Master Science et technologie du logiciel

Rapport Projet DAAR

Indexation de CVs dans Elasticsearch

Réalisé par:
Azzoug Koceila
Sofiane Braneci

2021/2022

Table des matières

1 Introduction

L'informatique moderne et l'internet ont rendu accessible une grande quantité d'informations. La capacité à rechercher efficacement ces informations est fondamentale pour les traiter. À l'époque où nous vivons, la quantité de données générées chaque jour est exponentielle, et le temps nécessaire pour retrouver l'information pertinente est proportionnel à la quantité des données dans lesquelles nous effectuons la recherche. Toutes les applications modernes doivent être conçues dans un souci de performance afin de répondre aux besoins des usagers, et la recherche est un processus clé pour répondre à chaque requête. Presque toutes ces applications utilisent une sorte de solution de gestion et de stockage des données, et la plupart d'entre elles s'appuient sur des systèmes de bases de données comme MySql, MongoDB. En revanche, ces dernières années, Elastisearch a trouvé sa place dans les plus grandes entreprises mondiales comme Facebook, Netflix, Deezer, Microsoft ...etc. Cela est due a ses performances en recherche et analyse de données car il opère en quasi temps réel et sur de grands volumes de données, elle est devenu la solution de recherche la plus populaire. Les indexs sont utilisés pour accélérer la localisation des données. Dans ce projet nous utiliserons Elasticsearch comme solution pour indexer des CV, Elasticsearch est un moteur de recherche open source hautement évolutif. Bien qu'il ait commencé comme un moteur de recherche de texte, il est en train d'évoluer comme un moteur analytique, qui peut supporter non seulement la recherche mais aussi des agrégations complexes.

2 Plan de développement

Le but du projet étant d'indexer des CVs sur Elastic search dans le but de faire des recherches efficaces et optimisées. La démarche entrepris consiste a :

— Indexation de documents

1. Parsing des documents en string et base64
2. Stockage des documents sous forme d'un JSON contenant

```
{
"filename": "filename-UUID.pdf",
"textContent": "file as String",
"content": "file as Base64 string"
}
```

— Recherche : recherche de tous les CVs contenant une compétence

3 Indexation des documents

3.1 Parsing

Le besoin de cette partie était d'indexer un CV d'une manière a faire une recherche efficace et de garder le document de manière a pouvoir l'instaurer au moment

voulu (modèle et contenu).
La classe *PDFParser* s'occupe de tous le parsing.

— parse(byte [] bytes) : ici, on utilise la bibliothèque **PDFBox** qui charge le fichier initialement sous forme d'un tableau de Bytes puis invoque **PDFTextStripper** qui extrait le contenu en string.

— encodePdfBase64(InputStream fileInputStream) : La requete d'ajout d'un document passe le CV sous forme de InputStream. Cette méthode transforme les bytes de ce stream en une chaine de caractères encodé en Base64.

— decode(String strEncoded,String saveRep) : cette méthode à pour but d'instaurer le contenue orgininal du document passée ici en paramètre strEncoded(encodé en base64)et le sauvegarde dans le répertoire de destination saveRep.

3.2 Stockage et indexation

Pour Pouvoir intéragir avec Elasticsearch, on a besoin d'un client RestHighLevelClient a travers le quel on peut emettre des requetes et recevoir des réponses.
La classe Config annoté par **@Configuration** contenant un **Bean** se charge de créer ce client et le rajoute au contexte d'exécution au lancement de l'application.
Les étapes ci-dessous résume les fonctionnalitées principales implémentées dans IndexHandler ;

1. Création d'index :
   La méthode **createIndex** qui prend en paramètre le nom souhaité pour l'index s'occupe de cette fonctionnalité, On utilise le client crée précedemment pour vérifier dans un premier temps l'existence de l'index. Si ce dernier n'existe pas, on crée une requete de type **CreateIndexRequest** ayant comme paramètres le nombre de shards ainsi que le nombre de replicas.
2. Ajout de document : Cette fontionnalité est assuré par la méthode handleUpload prenant en paramètres un objet de type MultiPartFile qui permet de récupérer le fichier en Bytes ou InputStream.
   (a) Parser le fichier en string en utilisant PDFParser
   (b) Encoder le fichier en Base64 en utilisant PDFParser
   (c) On construit le document a insérer contenant les trois attributs filename, textContent, content.
   (d) On crée la requete IndexRequest avec notre index et le document crée précédemment
   (e) le client se charge d'envoyer la réponse et de collecter la réponse **IndexResponse**
3. Recherche : La recherche s'effectue au niveau de la méthode seach prenant en paramètres un mot clé 'skill'.
   (a) Construire une requête de type MatchQuery afin de rechercher tous les documents qui match skill dans leur attribut 'textContent'
   (b) On envoie la requete a travers notre client, on récupère la réponse sous forme de SearchResponse

(c) A partir de la réponse, on extrait les SearchHit(contenant les attributs dont on a besoin).

(d) On parcourt la réponse(qui correpond a un tableau) pour construire une liste de CV(CV : structure contenant le nom du fichier, l'encodage en base64).

(e) On décode chaque CV en enregistrant le résultat(fichier PDF) dans le répertoire Docs et on renvoie au client les noms des CVs résultants.

4 API

Notre API est implémenté par CVController, elle contient deux **Endpoints**, upload et seach/skill qui servent respectivement a ajouter un CV et chercher un mot clé parmi tous les documents indexés.

— "/upload" : prend en charge les requetes HTTP POST contenant le ficher à indexer, la méthode correspondante a ce mapping invoque la service handleUpload de indexHandler.

— "/search/skill : prend en charge les requetes HTTP GET invoquant le service search en lui passant 'skill' en paramètre, la réponse étant une liste de String correspondant au noms des fichiers résultants.

5 Quelque tests d'ajout et recherche

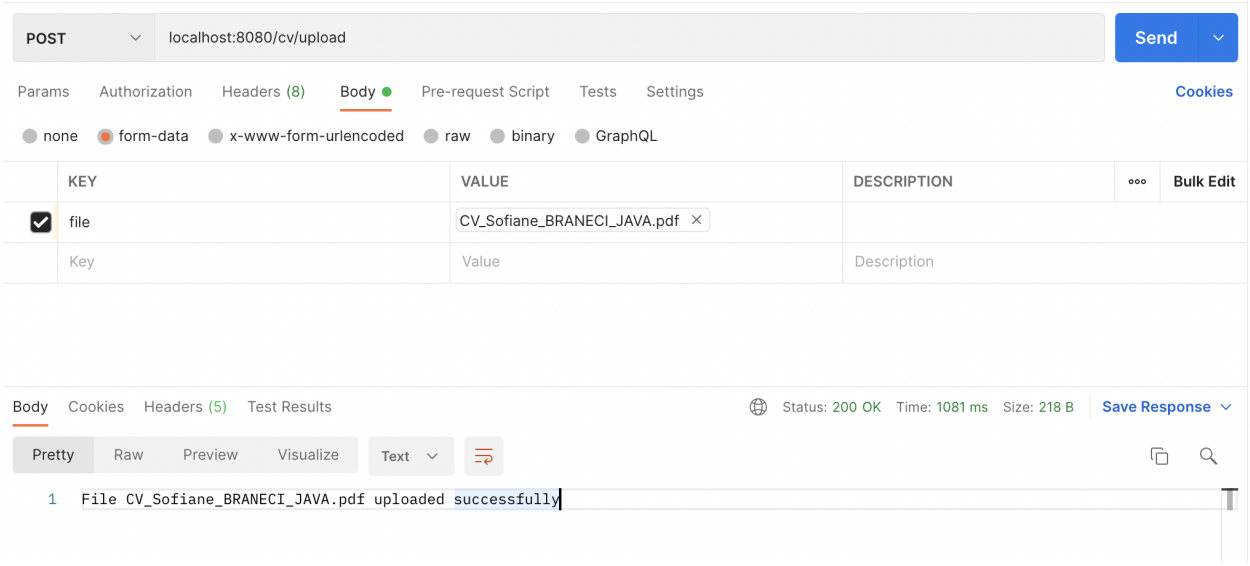

FIGURE 1 – Ajout d'un cv et réponse du serveur.

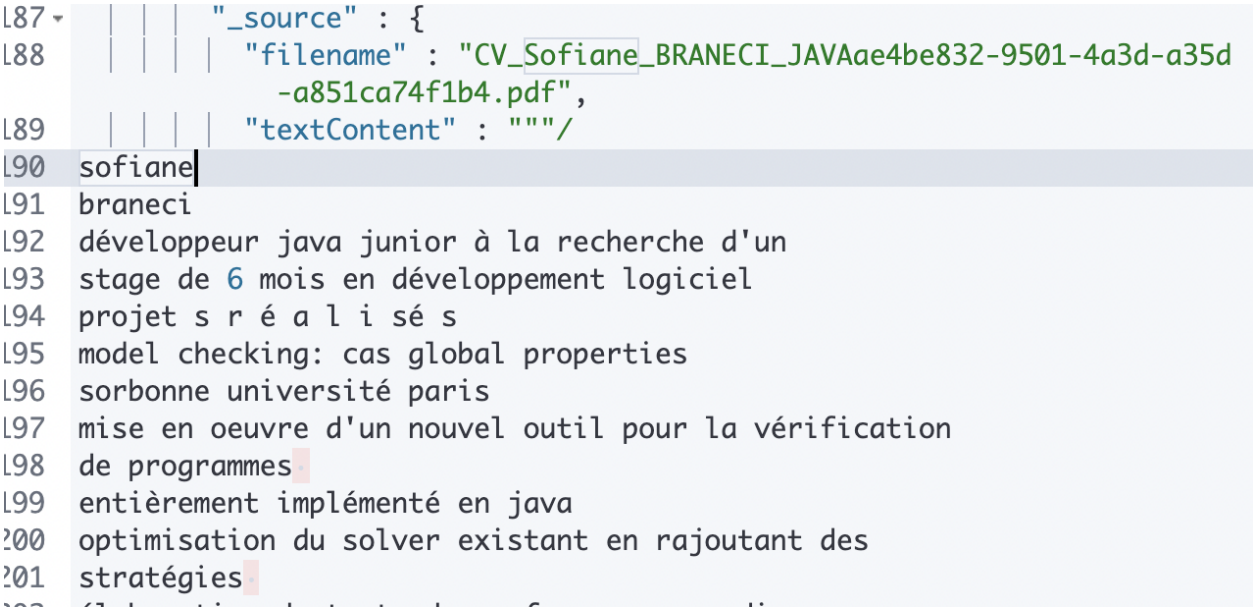

FIGURE 2 – Preuve d'ajout sur elasticsearch.

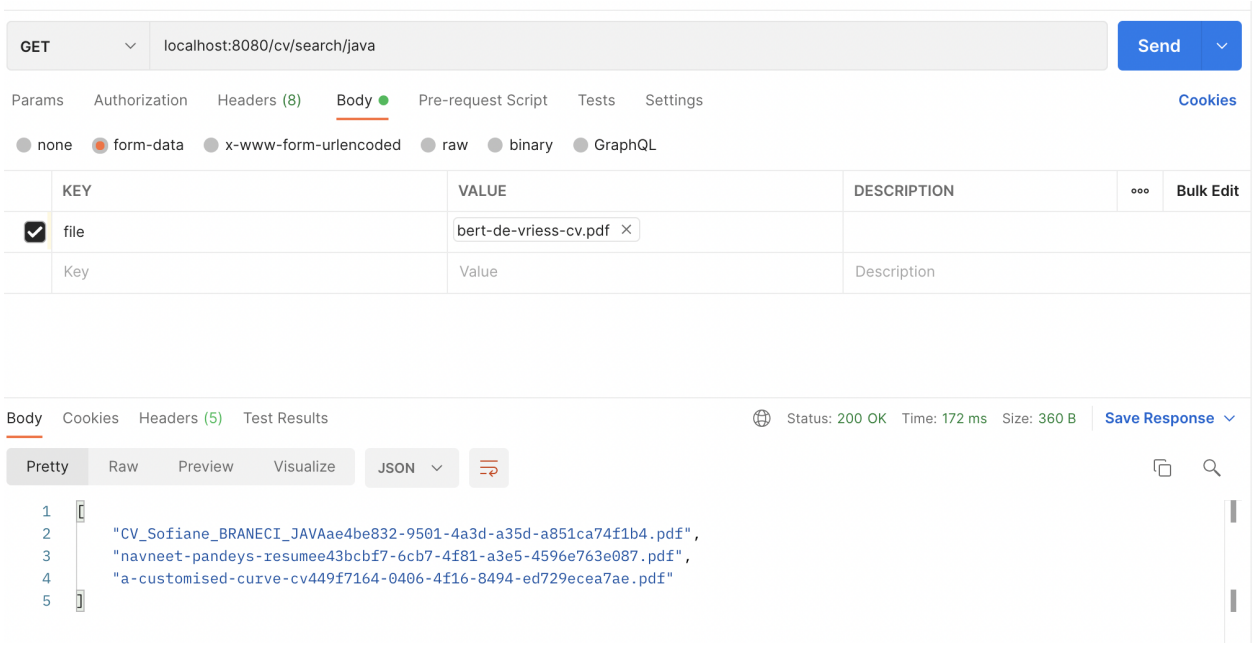

FIGURE 3 – Recherche CV contenant JAVA.

6 Conclusion

L'indexation des données est une pratique très efficace dans le domaine de recherche de l'information, elle nous permet entre autres de réduire notre espace de recherche, ce qui nous fait gagner énormément de temps,particulièrement quand on manipule une masse importante de données, c'est ce que nous propose aujourd'hui Elasticsearch, stocker, rechercher et analyser d'énormes volumes de données rapidement et en temps quasi réel pour retourner des réponses en quelques millisecondes.