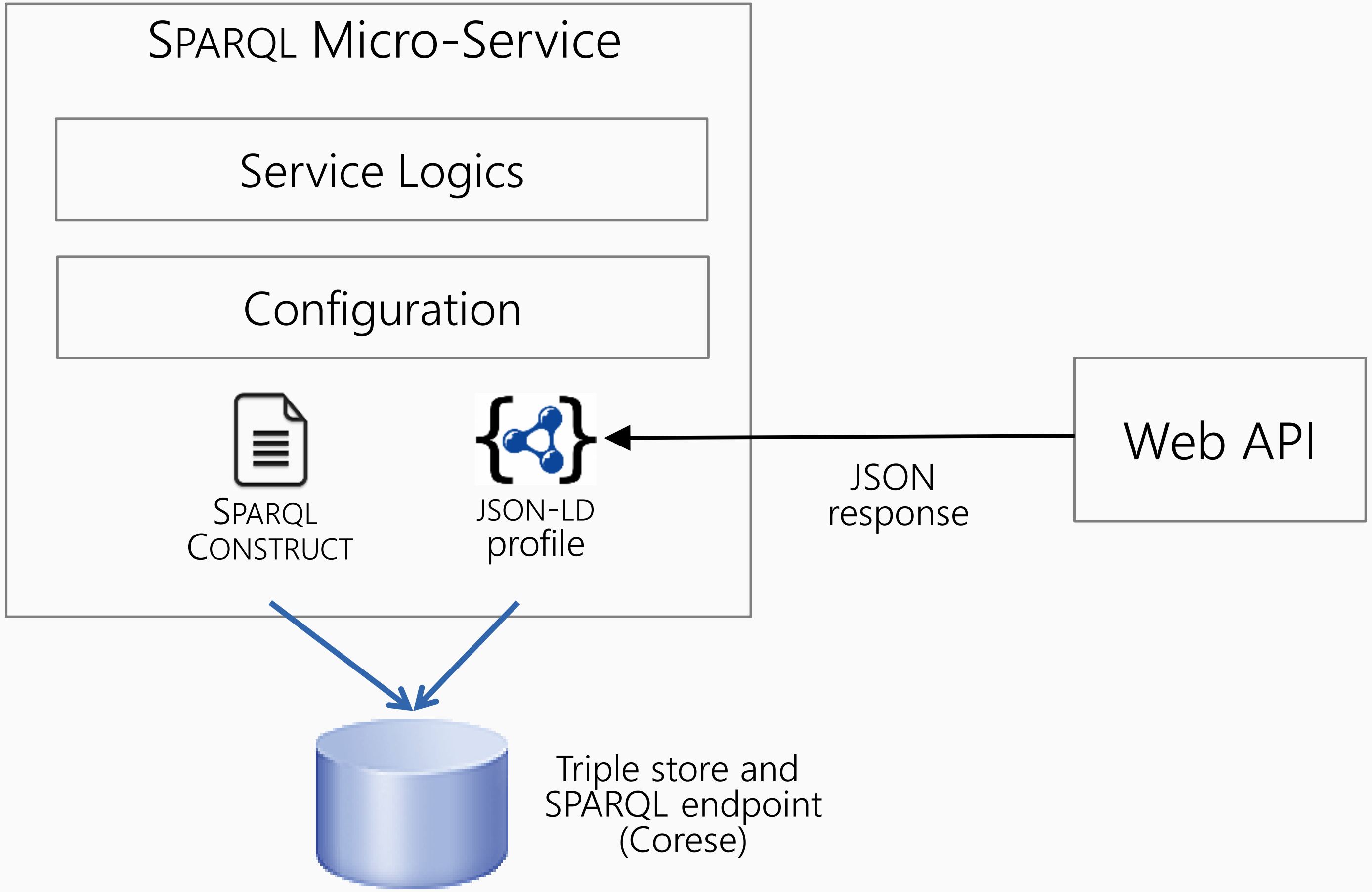
```
{
  "@context": {
    "@vocab": "http://ns.inria.fr/sparql-micro-service/api#"
  }
}
```

```
[] <http://ns.inria.fr/sparql-micro-service/api#id>    "53735656";
   <http://ns.inria.fr/sparql-micro-service/api#title> "Brooklyn Bridge sunset";
```

```
@prefix api: <http://ns.inria.fr/sparql-micro-service/api#>

[] api:id      "53735656";
   api:title   "Brooklyn Bridge sunset";
```

Translating the Web API response to RDF



Advanced mapping with SPARQL (optional)

```
@prefix api: <http://ns.inria.fr/sparql-micro-service/api#>

[] api:id      "53735656";
   api:title   "Brooklyn Bridge sunset";
```

SPARQL

```
PREFIX schema: <http://schema.org/>
CONSTRUCT {
  ?photoUri
    a            schema:Photograph;
    schema:name  ?title;
}
WHERE {
  ?result
    api:id      ?photoId;
    api:title   ?title;

  BIND(IRI(concat("http://example.org/ld/photo/", ?photoId)) AS ?photoUri)
}
```

```
@prefix schema: <http://schema.org/>

<http://example.org/ld/photo/53735656>
    a           schema:Photograph;
    schema:name "Brooklyn Bridge sunset".
```

Quick-start guide

Main approach to create a SPARQL µ-service

Read the API documentation:
find out the API service that does what you want to do

Create a basic SPARQL micro-service:
a JSON-LD profile translates the API response into “raw” RDF (namespace “api”)

Define a mapping to an RDF vocabulary:
figure out an appropriate vocabulary for your use case

Write the mapping in a CONSTRUCT query
and test the service.

My first SPARQL micro-service

Goal: find music albums on Deezer, that match a keyword

Find out the right API query

Create a basic SPARQL micro-service

Decide of a mapping to an RDF vocabulary

Write the mapping in a CONSTRUCT query

Check the Web API documentation

https://developers.deezer.com/api/search#connections

Choose a service to be fulfilled by the SPARQL micro-service

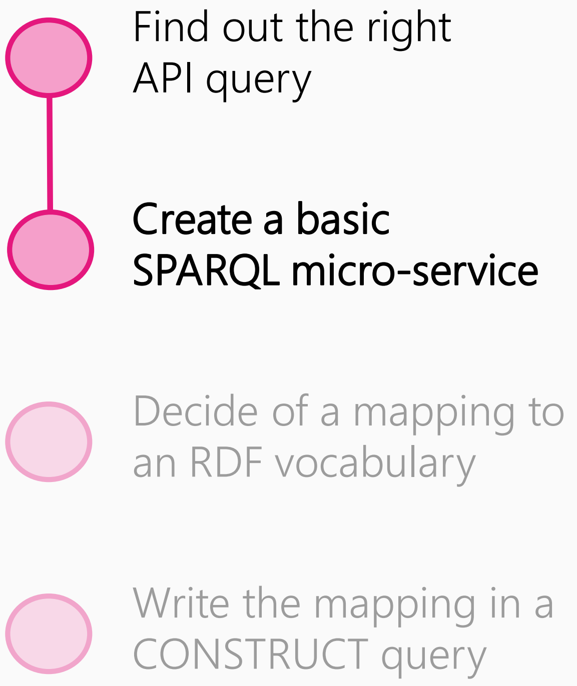
Query music albums by keyword (search > Search Methods)

Find out the right query to do this

https://api.deezer.com/search/album?q=eminem

My first SPARQL micro-service

Documentation:
https://s.42l.fr/smsdoc



Find out the right API query

Create a basic SPARQL micro-service

Decide of a mapping to an RDF vocabulary

Write the mapping in a CONSTRUCT query

1. CD to the directory of deployed services.

2. Create directory `deezer/findAlbums`

3. In `deezer/findAlbums`, create file `config.ini` with 2 properties:

```
api_query = "https://api.deezer.com/search/album?q={keyword}"
custom_parameter[] = keyword
```

Create file profile.jsonld

```
{ "@context": {
    "@vocab": "http://ns.inria.fr/sparql-micro-service/api#"
}}
```

Query the SPARQL micro-service (using Yasgui):

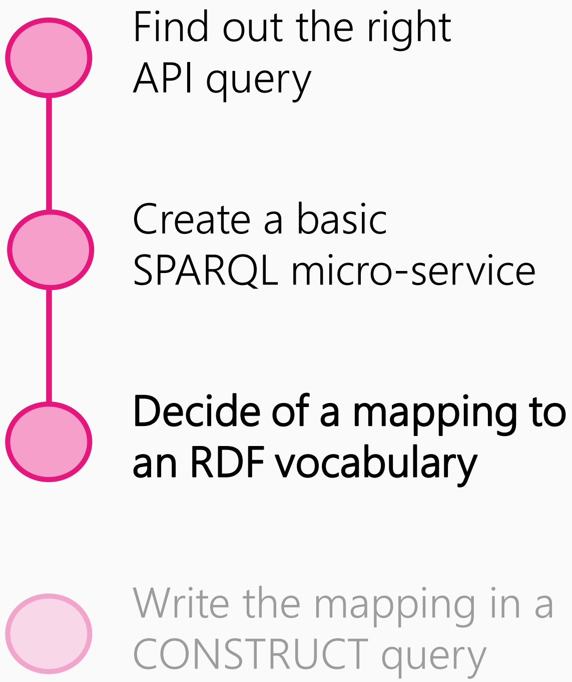
Endpoint URL:

```
http://localhost/service/deezer/findAlbums?keyword=eminem
```

Query:

```
select * where { ?s ?p ?o. }
```

My first SPARQL micro-service



Find out the right API query

Create a basic SPARQL micro-service

Decide of a mapping to an RDF vocabulary

Write the mapping in a CONSTRUCT query

Find appropriate vocabularies

- Schema.org https://schema.org/docs/full.html
- Wikidata https://wikidata.org
- LOV (Linked Open Vocabularies) https://lov.linkeddata.es/dataset/lov/
- Specialized ontology portals…

Schema.org

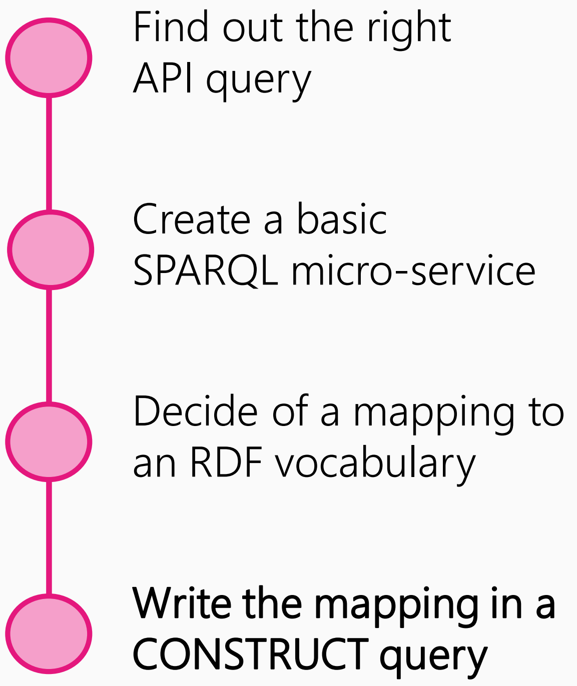
Thing > CreativeWork > MusicPlaylist > MusicAlbum

Thing > Organization > PerformingGroup > MusicGroup

Thing > CreativeWork > MusicRecording

…

My first SPARQL micro-service



Find out the right API query

Create a basic SPARQL micro-service

Decide of a mapping to an RDF vocabulary

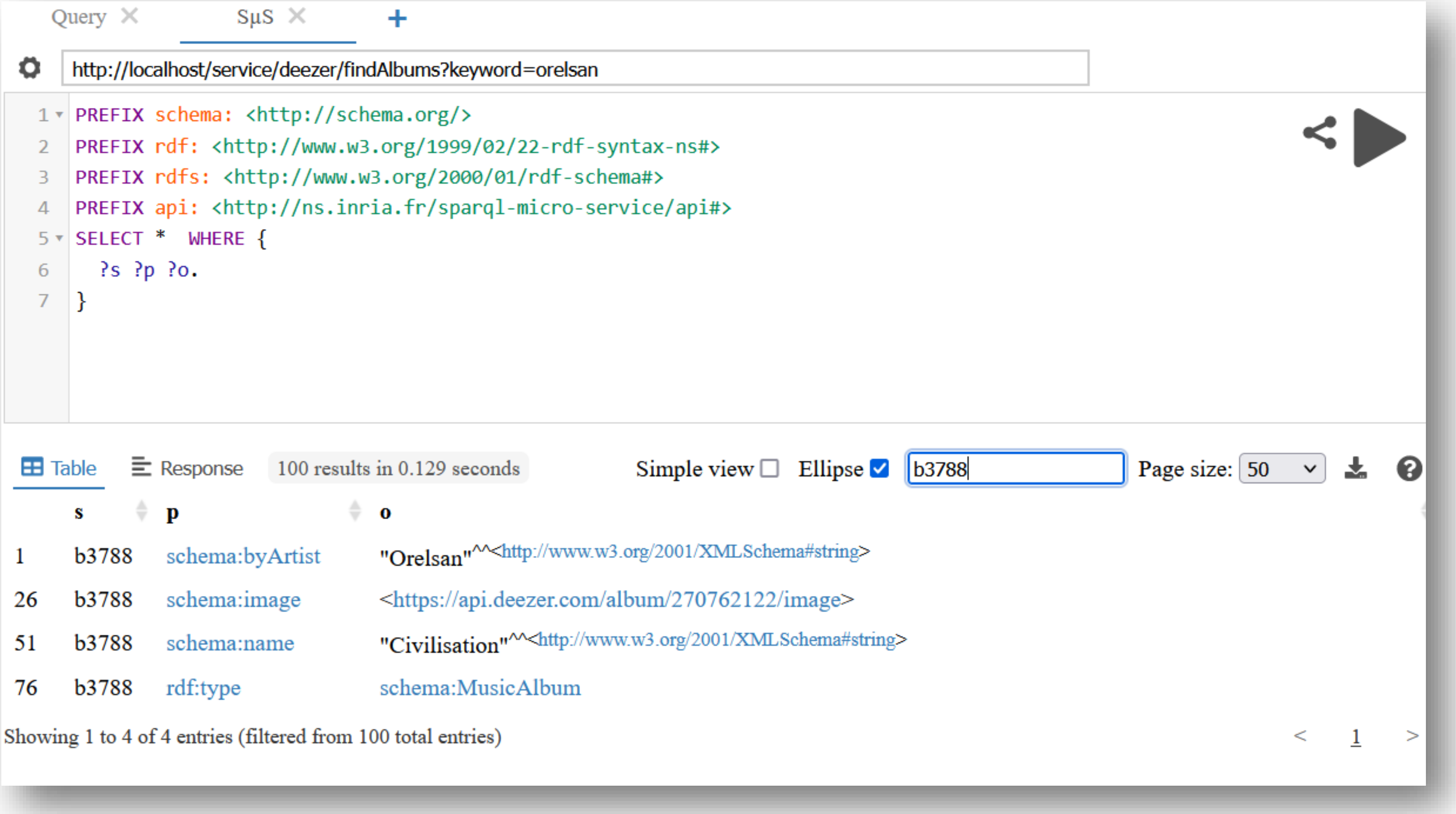
Write the mapping in a CONSTRUCT query

```
PREFIX schema: <http://schema.org/>
CONSTRUCT {
   []
     a                 schema:MusicAlbum;
     schema:name       ?albumTitle;
     schema:image      ?imageUri;
     schema:byArtist   ?artistName
}
WHERE {
  ?album
     api:title  ?albumTitle;
     api:cover  ?image;
     api:artist [ api:name ?artistName ].

     bind(iri(?image) as ?imageUri)
}
```

My first SPARQL micro-service

Query the SPARQL micro-service with Yasgui



http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX api: <http://ns.inria.fr/sparql-micro-service/api#>
SELECT *  WHERE {
  ?s ?p ?o.
}
```

100 results in 0.129 seconds

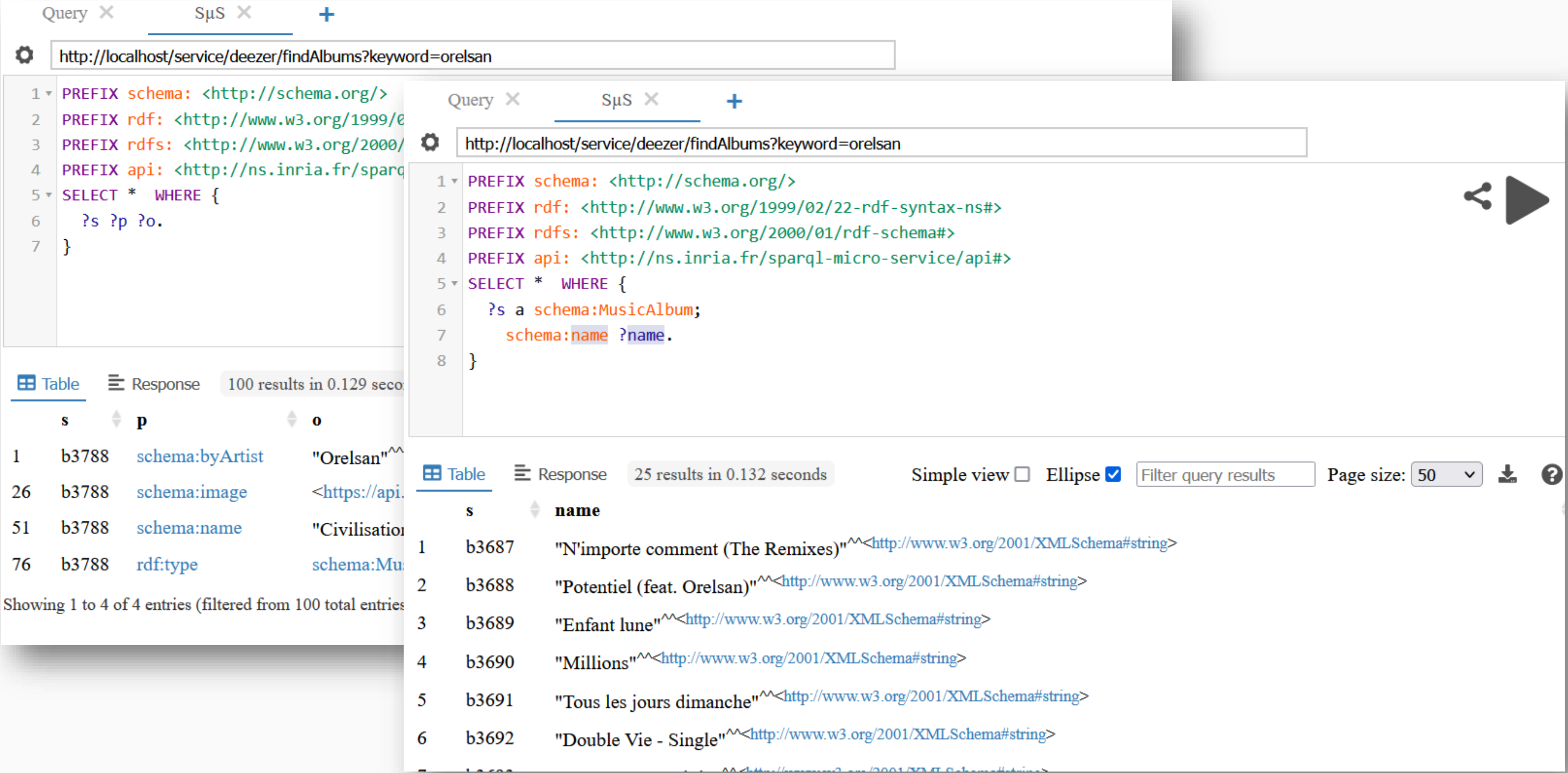
Simple view ☐ Ellipse ☑ b3788 Page size: 50

	s	p	o
1	b3788	schema:byArtist	"Orelsan"^^<http://www.w3.org/2001/XMLSchema#string>
26	b3788	schema:image	<https://api.deezer.com/album/270762122/image>
51	b3788	schema:name	"Civilisation"^^<http://www.w3.org/2001/XMLSchema#string>
76	b3788	rdf:type	schema:MusicAlbum

Showing 1 to 4 of 4 entries (filtered from 100 total entries)

My first SPARQL micro-service

Query the SPARQL micro-service with Yasgui



http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/0
PREFIX rdfs: <http://www.w3.org/2000/
PREFIX api: <http://ns.inria.fr/sparq
SELECT *  WHERE {
  ?s ?p ?o.
}
```

100 results in 0.129 seco

	s	p	o
1	b3788	schema:byArtist	"Orelsan"^^
26	b3788	schema:image	<https://api.
51	b3788	schema:name	"Civilisatio
76	b3788	rdf:type	schema:Mu

Showing 1 to 4 of 4 entries (filtered from 100 total entries

http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX api: <http://ns.inria.fr/sparql-micro-service/api#>
SELECT *  WHERE {
  ?s a schema:MusicAlbum;
    schema:name ?name.
}
```

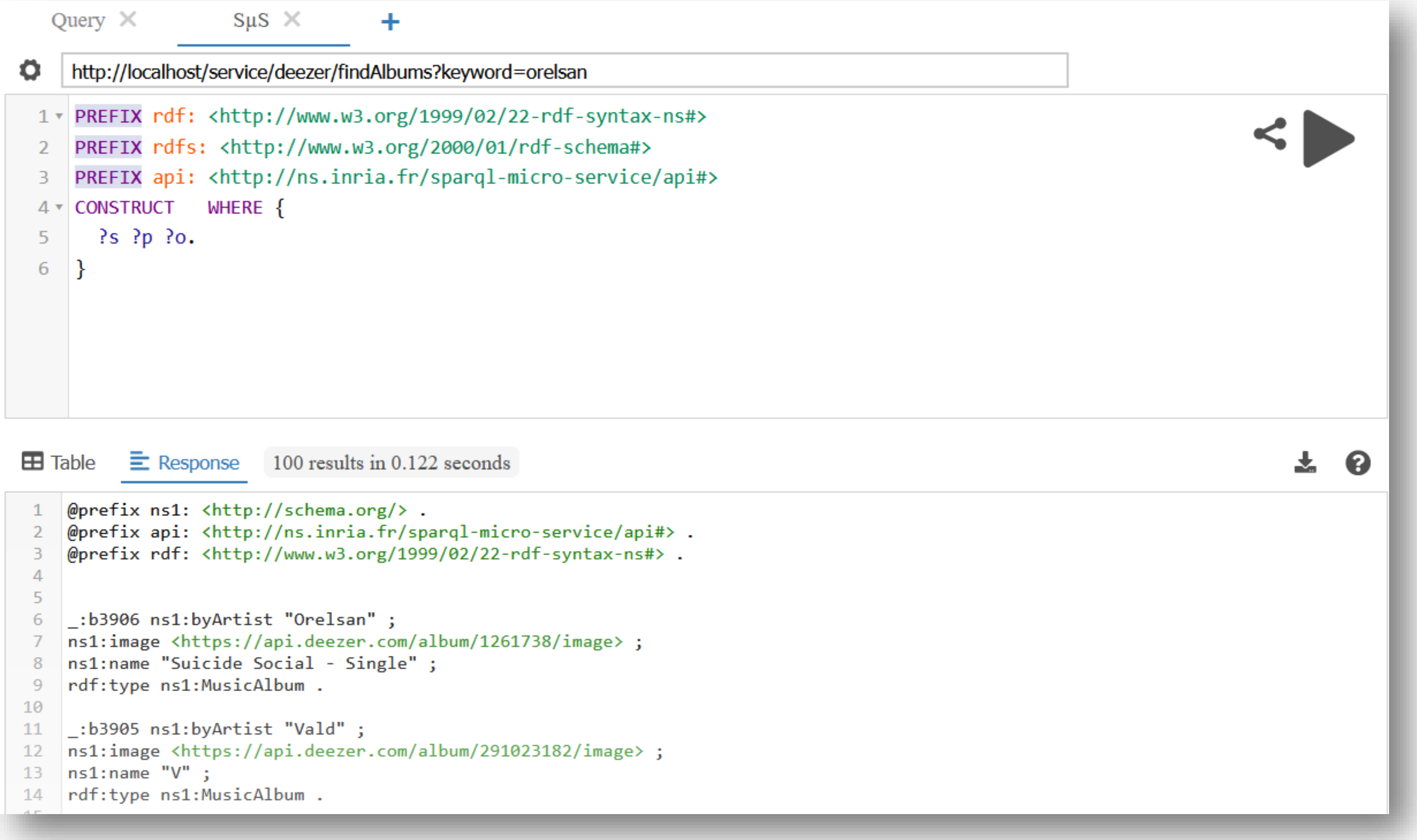
25 results in 0.132 seconds

Simple view ☐ Ellipse ☑ Filter query results Page size: 50

	s	name
1	b3687	"N'importe comment (The Remixes)"^^<http://www.w3.org/2001/XMLSchema#string>
2	b3688	"Potentiel (feat. Orelsan)"^^<http://www.w3.org/2001/XMLSchema#string>
3	b3689	"Enfant lune"^^<http://www.w3.org/2001/XMLSchema#string>
4	b3690	"Millions"^^<http://www.w3.org/2001/XMLSchema#string>
5	b3691	"Tous les jours dimanche"^^<http://www.w3.org/2001/XMLSchema#string>
6	b3692	"Double Vie - Single"^^<http://www.w3.org/2001/XMLSchema#string>

My first SPARQL micro-service

Query the SPARQL micro-service with Yasgui



http://localhost/service/deezer/findAlbums?keyword=orelsan

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX api: <http://ns.inria.fr/sparql-micro-service/api#>
CONSTRUCT   WHERE {
  ?s ?p ?o.
}
```

100 results in 0.122 seconds

```
@prefix ns1: <http://schema.org/> .
@prefix api: <http://ns.inria.fr/sparql-micro-service/api#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


_:b3906 ns1:byArtist "Orelsan" ;
ns1:image <https://api.deezer.com/album/1261738/image> ;
ns1:name "Suicide Social - Single" ;
rdf:type ns1:MusicAlbum .

_:b3905 ns1:byArtist "Vald" ;
ns1:image <https://api.deezer.com/album/291023182/image> ;
ns1:name "V" ;
rdf:type ns1:MusicAlbum .
```

Suggested APIs

Deezer	https://developers.deezer.com/api
MusicBrainz	https://musicbrainz.org/doc/Development/JSON_Web_Service (Example: beta.musicbrainz.org/ws/2/artist/?fmt=json&query=name:eminem)
SoundCloud	https://developers.soundcloud.com/docs/api/explorer/open-api
Flickr	https://www.flickr.com/services/api/
IMDB	https://imdb-api.com/
Youtube	https://developers.google.com/youtube/v3/getting-started https://developers.google.com/youtube/v3/docs/
Twitter	https://developer.twitter.com/en/docs/api-reference-index https://developer.twitter.com/en/docs/basics/authentication/oauth-2-0/application-only
Facebook	Create an application: https://developers.facebook.com/apps/ API documentation: https://developers.facebook.com/docs/graph-api Test: https://developers.facebook.com/tools/explorer/
Instagram	https://developers.facebook.com/docs/instagram-api Only for companies (!!!!!!!)
Spotify	https://developer.spotify.com/documentation/web-api/

🟩 Easy (some services require) no key

🟦 Easy, need one developer key

🟧 Developer key + long lasting token

🟥 Developer key + token

⬜ Not evaluated

Suggested APIs

LinkedIn	https://docs.microsoft.com/fr-fr/linkedin/
Snapchat	https://developers.snapchat.com/api/docs/ https://docs.snapchat.com/docs/api/web/
Lignes d'Azur	https://data.lignesdazur.com/dataset
PokéAPI	https://pokeapi.co/docs/v2

Need inspiration ?
https://rapidapi.com/
https://api.gouv.fr/

🟩	Easy (some services require) no key
🟦	Easy, need one developer key
🟧	Developer key + long lasting token
🟥	Developer key + token
⬜	Not evaluated

Environment

With Docker

Instructions	https://github.com/frmichel/sparql-micro-service/tree/master/deployment/docker
SPARQL µ-services URL	http://localhost/service/<api>/<service>?...
YASGUI	file:///home/user/yasgui.html

With VirtualBox and a pre-installed virtual machine

Virtual Machine	https://sms.i3s.unice.fr/~smshandson/material/sms-hands-on-polytech.ova
Login	user / user
Path for services	/home/user/public_html/services
SPARQL µ-services URL	http://localhost/sparql-ms/<api>/<service>?...
YASGUI	file:///home/user/yasgui.html