

Web sémantique et web des données

Projet

Dépot git : [GitHub](#)

Collecte des données

Pour ce projet, on choisit d'utiliser des données concernant des albums musicaux. Pour cela, on utilise le site [Rate Your Music](#), qui répertorie les sorties d'albums et permet à des utilisateurs de déposer des *reviews*, noter des albums et gérer leur collection. Le site maintient aussi des classement des meilleurs albums pour des périodes données, calculé en utilisant une formule prenant en compte la moyenne des notes données par les utilisateurs, et le nombre de notes et de reviews.

Malheureusement, *RYM* ne permet pas d'exporter directement des données issues du site en format tabulaire. On utilise donc un *scraper* Python ([GitHub - dbeley/rymscraper: Python API to extract data from rateyourmusic.com.](#)) pour extraire les données issues des 60 premières pages du site, i.e. les 2400 meilleurs albums répertoriés. Pour chaque individu, on obtient 8 variables :

- Le rang dans son classement général, entier de 1 à 2400
- L'artiste
- Le nom de l'album
- La date de sortie (pour certains individus, seulement l'année, pour d'autre la date précise)
- Le(s) genre(s) de l'album
- La note moyenne de l'album
- Le nombre de note
- Le nombre de *reviews*

Modélisation en RDF(S)

En utilisant OpenRefine, on transforme les données pour les rendre plus utilisables :

1. On modifie les genres, pour les séparer en utilisant un point-virgule.
2. On unifie le format de date en conservant uniquement l'année de sortie (et en supprimant jour et mois lorsque présent).
3. On transforme le type des attributs pour correspondre aux données : suppression des virgules dans les chiffres puis conversion vers type numérique.

4. On sépare les artistes, pour les albums qui sont des collaborations entre plusieurs artistes : il n'existe pas de convention claire sur ces données pour le séparateur entre différents artistes. Parfois le '/' ou ',' sont utilisés, des fois 'featuring' ou 'with', et d'autre '&'. On doit donc corriger ces données manuellement, et de la même manière que les genres, on utilise le point-virgule comme séparateur.

Pour le schéma RDFS, on crée 3 classes : **:Album**, **:Artist** et **:Review**. **Album** est lié à **Review** par une relation **:hasReview**, et à **Artist** par **:wasMadeBy**.

- **Album** a aussi les propriétés **:wasReleasedIn** et **:hasGenre**, qui ont pour range **xds:int** et **xds:string** respectivement.
- **Review** a les propriétés **:hasNumberOfRatings**, **:hasNumberOfReviews**, **:isRank**, de range **xsd:int** et **:isRated** de range **xds:decimal**.

De plus, pour **Album** et **Artist**, on renseigne la propriété standard `rdfs:label` afin de fournir les noms.

Après avoir exporté ce schéma, on utilise des RegEx sur le fichier turtle pour scinder les chaînes de caractères non-atomiques, par exemple pour les genres :

hasGenre : "Post Punk; Punk Rock" devient hasGenre: "Post Punk", "Punk Rock"

On définit le vocabulaire en turtle :

```

7
8 :Album a rdfs:Class.
9 :Review a rdfs:Class.
10 :Artist a rdfs:Class.
11
12 :hasGenre a rdfs:Property;
13     rdfs:domain :Album.
14
15 :hasReview a rdfs:Property;
16     rdfs:domain :Album;
17     rdfs:range :Review.
18
19 :wasMadeBy a rdfs:Property;
20     rdfs:domain :Album;
21     rdfs:range :Artist.
22
23 :wasReleasedIn a rdfs:Property;
24     rdfs:domain :Album.
25
26 :hasNumberOfRatings a rdfs:Property;
27     rdfs:domain :Review.
28
29 :hasNumberOfReviews a rdfs:Property;
30     rdfs:domain :Review.
31
32 :hasRank a rdfs:Property;
33     rdfs:domain :Review.
34
35 :isRated a rdfs:Property;
36     rdfs:domain :Review.
37

```

Fig 1 : RDFS dans le fichier Turtle.

On ne modifie pas le préfixe de notre ontologie.

Point d'accès SPARQL

Comme préconisé par le sujet, on utilise simplement Apache Jena Fuseki en tant que serveur pour mettre en place un point d'accès SPARQL.

On utilise "rym_onto" comme nom de dataset, ce qui a une incidence sur l'URL du point d'accès.

Pour *fetch* le point d'accès SPARQL depuis l'application JS, on utilise le code de [d3-sparql](#), adapté pour les besoins de l'application.

Elaboration de requêtes SPARQL et accès à des données liées

Dans le cadre de notre application, on utilise 3 requêtes SPARQL, dont 2 paramétriques :

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://127.0.0.1:3333/>
```

```
SELECT ?year WHERE {
  ?sub rdf:type :Album ;
      :wasReleasedIn ?year.
}
```

Requête pour obtenir les années de sortie des albums.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://127.0.0.1:3333/>

SELECT ?label ?rank (GROUP_CONCAT(DISTINCT ?label_artist; SEPARATOR=';') AS ?label_artists)
(SAMPLE(?genre) AS ?genres) ?year WHERE{
  ?album a :Album;
  rdfs:label ?label;
  :hasReview ?review;
  :hasGenre ?genre;
  :wasReleasedIn ?year;
  :wasMadeBy ?artist.
  ?artist rdfs:label ?label_artist.
  ?review :hasRank ?rank.
  FILTER( ${filter ? "false" : "true"} || ?year >= ${x0} && ?year <= ${x1})
}
GROUP BY ?label ?rank ?year ORDER BY ASC(?rank)
```

Requête pour obtenir les informations générales des albums de notre base de données. On inclut également un filtre paramétrique servant à filtrer les albums par année de sortie.

```

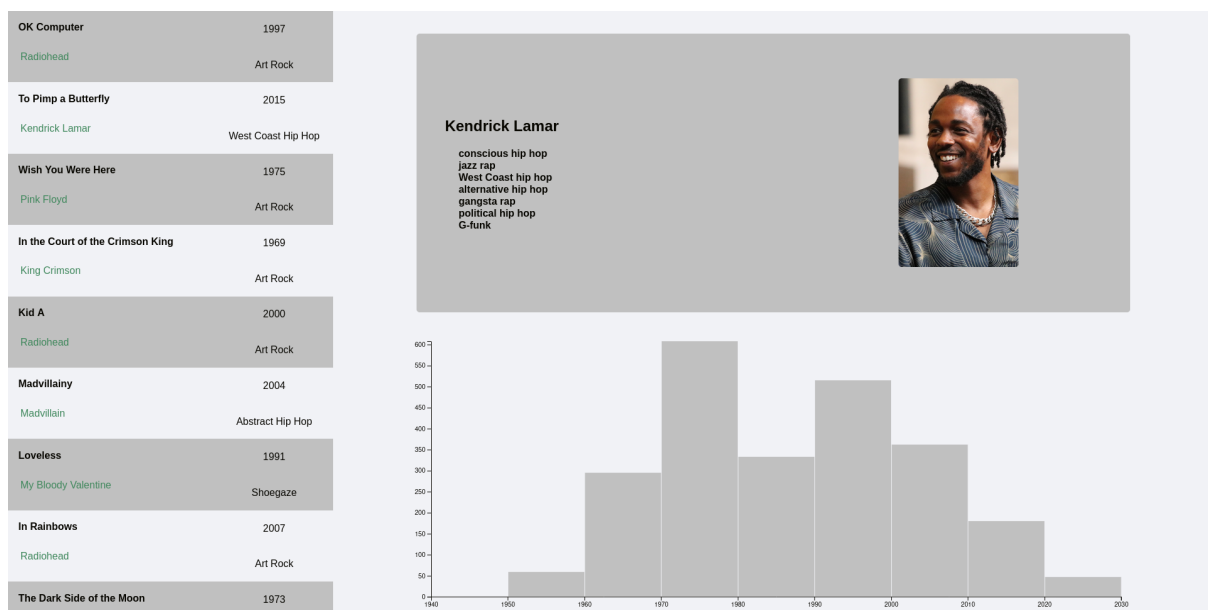
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX : <http://127.0.0.1:3333/>

SELECT ?image (GROUP_CONCAT(DISTINCT ?genre; SEPARATOR='<br>') AS ?genres) WHERE {
  ?album a :Album;
  rdfs:label "${album}"@en;
  :wasMadeBy ?artist.
  ?artist rdfs:label "${artist}"@en.
  SERVICE <https://query.wikidata.org/sparql> {
    ?sub wdt:P31 wd:Q482994;
    wdt:P175 ?f_artist.
    ?f_artist rdfs:label|skos:altLabel "${artist}"@en;
    wdt:P18 ?image.
    OPTIONAL {
      ?f_artist wdt:P136 ?f_genre.
      ?f_genre rdfs:label ?genre.
      FILTER(lang(?genre) = 'en').}
    }
} GROUP BY ?image`

```

Requête fédérée s'appuyant sur Wikidata pour récupérer des informations sur un artiste/groupe, plus précisément une image dépliant l'artiste et les genres musicaux auxquels il/elle est associée. On peut noter l'utilisation de la propriété *skos:altLabel* avec une construction RegEx permettant de récupérer un artiste grâce à un alias. On utilise également le mot clé **OPTIONAL** pour pouvoir récupérer l'image lié à un artiste/groupe dans le cas où les genres associés ne sont pas spécifiés.

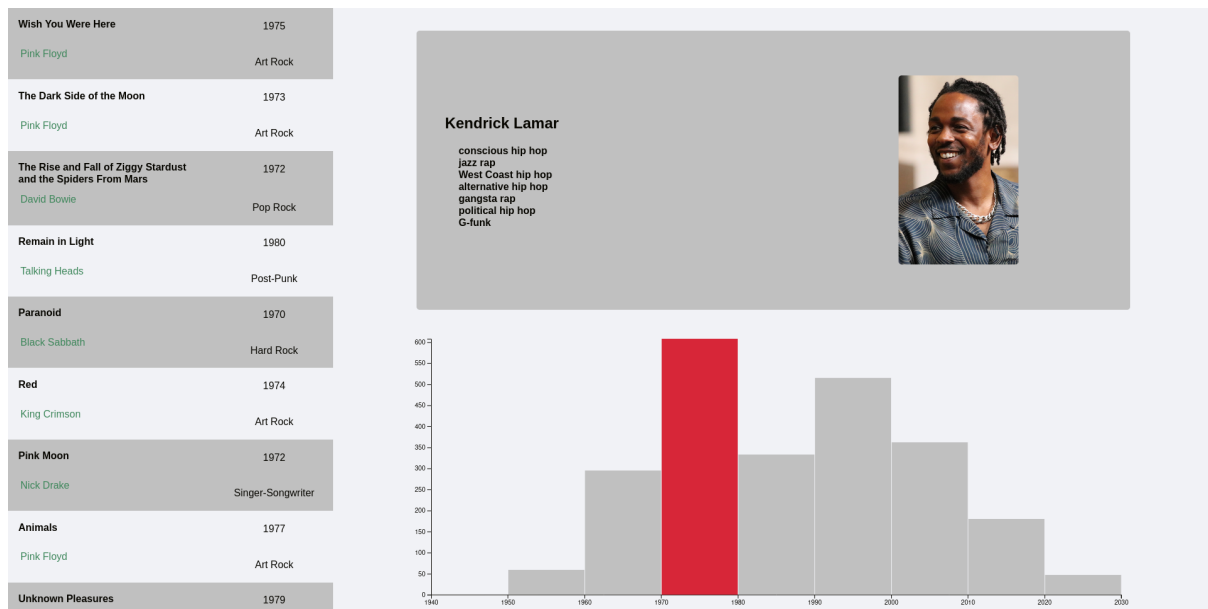
Application Web



Page d'accueil de l'application Web réalisée

L'application se divise en trois parties :

- En bas à droite, un histogramme permet de visualiser le nombre d'album par décennie présent dans notre dataset. Ce diagramme, réalisé avec d3.js, est interactif : il permet la sélection d'une décennie particulière via un clic sur la barre associée (et la dé-sélection à l'aide d'un second clic)
- À droite, une liste permet d'énumérer les albums présents dans le dataset, ainsi que les informations associées : titre, année de sortie, genre et artistes/groupes. Cette liste évolue si une décennie est sélectionnée via l'histogramme, pour n'afficher que les albums issue de la décennie en question. De plus, les noms d'artistes/groupes sont cliquables.
- Lorsqu'un nom d'artiste/groupe est cliqué, la vue en haut à droite affiche des informations collectées sur Wikidata qui lui sont liées : si un lien avec une ressource Wikidata a pu être effectué, on affiche une image décrivant l'artiste/groupe ainsi que la liste des genres qui lui sont associés.




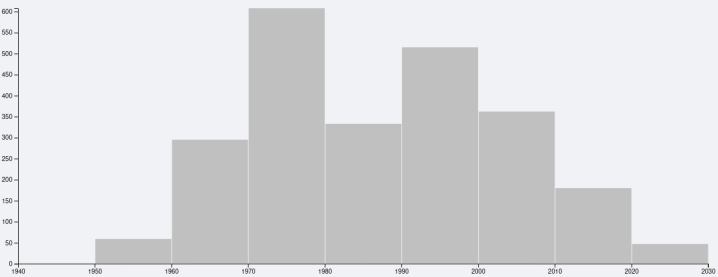
Les années 70 sont sélectionnées via le diagramme : la liste de gauche évolue.

Future Days	1973
Can	Krautrock
Midnight Marauders	1993
A Tribe Called Quest	East Coast Hip Hop
Deathconsciousness	2008
Have a Nice Life	Post-Punk
Black Sabbath	1970
Black Sabbath	Hard Rock
Rubber Soul	1965
The Beatles	Pop Rock
Ágætis byrjun	1999
Sigur Rós	Post-Rock
Carrie & Lowell	2015
Sufjan Stevens	Singer-Songwriter
Tago Mago	1971
Can	Experimental Rock
Innervisions	1973

Sufjan Stevens

western classical music
 electronic music
 baroque pop
 indie rock
 chamber pop
 alternative rock
 indie pop
 indie folk
 lo-fi music
 folk-pop



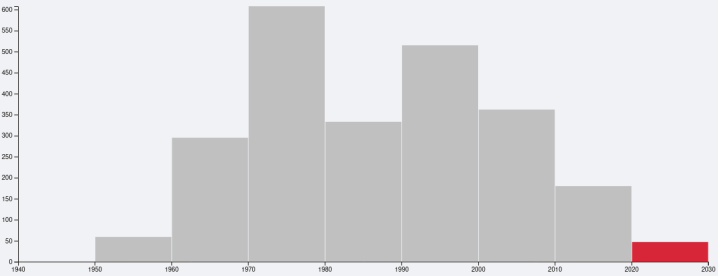


On accède aux informations d'un autre artiste : une image et les genres sont disponibles.

Ants From Up There	2022
Black Country, New Road	Art Rock
Sometimes I Might Be Introvert	2021
Little Simz	Conscious Hip Hop
LP!	2021
JPEGMAFIA	Hardcore Hip Hop
Hellfire	2022
black midi	Brutal Prog
Dragon New Warm Mountain I Believe in You	2022
Big Thief	Folk Rock
Cheat Codes	2022
Danger Mouse	East Coast Hip Hop
By the Time I Get to Phoenix	2021
Injury Reserve	Experimental Hip Hop
アダンの風 (Windswept Adan)	2020
Ichiko Aoba [青葉市子]	Singer-Songwriter
Microphones in 2020	2020

JPEGMAFIA





Pour cet artiste, seulement une image est disponible.

Instructions

Pour déployer l'application web, il suffit simplement de charger les données *data.ttl* sur Apache Jena Fuseki, en utilisant "**rym_onto**" comme nom de projet.

On peut ensuite utiliser le fichier *index.html* localement.