# Vers l'internationalisation d'un moteur de recherche juridique

Mohamed Chaieb

# Vers l'internationalisation du moteur de recherche

Mohamed
Chaieb

Sous la direction de Mathieu Loiseau

Réalise au sein de l'entreprise : Luxia SAS
Sous la direction de Fabien Tiret

UFR LLASIC
Département I3L

---

Mémoire de master 2 mention Sciences du langage - 20 crédits

Parcours : Industrie de la langue, orientation professionnelle

Année universitaire 2019-2020

## Remerciements

*Au terme de ce travail, je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont contribué au bon déroulement de mon projet de fin d'études.*

*Je tiens particulièrement à remercier Monsieur Martin, responsable de Luxia, pour m'avoir accueilli au sein de son équipe et pour m'avoir encadré et orienté durant mon stage.*

*Je remercie également mon tuteur Monsieur Tiret Fabien, développeur chez Luxia, pour sa patience, son assistance, ses directives et ses conseils précieux.*

*Mes remerciements s'adressent aussi à mon enseignant référent Monsieur Loiseau Mathieu, pour l'aide et le soutien qu'il m'a accordé tout au long de ce stage.*

*Avec beaucoup d'égard, je ne manque pas d'exprimer ma grande reconnaissance à tous les enseignants et administrateurs de l'Université Grenoble Alpes et tous les membres du jury pour avoir accepté de juger ce modeste travail.*

**DÉCLARATION ANTI-PLAGIAT**

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

PRENOM : Mohamed

NOM : CHAIEB

DATE : 2019/2020

# Sommaire

# Introduction

L'objectif de mon travail est de réaliser une solution pour rendre les services proposés par l'entreprise Luxia accessibles en multilingue. En effet il s'agit d'internationaliser son moteur de recherche sollicité par une clientèle diversifiée dans la recherche juridique.

De ce fait, ce projet synthétise le travail que j'ai réalisé pour répondre à ses besoins spécifiques au domaine juridique. Il est composé de quatre chapitres qui sont organisés comme suit :

- Le premier chapitre est consacré à la présentation du sujet, suivi par la présentation de l'organisme d'accueil et son secteur d'activité.
- Le deuxième chapitre est consacré à l'analyse et la spécification des besoins.

  Durant ce chapitre j'ai cité les fonctionnalités principales que la solution doit offrir.
- Dans le troisième chapitre, j'ai détaillé la phase de conception et les techniques qui ont servi à la réalisation de mon projet.
- Enfin, le mémoire est couronné par une conclusion générale récapitulant le travail réalisé et mentionnant les perspectives.

# Partie 1

# Présentation générale du projet

# I. Chapitre 1. Présentation générale du projet

## 1. Présentation du projet

Ce projet s'inscrit dans le cadre de la réalisation d'un stage de fin d'études intitulé «Internationalisation du moteur de recherche juridique » au sein de l'entreprise Luxia.

Il vient conclure mon cycle de formation en master professionnel en industrie de la langue à l'Université Grenoble Alpes.

Le projet a été proposé par la start-up Luxia. Il a été mené sous la tutelle de M. Tiret Fabien, ingénieur développeur chez la dite entreprise.

## 2. Présentation du stage

Au cours de ce chapitre, je vais présenter dans un premier temps l'entreprise d'accueil « LUXIA » et son secteur d'activité, dans un deuxième temps je donnerai un aperçu des services proposés par la start-up et son public cible. Enfin j'aborderai le projet ainsi que la problématique, afin de proposer une solution adaptée aux besoins de l'entreprise.

### 1. Luxia

Luxia est une société de développement informatique offrant des solutions destinées principalement aux avocats et juristes, généralement à tous les utilisateurs du droit qui font recours aux codes de loi et articles de droit.

### 2. Stage

- ***Sujet***

**Internationalisation du moteur de recherche juridique**

L'objectif du stage est de rendre les services proposés par Luxia accessibles en multilingue, à la fois coté documents (disponibles dans différentes langues, pas toujours traduits) et coté utilisateurs, qui peuvent faire des recherches en différentes langues.

## 1. Le secteur d'activité

Luxia est une entreprise impliquée dans la LegalTech, parmi les spécialistes européens du Big Data juridique, un marché en plein essor où la demande est nombreuse et pointue.

En tant qu'une entreprise dans le domaine juridique, Luxia facilite l'accès aux données juridiques, en offrant des différentes solutions de droits.
Des millions des données sont traitées par Luxia, ce qui donne une grosse base de données juridique accessible à son public.

Le but de son intégration dans la LegalTech est d'aider les conseils juridiques des avocats ou voir les remplacer par un système automatisé.

Pour imposer son existence dans la LegalIT[1], Luxia utilise des différents technologies et concepts notamment le traitement automatique de la langue (TAL) qui fait partie de l'intelligence artificielle dont l'objectif est d'analyser et traiter les textes.

- LegalTech

« La technologie légale ou LegalTech est un concept qui a apparut dans les années 1960 et 1970. »[1]

LegalTech, technologie juridique ou technologie au service de droit, concept anglophone : Legal Technology, combine l'utilisation de la technologie et les logiciels pour proposer des services juridiques.

Les LegalTech sont des technologies juridiques qui mettent en valeur les techniques qui aident à automatiser les services juridiques, à la fois coté document, coté fonctionnement et coté relation avec les spécialistes du droit.
Grace à ses bibliothèques de données volumineuses, les LegalTech deviennent de plus en plus utilisées par les entreprises du droit, spécialement les petites startups.

Elles sont également utilisées par des particuliers qui profitent de cette technologie afin d'améliorer leurs services en offrant des solutions et des conseils juridiques satisfaisants leurs utilisateurs.

[1] LegalIT : legal informatics, un autre synonyme de LegalTech

- Big Data juridique

L'utilisation et le traitement des grandes quantités des données ont dépassé la capacité de stockage et de calcul des systèmes informatiques.

Pour cela, une nouvelle approche est apparue pour gérer et traiter ces grosses quantités d'information. Nous parlons ici du Big Data, ce domaine qui grâce à des nouvelles technologies d'analyses et statistiques des données notamment l'apprentissage machine, facilite le traitement sur les données.

Cette approche est utilisée dans plusieurs domaines, comme la santé, le transport, la gestion énergétique, et aussi dans le domaine juridique.

Le Big Data Juridique est donc un ensemble d'applications appliqué à des données judiciaires afin de les analyser et les traiter et ceci pour faciliter leurs utilisations dans d'autres secteurs d'activités, spécialement le LegalTech.

## 2. Les données traitées par Luxia

Luxia offre deux services dans le domaine juridique, Regmind et Alinéa by Luxia.

Ces deux services, qui utilisent le même moteur de recherche, ont comme mission de faciliter l'accès aux ressources juridiques grâces à différentes fonctionnalités mises à disposition de leurs publics, notamment la visualisation des actualités juridiques, la création de veilles ou d'alertes pour être à jour par rapport à toute évolution judiciaire avec un outil de comparaison entre les versions des documents.

Les donnés traitées par ces deux services diffèrent en fonction du public ciblé, Regmind se spécialise dans les documents bancaires et financiers, alors que Alinéa by Luxia pointe vers des textes de droits généralistes.

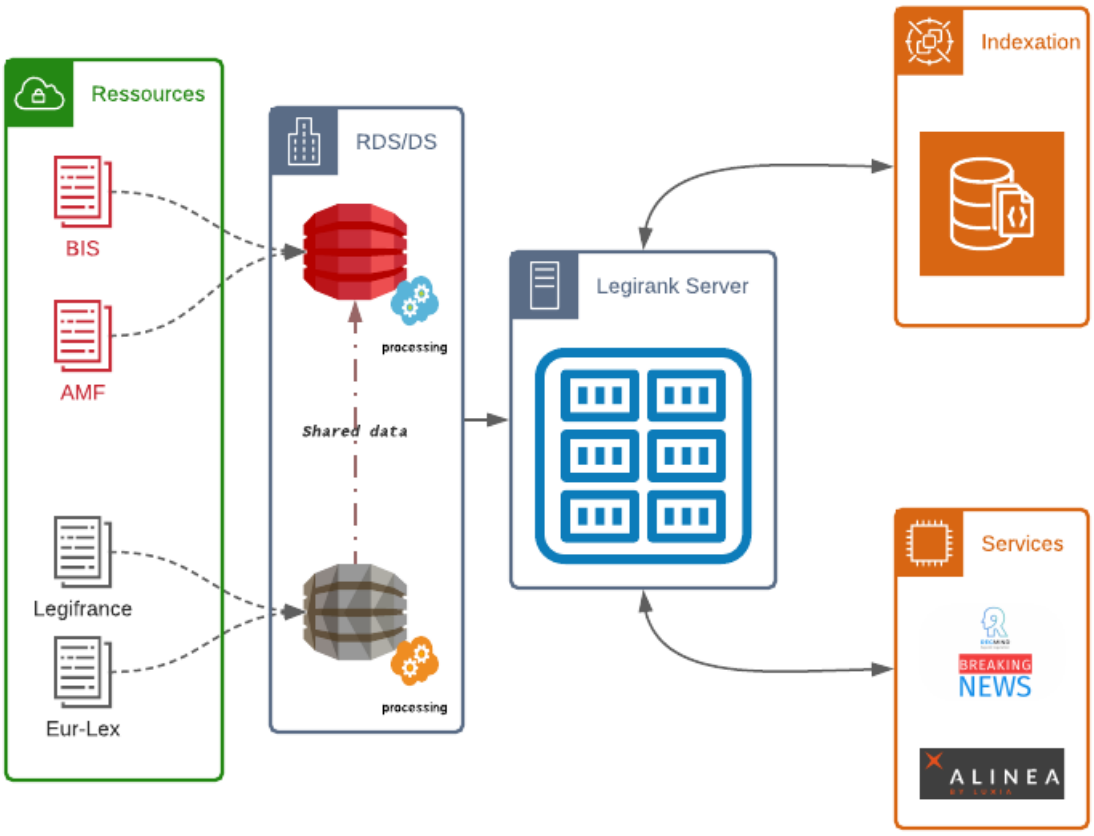


Figure 1 Données entrantes/services sortants de Luxia.

Le schéma ci-dessus présente le flux de donnés traités par le serveur Legirank de Luxia. Grâce à ses moteurs de recherche elle fait régulièrement l'extraction des actualités et les stocke en fonction de leurs natures/types dans ses bases de données.

## 1. Services fournis

- **Regmind**

  *Banque et finance*

  Regmind permet l'accès à l'ensemble de la régulation Européenne et nationale et aux réglementations bancaires, sous la forme de textes de référence multilingues, dossiers thématiques, de veille et les derniers événements avec une mise à jour quotidienne.

  On y trouve toutes les règles, la jurisprudence, les sanctions, les

recommandations et même les guidelines dans la langue d'origine du régulateur. Ainsi, Regmind offre la possibilité d'organiser le travail grâce aux dossiers avec différents modes de classement.

Dans Regmind, les formats proposés sont généralement des textes, et aussi sous forme des tables des matières pour les codes de Legifrance. Actuellement, réservé aux directions juridiques des banques et aux sociétés qui testent en avant-première toutes ses avancées technologiques.

- **Alinéa by Luxia**

  *Textes légaux généraux*

Alinéa est une solution destinée aux Juristes, avocats indépendants et aux étudiants.

Elle utilise un ensemble commun des données avec Regmind, notamment le journal officiel, les codes, les lois et les règlementations européennes.

Mais Alinéa contient également des documents absents de Regmind, par exemple « les sanctions de la CNIL [2] »[2].

En effet en tant qu'une open data juridique classique, toutes les lois indexées sont généralistes.

Elle donne l'accès en français et anglais à l'ensemble des sources du droit européen avec son moteur de recherche avancé.

## 2. Données entrantes

Luxia met à disposition son moteur de recherche qui utilise un ensemble fiable de ressources accessibles gratuitement au public :

[2] Commission nationale de l'informatique et des libertés : Protéger les données personnelles, accompagner l'innovation et préserver les libertés individuelles.

- **Legifrance**

  « C'est le site officiel du gouvernement français pour la diffusion des textes législatifs et réglementaires et des décisions de justice des cours suprême et d'appel de droit français. »[3]

  Il met à disposition du public une variété de données, notamment : La constitution, les codes, les lois et les actes à caractère règlementaire émanant des autorités de l'État, ainsi que les jurisprudences et des ensembles de publications officielles :

  a) L'édition « Lois et décrets » du Journal officiel de la République française

  b) Les bulletins officiels des ministères

  c) Le Journal officiel des Communautés européennes.[3]

- **Eur-Lex**

  Eur-Lex est l'une des branches du portail de l'union européenne donnant l'accès aux éléments suivants:

  - journal officiel de l'Union européenne
  - droit de l'Union Européenne
  - actes préparatoires (propositions législatives, rapports, livres verts, livres blancs, etc.)
  - jurisprudence de l'Union européenne (arrêts, ordonnances, etc.)

Legirank indexe les documents traitant du droit de la finance issus des sources ci-dessus, mais indexe également des documents issus des sources suivantes :

[3] Le Journal officiel de l'Union européenne ou JOUE est une publication périodique publiée par l'Office des publications de l'Union européenne.

- **BIS**

  Bank for International Settlements est une organisation financière internationale, qui s'occupe de finance internationale.

  Elle offre des publications et des études purement bancaires utiles pour Regmind, qui intéressent les banques pour avoir une prévision sur l'évolution des politiques, des études compétitives et des comparaisons par pays.

- **AMF**

  « L'Autorité des marchés financiers, est une institution financière et une autorité administrative indépendante française. » [4]

  Parmi ses missions, l'AMF garantit le bon fonctionnement des marchés d'instruments financiers[4] ce qui aide les clients à trouver leur stabilité financière.

  Elle est considérée parmi les axes de ressources de Regmind. Les banques ou même les institutions financières se servent de ses publications et de ses études pour garantir leur stabilité sur le marché financier.

  Ainsi, elle propose des analyses, des prises de position et des décisions pour agir dans l'intérêt général des banques.

## 3. Indexation de ressources

L'indexation désigne l'action du robot (c'est-à-dire d'un programme) d'un moteur de recherche qui passe sur un site, le parcourt et indexe son contenu. Lorsque l'on dit que le robot (le spider) indexe un site, cela signifie qu'il visite le site, en copie le contenu et le stocke dans les serveurs du moteur de recherche.[5]

Pour indexer ses données juridiques, Luxia applique un processus de traitement et d'analyses sur les documents extraites depuis sa base de données.

[4] Marché d'instruments financiers (MIF): Gère les opérations boursières entre les banques.

Le processus de préparation des documents pour l'indexation à pour objectif de définir le langage documentaire de tel document. Cette phase permet de choisir les bons éléments à indexer. Il s'agit de définir des champs spécifiques qui contiennent des parties déterminées pour chaque article, notamment le titre par exemple et son contenu, ainsi identifier pour chaque document sa langue en ajoutant cette valeur dans un champs ''langue''.

A partir de cette forme, les documents sont indexés dans un seul et unique index qui applique un analyseur français sur tous les champs des documents.

## 4. Mission et problématique

### *1. Mission principale*

La création d'un seul index pour des documents de langues différentes influence la recherche. Comme certains mots s'écrivent de la même manière dans différentes langues, le taux de correspondance va être de plus au moins élevé au niveau de la recherche, ce qui préserve le rappel. Mais dans certains cas, cela produit un taux de bruit important puisque certains termes pourront se trouver dans une autre langue, exemple :

Une recherche de ''commercial commission '' va renvoyer directement sur des documents en français sur la tête de la liste, pourtant l'utilisateur voulait une recherche en anglais. Donc, une recherche dans des documents qui ne peuvent pas convenir influence sur le temps du traitement de la requête, ce qui produit une autre insuffisance.

***Problématique :***

En vue de l'internationalisation de la solution, Luxia a besoin de rendre le moteur de recherche plus robuste en anglais, pour commercialiser la solution sur les pays européens et avoir de meilleurs résultats sur les documents européens et nouveaux clients anglophones.

Le premier objectif est de trouver une stratégie qui aide à montrer les différents documents possibles en fonction de la langue souhaitée par l'utilisateur, pour ensuite avoir de meilleurs résultats sur les documents européens. C'est aussi en vue de grandir nos donnés en fonctions du pays où on va commercialiser la solution.

## 2. *Missions annexes*

### 2.1 Etude technique

Cette internationalisation permet également une étude technique pour l'étude de faisabilité du passage de Solr[5] à Elasticsearch[6].

Actuellement le moteur de recherche utilise l'apache Solr pour l'indexation des documents.

Cette approche qui est basée sur la Bibliothèque Lucene[7], est considérée comme une base de données spécialisée dans la recherche d'information.

Cette migration technologique à pour but de tester les apports d'un nouvel outil qui est appliqué dans plusieurs domaines d'indexation de l'information.

Le développement en Elasticsearch offres des nombreuses avantages tel que une recherche large (recherche sur un grand ensemble de données) sur toutes sortes de documents quelle que soit sa configuration ou sa définition. Ainsi grâce à son architecture, il offre une capacité d'analyse et de synthèse importante surtout au niveau de partage de l'indexation entre les différents utilisateurs ce qui confirme la rapidité.

### 2.2 Gestion d'entité spécifique : les dates

Pour prendre en main l'outil, j'ai fait un traitement spécifique sur l'ensemble des documents.

Les données que j'ai indexées sont au format JSON, elles possèdent un format de date particulier, cette forme correspond à une suite de chiffres collés qui est insupportable par elasticsearch.

Le fonctionnement du script est comme suit :

- Parcourir les documents en lecture
- Identifier le champ dates (date de création, date de dernière modification, etc.)

[5] Solr : plateforme logicielle de moteur de recherche.
[6] Elasticsearch : logiciel pour l'indexation et la recherche des données, comme Solr il utilise le système d'indexation lucene.
[7] Lucene : bibliothèque java qui possède une puissante fonctionnalité d'indexation et recherche.

- Appliquer sur les valeurs des champs repérés une suite d'opération d'extraction des chiffres par position.
- Transformer les valeurs extraites au format datetime[8].
- Génération d'une date en isoformat[9].
- Recouvrir les documents avec le bon format

## 5. Méthodologie de travail

- Déroulement du stage

Le stage s'est déroulé de 02 mars à 28 aout 2020. Dès le début et jusqu'au le 15 mars, le travail était dans les locaux de Luxia. A partir du 16 mars nous avons entamé le télétravail général jusqu'à la fin du confinement, puis nous avons repris progressivement le travail normal dans les locaux.

- L'organisation au quotidien

Le déroulement de travail s'inscrivent dans la méthode agile[10], nous procédions par des Sprints[11] hebdomadaires et des stand-up meeting de 20 à 30 minutes par jours.

Pour chaque stand-up meeting :

*. Définition de l'objectif et des taches.*

*. Plan de réalisation*

Le suivi de la réalisation faisant dans github (tickets + milestones) a pour but :

*. Définir les tâches répondant à l'objectif du Sprint*

*. Le plan de réalisation de ces taches (sous taches techniques, ordonnancement…)*

[8] datetime : librairie python pour le type de base date et heure.
[9] isoformat : norme ISO 8601, pour une représentation numérique de la date et l'heure.
[10] Méthode agile : méthode de travail collaborative, sert à fixer les objectifs en devisant le projet en sous-projet.
[11] Sprints : est un cycle (période de temps) durant la quelle on définit les objectifs.

Cette méthode est indispensable pour le suivi de mon projet, elle permet de faire un point complet sur l'avancement de mes travaux, avec aussi des phases de contrôle du déroulement du projet selon l'organisation planifiée.

- Les différentes étapes du travail

Le travail comprend deux phases :

**Analyse des besoins :** afin de proposer une solution adéquate, il me faut tout d'abord étudier les besoins du client et ceci en recensant aussi bien ses besoins fonctionnels et que non fonctionnels. Cette phase était réalisée durant les premiers sprints de 3 mars 2020 à 5 mai 2020.

**Conception et développement :** C'est une étape dans laquelle, je mets en œuvre les besoins fonctionnels et leurs contraintes et je modélise le système à réaliser pour clarifier les tâches à accomplir dans la partie développement. Cette phase se termine par une partie qui comprend la programmation et les tests de validations. Elle e été accompli pendant les derniers sprints de juin et jusqu'au juillet 2020.

# Partie 2

# Analyse et spécifications

# II. Chapitre 2. Analyse et spécifications

## Introduction

Cette phase de mon étude est la plus importante, puisque c'est la première étape du développement de mon projet, au cours de laquelle les fonctionnalités que la solution devrait satisfaire seront déterminées.

Afin de proposer une solution adéquate, je présente dans un premier temps les attentes du public. Ensuite je vais dégager dans la deuxième partie les différents besoins fonctionnels et non fonctionnels à savoir les fonctionnalités requises par les utilisateurs.

### 1. Les attentes du client

Comme il s'agissait de reprogrammer le back-end, l'analyse du front-end doit répondre aux différentes besoins du client.


**Figure 2 interface principale sur Regmind**

La figure ci-dessus présente la première interface présentée sur Regmind.

Lors de lancements de Regmind, une interface d'accueil apparaît, elle offre la possibilité de :

- Faire une recherche sur les documents juridiques.
- Accéder aux veilles créées
- Accéder l'espace de travail (les dossiers et les documents créés)
- Accéder aux notes et articles favoris.
- Etc.

Regmind offre sa boite de recherche à son public afin de lancer ses recherches.

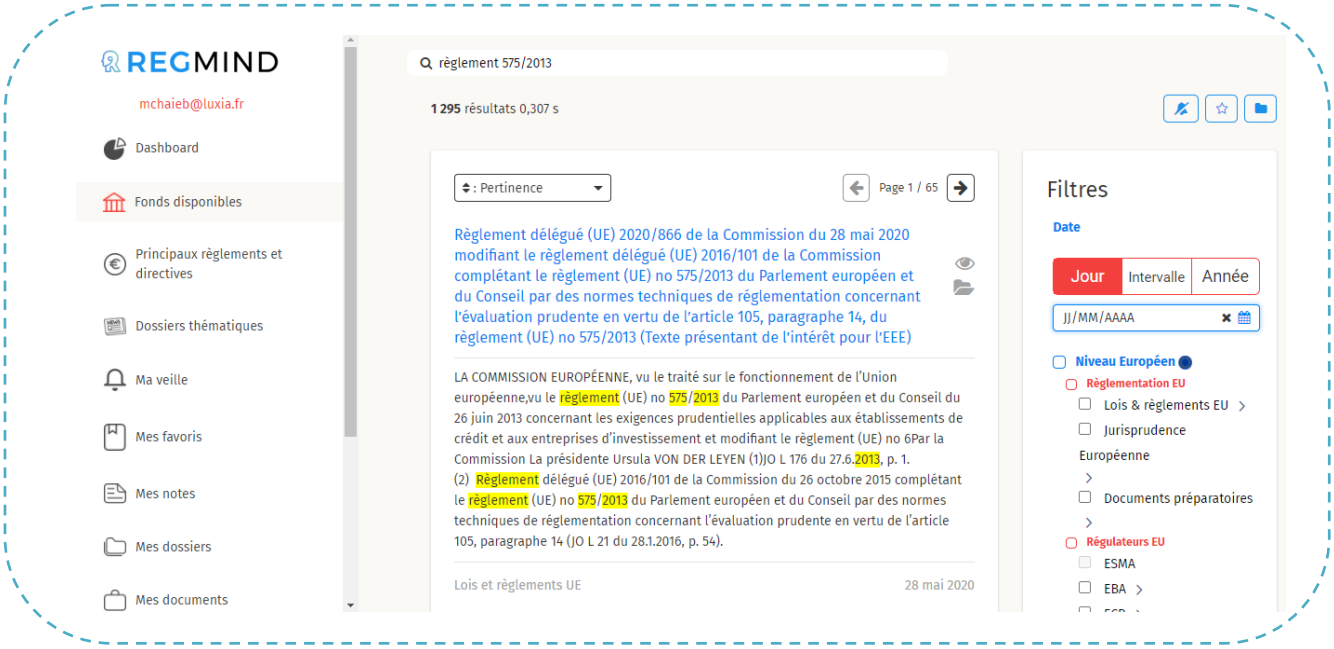


**Figure 3 Résultat de recherche**

Cette interface permet à l'utilisateur de faire une recherche multi-termes pour pouvoir traiter le cas ou il fournit des mots-clés, on parle ici d'une recherche par mot. Et dans un second temps le cas ou l'utilisateur fournit une phrase exacte c'est-à-dire une recherche par phrase.

Dans la figure 3 ci-dessus, l'utilisateur lance une recherche sur le règlement numéro 575/2013, le moteur de recherche répond en offrant 17 résultats concernant ce règlement.

Cette recherche multi-termes nécessite qu'on ordonne les résultats par champs afin de montrer en priorité les documents les plus proches à la demande de l'utilisateur.

Puisque le serveur Legirank indexe des millions des documents, les résultats de la recherche peuvent être trop nombreux, pour cela il faudra pouvoir les filtrer.


**Figure 4 : recherche par facette**

Grâce aux filtres proposés par Regmind, l'utilisateur a toujours la possibilité de faire une recherche par facette[12], en donnant le moyen de filtrer l'ensemble de données en associant des critères et des mots-clefs à sa recherche ou de la fixer sur une date déterminée.

Le but de la recherche par facette est d'optimiser le moteur de recherche, il permet l'utilisateur le pouvoir de sélectionner les critères de sa recherche pour ensuite renvoyer exactement ce qui correspond à ses attentes.

[12] Facette : une technique qui permet d'intégrer des informations à une recherche sous formes des filtres.

La figure 5 ci-dessous illustre les différents filtres disponibles dans le moteur de recherche.

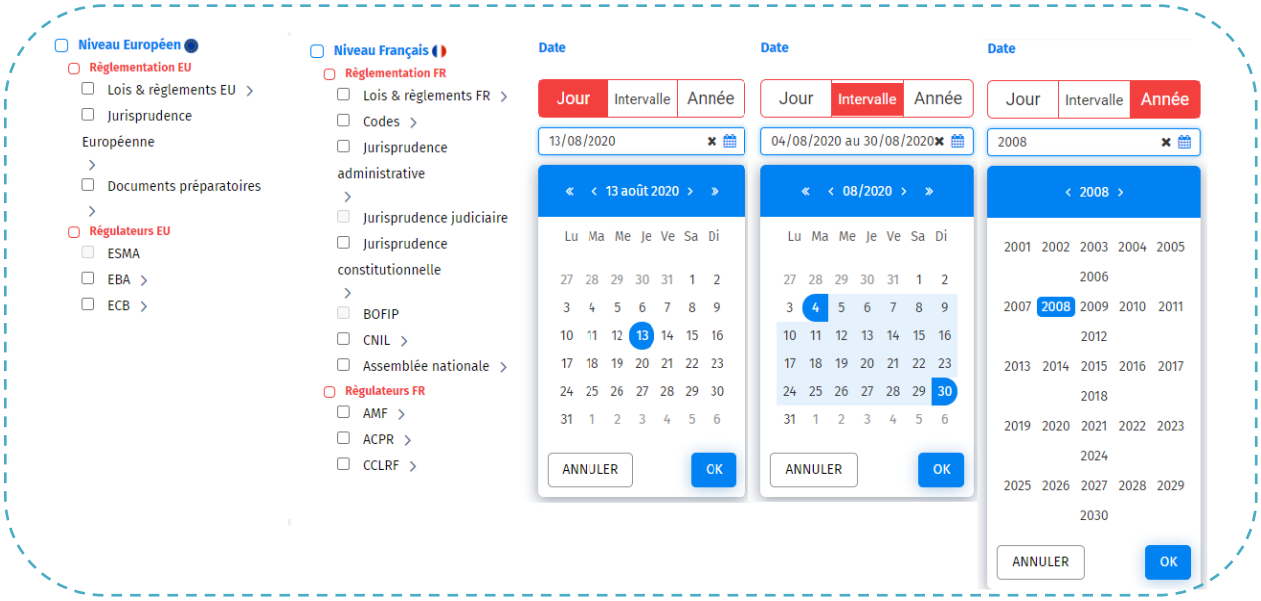


Figure 5 : Filtres disponibles

La recherche par facette demande d'avoir plusieurs champs afin de répondre à toutes les besoins des clients, et éventuellement de traiter différemment certaines entités, notamment :

- Filtrage par fond, français & européen
- Filtrage par type, lois, codes, jurisprudence, etc.
- Sélections des dates soient par jour, année, ou même dans un intervalle de temps.

## 2. Analyses des besoins

### i. Besoins non fonctionnels :

Le système doit être flexible et évolutif afin de répondre aux éventuelles demandes des utilisateurs. De plus il doit fournir des résultats adéquats aux attentes.

Pour garantir une bonne performance du moteur de recherche, notre solution doit s'appuyer sur les deux axes suivants :

- Rapidité : Obtention rapide du bon résultat grâce aux différents algorithmes de recherche.

- Précision et rappel : Un taux de pertinence élevé.

### ii. Besoins fonctionnels :

Les fonctionnalités que le moteur de recherche doit offrir au public se résument dans les points suivants :

- ***Recherche en multi-termes :*** Il s'agit de faire une recherche de divers termes sur plusieurs champs à la fois, notamment sur les titres exacts, contenus exacts, partie du titre ou du contenu et même sur les numéros des documents. La recherche en multi-termes est toujours utilisée dans le processus de la recherche et s'applique sur plusieurs champs indexés.
- ***Trier sur plusieurs champs :*** Le moteur de recherche doit fournir la possibilité de faire le classement du résultat de la recherche par ordre de pertinence en fonction des champs proches à la requête.
- ***Recherche et filtrage par facette :*** Consiste à augmenter les techniques de recherche afin d'affiner les résultats. L'utilisateur possède la faculté d'appliquer plusieurs filtres sur ses requêtes dans le but de fixer sa recherche sur des critères différents.

Quelles que soient les requêtes du public, le système doit effectuer le processus de la recherche sur tous les champs possibles.

Interroger plusieurs champs à la fois augmente la possibilité de trouver la bonne information, donc en garantissant un taux de pertinence et rappel élevé.

Pour que le client trouve facilement ce qu'il veut, le moteur de recherche met à sa disposition plusieurs façons de recherche dont il a besoin.

#### *Les types de recherches :*

- **Recherche par mot**

La recherche par mot va rechercher dans les documents l'existence de ce mot, et renvoyer les résultats classés par pertinences (du document le plus pertinent au moins pertinent) et en deuxième critère le plus récent au plus ancien.

Ce type de recherche s'applique sur le titre ou/et contenu.

- **Recherche par phrase**

La recherche plein texte par phrase permet de recherche plusieurs mots consécutifs devant apparaitre dans le titre, dans le contenu ou dans ses mots-clefs associés.

Cette recherche sur les titres&contenus est considérée comme une recherche exacte, où elle permet de rechercher dans les documents dont les titres&contenus sont les plus proches de la valeur saisie, plus la requête est proche du document, plus le score de ce document est élevé et sera afficher le premier. Ainsi, la recherche des mots de la phrase va rechercher dans les documents l'existence de ces mots, et renvoyer les résultats classés par pertinences et par ordre chronologique.

Ce type de recherche est utilisé éventuellement pour la recherche par numéro, où elle s'applique généralement sur un champs défini (numéro). Mais aussi dans certains cas quand le numéro figure dans le titre ou dans le contenu, on a besoin de faire la recherche sur d'autres champs, notamment le titre ou le contenu, afin d'augmenter la précision

La recherche par numéro permet de retrouver un document juridique faisant référence à un numéro connu, en renvoyant les documents dans l'ordre le plus pertinent et en deuxième critère du document le plus récent au plus ancien.

Exemple des formes des numéros que le moteur de recherche est capable de reconnaître et pour lesquelles il va lancer une recherche précise par n° :

Articles : L465-1, 1183, R123-208-2, L.621-15,…

Numéros de texte: 2017-3, 2019-28

Codes : CSAC1906270S, …

Directives UE:2013/36/UE, …

Règlements UE: 575/2013, …

Numéros d'arrêts ou de pourvois: 18/159091, 17-26974, …

- **Recherche par date**

La recherche par date permet de mettre en tête des résultats les documents ayant comme date un jour, un mois ou une année spécifique par ordre décroissant.

## Conclusion

Dans ce chapitre, nous avons vu les différents besoins dégagés afin de trouver la meilleure solution pour satisfaire le public.

Dans le chapitre suivant, je présente la phase de conception avec l'architecture elasticsearch utilisée.

# Partie 3

# Conception

# III. Chapitre 3. Conception

## Introduction

Ce chapitre est consacré à la phase de conception, une phase cruciale dans le cycle de développement d'un projet. Je présente d'abord le mécanisme général du système d'indexation avec Elasticsearch.

Je détaille ensuite les différents éléments de la conception de la solution proposée.

### 1. L'indexation dans Elasticsearch

Elasticsearch offre un mécanisme pour bien configurer l'index avant sa création, la figure ci-dessous présente les deux étapes que je les ai appliqué pour définir la structure de l'index utilisé.

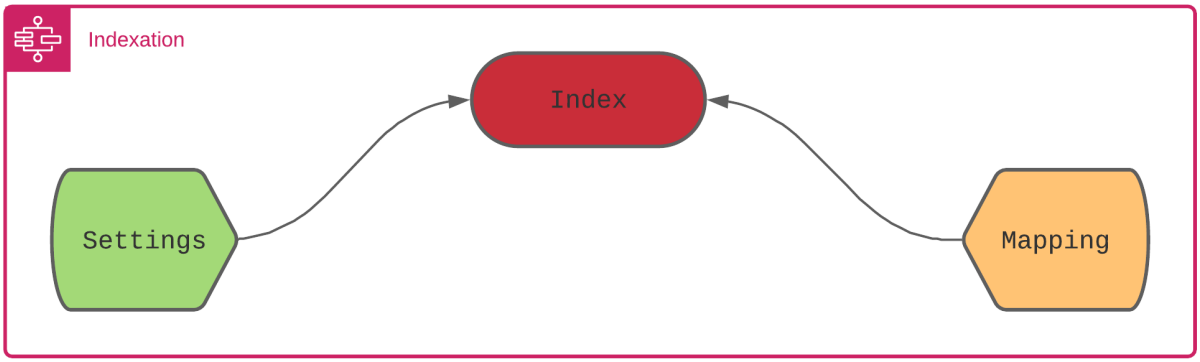


Figure 6 : architecture générale de l'indexation avec Elasticsearch

L'indexe est composé de deux parties :

*Settings* : « Paramètre de définition de l'index, il peut être statique ou dynamique. » [6]

*Mapping* : « Le Mapping est le processus de définition de la manière dont un document et les champs qu'il contient sont stockés et indexés. » [7]

## 2. Processus d'indexation

L'indexation consiste tout d'abord à définir un ensemble d'instructions afin de créer la structure de l'index.

La structure de l'index est composée de deux éléments : Settings et le Mapping, ces deux facteurs sont nécessaires afin d'intégrer les analyseurs utilisés et fixer les types des éléments à indexer.

### I. Settings :

Le paramétrage consiste à définir les traitements qui seront appliqués pour constituer l'index, il comporte la configuration des analyseurs, les tokenizeurs et les filtres appliqués.

***Les analyseurs :***

Dans cette phase, et pour la création de mon index, j'ai configuré deux analyseurs différents :

- ***"analyzer_exact_search"***
  Utilisé pour la recherche par terme exact.
- ***"analyzer_inaccurate_search"***
  Utilisé pour la recherche par terme proche.

Selon les besoins étudiés dans le chapitre II, il est recommandé d'utiliser deux types différents d'analyseurs, l'analyzer_exact_search s'applique dans la recherche par terme exact (par phrase) tandis que l'analyzer_inaccurate_search est consacré à la recherche par mot.

### *Les filtres :*

L'utilisation des filtres diffère d'un analyseur à un autre, pour l'analyseur de la recherche par terme exact (analyzer_exact_search), j'ai utilisé trois filtres :

- ***"lowercase ":*** ce filtre permet de changer les tokens en minuscule.
- ***"french_stop" :*** pour supprimer les mots vides d'un flux de tokens, j'ai utilisé les mots vides par défaut fournis par elasticsearch.
- ***"english_stop"*** **:** pour supprimer les mots vides d'un flux de tokens, j'ai utilisé les mots vides par défaut fournis par elasticsearch.

Concernant la recherche par terme proche, j'ai utilisé les filtres suivants dans l'analyseur analyzer_inaccurate_search :

- **"french_elision"** *:* ce filtre permet de supprimer l'élément vocalique final au début de chaque token.
- **"french_stemmer"** *:* ce filtre qui gère des mots à partir de leurs formes fléchies, il applique une suppression de la forme plurielle, les conjugaisons, et les accords de genre sur le mot.
- **"lowercase"**
- **"french_stop"**
- **"english_stop"**

C'est deux analyseurs utilisent le même tokenizeur que j'ai définit comme suit :

### *Le tokenizeur :*

- ***"my_standard"***
  Tokenizeur de type whitespace, divise le texte en terme, il utilise l'espace comme séparateur.

### *Algorithme de french stemmer :*

Pour le filtre french stemmer, j'ai utilisé l'algorithme french[8].

Cette approche consiste à supprimer les variations morphologiques associées au nombre (singulier vs pluriel), au genre (masculin ou féminin) et à la grammaire.

Etapes de fonctionnement :

J'ai essayé de définir seulement les étapes nécessaires de l'algorithme, le reste de détails est accessible via le lien de la source.

**1- Suppression du suffixe standard**

- Rechercher le suffixe le plus long parmi les suffixes suivants et le supprimer.

Suffixe :

| | | | | | | |
|---|---|---|---|---|---|---|
| ince(s) | ique(s) | isme(s) | able(s) | iste(s) | eux | atrice(s) |
| ateur(s) | ation(s) | logie(s) | usion(s) | ution(s) | ence(s) | ement(s) |
| ité(s) | if(s) | ive(s) | eaux | aux | euse(s) | issement(s) |
| amment | emment | ment(s) | | | | |

Exécuter l'étape 2a si aucune terminaison n'a été supprimée à l'étape 1, ou si l'une des terminaisons suivantes (amment, emment, ment, ments) a été trouvée.

**2a- Les suffixes des verbes commençant par i**

- Rechercher le suffixe le plus long parmi les suffixes suivants et, s'il est trouvé et précédé d'une consonne, le supprimer.

Suffixe :

| | | | | | |
|---|---|---|---|---|---|
| îmes | ît | îtes | i | ie(s) | ir |
| ira(s) | irai | iraient | irais | irait | irent |
| irez | iriez | irions | irons | iront | is |
| issaient | issais | issait | issant(e) | issant(s) | issantes |
| isse | issez | issiez | issions | issons | it |

Effectuer l'étape 2b si l'étape 2a a été effectuée, sans réussir à supprimer un suffixe.

**2b- Autres suffixes verbaux**

- Rechercher le suffixe le plus long parmi les suffixes suivants et effectuer une suppression.

Suffixe :

| | | | | | | |
|---|---|---|---|---|---|---|
| ions | é(e) | ées | és | èrent | er | erai |

| | | | | | | |
|---|---|---|---|---|---|---|
| eraient | erais | erait | era(s) | erez | eriez | erions |
| erons | ez | iez | âmes | ât | âtes | a |
| ai | aient | ais | ant(e) | antes | ants | as |
| asse(s) | assent | assiez | assions | | | |

**3 – Remplacer le dernier Y par i ou le ç final par c**

Si l'étape 3 n'a pas modifié le mot, effectuer l'étape 4

**4 – Suffixes résiduels**

- Si le mot se termine par s, non précédé de a, i, o, u, è ou s, le supprimer.
- Rechercher le suffixe le plus long parmi les suffixes suivants et effectuer une suppression ou un replacement.

Suffixe :

| | | | | | | |
|---|---|---|---|---|---|---|
| ion | ier | ière | Ier | Ière | e | ë |

**5 – Suppression des doubles**

- Si le mot se termine par enn, onn, ett, ell ou eill, supprimer la dernière lettre

**6 – Suppression des accents**

- Si les mots se terminent par é ou è suivis au moins d'une consonne, supprimer l'accent du e.

**Finalement**

- Transformez les lettres I, U et Y restantes du mot en minuscules.

### *Limitation de french stemmer :*

Cet algorithme possède certaines limites au niveau de traitement de la langue française, par exemple si on applique cet algorithme sur les mots va et aller il produit deux racines différentes.

Ces approximations sont acceptables pour le problème traité, parce que cette insuffisance existe sur les verbes or les utilisateurs rarement qu'ils écrivent des requêtes avec des formes verbales.

### II. Mapping

Le mappage, utilisé pour définir la manière d'indexer et stocker les documents et leurs champs.

Ce processus s'applique pour définir quels paramètres (chaine de traitement) sont appliqués à quel champs :

- Définir quels champs textes doivent être traités comme des champs full text ou keyword (mot-clef)
- Spécifier les champs qui contiennent des nombres, des dates ou d'autres types spéciaux avec tous leurs différents formats.
- Spécifier pour chaque champs un analyseur.

***Les Fichiers Settings_1.json et Mapping_1.json dans l'annexe représentent le setting et le mapping que je les créé pour réalisé mon index.***

## 3. Traitement des requêtes

Après avoir créé et rempli l'index de différents documents, il ne reste que traiter les requêtes des utilisateurs.

Avant de commencer cette partie, je définis l'indexation et ses avantages.

Indexer des documents consiste à créer un index contenant ces documents. Il s'agit aussi d'affecter aux documents des indices et des analyses sur leurs contenus. En plus, l'indexation possède le pouvoir d'analyser par le même procédé les textes et les requêtes de l'utilisateur.

Pour un traitement efficace sur les requêtes, il faut savoir auparavant la cible de l'utilisateur.

Le traitement diffère d'une requête à l'autre, en fonction de son contenu, le processus de recherche exécute les meilleures fonctions.

Pour cela, j'ai réalisé beaucoup des tests sur plusieurs types des requêtes fournis par elasticsearch, notamment :

La Multi-match-query, ayant pour but de lancer une recherche dans plusieurs champs à la fois. La façon d'exécuter cette fonction dépend du paramètre *type* de la fonction.

### *Test et Réalisation*

Pour intégrer les requêtes multi-match-query, j'ai utilisé la bibliothèque elasticsearch-dsl[13].

Cette bibliothèque permet de créer des requêtes elasticsearch de façon claire et simple.

*Requêtes développées :*

```
query_title_content = Q ("multi_match", query =value, fields = ["title.proche", "content.proche"], type ="best_fields", analyzer= "analyzer_inaccurate_search", operator = "or")

query_exact_title_content = Q ("multi_match", query =value, fields = ["title.exact", "content.exact"], type ="phrase", analyzer = "analyzer_exact_search", operator = "or")

query_num = Q ("multi_match", query =value, fields = ["num_s ", "num_ss", "nnum_s"], type ="best_fields", analyzer = "analyzer_exact_search", operator = "or")
```

**query_search_final = query_exact_title_content| query_title_content | query_num**

A chaque exécution, le moteur de recherche exécute la requête query_search_final, qui englobe les trois requêtes (query_exact_title_content, query_title_content, query_num).

La concaténation des trois requêtes est une bonne solution pour interroger tous les champs possibles d'un document en une seule fois, ce qui donne lieu à une augmentation de pertinence et rapidité.

---

[13] Elasticsearch-dsl : bibliothèque python qui sert à écrire et exécuter des requêtes elasticsearch.

### *a. Requête proche du contenu ou du titre*

➢ **Exemple de requête**

Query = modification des exigences prudentielles applicables aux établissements de crédit.

Résultat obtenu (*top 3 des documents trouvés)*

**Document 1**

| Champs | similarité par champs | Score tf*idf [14] |
|---|---|---|
| **content.proche** | exigent | 4.04 |
| **content.proche** | établ | 3.74 |
| **content.proche** | prudentiel | 3.40 |
| **content.proche** | modif | 2.76 |
| **content.proche** | cred | 2.68 |
| **content.proche** | applic | 1.65 |
| **title.proche** | modif | 1.46 |

**Score de correspondance : 18.29**

**Document 2**

| Champs | similarité par champs | Score tf*idf |
|---|---|---|
| **content.proche** | exigent | 4.14 |
| **content.proche** | établ | 3.30 |
| **content.proche** | prudentiel | 2.81 |
| **content.proche** | modif | 2.39 |
| **content.proche** | cred | 2.21 |
| **content.proche** | applic | 2.15 |
| **title.proche** | modif | 1.25 |

**Score de correspondance : 17.04**

**Document 3**

| Champs | similarité par champs | Score tf*idf |
|---|---|---|
| **content.proche** | prudentiel | 4.68 |
| **content.proche** | exigent | 3.97 |
| **content.proche** | établ | 3.03 |
| **content.proche** | cred | 2.07 |
| **content.proche** | modif | 1.94 |
| **content.proche** | applic | 1.12 |

**Score de correspondance : 16.85**

[14] Tf-idf : term frequency–inverse document frequency, score qui détermine le poids d'un mot dans un corpus.

➢ **Synthèse**

D'après les résultats obtenus, on remarque qu'aucun document n'a matché l'intégralité de la requête, donc le système a fait une recherche proche sur les termes.

D'autre part, on remarque aussi que les documents ayant un score maximal sont en tête de la liste. Ce classement est du au calcul des scores effectué sur les champs. Le *document1* inclut le maximum de termes proches de la requête ce qui explique que le score de correspondance (la somme des poids calculée) est élevé par rapport aux autres documents.

Le calcul de score d'un terme est calculé en fonction de la fréquence de ce terme dans le corpus et le nombre de fois où il apparait dans le document.

Plus les termes de la requête apparaissent dans le document, plus le score de ce document augmente et prend la tête de la liste.

Remarque : Pour augmenter manuellement le score d'un document, on peut ajouter des poids à chaque champs, ce qui influence la valeur de TF*IDF.

Exemple : fields= ["title.proche^5", "content.proche^2"]

### b. Requêtes exactes du contenu ou du titre

➢ **Exemple de requête**

Requête 1 : Modifications des règles de fonctionnement de la chambre de compensation et du système de règlement livraison.

Requête 2 : L'activité des conseillers en investissements financiers.

➢ **Résultat obtenu**

***Cas requête 1***

**Document1**

| Champs | similarité par champs | Score tf*idf |
|---|---|---|
| **title.exact** | modifications des règles de fonctionnement de la chambre de compensation et du système | 18.08 |

| | | |
|---|---|---|
| | de règlement livraison | |
| **title.proche** | regl | 4.87 |
| **content.proche** | modif | 2.69 |

Score de correspondance : 46.47

Document2

| Champs | similarité par champs | Score tf*idf |
|---|---|---|
| **title.exact** | modifications des règles de fonctionnement de la chambre de compensation et du système de règlement livraison | 14.95 |
| **title.proche** | chambr | 2.13 |
| **content.proche** | regl | 5.85 |

Score de correspondance : 44.96

*Cas requête 2*

Document1

| Champs | similarité par champs | Score tf*idf |
|---|---|---|
| **content.exact** | l'activité des conseillers en investissements financiers | 18.39 |
| **title.proche** | activ | 5.47 |
| **title.proche** | invest | 4.22 |

Score de correspondance : 36.64

Document2

| Champs | similarité par champs | Score tf*idf |
|---|---|---|
| **title.proche** | activ | 5.47 |
| **title.proche** | invest | 4.28 |
| **content.proche** | financi | 2.57 |

Score de correspondance : 18.25

- **Synthèse**

D'après les résultats de la requête 1, le moteur de recherche à réussi à matcher exactement la requête avec les deux titres de deux documents, donc cette fois-ci le système a fait une recherche exacte sur les termes.

Egalement, on remarque aussi une correspondance dans les titres proches et la requête, c'est tout à fait normal puisque le terme proche est inclut dans le terme exact.

Pour le deuxième cas de recherche, la similarité par champs exacte est apparue seulement dans le premier résultat, le premier document possède un score de 36.64 de similarité où la phrase passée par la requête est inclues exactement dans son contenu. Tant que le deuxième document qui arrive avec un score de 18.25 c'est presque la moitié contient seulement quelques termes de la requête, ce qui explique la chute du score.

Depuis ces résultats, on peut conclure que le moteur de recherche réussit toujours à donner des bons résultats à l'utilisateur, quelque soit la valeur passée dans la requête, il y aura dans tous les cas un retour positif.

### c. Requêtes sur les numéros

C'est le même principe, cette requête s'exécute toujours avec les deux requêtes précédentes.

Son objectif est de renvoyer les documents dont le numéro figure dans la recherche.

La seule différence c'est que les champs numéros utilisent toujours un seul analyseur, l'**analyzer_exact_search.** L'application de cet analyseur dans ce type de recherche est nécessaire, son objectif est de matcher exactement la requête comme elle est.

Une recherche proche sur les numéros peut provoquer une perte de précision, dans le cas où l'utilisateur tape par exemple une recherche du numéro ''r125-3'', cette dernière va renvoyer sur les documents où ''r125'' et/ou ''3'' existent dans leurs contenus, titres ou numéros.

- **Exemple de requête**

  Requête = Article r125-3

- **Résultat obtenu**

| Champs | similarité par champs | Score tf*idf |
|---|---|---|
| **title.exact** | article r125-3 | 8.43 |
| **title.proche** | r125-3 | 6.79 |
| **num_s** | r125-3 | 4.52 |

- **Synthèse**

  Comme nous remarquons ici, le processus de recherche a renvoyé dans la tête de liste le document dont son numéro = r125-3.

  Le moteur de recherche a combiné les résultats de trois requêtes ensembles (query_exact_title_content, query_title_content, query_num) où il a exécuté une recherche exacte sur le titre avec un score de similarité égale à 8.43, aussi une recherche proche de ''r125-3'' sur le même champs, et une recherche sur le numéro avec un score de 4.52.

  La recherche exacte sur le titre possède le score le plus élevé parce que le champs titre contient exactement la valeur passée par la requête. Si la requête était seulement ''r125-3'', on trouvera le champs num_s en tête de la liste avec le score qui pèse le plus.

### d. Requêtes basées sur les facettes

Ce type de requête s'exécute sous la forme de filtres ajoutés au résultat.

Selon les besoins spécifiés, l'utilisation des facettes est utile dans le cas d'une recherche par date, par fond ou par nature, le moteur de recherche exécute le processus de filtrage à chaque demande.

**Implémentation :**

Avant d'entrer dans le processus de recherche proche ou exacte, le système vérifie s'il en a des filtres ajoutés par l'utilisateur ou pas.

Si aucun filtre n'a été détecté, le moteur lance une recherche seulement sur la valeur passée par la requête.

Sinon, et avant de lancer la recherche, le système prend les valeurs sélectionnées sur la facette (Figure 5) et les passe dans le constructeur de la classe *DocSearchFacets*

```
Search = DocSearchFacets

(self.es_client,self.index,params_values,self.lang,filters=fil
tres)
```

Ensuite, après le filtrage, il lance la recherche sur la requête.

### e. Requête multilingue

Le but de réaliser ce type de requête, est de montrer les différents documents possibles en fonction de la langue souhaitée par l'utilisateur. Pour cela j'ai travaillé sur deux possibilités :

**Option 1 : Deux indexes séparés par langue :**

Avant l'indexation, il s'agit de créer deux indexes, un index anglais et un index français dont chacun utilise respectivement un analyseur anglais et un analyseur français. L'indexation des documents dans deux indexes séparés se fait en fonction de chaque langue du document.

Lors d'une recherche, le système va déclencher d'abord l'algorithme de détection de langue, puis en fonction de la langue détectée, le moteur de recherche va changer d'index.

Ce traitment que je l'ai développé utilise un processus d'inférence d'un pipeline d'ingestion qui fait référence à un modèle d'identification de langue.

Pour ce traitement il a fallu utiliser le model *lang_ident_model_1* qui prend la valeur de la requête, puis à l'aide d'un pipeline elasticsearch il exécute le processeur de détection sur cette valeur puis il renvoi la langue qui possède le score le plus élevé.

Le processeur de détection de la langue proposé par elasticsearch est composé d'une phase de modélisation et une phase de classification.

La création du modèle (modélisation)  sert à identifier et stoker les mots ou les N-grammes les plus fréquents.

La deuxième étape, celle de la classification s'agit de comparer le document d'entrés aux différents modèles de langue de référence.

- Point fort :

Avec cette idée, la précision sur les résultats sera élevée, puisque le moteur de recherche va changer automatiquement d'index en fonction de la langue, il aura donc uniquement les documents de la langue détectée.

Prenons l'exemple cité dans la problématique, la recherche de ''commercial commission'' au début avec un seul index a renvoyé sur des documents en français et en anglais à la fois, sans prendre compte de la langue.

Par contre, dans le cas d'une recherche sur deux indexes différents, le moteur de recherche a réussi (après un test effectué sur cette phrase) à montrer des résultats qui correspondent à la langue de la requête en changeant d'index (index anglais).

```
en, 0.8767139254372106 index= en
```

- Point faible :

Malgré son efficacité niveau précision, cette solution possède une insuffisance non négligeable, c'est le cas d'une recherche par numéro.

En effet le processeur de détection de langue ne va pas reconnaitre la bonne langue de la requête si elle contient seulement un numéro, il va renvoyer un résultat quelconque qui peut être français, anglais, ou n'importe quelle autre langue. Dans ce cas, le moteur de recherche va changer son index en fonction d'une langue non fiable, ce qui va entrainer une recherche dans un autre index où le numéro demandé ne figure jamais.

Exemple : le numéro ‘’r125-3’’ existe seulement dans l’index français, et le processeur de détection de la langue a demandé un changement d’index vers l’anglais.

**Option 2 : un seul index avec deux différents analyseurs de langue :**

Cette solution consiste à définir un analyseur par langue. L’idée est comme suit :

Pour chaque champs qui peut être en deux langues différentes (contenu ou titre), création de :

- Un sous champs français qui pointe sur le champs père avec un analyseur français.
- Un sous champs anglais qui pointe sur le même champs père mais avec un analyseur anglais.

Au moment de l’indexation, un seul index va contenir tous l’ensemble des documents de différentes langues.

Ensuite pendant la recherche, le changement sera au niveau des analyseurs. L’échange des analyseurs se fait en fonction de la langue de la requête. Si l’utilisateur tape une requête en français, le moteur de recherche exécute l’analyseur français, si la requête est en anglais, l’analyseur anglais sera exécuté.

Comme il est bien indiqué dans la partie annexe Mapping_2, chaque sous-champs possède un seul analyseur, donc pour changer d’un analyseur à l’autre, il suffit de changer l’ensemble des champs à interroger.

- Point fort :

Cette solution garantit toujours un score de correspondance élevé. Les sous-champs seront dans tous les cas interrogés ensembles par la requête, donc le moteur de recherche quelle que soit la requête de l’utilisateur il trouvera toujours une similarité soit en français ou en anglais.

Grace à cette idée, la précision sur les documents sortants sera élevée, également pour le rappel où le moteur de recherche renvoi toujours les articles proches de la requête.

- Points à améliorer

Afin de rendre cette solution efficace à 99.99%, il faut réaliser plusieurs tests avec des documents de langues différentes. Pour l'instant j'ai testé cette solution sur un ensemble de documents réduit (environs 720 articles).

# Partie 4

# Conclusion générale et perspectives

# IV. Chapitre 4. Conclusion et perspective

Mon travail consiste à internationaliser le moteur de recherche de Luxia qui m'a donné l'opportunité de travailler sur un sujet complet depuis la phase de conception jusqu'au la phase de réalisation de la solution.

Pour réaliser ce projet, il a fallu étudier différentes solutions et les comparer pour aboutir au choix de la solution appropriée « indexation et recherche multilingues ».

Ce choix se justifie par l'avantage de deux options possibles : une garantie d'un taux de correspondance élevé.

Ce projet m'a permis d'améliorer mes connaissances théoriques acquises durant mes deux ans d'études universitaires à l'université Grenoble Alpes, ainsi que la maitrise de nouveaux outils d'indexation et de programmation. Il a été une expérience bénéfique pour moi, car il m'a initié à la vie professionnelle.

On peut conclure que le travail effectué, dans le cadre de ce projet de fin d'études m'a été bénéfique sur plusieurs niveaux :

En premier lieu, cette étude m'a permis d'enrichir mes connaissances dans le domaine d'indexation, et de découvrir ces nouvelles notions.

En deuxième lieu, ce projet m'a permis d'acquérir une expérience théorique et pratique de grande valeur, vu qu'il se présente comme une occasion de prendre la responsabilité de l'étude et du développement d'un outil d'indexation.

En troisième lieu, il m'a permis de savoir l'importance de la phase d'analyse d'un tel projet informatique et surtout l'importance de la phase de conception.

Comme tout travail, ce travail n'est pas exhaustif, il manque certaines fonctionnalités à améliorer à l'avenir.

L'utilisation d'Elasticsearch pour cette solution est encore étendue, il existe des points à améliorer, notamment :

- Le détecteur de la langue, dans certaines langues, le modèle de détection est assez précis sur des textes courts, mais il est plus difficile à identifier la langue sur des courts flux pour les langues similaires.
- Le stemmer utilisé au niveau d'analyseur par terme proche, il possède certaines insuffisances dans certains cas, même elles sont négligeables dans notre cas, mais ça reste toujours un point à améliorer.
- Chercher une troisième possibilité de recherche multilingue, penser à interroger simultanément deux indexes à la fois.

Ce projet a été une expérience bénéfique du point de vue des relations humaines car j'ai appris à travailler dans une équipe disciplinée et dans un environnement de travail réel et efficace.

# Bibliographie


[1] R. Dale, « Law and Word Order: NLP in Legal Tech », *Nat. Lang. Eng.*, vol. 25, n° 1, p. 211‑217, janv. 2019, doi: 10.1017/S1351324918000475.

[2] « CNIL | ». https://www.cnil.fr/fr (consulté le sept. 04, 2020).

[3] « Légifrance », *Wikipédia*. juill. 22, 2020, Consulté le: sept. 04, 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=L%C3%A9gifrance&oldid=173164408.

[4] « Autorité des marchés financiers (France) », *Wikipédia*. août 22, 2020, Consulté le: sept. 04, 2020. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Autorit%C3%A9_des_march%C3%A9s_financiers_(France)&oldid=174029208.

[5] « Indexation et Index Google : Comment ça Marche ? | Optimiz.me », *Optimiz*. https://optimiz.me/indexation-comprendre-le-fonctionnement-de-lindex-google/ (consulté le sept. 03, 2020).

[6] « Index modules | Elasticsearch Reference [7.9] | Elastic ». https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules.html#index%20modules-settings (consulté le sept. 04, 2020).

[7] « Mapping | Elasticsearch Reference [7.9] | Elastic ». https://www.elastic.co/guide/en/elasticsearch/reference/7.9/mapping.html (consulté le sept. 04, 2020).

[8] « French stemming algorithm ». http://snowball.tartarus.org/algorithms/french/stemmer.html (consulté le sept. 03, 2020).

# Table des annexes

# Annexe 1
# Settings_1.json

```
{
"settings": {"index.mapping.ignore_malformed": "true",
            "analysis": {
                        "analyzer": {
                                    "analyzer_exact_search": {
                                                "type": "custom",
                                                "tokenizer":"my_standard",
                                                "filter": [
                                                "lowercase",
                                                "french_stop",
                                                "english_stop"]
                                    },
                                    "analyzer_inaccurate_search":{
                                                "type": "custom",
                                                "tokenizer": "my_standard",
                                                "filter": ["lowercase",
                                                "french_elision",
                                                "french_stop",
                                                "french_keywords",
                                                "french_stemmer"]
                                    }
                        },
            "tokenizer": {
                        "my_standard":{
                                    "type": "char_group",
                                    "tokenize_on_chars": ["whitespace","\n"],
                                    "max_token_length": 5}
                        },

            "filter": {
                        "english_stop": {
                                    "type": "stop",
                                    "stopwords": "_english_"},

                        "french_stop": {
                                    "type": "stop",
                                    "stopwords": "_french_"},

                        "french_elision": {
                                    "type": "elision",
                                    "articles_case": "true",
                                    "articles": ["l", "m", "t", "qu", "n", "s", "j", "d", "c", "jusqu", "quoiqu", lorsqu",
"puisqu"]},
                        "french_stemmer": {
                                    "type": "stemmer",
                                    "language": "french"},
            }
          }
       }
}
```

# Annexe 2
# Mapping_1.json

```
{
              "mappings": {
 "properties": {
      "content": {
            "type": "text",
            "fields":{
                        "proche":{
                                    "type":"text",
                                    "analyzer":"analyzer_inaccurate_search" },
                        "exact":{
                                    "type":"text",
                                    "analyzer":"analyzer_exact_search" }
                        }
  },

  "title":{
            "type": "text",
            "fields":{
                        "proche":{
                        "type":"text",
                        "analyzer":"analyzer_inaccurate_search" },
                        "exact":{
                        "type":"text",
                        "analyzer":"analyzer_exact_search" }
            }
  },

  "num_s": {
    "type": "text",
    "analyzer": "analyzer_exact_search"
  },
  "num_ss": {
    "type": "text",
    "analyzer": "analyzer_exact_search"
  },
  "nnum_s": {
    "type": "text",
    "analyzer": "analyzer_exact_search"
     },
  "dt": {
    "type": "date",
    "format": "d MMMM yyyy|| MMM y ||MM-yyyy ||MM/yyyy||yyyy-MM||dd-MM-yyyy||yyyy-MM-dd
    "locale": "fr"
  },
  "id":{
    "type":"long"
  },
  "nature_s":{
     "type":    "keyword"
  },
```

```
    "corpus":{
      "type":     "text",
      "analyzer": "analyzer_exact_search",
    "fielddata": true
    }
   }
  }
}
```

# Annexe 3
# Settings_2.json

```
{
 "settings": {"index.mapping.ignore_malformed": "true",
        "analysis": {

          "analyzer": {
            "fr_analyzer_exact_search": {
              "type": "custom",
              "tokenizer":"my_custom_char_group",
              "filter": [
                "lowercase",
                "english_stop", ]
            },

            "fr_analyzer_inaccurate_search":{
              "type": "custom",
              "tokenizer": "my_standard",
              "filter": ["french_elision",
                      "lowercase",
                      "french_stop",
                      "french_keywords",
                      "french_stemmer"
                      ]
            },
            "en_analyzer_exact_search": {
              "type": "custom",
              "tokenizer":"my_custom_char_group",
              "filter": [
                "lowercase",
                "english_stop",
                "french_stop"
              ]
            },

            "en_analyzer_inaccurate_search":{
              "type": "custom",
              "tokenizer": "my_standard",
              "filter": [
                      "lowercase",
                      "english_stop",
                      "english_keywords",
                      "english_stemmer",
                      "english_possessive_stemmer"
                      ]
            }
          },
          "tokenizer": {

            "my_custom_char_group":{
              "type":"char_group",
              "tokenize_on_chars":[
                "whitespace"
              ]
```

```
            },
            "my_standard":{
              "type": "char_group",
              "tokenize_on_chars": [
                "whitespace",
                "\n"
              ],
              "max_token_length": 5
            }
          },
          "filter": {
            "english_stop": {
              "type": "stop",
              "stopwords": "_english_"
            },
            "french_stop": {
              "type": "stop",
              "stopwords": "_french_"
            },

             "english_stemmer": {
              "type": "stemmer",
              "language": "english"
             },
            "english_keywords": {
              "type": "keyword_marker",
              "keywords": []
            },
            "english_possessive_stemmer": {
             "type":       "stemmer",
             "language":   "possessive_english"
            },

             "french_elision": {
                "type": "elision",
                "articles_case": "true",
                "articles": ["l", "m", "t", "qu", "n", "s", "j", "d", "c", "jusqu", "quoiqu", "lorsqu", "puisqu"]

             },
             "french_stemmer": {
              "type": "stemmer",
              "language": "french"
             },
            "french_keywords": {
              "type": "keyword_marker",
              "keywords": []
            }

          }
        }
    }
```

# Annexe 4
# Mapping_2.json

```
{
"mappings": {

  "properties": {

    "content": {
      "type": "text",
       "fielddata": true,
      "fields": {
        "en": {
          "type": "text",
          "fielddata": true,
          "fields":{
           "exact":{
             "type":"text",
             "analyzer":"en_analyzer_exact_search",
             "fielddata":true
           },
           "proche":{
             "type":"text",
             "analyzer":"en_analyzer_inaccurate_search",
             "fielddata":true
           }
          }
        },
        "fr": {
          "type": "text",
          "fielddata": true,
          "fields":{
           "exact":{
             "type":"text",
             "analyzer":"fr_analyzer_exact_search",
             "fielddata":true
           },
           " proche ":{
             "type":"text",
             "analyzer":"fr_analyzer_inaccurate_search",
             "fielddata":true
           }
          }
        }
     }

    },



    "title": {
      "type": "text",
       "fielddata": true,
      "fields": {
        "en": {
```

```
        "type": "text",
        "fielddata": true,
        "fields":{
         "exact":{
          "type":"text",
          "analyzer":"en_analyzer_exact_search",
          "fielddata":true
         },
         " proche ":{
          "type":"text",
          "analyzer":"en_analyzer_inaccurate_search",
          "fielddata":true
         }
        }
      },
      "fr": {
        "type": "text",
        "fielddata": true,
        "fields":{
         "exact":{
          "type":"text",
          "analyzer":"fr_analyzer_exact_search",
          "fielddata":true
         },
         " proche ":{
          "type":"text",
          "analyzer":"fr_analyzer_inaccurate_search",
          "fielddata":true
         }
        }
      }
   }

},
"num_s": {
  "type": "text",
  "analyzer": "fr_analyzer_exact_search",
   "fielddata": true
},
"num_ss": {
  "type": "text",
  "analyzer": "fr_analyzer_exact_search",
   "fielddata": true
},
"nnum_s": {
  "type": "text",
  "analyzer": "fr_analyzer_exact_search",
   "fielddata": true
},
"dt": {
  "type": "date",
  "format": "d MMMM yyyy|| MMM y ||MM-yyyy ||MM/yyyy||yyyy-MM||dd-MM-yyyy,
  "locale": "fr",
  "ignore_malformed": "false"
},
"id":{
  "type":"long"
},
"nature_s":{
```

```
      "type":     "keyword"
    },
    "corpus":{
      "type":     "text",
      "analyzer": "fr_analyzer_exact_search",
    "fielddata": true
    }
  }
 }

}
```

# Annexe 5

## Détailles sur le calcul de score (tf-idf)

Le tf-idf est proportionnel au nombre de fois que le terme apparu dans le document et il est compensé par le nombre totale des documents dans le corpus.

Idf = log (1 + (N - n + 0.5) / (n + 0.5)), où :

- n : nombre des documents contenant le terme.
- N : nombre totale des documents.

Tf = freq / (freq + k1 * (1 - b + b * dl / avgdl)), où :

- freq : occurrence du terme dans le document.
- k1 : paramètre de saturation du terme.
- b : paramètre de normalisation de langueur.
- dl : longueur du champ (approximative)

  avgdl : longueur moyenne du champ.