Développement des Algorithmes d'Applications Réticulaires

Choix A : Moteur de recherche d'une bibliotheque

Authors :
Amine KHEDDAR
Sofiane BELKHIR

Professor :
BM. Bui-Xuan

April 17, 2022

Table des matières

1 Introduction

Les moteurs de recherche portent essentiellement sur des filtres pour la richesse des informations disponibles sur Internet. Ils permettent aux utilisateurs de trouver rapidement et facilement des informations d'intérêt ou de valeur réelles, sans avoir à parcourir de nombreuses pages Web non applicables. Il y a beaucoup de filtrage à faire - il y a trois ans, en 2004, le nombre de pages dans l'index de Google a dépassé le nombre de personnes sur la planète, atteignant un chiffre stupéfiant de plus de 8 milliards. Avec autant de contenu, Internet serait essentiellement impraticable sans moteurs de recherche, et les internautes se noieraient dans une mer d'informations non applicables et de messages marketing stridents.

Les moteurs de recherche fournissent aux utilisateurs des résultats de recherche qui conduisent à des informations pertinentes sur des sites Web de haute qualité. Le mot clé ici est « pertinent ». Pour atteindre et conserver des parts de marché dans les recherches en ligne, les moteurs de recherche doivent s'assurer qu'ils fournissent des résultats pertinents pour les recherches de leurs utilisateurs. Ils le font en maintenant des bases de données de pages Web, qu'ils développent en utilisant des programmes automatisés connus sous le nom de « araignées » ou « robots » pour collecter des informations. Les moteurs de recherche utilisent des algorithmes complexes pour évaluer les sites Web et les pages Web et leur attribuent un classement pour les expressions de recherche pertinentes. Ces algorithmes sont jalousement gardés et fréquemment mis à jour.

Dans ce projet, il nous est donc demandé de réaliser un moteur de recherche sur la base de livres de Gutenberg, étant un site contenant une grosse quantité de livres diffèrent. Ainsi, il s'agira d'utiliser des technologies et des algorithmes afin d'obtenir un moteur de recherche efficace en proposant des résultats pertinents (en proposant des résultats adaptés à chaque recherche) sur une grosse quantité de données.

2 Problématique

Une recherche directe sur une base de données volumineuses nécessite un temps considérable vu qu'on doit parcourir tous les livres un par un et dans chaque livre mot par mot, pour s'applique à ce problème, on doit utiliser le principe de fonctionnement des moteurs de recherche :

- Une indexation du contenu livres dans une base de données, est effectuée préalablement à la recherché.
- Ainsi, le moteur restitue, en fonction de ses paramètres une liste de pages correspondant aux mots recherchés.

3 Architecture logiciel

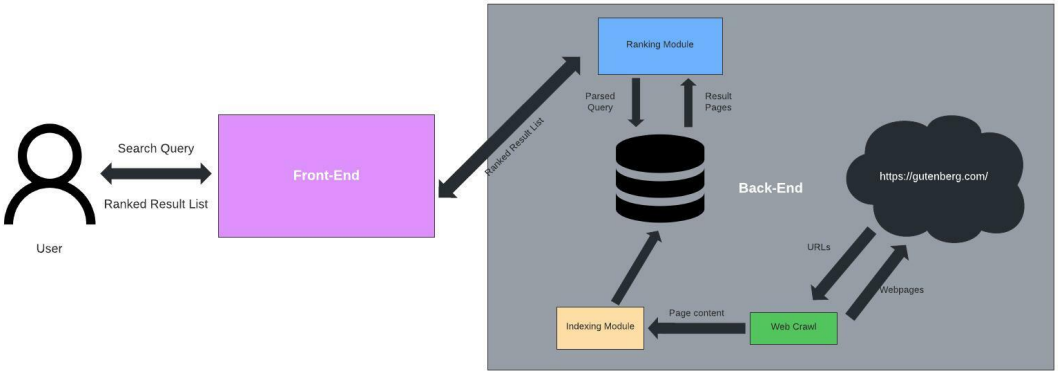

FIGURE 1 – Schéma illustrant global l'architecture de l'application

L'architecture utilisé dans la figure 3 est appelée RESTful est un style d'architecture permettant de construire des applications Web. Il s'agit de séparer le front et le backend dans deux composants différents, et les faire communiquer via des requêtes HTTP à travers des API développée au niveau du backend selon les services demandés. De plus, on a utilisé l'api du site https ://gutendex.com/pour récupérer l'ensemble des livres au moment de l'indexation.

4 Technologie utilisée

4.1 Python Django

En ce qui concerne le développement backend , Python fait partie des choix les plus populaires que font les entreprises, ainsi que les développeurs. Récemment, de plus en plus de personnes sont passées à Python. Après sa mise en oeuvre pour les plates-formes de produits par certains des plus grands noms comme Instagram, YouTube, Quora, Pinterest, Facebook, Reddit, Netflix, Google et Spotify, Python est devenu le nouveau favori.

4.2 Ionic - Angular

Devant une concurrence framework aussi intense, le besoin d'applications Web conviviales et interactives devient très crucial pour favoriser le succès de l'entreprise. Angular a servi de cadre idéal pour diriger l'industrie pour la production de solutions évolutives qui répondent aux besoins des gens. Angular est connu comme un framework Javascript open-source qui a été développé pour la première fois en 2009

par le développeur Google Misko Hevery. La structure basée sur CSS, HTML et JS afin de faciliter le développement front-end. Les applications Web Angular JS ont connu un tel succès qu'environ 8 400 sites Web prennent en charge ce cadre. Certaines de ces sociétés incluent également Upwork, PayPal et Netflix. Angular possède de nombreuses fonctionnalités qui permettent aux professionnels du développement d'applications Web de concevoir des sites Web avec un codage minimal. L'obligation d'écrire des setters et des getters dans différents modèles de données est supprimée. Moins de codage permet d'économiser beaucoup de temps et d'efforts.

Ainsi, la concaténation de ces deux outils va nous permettre de mettre en place l'architecture suivante :

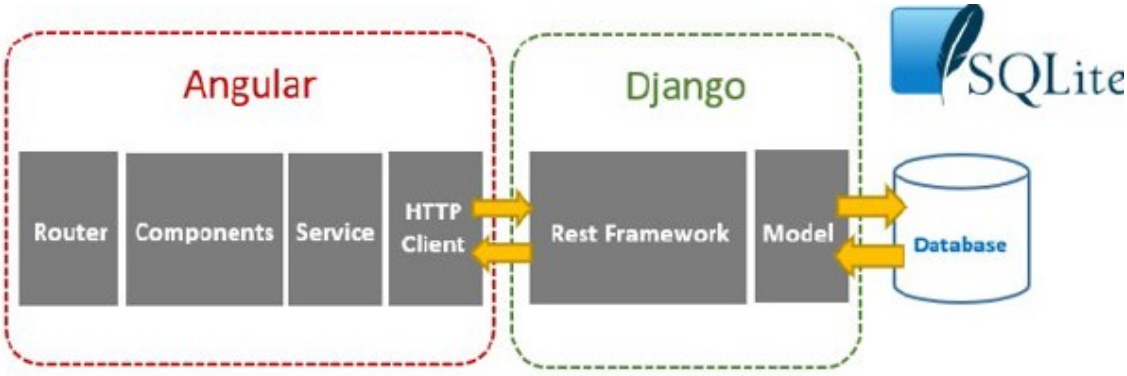

FIGURE 2 – Schéma illustrant l'architecture de l'application

- Django exporte les Apis REST à l'aide de Django Rest Framework et interagit avec la base de données SQLite à l'aide du modèle Django.
- Angular envoie des requêtes HTTP et récupère les réponses HTTP , affiche des données sur les composants.

5 Back-End

La partie backend représente le noyau du projet, cette partie implémente les différents algorithmes qu'on verra ci-dessous qui permettent d'implémenter le moteur de recherche proposer aux utilisateurs.

Les applications Web Django accèdent et gèrent les données via des objets Python appelés modèles. Les modèles définissent la structure des données stockées, y compris les types de champs et éventuellement leur taille maximale, les valeurs par défaut,

les options de liste de sélection, le texte d'aide pour la documentation, le texte d'étiquette pour les formulaires, etc.
La définition du modèle est indépendante de la base de données sous-jacente, il est possible d'en choisir un parmi plusieurs dans le cadre des paramètres du projet. Une fois avoir choisi la base de données à utiliser pour le projet, il suffira d'implémenter la structure du modèle et Django s'occupera de la communication avec la base de données.
Ici, la base de données que nous allons utiliser est une base **Sqlite**. Cette base devra contenir dans le futur un nombre total de **1664 livres**.

5.1 Structure de données

listOFBooks : qui définit une donnée indexer avec :

```
id = models.IntegerField(null=False, primary_key=True)
# id du livre dans la base du site gutenberg
author = models.CharField(max_length=150)
# auteur du livre
language = models.CharField(max_length=5, default="")
# la langue
title = models.CharField(max_length=255, blank=False)
# titre du livre
imageBook = models.CharField(max_length=355,default="")
# Image de la face avant du livre
text = models.TextField(default="")
# contenu du livre ou sommaire + titre
crank = models.FloatField(default="0.0")
# score du livre pour le classement
occurence = models.IntegerField(default="0")
# occurence d'un mot qui sera actualise durant une recherche
```

BookIndex : qui représente le résultat de l'algorithme décrit dans la sections ci-dessous les données sont stockés comme suit :

```
id = models.IntegerField(null=False, primary_key=True)
title = models.CharField(max_length=50000000, default="")
wordOcc = models.CharField(max_length=50000000, default="")
# nombre d'occurence des mots d'un livre
```

BookGraphJaccard : qui représente le résultat de l'algorithme de la distance de Jaccard avec les mots des livres indexer precedement :

```
id = models.IntegerField(null=False, primary_key=True)
neightbors = models.CharField(max_length=50000000, default="")
# graphe de jaccard traduit par un tableau regroupant tous les
                                    voisins d'un livre
```

Le diagramme suivant décrit le flux de données principal et les composants requis lors du traitement des requêtes et des réponses HTTP. Comme nous avons déjà implémenté le modèle, les principaux composants que nous allons créer sont :

— Mappeurs d'URL pour transférer les URL prises en charge (et toute information codéedans les URL) vers les fonctions d'affichage appropriées.

— Afficher les fonctions pour obtenir les données demandées des modèles, créer despages HTML qui affichent les données et renvoyer les pages à l'utilisateur pour qu'il lesaffiche dans le navigateur.

— Modèles à utiliser lors du rendu des données dans les vues ainsi que leurs créations.

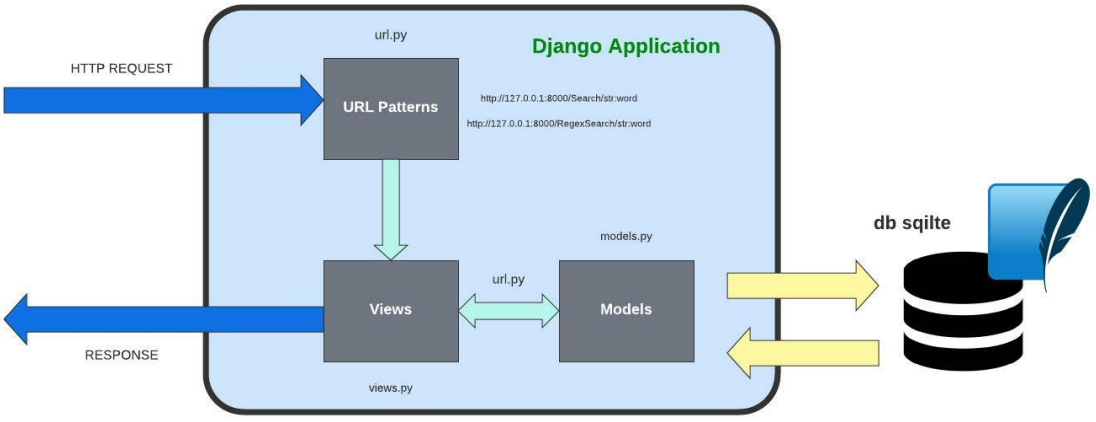

FIGURE 3 – Schéma illustrant l'architecture du Back-End

5.2 Indexation

L'indexation est un moyen d'optimiser les performances d'une base de données en minimisant le nombre d'accès au disque requis lors du traitement d'une requête. Il s'agit d'une technique de structure de données qui est utilisée pour localiser et accéder rapidement aux données d'une base de données.

Pour effectuer cette indexation nous avons decider d'utiliser la bibliothèque **stop word** afin que pour chaque livre, on récupère un dictionnaire contenant tous les mots du livre par nombre d'occurences du mot. On utilise ensuite des formulaires Django standard pour transmettre les données à l'aide d'un sérialiseur Django REST Framework avant de l'enregistrer dans la base de données.

Une fois l'indexation effectuée et en se basant sur les dictionnaires obtenus, on peut maintenant implémenter les fonctionnalités de recherche et de classification qui décrivent les fonctionnalités de base de notre moteur de recherche.

5.3 Recherche simple

Afin d'effectuer une recherche simple, on utilise la table ***BookIndex*** obtenu précédement afin de récupérer la données wordOcc pour ensuite la convertir en dictionnaire avec la fonction ***ast.literal_eval(book.wordOcc)***. Ainsi, pour chaque mot que l'utilisateur va entrer, on retourne tous les livres où celui-ci figure dans leur dictionnaire respecetive. La complexité de cet algorithme est en $O(n)$ où n est le nombre de livre indexer qui contiennent le mot.

5.4 Recherche avancée

Pour cette algorithme, nous allons utiliser une expression régulière en entrer afin de retourner la liste des livres qui contient au moins un mot accepté par cette Regex.

Pour optimiser cette recherche, nous allons reprendre l'algorithme aho-ullman implémenter dans le premier projet de l'UE. Afin d'adapter cet algorithme avec notre projet actuel, nous avons modifié la sortie du programme afin qu'il puisse retourner le nombre d'occurences des mots qui valide cette regex. Cependant, nous n'avons pas réussi à modifier l'entrée de l'algorithme pour qu'il puisse prendre seulement une liste de mots contrairement à un fichier.

5.5 Algorithme de suggestion

L'indice de Jaccard, également connu sous le nom de coefficient de similitude de Jaccard, est une statistique utilisée pour comprendre les similitudes entre les ensembles d'échantillons. La mesure met l'accent sur la similitude entre les ensembles d'échantillons finis et est formellement définie comme la taille de l'intersection divisée par la taille de l'union des ensembles d'échantillons.

Semblable à l'indice de Jaccard, qui est une mesure de similitude, la distance de Jaccard mesure la dissimilarité entre les ensembles d'échantillons. La distance de Jaccard est calculée en trouvant l'indice de Jaccard et en le soustrayant de 1, ou en divisant les différences à l'intersection des deux ensembles. La formule de la distance de Jaccard est représentée par :

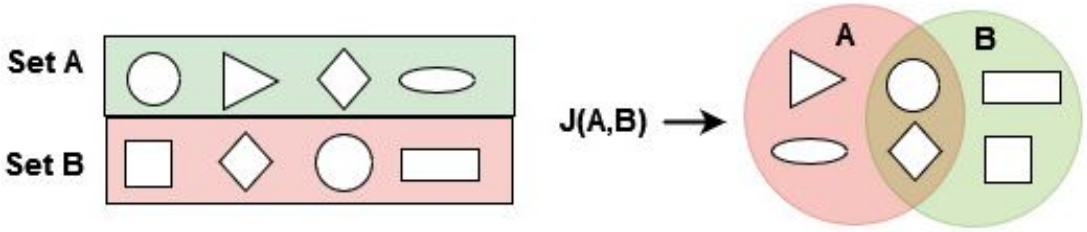

FIGURE 4 – Schéma illustrant la distance de Jaccard entre deux ensembles A et B

Cet algorithme va nous permettre de construire un arbre appelé l'arbre de Jaccard qui pour chaque livre l lui associe un ensemble de livres S tel que $\forall s \in S d(l, s) \leq \alpha$ où α est un seuil défini au préalable. Cet arbre sera ensuite stocké dans la table BookGraphJaccard afin de ne pas le recréer à chaque recherche.

L'algorithme parcourt la liste L des livres retourner lors de la recherche (simple ou avancée) et $\forall l \in L$ renvoie la liste des neighbour(l) du graphe de jaccard enregistrer (pour un seuil fixer à 0.65 qui a été determiné après plusieurs tests de comparaison). La complexité de l'algorithme est en $O(n * m)$ où n est le nombre de livres (sommet du graphe) et m le nombre total.

5.6 Algorithme de classement

Une classification des moteurs de recherche en catégories nous aide à trouver plus rapidement des éléments d'information spécifiques en organisant ces informations de manière significative et en identifiant les catégories dans lesquelles les informations appartiennent pour rechercher ces informations classifiées.

Cet algorithme va donc classer les livres obtenus par la recherche simple ou la recherche avancée avec plusieurs critères de classement, pour notre cas on a opté pour les classifications suivantes :

— nombre de livre : permets de classer par ordre lexicographique les livres par le nom de leurs auteurs.
— occurence : permets de classer par ordre lexicographique les livres sur sa langue de saisit.
— score (crank) : cette fonctionnalités inclus d'utilisation d'un algorithme de closeness qui est exprimer par la formule :

$$\boldsymbol{crank(v)} = \frac{n-1}{\sum_{u \neq v} d(u,v)}$$

cette formule représente la mesure de la centralité dans un réseau, calculée comme l'inverse de la somme de la longueur des chemins les plus courts entre le noeud et tous les autres noeuds du graphe. Ainsi, plus un noeud est central, plus il est proche

de tous les autres noeuds. l'algorithme retournera alors les résultats de la rechercher classer par ordre croissants du crank.

6 Front-End

6.1 Recherche simple et avancée

Le site se constitue simplement d'une barre de recherche avec un bouton afin de switcher entre une recherche simple à une recherche avancée (et vis versa). Il suffira donc à l'utilisateur d'entrer un mot ou une expression regulière puis d'appuyer sur entrer (ou appuyer sur la loupe) pour lancer la recherche. L'exemple suivant montre le résultat de la recherche du mot="alice".

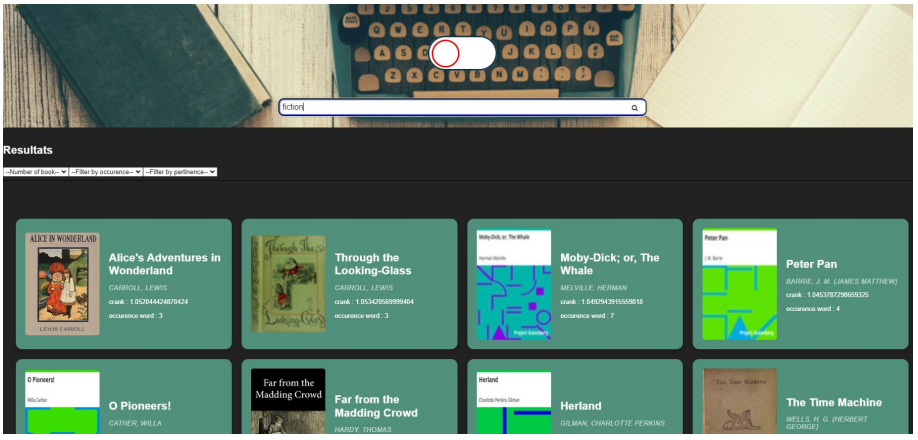

FIGURE 5 – Visuel de la rubrique de suggestion du Front-End

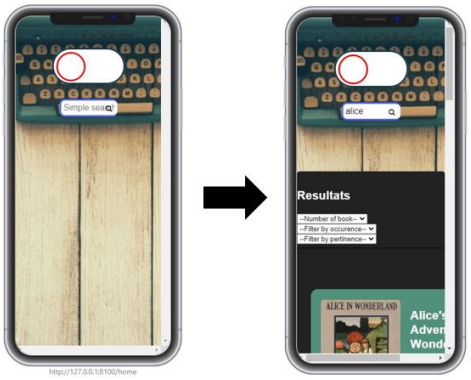

FIGURE 6 – Simulation de l'application sur mobile avec Xcode

6.2 Classement

Afin d'aider l'utilisateur dans son choix de livres, nous proposons une fonctionnalité de classement qui s'effectue durant le traitement de données du front avec 3 different filtre. Un filtre permettant de récupérer seulement les n premiers livres car certaines recherches peut engendrer un grand nombre en résultat. Un filtre qui classe les livres par nombre décroissant d'occurrence du mot de chaque livre. Enfin, un filtre de pertinence avec un classement par nombre décroissant de la valeur de ***crank*** pour chaque livre.

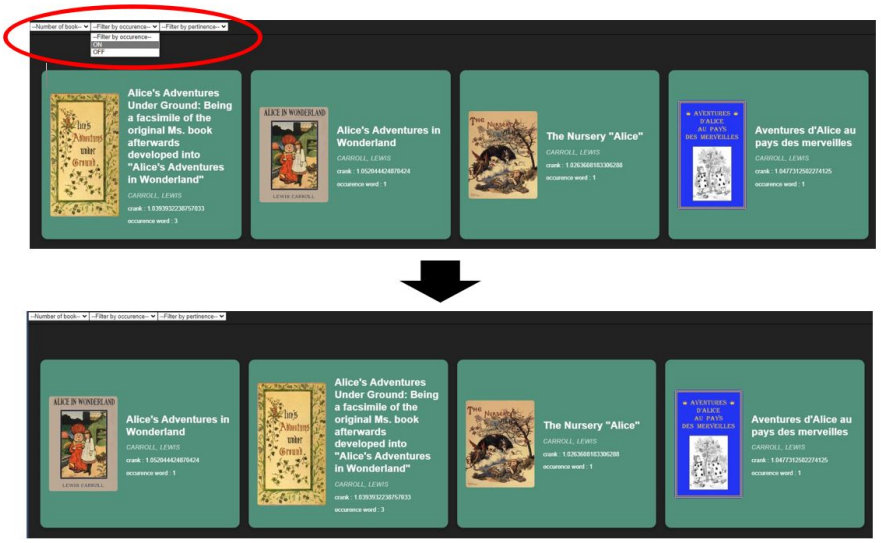

FIGURE 7 – Visuel du filtrage pouvant être effectué

6.3 Suggestion

En dessous des résultats de la recherche précédente, on pourra retrouver une suggestion de livre qui est déterminé par notre Graphe de Jaccard.

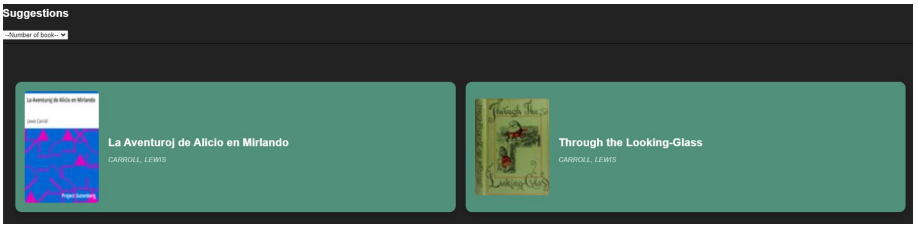

FIGURE 8 – Visuel de la rubrique de suggestion du Front-End

7 Tests

7.1 Tests unitaires

Afin de vérifier le bon fonctionnement et l'intégrité de la structure, nous avons mis en place des tests unitaires. Il s'agira ensuite de regarder si les données ont été correctement ajouté dans la base de données afin de s'assurer que le programme exécute exactement les fonctionnalités ainsi que toutes les opérations sur la base de donnée.

```
serializer = listOFBooksSerializer(data={
                        "id": 9999,
                        "title": "test unitaire",
                        "author": "Amine Kheddar",
                        "language": "fr",
                        "imageBook": "pas",
                        "text" : "ceci est un test "
                    })
if serializer.is_valid():
    serializer.save()
    self.stdout.write(self.style.SUCCESS('['+time.ctime()+']
                                        Successfully added test 1 '))
```

	id	title	author	language	text	imageBook	crank	occurence
	9999	test unitaire	Amine Kheddar	fr	ceci est un test	pas	0	0

FIGURE 9 – Vérification des données entrer dans la base de données

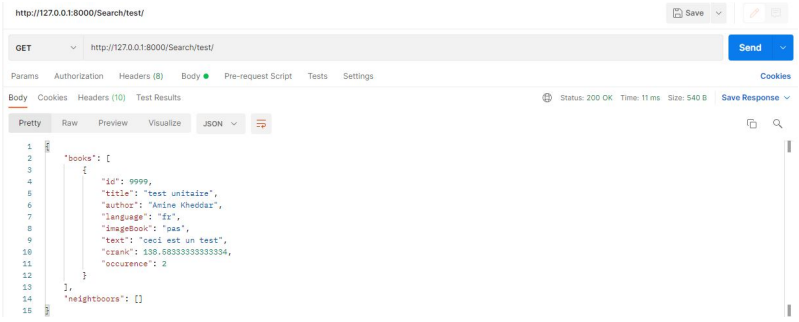

FIGURE 10 – Test effectuer sur postman pour la vérification du fonctionnement du Back-End

Pour la distance de jaccard, nous avons également vérifié la cohérence des résultats sur quelque livre avant de l'appliquer entièrement sur toute la base.

7.2 Tests de performances

Initialement, nous avons decider d'utiliser les mots du résumé ainsi que du titre du livre pour effectuer nos recherches sur le moteur de recherche. En faisant cela, les temps d'exécution et les temps d'indexage et calcul de la distance de jaccard ainsi que du crank sont beaucoup plus courts et nous ont permis d'effectuer plus facilement les tests lors du dévelopemment du projet.

Pour tester la performance des différentes recherches, nous avons calculé le temps d'exécution que prend le back end pour envoyer les données de résultat de la recherche :

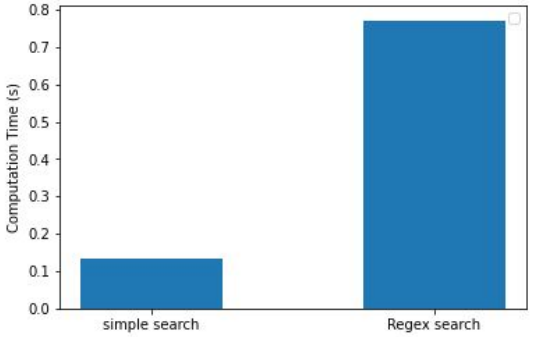

FIGURE 11 – Comparaison des temps d'éxecution entre la recherche simple et avancée pour 50 livres

On remarque à partir de ces résultats que le temps de réponse pour la recherche simple est beaucoup plus rapide que la recherche avancée. Cela s'explique facilement car contrairement a la recherche simple qui vérifie directement dans les dictionnaires d'indexations des livres la présence du mot, la recherche avancée utilise notre egrep sur chaque livre. Egalement, pendant cette recherche, chaque texte du livre est stocké dans un fichier en local pour pouvoir le faire fonctionner avec notre egrep, ce qui augmentera le temps de réponse.

Pour les 1664 livres dans la base, le temps de réponse pour une recherche avancée peut aller jusque plus de 5 minutes !

8 Conclusion

La réalisation de ce projet nous a permis de comprendre comment classer efficacement les résultats et surtout l'importance que peut apporter les différentes classifications d'une recherche. En effet, il serait moins cher et plus facile pour un site de moteur de recherche d'afficher simplement des pages ou des résultats au hasard, par nombre de mots, par fraîcheur ou par l'un des divers systèmes de tri faciles.

Cependant la raison pour laquelle ils ne le font pas est principalement pour l'utilisateur car celui ne l'utilisera pas. Les utilisateurs que nous devons essayer de satisfaire ne seront pas attirer par ce moteur si ce celui ne propose pas une précision de résultat et d'aide optimal.