Rapport DAAR Choix A

Moteur de recherche

Binôme

AOURTILANE Khaled

SOUMARE Nicolas

Plan

Introduction

Aujourd'hui, avec la prolifération des données numériques, la gestion et la recherche efficace de ces données sont devenues des défis majeurs pour les entreprises et les particuliers. Les algorithmes et les techniques de recherche d'informations ont donc pris une place prépondérante dans la gestion de ces données.

Les grandes entreprises du domaine, telles que Google, ont ainsi développé des techniques innovantes pour gérer leur immense base de données et offrir des résultats de recherche pertinents et rapides à leurs utilisateurs. Ces entreprises ont notamment recours à des techniques d'indexage et d'analyse de données pour fournir des résultats pertinents et personnalisés en temps réel.

La recherche d'informations dans un grand volume de données textuelles peut être fastidieuse et chronophage, en particulier lorsque l'utilisateur ne sait pas exactement où se trouve l'information recherchée. Il est donc essentiel de fournir un outil efficace de recherche en texte intégral pour permettre aux utilisateurs d'accéder rapidement à l'information souhaitée.

Dans ce contexte, notre projet a pour objectif de développer une application web avec un moteur de recherche permettant aux utilisateurs d'accéder plus rapidement à des documents textuels stockés dans une base de données. L'objectif principal de ce projet était de fournir un moyen efficace de rechercher des documents en utilisant des mots-clés et des expressions régulières pour trouver des correspondances exactes.

Le développement de cette application web a nécessité l'utilisation de compétences en algorithmique pour développer un moteur de recherche performant, ainsi qu'en développement web pour créer une interface utilisateur conviviale. En outre, la gestion de la base de données a été un élément important de ce projet pour stocker et récupérer les données nécessaires pour la recherche.

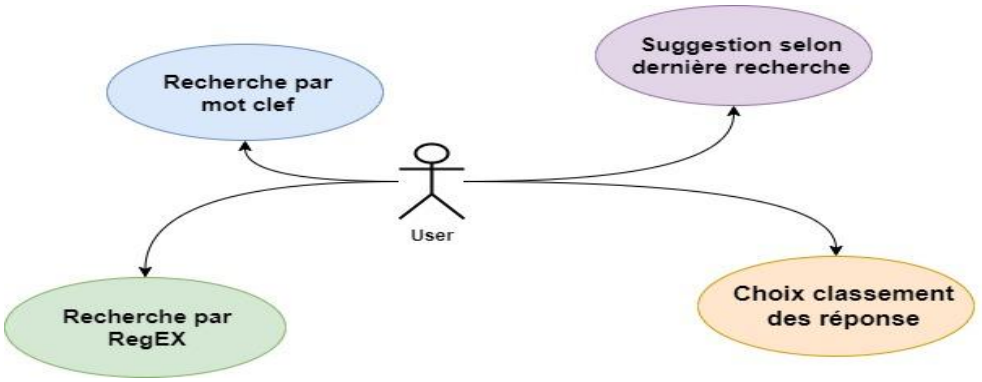
Dans ce rapport, nous présenterons en détail l'implémentation de l'application web, notamment le développement du moteur de recherche et la configuration de la base de données. Nous décrirons également les résultats obtenus, les difficultés rencontrées ainsi que les perspectives d'amélioration pour le projet.

Problématique

Le problème auquel nous faisons face est celui de la recherche efficace de documents textuels dans une bibliothèque de livres numériques. Avec une taille de 1664 livres et une taille minimale de 10 000 mots par livre, la quantité de données à traiter peut être immense et peut rendre la recherche manuelle fastidieuse et chronophage. Par conséquent, il est

essentiel de mettre en place un moteur de recherche performant et efficace qui permettra à l'utilisateur de trouver rapidement les documents pertinents en fonction de leurs besoins et de leurs intérêts. La principale difficulté est de gérer efficacement l'indexation des données et de garantir une pertinence optimale des résultats de recherche tout en maintenant des performances acceptables pour l'application web/mobile. De plus, il est important de proposer une expérience utilisateur fluide et intuitive pour faciliter l'utilisation de l'application et maximiser son adoption par les utilisateurs. Dans la suite de cet article, nous aborderons plus en détail les défis techniques et fonctionnels qui se posent pour la mise en place de ce moteur de recherche.

Use case



Architecture Logicielle

Technologies utilisées:

La partie architecture de ce projet est constituée d'une architecture à deux niveaux : une couche front-end et une couche back-end.

La couche front-end utilise le framework Angular CLI, qui est un framework open-source développé en TypeScript, et qui permet de créer des applications web dynamiques et réactives. Angular CLI est un choix judicieux pour le front-end car il offre une grande flexibilité dans la gestion de l'interface utilisateur et la manipulation des données.

La couche back-end est développée en Python avec le framework Django, qui est également open-source. Django est un framework de haut niveau qui permet de créer des applications web complexes rapidement et efficacement. Il offre des fonctionnalités avancées pour la gestion des bases de données, la sécurité et l'authentification des utilisateurs, ainsi que la manipulation des données.

Le système de gestion de base de données choisi pour ce projet est SQLite, qui est un système de gestion de base de données relationnelle léger, rapide et simple à utiliser. SQLite est un choix judicieux pour ce projet car il ne nécessite pas de configuration de serveur et est facilement intégrable dans une application web.

L'architecture globale est basée sur un modèle client-serveur, où le client est représenté par le front-end et le serveur est représenté par le back-end. La communication entre les deux couches est assurée par une API REST, qui permet d'échanger des données au format JSON entre le client et le serveur.

En résumé, l'architecture globale de ce projet est basée sur une approche moderne, avec des technologies éprouvées telles que Angular CLI, Django et SQLite. Cette architecture offre une grande flexibilité, une facilité de développement et une scalabilité, et permet de répondre aux exigences du projet de manière efficace et efficience.

Conception BDD

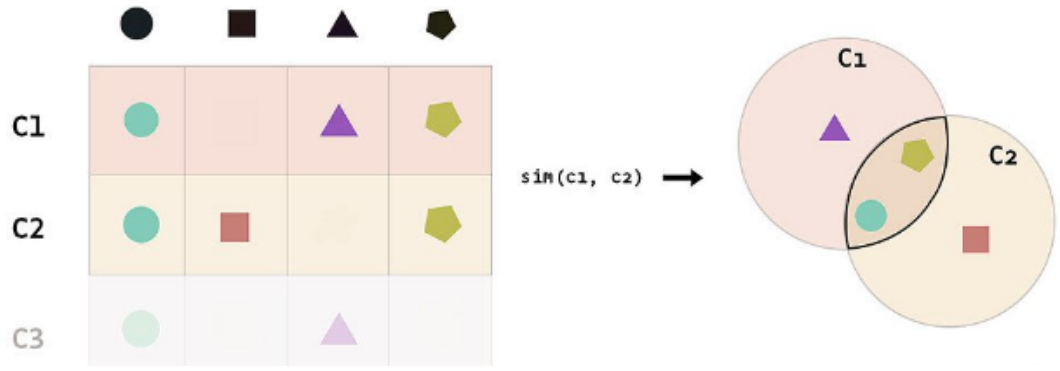
Pour la conception de notre base de données, nous avons pris en compte les exigences spécifiques de notre système de recherche de livres. Nous avons donc créé deux tables pour stocker les informations sur les livres et leurs index. La première table contient toutes les informations sur chaque livre, telles que le titre, l'auteur, la langue et le crank.

La deuxième table est utilisée pour l'indexation de nos livres, elle contient l'occurrence de chaque mot dans le livre, ce qui nous permet de faire des recherches efficaces et précises sur la base de données. Grâce à cette indexation, les utilisateurs peuvent rechercher un livre en utilisant des mots clés pertinents pour trouver les résultats les plus appropriés.

```
Table bookObject
{
id = Identifiant du livre
author  = Le nom de l'auteur
language = La langue de saisie
title = Le titre du livre
text = Le lien vers le livre en .txt
crank = Le score du livre
coverBook = Le lien vers l'image de
couverture du livre
}
```

```
Table bookIndex
{
id = Identifiant de l'index
word = Le mot indexé
idBook = L'identifiant du livre où
se trouve le mot
occurrence = Le nombre d'occurrence
du mot dans le livre
}
```

En plus de ces deux tables, nous avons créé une table pour stocker notre graphe de jacquard. Cette table contient les ID des livres voisins, permettant ainsi aux utilisateurs de découvrir d'autres livres similaires à ceux qu'ils consultent actuellement. Cette fonctionnalité facilite l'exploration de notre catalogue de livres et permet aux utilisateurs de découvrir de nouveaux titres qu'ils pourraient aimer.



En plus du graphe de Jaccard, nous avons également utilisé l'algorithme CRANK (Common Ranking Algorithm for Network and Knowledge graph) pour calculer un score pour chaque livre en fonction de sa similarité avec d'autres livres. Le score CRANK est basé sur l'analyse de la structure de notre graphe de Jaccard, ce qui nous permet de

déterminer la pertinence de chaque livre par rapport aux autres. Ce score sera utilisé pour recommander des livres similaires aux utilisateurs en se basant sur leur historique de lecture. Plus le score est élevé, plus le livre est considéré comme étant similaire aux autres livres de la bibliothèque. Cette approche permettra de suggérer des livres pertinents et intéressants pour les utilisateurs en fonction de leurs préférences de lecture.
La formule de Crank est la suivante :

*Crank(livre) = somme(jacquard_score(livre, voisin) * score(voisin)) pour tout voisin du livre*

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

En résumé, notre conception de base de données prend en compte les besoins spécifiques de notre système de recherche de livres et nous avons utilisé des techniques d'indexation efficaces pour permettre des recherches rapides et précises.

Conception et algo

Indexation

L'indexation des données des livres est l'un des piliers de notre projet pour garantir une recherche rapide et efficace. Pour cela, nous avons mis en place une technique d'indexation basée sur l'analyse des mots-clés dans chaque livre. Lorsqu'un livre est ajouté à notre base de données, nous parcourons chaque mot dans son texte et l'associons à l'identifiant unique de ce livre. De cette façon, lorsqu'un utilisateur effectue une recherche, nous pouvons rapidement accéder aux livres pertinents en utilisant l'index correspondant aux mots-clés de sa requête.

Cette technique d'indexation nous permet d'optimiser les performances en réduisant le nombre de recherches effectuées sur les textes des livres. En effet, au lieu de parcourir l'ensemble des livres à chaque recherche, nous pouvons nous concentrer sur les livres pertinents qui ont été préalablement indexés. En outre, en utilisant des algorithmes d'indexation de mots-clés avancés, nous pouvons améliorer la précision des résultats de recherche et minimiser les temps de réponse.

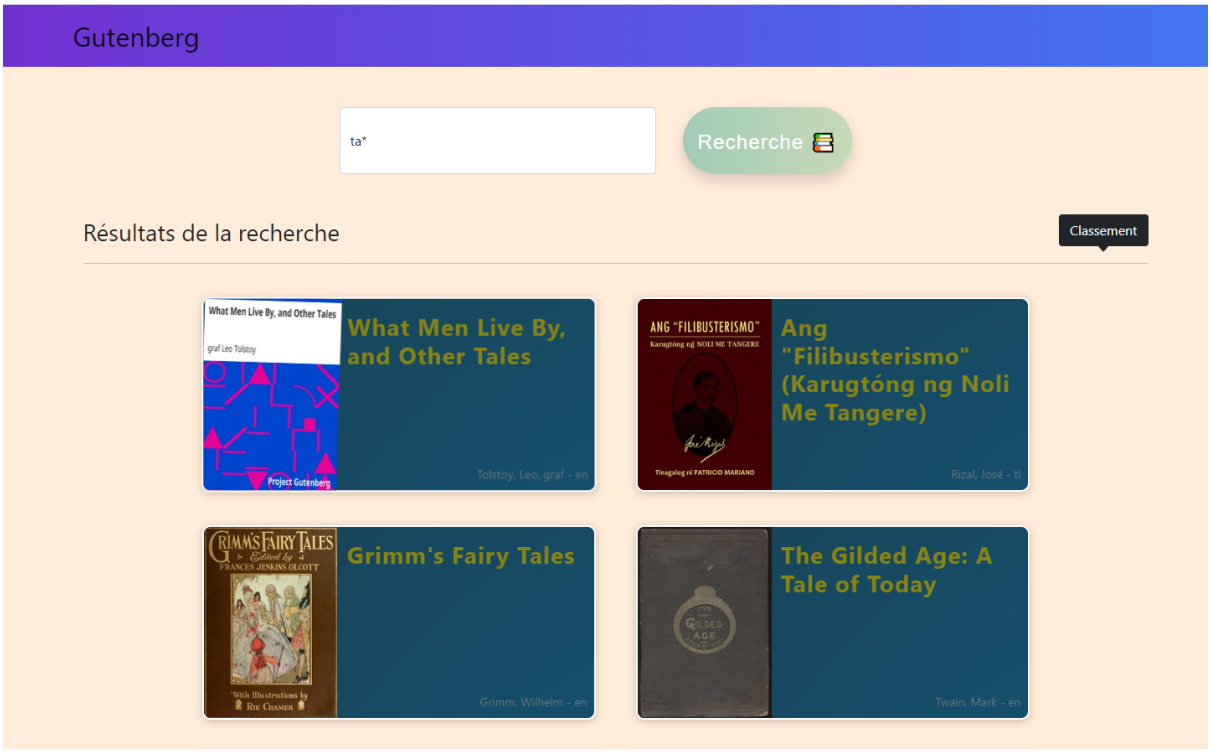
Fonctionnalitées

Recherche simple

Pour la recherche simple, nous avons développé un algorithme qui prend en entrée une chaîne de caractères (mots clés) et recherche dans la table d'index pour retourner une liste de livres contenant ces mots clés. L'algorithme parcourt la table d'index et récupère l'ID des livres contenant les mots clés. Ensuite, il parcourt la table de livres pour récupérer les informations sur ces livres et les renvoie sous forme de liste.

Recherche avancée

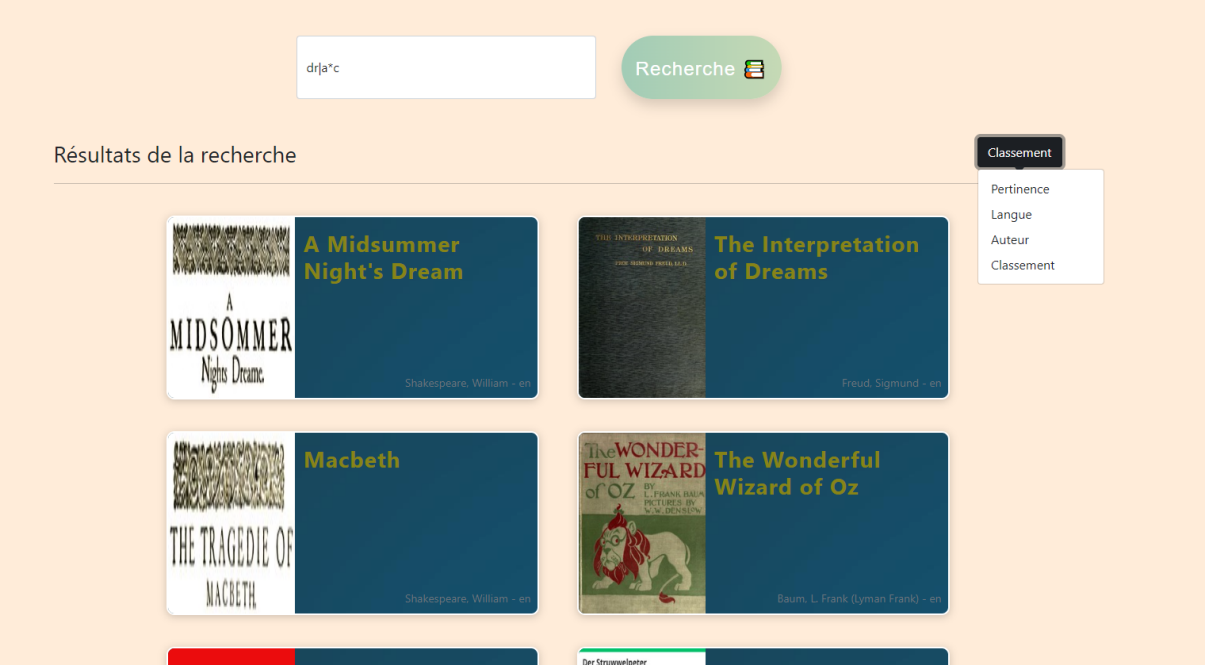
La fonctionnalité de recherche avancée permet d'effectuer une recherche en utilisant des expressions régulières (regex). L'algorithme utilise la même méthode que la recherche simple, mais avec l'ajout d'un jar externe permettant de traiter les expressions régulières.. Nous avons développé un algorithme qui prend en entrée une expression régulière et recherche dans une base de mots prédéfinie de mot anglais, puis on trouve dans la table d'index les livres contenant les mots résultant de l'expression régulière. L'algorithme utilise la fonction de recherche SQLite pour effectuer la correspondance et renvoie une liste de livres contenant des correspondances.



Classement

Nous avons implémenté une fonctionnalité de classement des livres pour permettre aux utilisateurs de trier les livres en fonction de différents critères tels que l'auteur, la langue ou la pertinence. Pour ce faire, nous avons développé un algorithme qui prend en entrée un critère de tri et trie les livres en fonction de ce critère. L'algorithme utilise des requêtes

SQL pour trier les livres en fonction du critère de tri sélectionné et renvoie une liste triée de livres.



Suggestion

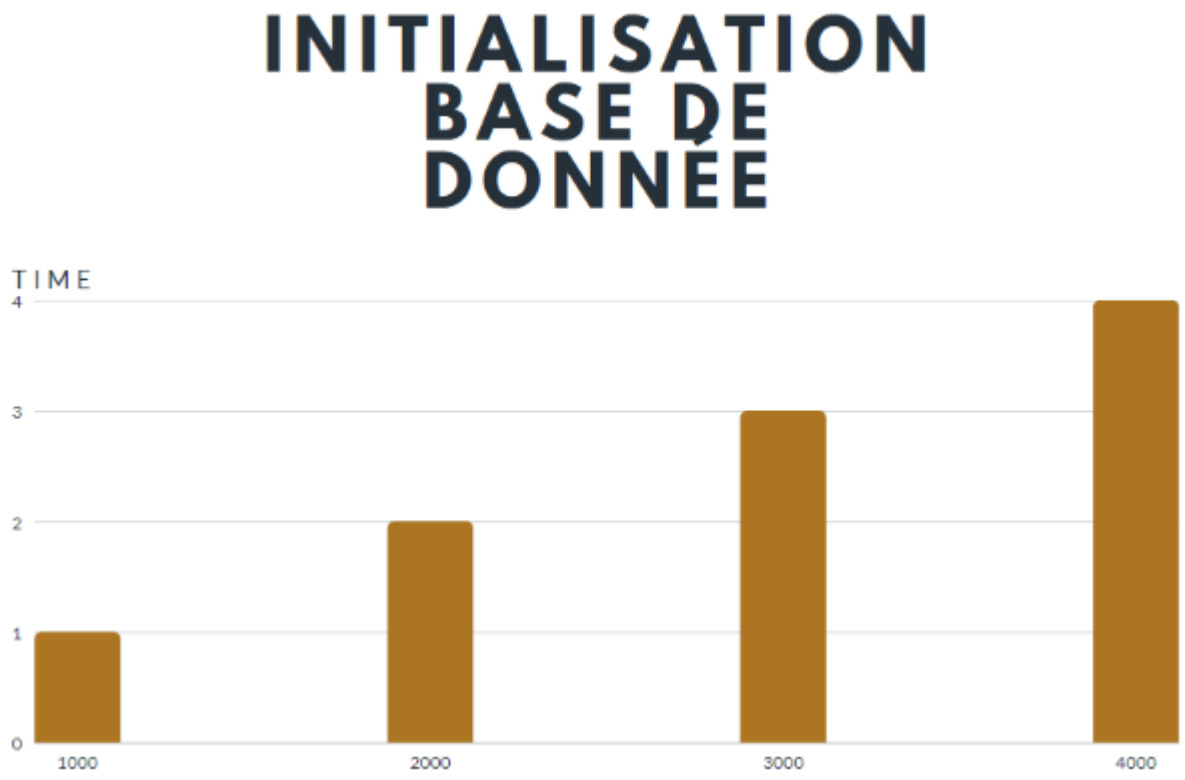
Pour la fonctionnalité de suggestion de livres, nous avons utilisé la technique Crank pour trouver des livres similaires en fonction du graphe de Jacquard. Nous avons développé un algorithme qui prend en entrée un livre et utilise le graphe de Jacquard pour trouver les livres similaires. L'algorithme calcule le score Crank pour chaque livre voisin en fonction du graphe de Jacquard et renvoie une liste de livres triée par score Crank décroissant.

Nous avons implémenté ces fonctionnalités dans le back²-end Django et les avons exposées via des API RESTful. Le front-end Angular utilise ces API pour interagir avec la base de données et fournir une interface utilisateur conviviale pour les utilisateurs.

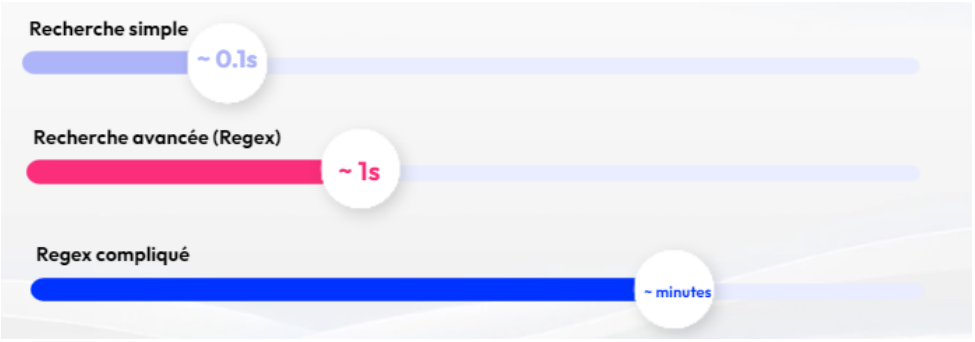
TEST

L'initialisation de la base de données consiste à créer les tables nécessaires pour stocker les informations sur les livres, ainsi que les index et les contraintes pour garantir l'intégrité des données. Ce processus peut être relativement rapide pour une petite quantité

de données, mais il peut prendre beaucoup plus de temps lorsque le nombre de livres est important.



Concernant les recherches voici une moyenne qu'on a observé en utilisant des regex plus ou moins complexes :



Amélioration

Lorsqu'il s'agit de créer un système de classement de livres efficace, il est crucial de trouver des moyens d'améliorer la vitesse de traitement tout en réduisant les coûts associés. Heureusement, il existe plusieurs méthodes qui peuvent aider à atteindre ces objectifs.

La première méthode consiste à utiliser des algorithmes de calcul distribué tels que Hadoop ou Spark. Plutôt que de traiter toutes les données sur un seul ordinateur, ces outils permettent de diviser le travail entre plusieurs ordinateurs, ce qui peut considérablement réduire le temps de traitement.

Une autre méthode consiste à optimiser les requêtes en utilisant des outils tels qu'Elasticsearch ou Apache Solr. Ces outils d'optimisation de requêtes peuvent accélérer les temps de recherche et réduire les coûts liés aux requêtes.

Une troisième méthode consiste à utiliser des services de cloud computing tels qu'Amazon Web Services ou Google Cloud Platform. En utilisant ces services, il est possible d'obtenir une puissance de traitement supplémentaire à un coût abordable.

Enfin, il est possible d'utiliser des algorithmes plus rapides pour calculer la centralité de betweenness ou la similarité de Jaccard. Il existe de nombreux algorithmes alternatifs qui peuvent être plus rapides que ceux que vous utilisez actuellement.

En combinant ces différentes méthodes, il est possible de créer un système de classement de livres efficace qui peut traiter des données de manière rapide et économique. Que vous utilisez des algorithmes de calcul distribué, des outils d'optimisation de requêtes, des services de cloud computing ou des algorithmes plus rapides, il est important d'explorer toutes les options pour trouver la solution la plus adaptée à vos besoins.

Conclusion et perspectives

En conclusion, le projet de recherche de livre par indexage est un système efficace pour trouver des livres pertinents en fonction des mots-clés saisis par l'utilisateur. En utilisant

une base prédéfinie de mots usuels en anglais, le système est capable d'élargir la recherche et de trouver des résultats plus pertinents. En utilisant une table d'index pour stocker les mots clés et les ID des livres correspondants, le système est capable de trouver rapidement les livres qui contiennent les mots clés. En calculant les occurrences de chaque mot clé pour chaque livre trouvé, le système est capable de déterminer les résultats les plus pertinents.

La fonctionnalité de suggestion de livres est un élément supplémentaire qui utilise la technique Crank pour trouver des livres similaires en fonction du graphe de Jacquard. Cela permet aux utilisateurs de découvrir de nouveaux livres qui peuvent correspondre à leurs intérêts et à leurs préférences de lecture.

Pour les perspectives d'optimisation, plusieurs pistes peuvent être envisagées. Tout d'abord, il serait intéressant d'explorer des techniques d'indexation plus avancées, telles que l'indexation inversée ou l'utilisation de trie pour améliorer la vitesse de recherche.

Ensuite, l'application de techniques de Machine Learning pour l'amélioration de la suggestion de livres en fonction des préférences des utilisateurs est une perspective intéressante. Il serait possible d'utiliser des algorithmes de clustering pour regrouper les utilisateurs ayant des préférences similaires, puis d'utiliser des techniques de recommandation pour suggérer des livres en fonction des livres les plus populaires parmi les groupes de préférences similaires.

En somme, le développement de cette application a été une expérience riche en enseignements. Elle nous a permis de mettre en pratique des concepts théoriques clés de l'informatique et de l'ingénierie logicielle, ainsi que de découvrir de nouvelles techniques et technologies. Les perspectives d'optimisation et d'amélioration sont multiples et ouvrent de nouvelles possibilités pour le développement de l'application dans le futur.

Lien de la vidéo

Lien vers la vidéo haute qualité:

https://youtu.be/_4vsbFi48Pg

Bon visionnage.