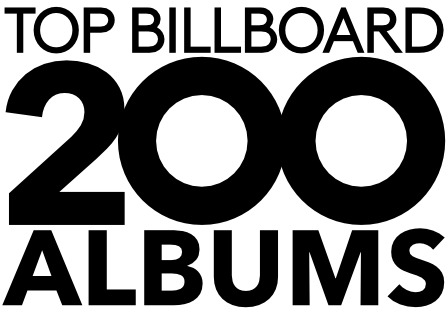
Lamy-Verdin TiphanieLim AnnieVichaikit Jean-Philippe

Documentation Technique



DSIA-4201C Data Engineering

Courivaud R. 2019-2020

Sommaire

[Présentation 3](#_Toc33315579)

[Docker 3](#_Toc33315580)

[Scrapy 3](#_Toc33315581)

[Mongo 4](#_Toc33315582)

[Flask 4](#_Toc33315583)

[ElasticSearch 4](#_Toc33315584)

# Présentation

Le projet à pour but d’afficher de manière actualisée le classement musical des meilleurs 200 albums de la semaine. Les données sont directement explorées et récupérées du site officiel [Billboard](https://www.billboard.com/charts/billboard-200), pour ensuite y traiter les informations pertinentes pour les réafficher à l’aide de graphiques. Il est également possible d’effectuer des recherches sur les albums ou artistes présents parmi ce top 200, s’afficheront alors plus de détails tels que leur positionnement de la semaine précédente, leur plus haut classement et durée à cette position.

# Docker

Afin de lancer notre application dans des conteneurs logiciels, nous avons utilisé Docker qui nous a notamment été utile pour monter une instance MongoDB qui sera accessible depuis internet. De même nous avons utilisé Docker pour conteneuriser ElasticSearch pour stocker et effectuer des requêtes sur nos données.

Un fichier docker-compose permet de générer nos images MongoDB et ElasticSearch.

# Scrapy

Nous utilisons le framework Scrapy qui permet l’exploration de sites web et y extrait des données de façon structurée. Ce framework de Spider peut gérer le Crawling et le Scraping de site web très efficacement.

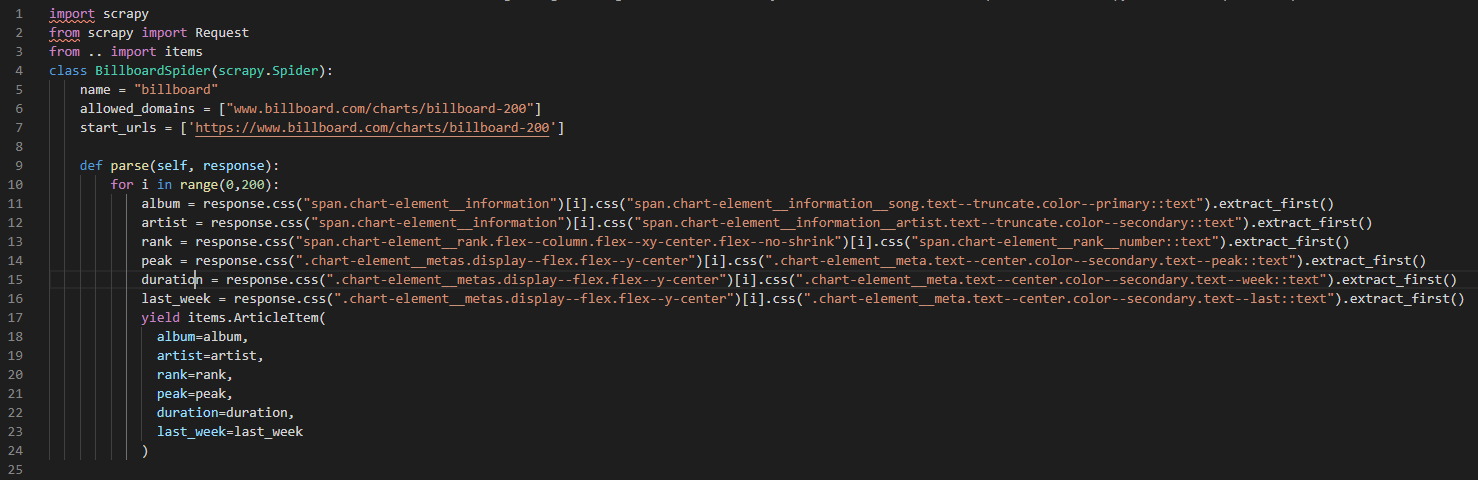
Spider

Notre spider *billboard.py* récupère directement les données de classement musical de la semaine depuis le site Billboard.

Il est possible de lancer la spider à l’aide de la commande :

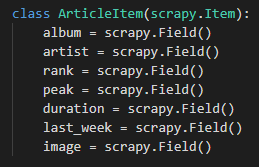
*scrapy crawl billboard*

La méthode *parse()* retourne un dictionnaire correspondant à chaque album présent dans le classement, contenant leurs informations respectifs : nom de l’album, artiste, rang, plus haut rang atteint, durée à ce haut rang, position de la semaine précédente.



Item

Nous structurons nos données en les récupérant sous la forme d’un modèle, défini dans la classe item. Chaque item, correspondant à chaque album du classement, défini dans le fichier *items.py* possède : nom de l’album, artiste, rang, plus haut rang atteint, durée à ce haut rang, position de la semaine précédente.

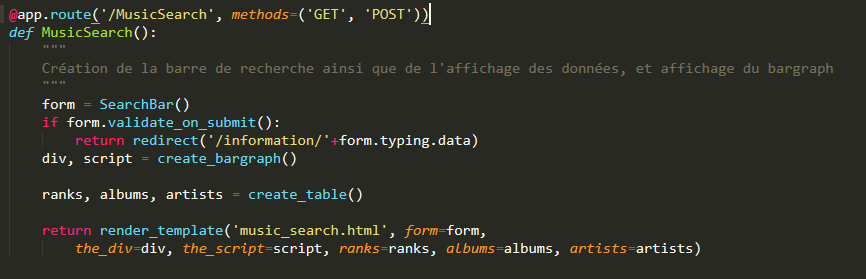


# MongoDB

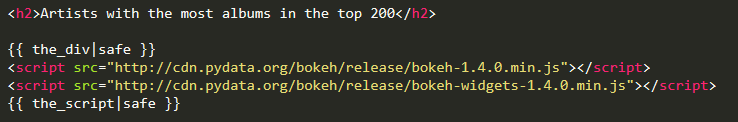
MongoDB est une base de données noSQL avec une structure très flexible et optimisée, adaptée au contexte de notre projet. Elle stocke les données sous la forme de document JSON et a pour avantage l’optimisation de la mémoire.

# Flask

Flask est un framework open-source de développement web nous permettant un affichage des données simple.



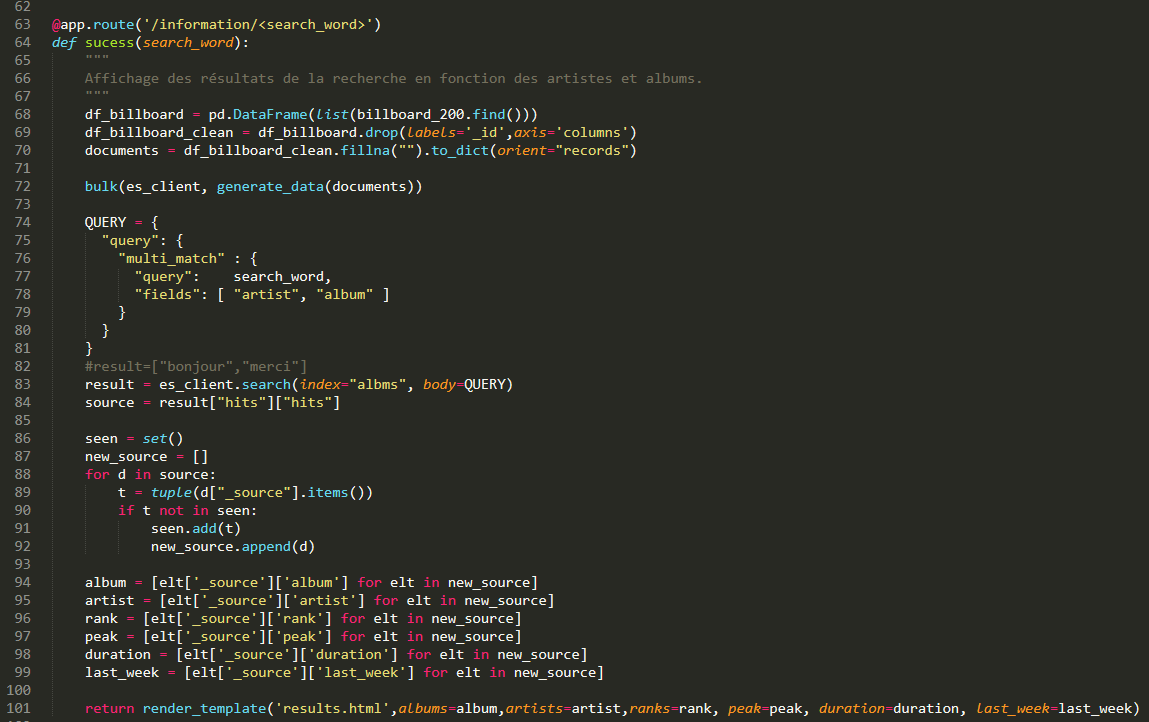
Le fichier *app.*py est utilisé pour l’affichage web. La fonction *MusicSearch* s’occupe de la génération d’une barre de recherche, d’un tableau ainsi que d’un bargraph, qui est envoyé au fichier  *music\_search.html*  par le biaies des variables dans la fonction *render\_template()*.



Dans le fichier *music\_search.html*, on récupère les données du backend vers le frontend grace à la synthaxe *{{ nom\_variable }}*. Il s’agit ici du bargraph que l’on récupère.

# ElasticSearch

ElasticSearch est un moteur de recherche permettant une recherche très efficace de données textuelles, numériques et géolocalisées.



Lorsque la recherche aboutit, les résult ats sont stockées dans la variable *« source »*. On traite ensuite les données pour supprimer les duplications, et on affiche le tout dans la fonction *render\_template()*.