Département Informatique
Promotion 2021-2022

Développement d'un moteur de recherche pour la documentation de l'équipe de développement

Rapport de stage présenté en septembre 2022
par

Jonathan MAROTTA

en vue de l'obtention du DUT Informatique

Stage effectué à efluid SAS

2bis, rue Ardant du Picq
CS 10100
57004 METZ CEDEX

Remerciements

Je tiens à remercier :

Francis GROSMANGIN pour la validation de mon stage au sein d'efluid SAS,

Alexis DMYTRYK pour ma prise en charge dans son secteur et la confiance qu'il m'a accordée en me proposant la thématique extrêmement intéressante de ce stage, ainsi que pour sa réponse appropriée à mon appétence à relever de nouveaux défis avec l'élargissement de la thématique initiale,

Fabien SACKSTEDER pour ses conseils avisés ainsi que sa démarche pédagogique sans faille, sans oublier le partage de son expertise et de sa passion dans une bonne humeur quotidienne,

Florian IGGIOTTI pour la soumission de ma candidature chez efluid SAS,

L'accueil général et la bienveillance de la part de l'ensemble des collaborateurs avec qui j'ai pu avoir un contact.

Préambule

Les mots suivis d'un astérisque (*) sont explicités dans le glossaire disponible à la fin de ce rapport.

Sommaire

Abstract

When access to a resource of information is complicated by the proliferation of distribution channels and with the sharp increase in volume of data, the development of a search engine is a solution that responds perfectly to this problem. Such a project revolves around three axes.

The first with the creation of a Web crawler, whose goal is to search for resources to be indexed. The implementation of several Web crawlers becomes necessary to deal with the specifications of each of the sources of information. Controlling their launch in multi-threading guarantees the speed and robustness of the application.

The second step is indexing the content. By choosing to use Apache Solr, we're using a stable and powerful tool to obtain personalized behavior. The configuration of a query analyzer that enables to use the full power of the Lucène syntax, or even the implementation of functionalities intended to assist the user (spelling correction and suggestion of keywords in real time, generation of relevant snippets, etc.) are some of the parameters to take into consideration before deploying Solr.

Finally, the implementation of a simple and elegant user interface provides an effective presentation of search results. An administration page allows remote control of the launch of crawlers, let us know their last successful launch date, or even offer the downloading of an audit report containing information on the use of the tool by users.

Introduction

La société efluid SAS, filiale d'UEM, la CDC (Caisse des Dépôts et Consignations) et enedis, développe depuis 2002 une solution progicielle éponyme à destination des entreprises productrices et gestionnaires d'énergie en France et à l'étranger.

L'outil ne cesse depuis lors d'être amélioré pour répondre aux besoins croissants de ses clients. Dans le même temps, les équipes de développement ont généré un ensemble de documentation interne visant à aider les membres des différentes équipes sur la multitude de thématiques et domaines d'applications relatifs au développement d'efluid.

L'accroissement du volume de ces données associé à un accès compliqué par la présence de nombreux canaux de diffusion rendent cependant cette ressource sous-exploitée.

Le problème étant récurrent depuis plusieurs années, la société souhaite développer un moteur de recherche interne permettant de valoriser la documentation existante et ainsi aider les développeurs en leur offrant un système de recherche efficace.

1. Présentation de l'entreprise

UEM (anciennement Usine d'Électricité de Metz) est un producteur et fournisseur français d'électricité et de gaz présent sur le marché messin depuis 1901, et sur le reste du territoire français via sa filiale Énergem depuis 2009.

C'est historiquement une entreprise locale de distribution d'électricité opérant sur un périmètre de 141 communes autour de la ville de Metz (Moselle), son actionnaire majoritaire.

Elle emploie plus de 300 personnes. Son siège social se situe au 2 place du Pontiffroy à Metz en Moselle depuis 1980.

Photographie n°1 : le siège social de la société UEM à Metz

1.1. Le groupe UEM

Le 1er janvier 2008, l'usine d'électricité de Metz (UEM) devient une société anonyme d'économie mixte locale (SAEML) et transfère ses activités de gestion des réseaux de distribution d'électricité dans une filiale dénommée Réséda pour respecter une directive européenne de 2007 imposant la séparation juridique des activités de distribution dans une entreprise intégrée.

S'en suit la création de plusieurs autres filiales :

- Réséda : gestionnaire des activités de gestion des réseaux de distribution d'électricité sur Metz et 141 communes environnantes. Il alimente plus de 180 000 points de services.

- Énergem : commercialisant l'électricité du groupe dans l'ensemble du territoire français, hors du périmètre d'UEM.

- Efluid SAS : développe le progiciel* efluid. Dans un premier temps pour l'usage unique d'UEM, puis à destination des grands acteurs de l'énergie en France et à l'étranger.

- Énergreen production : chargée du développement de la production d'énergie renouvelable.

1.2. La filiale efluid SAS

Photographie n°2 : bâtiment de la société efluid SAS à Metz

En charge du développement du progiciel efluid, elle regroupe plus de 200 personnes organisées dans plusieurs branches fonctionnelles.

Figure n°1 : Architecture fonctionnelle d'efluid

L'ensemble des fonctions présentes dans efluid a été conçu dans une logique de composants. Ainsi, lorsqu'un client exprime un nouveau besoin, celui-ci n'est pas implémenté directement, mais rendu possible grâce aux outils de paramétrage. Ce sont les composants qui évoluent, et qui offrent à chaque client d'efluid SAS une personnalisation de l'outil selon ses besoins.

1.3. Service composants transverses

Le service dans lequel j'ai été accueilli développe des composants qui seront utilisés par plusieurs autres services d'efluid SAS. Les briques logicielles développées doivent répondre aux besoins de tous les acteurs concernés.

2. Problématique

2.1. Un volume de données conséquent et la multiplication des canaux de diffusion

L'équipe de développement d'efluid SAS dispose d'un volume de documentation interne conséquent (plusieurs milliers de documents uniques). Créée par le personnel d'efluid SAS, cette ressource est disponible sur le réseau interne de l'entreprise via plusieurs sites internet.

La documentation est globalement constituée de pages Web même s'il existe aussi des fichiers .pdf, .docx, ... Chaque site propose cependant un style graphique propre, une arborescence parfois complexe avec des profondeurs de page* qui peuvent être élevées (fréquemment supérieure à 3, cf. Annexes A à G). Autant d'éléments qui contribuent à l'augmentation de la difficulté lors de la recherche d'une information.

S'il est relativement simple de créer de l'information, on se rend compte qu'il est toujours plus difficile de la traiter et de la rechercher. La multiplication des sources (structurées et non structurées) et des canaux de diffusion, ainsi que la forte augmentation des volumes d'information et de données compliquent l'accès à la ressource.

Comme dans 38% des sociétés (Arcalys, 24/02/2015), les employés d'efluid SAS trouvent difficile de retrouver un document dans le système d'archivage interne. Des études (LinkedIn, 15/05/2017) montrent qu'environ 20 à 30% du temps d'activité d'un collaborateur est gaspillé à la recherche d'informations : un manque à gagner pour l'entreprise.

Un effet pervers s'ajoute encore à cela : les rares personnes ayant la connaissance de la documentation (parce qu'ils ont participé à sa rédaction et sont donc conscients des processus de génération, de l'architecture de la documentation, des différents moyens d'accès, ou tout simplement de l'existence de telle ou telle ressource, ...) sont systématiquement sollicitées pour réussir à l'obtenir plus facilement.

2.2. Le besoin d’un moteur de recherche efficace

Des tentatives d'amélioration de l’accès à la documentation ont bien été mises en place. Nécessitant la participation de chacun, très peu de retours ont été obtenus (il est déjà difficile d’obtenir un retour par mail lorsqu’un collaborateur touché par le problème est sollicité sur le problème qu’il a rencontré).

La solution envisagée est alors la mise en place d'un moteur de recherche permettant la recherche simplifiée depuis un point d'accès unique, et ce sans que l’utilisateur ne sache à l'avance où la documentation se trouve, dans quelle base, dans quel applicatif ou sur quel lecteur réseau.

La recherche doit fournir des résultats les plus pertinents possibles par défaut, mais aussi permettre l'élaboration de requêtes complexes ou personnalisées grâce à l'utilisation d'une syntaxe particulière (*i.e.* : syntaxe Lucène* et utilisation de wildcards*) afin d'affiner encore la recherche.

De même, intégrer des améliorations de l'expérience utilisateur paraît intéressant (correcteur orthographique et suggestions de recherche "real-time"). *In fine* l’application sera ajoutée aux outils de développement et devra par conséquent respecter la charte graphique de l’entreprise.

Doivent également se poser des questions relatives à :

- La pérennité / robustesse :
    - Si l'utilisation de solutions externes doit être envisagée, depuis combien de temps existent-elles ?
    - Est-ce que leur développement est actif ?
    - Est-ce que le nouvel outil sera à même de prendre en charge l’intégration de la totalité de la documentation ?
- La rapidité :
    - Temps de réponse moyen lors de la recherche par un utilisateur ?
    - Durée pour crawler*/scraper/indexer la totalité de la documentation disponible sur l'intranet ?
- La bande passante : est-ce que l'outil peut poser un risque de dégradation de la performance du réseau interne ?
- La pertinence : qualité des résultats ?

Lorsque le moteur de recherche sera implémenté, il serait intéressant de pouvoir tracker l'utilisation qui en est faite :

- Qui utilise la ressource ?
- Quel volume / fréquence ?
- Quelle ressource sert le plus ?

Ceci dans le but d'améliorer encore la documentation elle-même, son accès, son usage.

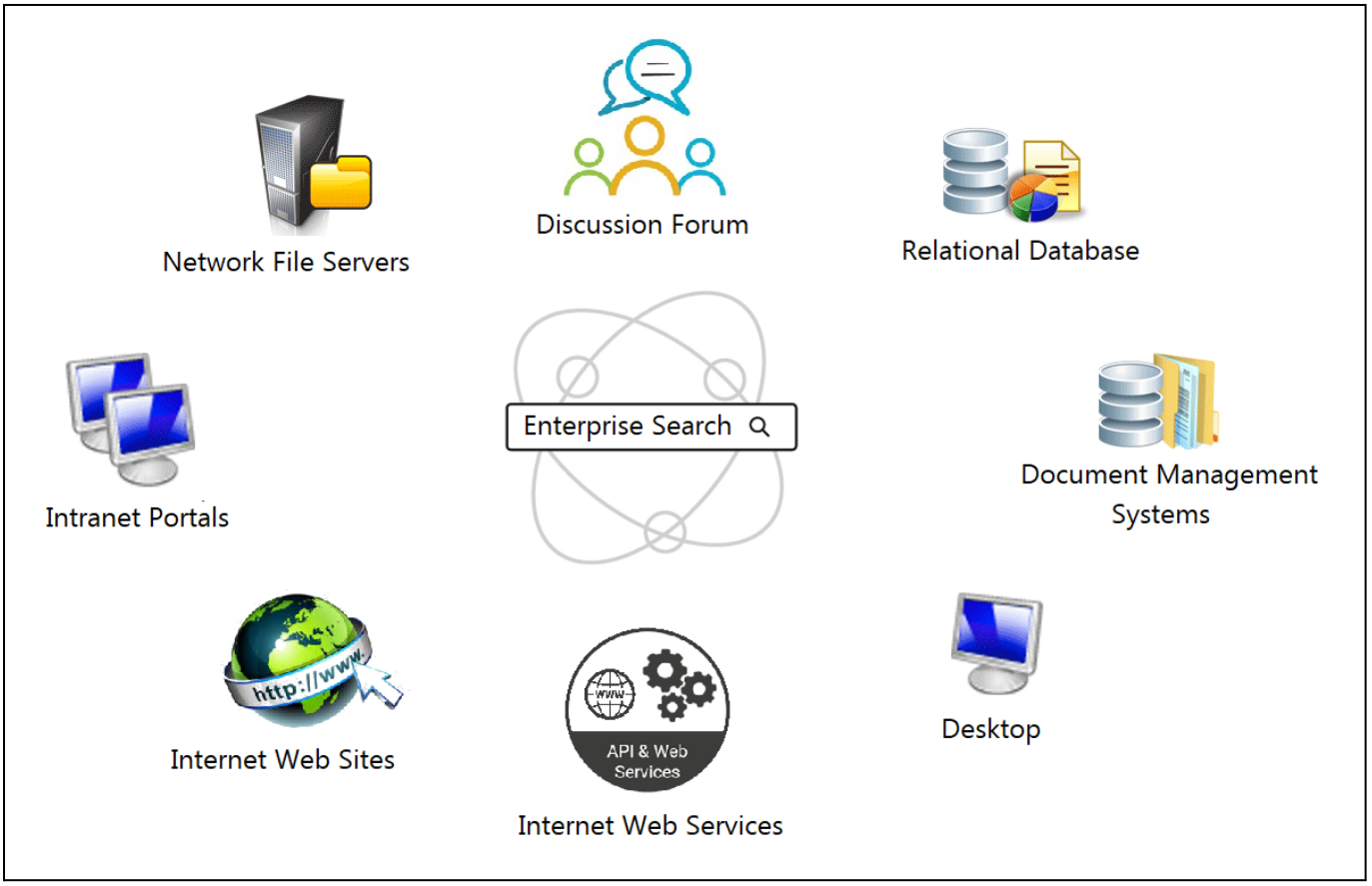


Figure n°2 : un moteur de recherche d'entreprise, un maillon essentiel de la GED (Gestion Électronique des Documents)

3. Analyse

3.1. La documentation interne chez efluid SAS

Comme énoncé précédemment, elle est accessible via plusieurs sites Web indépendants :

- documentation "archi" : représente une fraction de la documentation disponible. Il s'agit d'un site Web généré quotidiennement à partir de référentiels de code source git. Il est lui-même subdivisé en plusieurs sections :
    - documentation “archi” classique
    - documentation “archi” front
    - documentation Veille Techno / Revue de Presse
    - ADR (Architecture Decision Record)
- wikefluid : un Wiki interne utilisant Médiawiki, un moteur de wiki.
- eforum : un forum de discussion PHPBB, où les utilisateurs peuvent échanger autour de sujets précis, généralement autour de problèmes rencontrés.
- eroom : le système de GED (Gestion Électronique des Documents) interne
- suivefluid : gestionnaire de tickets

Chaque service possède des caractéristiques propres de :

- création,
- mise en page,
- accès (authentification par login et mot de passe requise pour certains d’entre eux)

On peut noter que le travail effectué n’a pas été réalisé sur les sources eroom et suivefluid.

Une visualisation de la structure simplifiée des sites Web étudiés est présentée en Annexes A à G .

3.2. Architecture d'un moteur de recherche

Un moteur de recherche permet de trouver plusieurs ressources à l'aide d'une requête formulée sous forme de mots-clés. Les résultats de cette requête s'affichent sur des pages de résultats de recherche.

Les moteurs de recherche sont assistés par des robots qui naviguent automatiquement sur les sites Web et sur Internet afin d'indexer des URL de page web pour qu'elles puissent apparaître sur les pages de résultats, correspondant à des mots-clés tapés par l'internaute.

L'architecture d'un moteur de recherche s'articule donc autour de trois composantes :

- un robot d'indexation*, constitué d'un Web crawler* (un logiciel qui explore automatiquement le Web à la recherche de ressources) associé à un Web scraper* (qui va extraire les données pertinentes desdites ressources),

- un système d'indexation du contenu, permettant la recherche inversée d'une ressource à partir de mots-clés présent dans son contenu,

- une interface de recherche permettant à l'utilisateur final de saisir une requête pour obtenir des résultats

3.2.1. Le robot d'indexation

Deux étapes intimement liées permettent de récupérer le contenu à indexer :

- Web crawling :

  En respectant la frontière* de crawl définie (*i.e.* les politiques d'exploration du Web), et à partir d'une URL fournie dans un fichier seed.txt*, le robot d'indexation se rend cycliquement sur tous les sites et pages autorisés qu'il trouve.

  Pour chaque page, il récupère les liens vers une ressource externe et les soumet à la frontière avant de les ajouter ou non à la liste des liens à parcourir.

  On se rend compte que la configuration de cette frontière est intrinsèquement liée aux pages qui seront parcourues et pour lesquelles une indexation sera possible.

- Web scraping :

  Lorsqu'une page est crawlée, son contenu doit être récupéré pour être indexé. La tâche est souvent compliquée par l'absence d'une standardisation du contenu. Chaque page étant construite différemment, le sélecteur CSS permettant d'identifier les différentes sections d'une page changent pour chaque site Web (e.g : un titre est caractérisé par une balise <h1> dans la documentation « archi » alors qu'il est présent dans les métadonnées* dans plusieurs autres services).

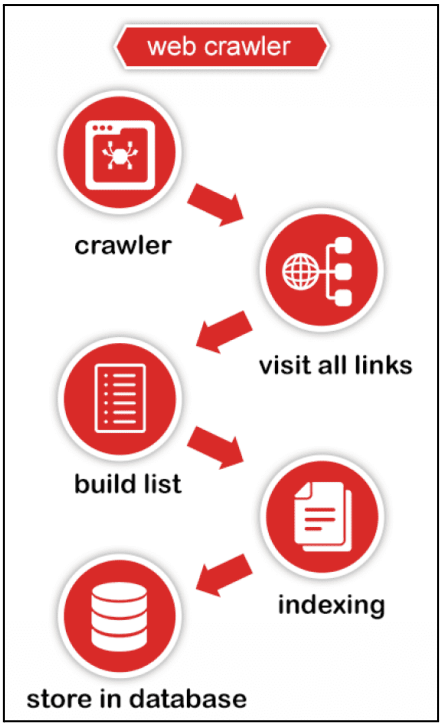


Figure n°3 : principe de fonctionnement d'un robot d'indexation - Web crawler

3.2.2. Système d'indexation

Après l'exploration du robot vient ensuite l'indexation de toutes les ressources récupérées. Cette action consiste à extraire les mots-clés considérés comme les plus pertinents, ou les plus significatifs, du site web afin de proposer ensuite un système de recherche inversée.

3.2.3. Interface de recherche

Après l'exploration et l'indexation, la dernière phase du fonctionnement du moteur de recherche est la recherche. Celle-ci équivaut à la requête effectuée sur le moteur.

L'interface de recherche doit donc permettre à l'utilisateur final d'entrer une requête constituée de mots-clés pour obtenir une liste de résultats.

Une mise en page élégante, l'intégration de fonctionnalités permettant de l'assister en cours de recherche, un tri pertinent des résultats fournis, ou encore le choix des informations à lui transmettre sont autant d'aspects à prendre en considération lors du design UX* pour améliorer l'expérience de l'utilisateur lors de l'utilisation de l'outil.

4. Réalisations

4.1. Mise en place d'un Proof of Concept

4.1.1. Premier essai avec des outils existants

Après une rapide présentation de la documentation “archi” classique sur laquelle j’allais travailler en priorité, il m'a été demandé de mettre en place une architecture pour obtenir un PoC* (Proof of Concept) du moteur de recherche.

J'ai apprécié la liberté sur le choix des technologies à utiliser, le droit d'utiliser des outils préexistants (du moment qu'ils étaient open-source et permettaient une accélération du projet), ainsi que la légèreté acceptée quant au style graphique que j’ai choisi.

Lors de ma prise d'information il était évident que deux acteurs principaux se partagent le marché en ce qui concerne l'indexation de contenu : Apache Solr et Elasticsearch. Le choix s’est finalement orienté vers la solution d’Apache qui est totalement open-source en plus d’offrir une maturité certaine, un développement actif avec un support assuré à long terme et une documentation extrêmement fournie.

Solr (prononcé “Solar”) est un sous-projet open source basé sur la bibliothèque de recherche et d’indexation Lucene d’Apache. Solr est basé sur Lucene Core et est écrit en Java. En tant que serveur de recherche, Apache Solr est l’un des outils les plus populaires pour l’intégration de moteurs de recherche verticaux.

Une fois l’installation d’Apache Solr effectuée, j’ai pu découvrir l’outil avec les exemples intégrés pour me familiariser avec les notions de core*, field*, collection*, document* au sens de Solr, puis une première approche de la configuration du schéma* Solr ou des requestHandlers*.

Le fait de travailler avec des données contenues et simples m’a permis de mieux intégrer ces notions et de pouvoir comprendre les mécanismes du query parser* ou de certains searchComponents* lors de la modification du fichier de configuration solrconfig.xml*.

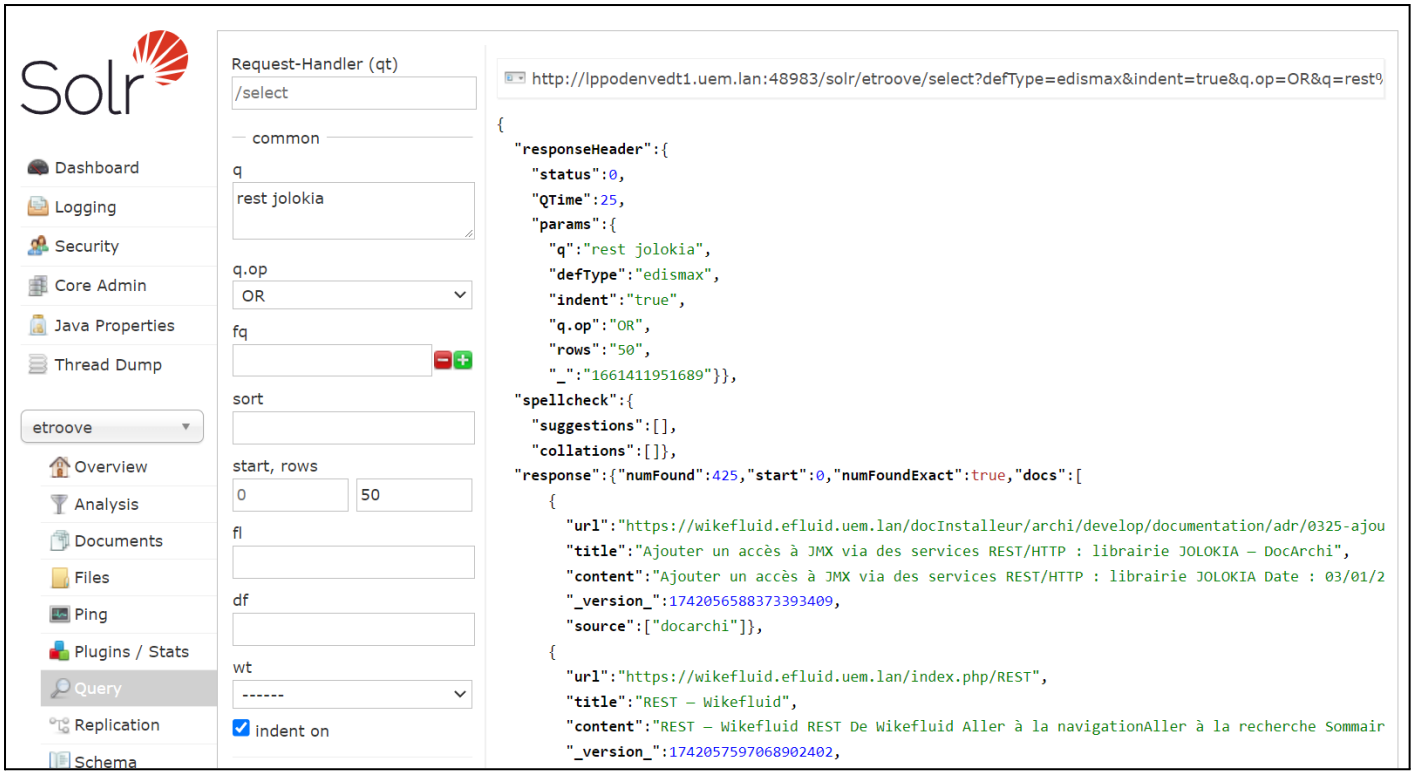


Figure n°4 : dashboard Web de Solr, une requête et la réponse retournée

J'ai ensuite essayé d'installer et configurer Apache Nutch et Apache Hadoop* (deux outils aux capacités reconnues utilisés par de grandes entreprises comme Google ou Yahoo!) pour obtenir une architecture utilisable.

Je me suis cependant confronté à une réelle difficulté quant au déploiement de ces outils sous Windows. Je n'ai pas eu plus de succès en me tournant vers StormCrawler et d'autres solutions préexistantes.

En effet, ces outils sont initialement conçus pour tourner sous environnement Linux et la configuration de mon poste pour les faire fonctionner sous environnement Windows ne m'était pas accessible puisqu'elle demandait une intervention du service gérant les postes de travail et les droits d'accès réseau.

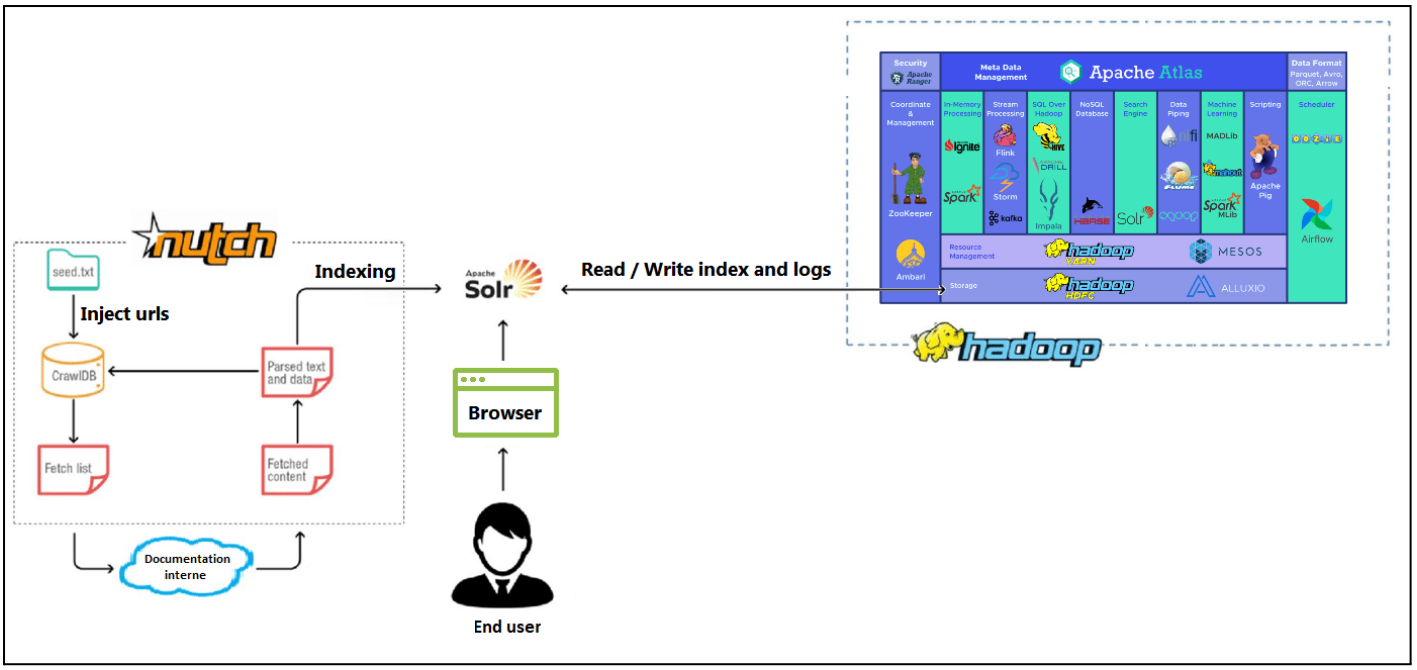


Figure n°5 : architecture pour le PoC utilisant Apache Solr, Nutch et Hadoop

Il a alors été décidé de se tourner vers une solution développée en interne, tout en conservant Solr pour la partie serveur de recherche.

Cette première approche m'a cependant permis de me familiariser avec la fonction de chaque composante d'un moteur de recherche, et un début de configuration de Solr.

Parcourir les fonctionnalités de Nutch et la manière de les implémenter m'a grandement inspiré lors du développement ultérieur du crawler.

4.1.2. Migration vers une solution développée en interne

Le projet a été nommé "etroove" pour rester dans la même terminologie que les composantes d'efluid.

Le développement a été effectué en Java, avec un projet Spring Boot multi-modules contenant :

- un client SolrJ (une API permettant l'interaction en Java avec Solr)
- un Web crawler / scraper (utilisation de la librairie jsoup)
- un serveur Web pour gérer l'interface utilisateur

Après la mise en place et configuration du projet sous IntelliJ, avec la découverte d'Artifactory, le gestionnaire universel de dépôts de binaires utilisé chez efluid, l'évolution de chaque module a été opérée de manière indépendante, chacun se voyant attribuer telle ou telle fonctionnalité au fur-et-à-mesure du stage.

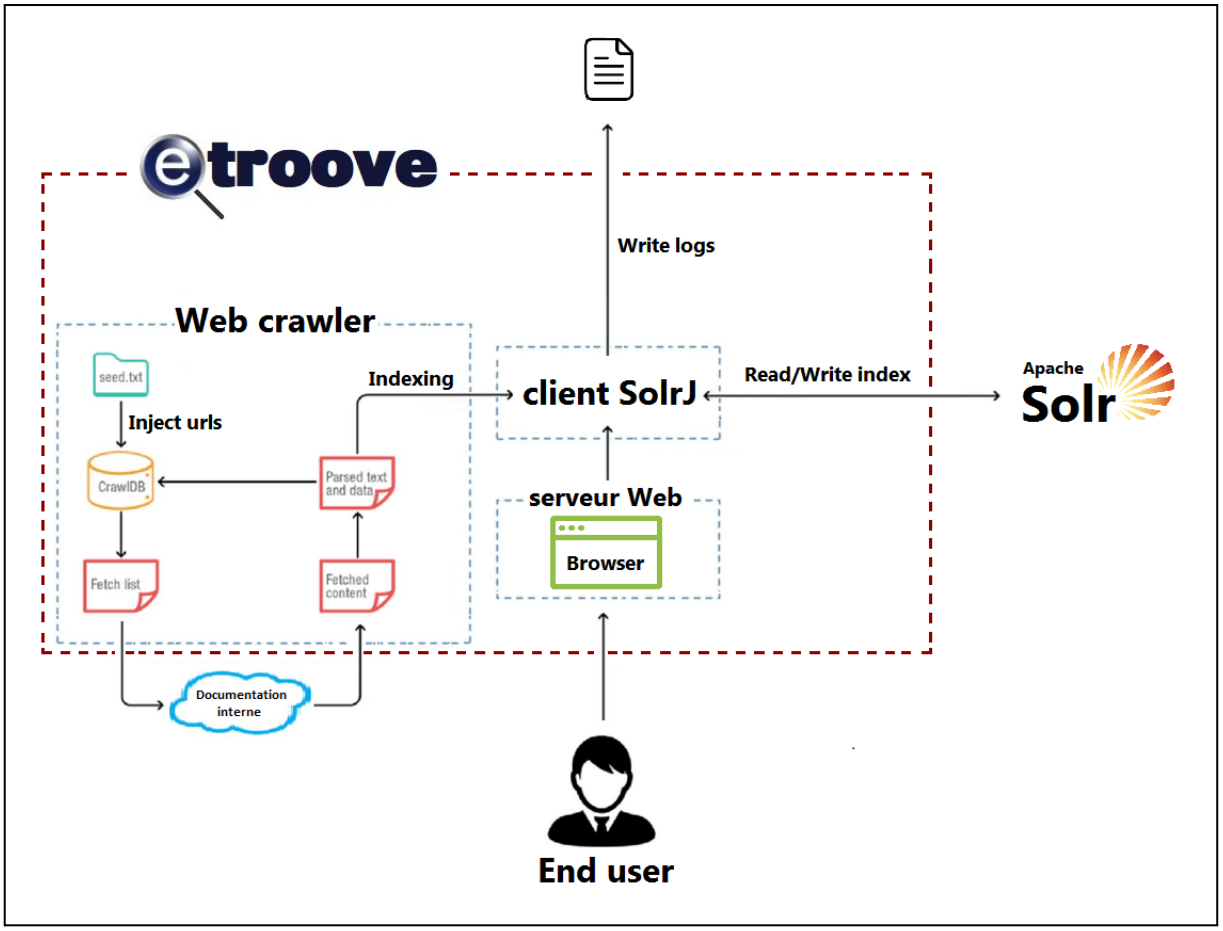


Figure n°6 : architecture du moteur de recherche etroove

J'ai d'abord travaillé sur le client SolrJ pour réussir à interagir avec Solr et obtenir :

- une connexion au serveur
- la récupération d'une réponse contenant le résultat de la recherche et ses données après l'envoi d'une requête à Solr
- l'indexation dans Solr de données brutes au format JSON, puis depuis un Bean* Java représentant un document issu de la documentation interne
- l'intégration d'informations dans des fichiers de logs lors du déroulement du programme
- l'intégration de la pagination des résultats (fonctionnalité supprimée *a posteriori* sur demande du tuteur de stage)

Le Web crawler n'étant pas développé dans un premier temps, j'ai fait le choix de continuer à utiliser les données fournies dans les exemples de Solr afin de faciliter l'intégration des nouvelles notions.

La configuration de Solr s'est déroulée en parallèle pour obtenir le comportement voulu avec notamment la définition de champs spécifiques (url, titre, source, catégorie, contenu).

L'interface de recherche est déployée via un serveur Tomcat embarqué dans le projet. Dans les premiers temps je me suis consacré à mettre en place :

- la génération de pages Web depuis le projet Spring Boot
- l'utilisation de Mustache pour les templates des pages Web
- l'utilisation de JavaScript Vanilla pour l'aspect dynamique des pages

Figure n°7 : résultats affichés sur l'interface utilisateur lors de la phase de PoC

4.2. Développement d'une solution interne

Une fois le PoC validé, j'ai poursuivi avec le développement d'un Web crawler pour la documentation "archi" classique. J'ai su atteindre au bout d'un mois cet objectif minimum convenu lors de l'entretien préalable au stage, ce qui a permis l'extension du projet à toutes les sources de documentation disponible en version Web.

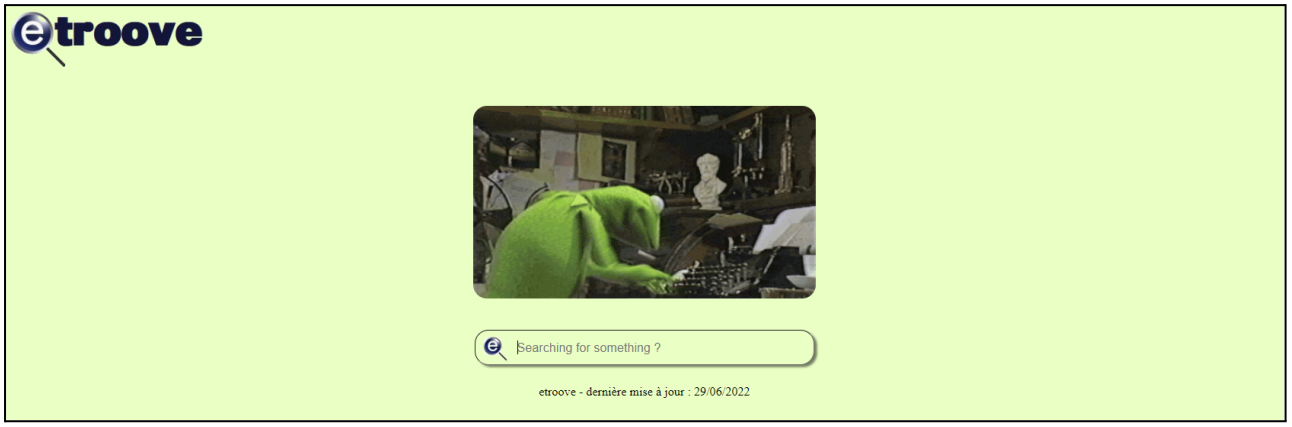


Figure n°8 : présentation de la page d'accueil du moteur de recherche "etroove" au 29/06/2022

Figure n°9 : présentation d'une page de résultats du moteur de recherche "etroove" au 29/06/2022

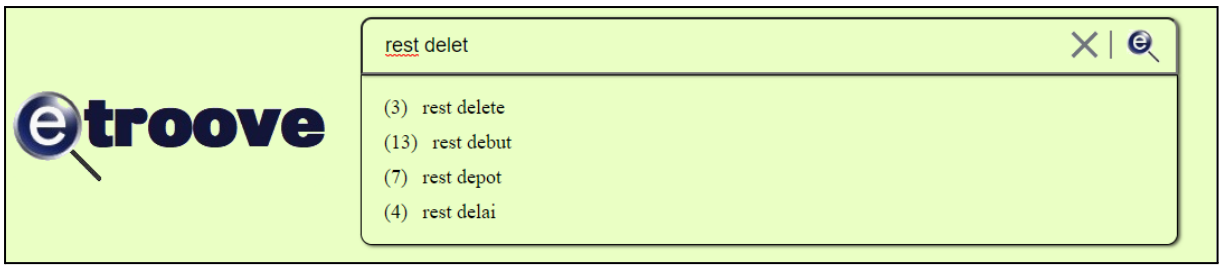


Figure n°10 : suggestions “realtime” lors de la saisie de mots-clés par l’utilisateur au 29/06/2022

Sur les dernières semaines, j’ai réussi à mettre en place une page d’administration permettant le contrôle des crawlers ainsi que la génération et la récupération d’un rapport d’audit au format CSV de l’utilisation de l’outil par les développeurs d’efluid SAS.

Il m’a également été demandé d’intégrer etroove dans la documentation existante.

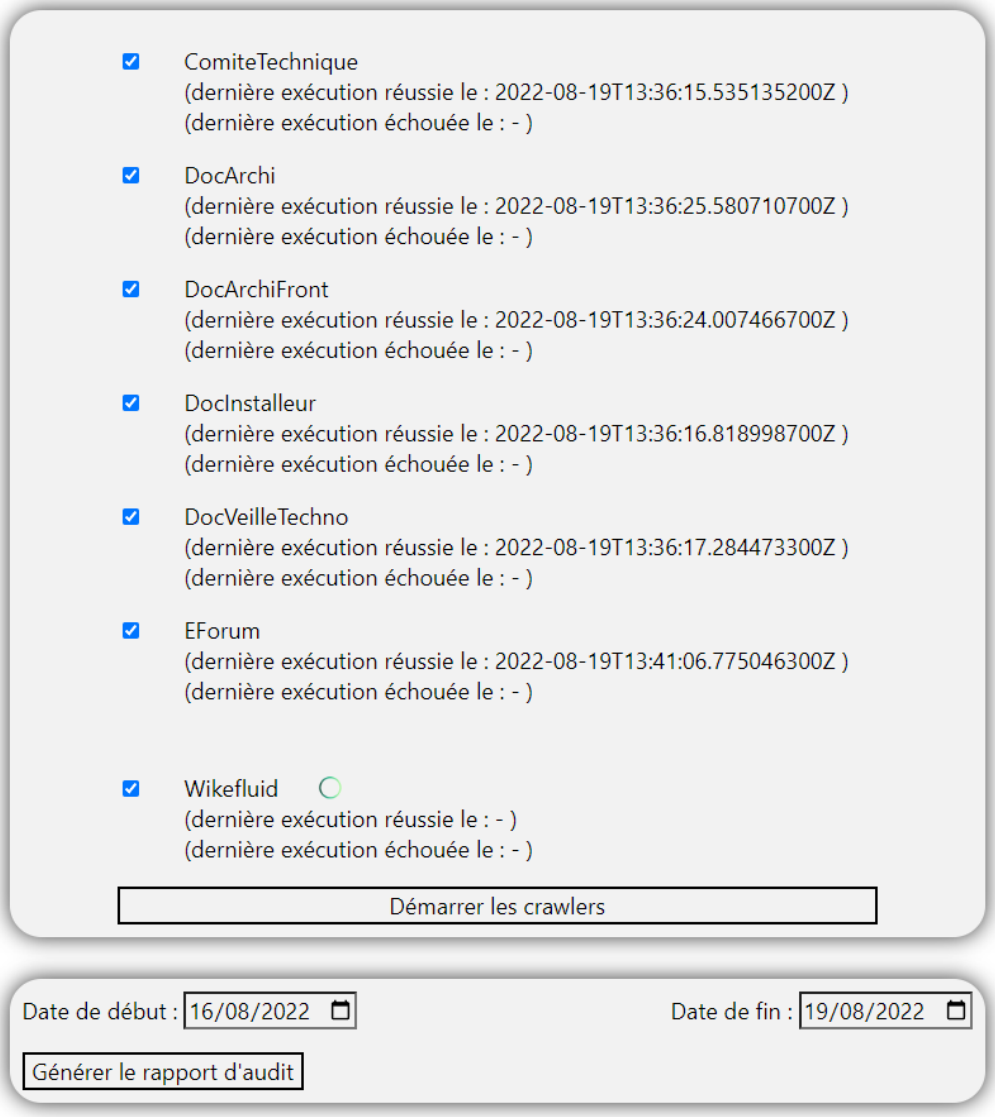


Figure n°11 : page d’administration après intégration

Ci-dessous un récapitulatif des actions menées en deuxième partie de stage sur les différentes composantes du projet.

Au niveau du client SolrJ :

- définition et paramétrage d'un second client SolrJ se connectant à un second core Solr dédié aux données nécessaires à la génération du rapport d'audit

- définition d'un nouveau Bean Java représentant les données relatives à une saisie utilisateur et servant au rapport d'audit

Configuration Solr :

- définition d'une UniqueKey* (l'URL) identifiant un document de manière unique et permettant sa mise à jour plutôt que la création d'un doublon lorsque l'on cherche à l'indexer à nouveau

- implémentation du Query Parser eDismax de Solr, permettant de profiter de toute la puissance de la syntaxe Lucène

- amélioration du positionnement des résultats de recherche avancée grâce à l'application :

    - de poids spécifiques pour chacun des champs,
    - de filtres de traitement sur la requête brute saisie par l'utilisateur ou au moment de l'indexation du contenu (utilisation et paramétrage des requestHandlers)

- configuration et prise en compte des fichiers :

    - synonyms.txt*
    - stopwords.txt*

- configuration et implémentation des fonctionnalités :

    - spellchecker* (correction des fautes d'orthographe),
    - suggester (autocomplétion et proposition de recherche fournissant un résultat)
    - highlighting* (génération dynamique d'un extrait pertinent du document, en relation avec les mots-clés saisis par l'utilisateur)

- création d'un second core et sa configuration pour indexer les données servant au rapport d'audit (la date et le contenu des requêtes saisies par les utilisateurs)

Web crawler / scraper :

- utilisation de la bibliothèque jsoup. Elle fournit une API très pratique pour la récupération des URL, l'extraction et la manipulation de leur contenu en utilisant le meilleur des méthodes DOM HTML5 et des sélecteurs CSS.

- frontière de crawl définie via des regex*

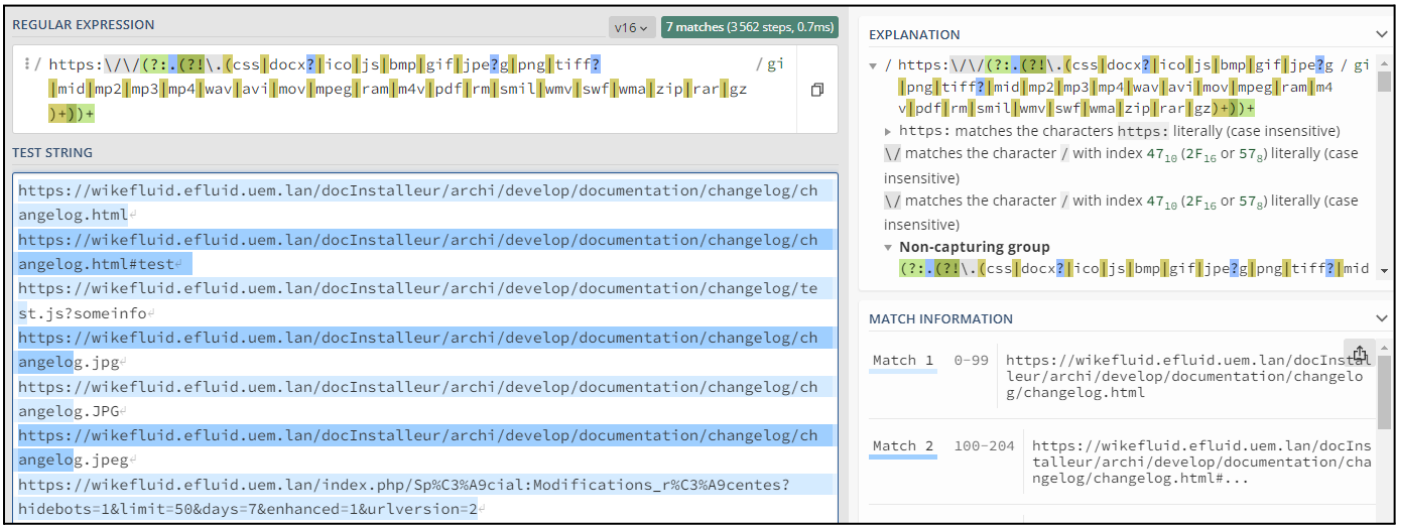


Figure n°12 : élaboration et contrôle des regex sur le site regex101.com
(ici contrôle des extensions refusées)

- chaque lien externe est soumis à la frontière de crawl

- chaque lien valide n'est exploré qu'une seule fois

- extraction du contenu en utilisant les sélecteurs CSS* disponibles dans le code source des pages, ou encore le slug* de l'URL pour extraire les catégories

- création de plusieurs crawlers spécifiques à chacune des sources d'information, pour répondre à la multitude de sources de documentation et leur architecture propre

- lancement asynchrone des crawlers (aucun impact sur le serveur Web qui reste disponible pour les utilisateurs)

- lancement sélectif des crawlers en multi-threading (optimisation de la durée de mise à jour des indexes)

- contrôle de l'état des crawlers : lorsqu'un crawler est lancé, impossibilité de le lancer à nouveau tant qu'il n'a pas fini sa tâche

- génération de fichiers de logs contenant notamment des informations sur les liens morts, ceux pour lesquels on note l'absence de titre, ceux dont l'accès nécessite une authentification préalable, …

- les mesures actuelles font état d'une durée totale d'exécution (depuis le crawling complet jusqu'à l'indexation de la totalité des contenus) portée à :

    - <1s pour la documentation "Comité Technique"
    - <1s pour la documentation "Veille Techno"
    - <1s pour la documentation "Installeur"
    - 3s pour la documentation "archi" front
    - 9s pour la documentation "archi" classique
    - 5min pour eforum
    - 19min pour Wikefluid

Puisque le lancement des crawlers est effectué en parallèle, avec une indexation near-real-time côté Solr, dès lors qu'un crawler a fini son exécution la mise à jour des données indexées est tout de suite effective et disponible via le moteur de recherche.

Ces opérations de mise à jour sont totalement invisibles pour l'utilisateur. Le service n'est aucunement perturbé, la procédure de mise à jour des indexes étant réalisée de manière asynchrone au niveau de l'application.

Il serait très facile de modifier le code pour lancer les crawlers de manière automatique et programmée plusieurs fois durant la journée et être ainsi à jour très rapidement lorsqu'une nouvelle documentation est créée et intégrée sur l'un des sites Web (option non implémentée à l'heure actuelle à la demande du tuteur de stage).

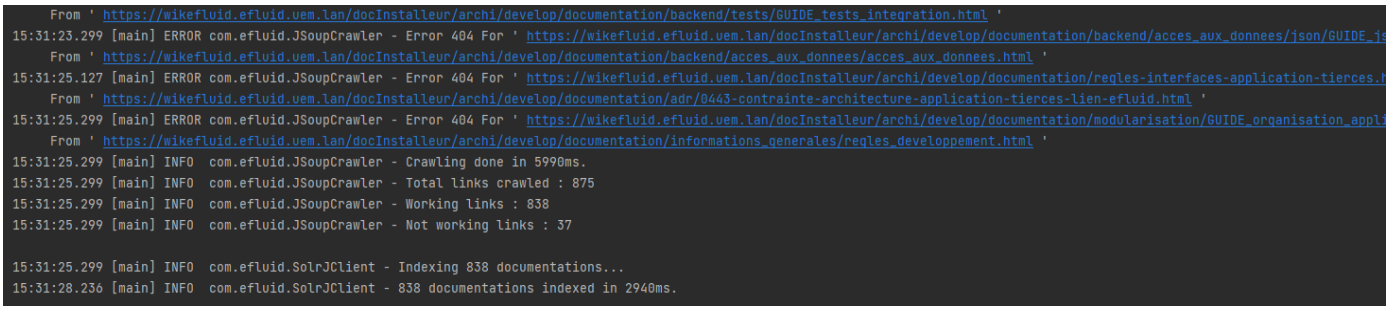


Figure n°13 : intégration d'informations dans les fichiers de logs et durée d'exécution des étapes de crawling et d'indexation

- des dispositions ont été mises en place afin de mesurer l'impact des crawlers sur le réseau et le serveur Apache hébergeant les sites de documentation interne.

    On observe que même en faisant tourner les crawlers en boucle pour essayer de saturer le réseau ou provoquer un ralentissement sur la navigation des sites, l'augmentation du trafic réseau ou la consommation des ressources du serveur ne sont pas problématiques (cf. Annexe P).

Interface utilisateur et page d'administration :

- intégration à la documentation actuelle

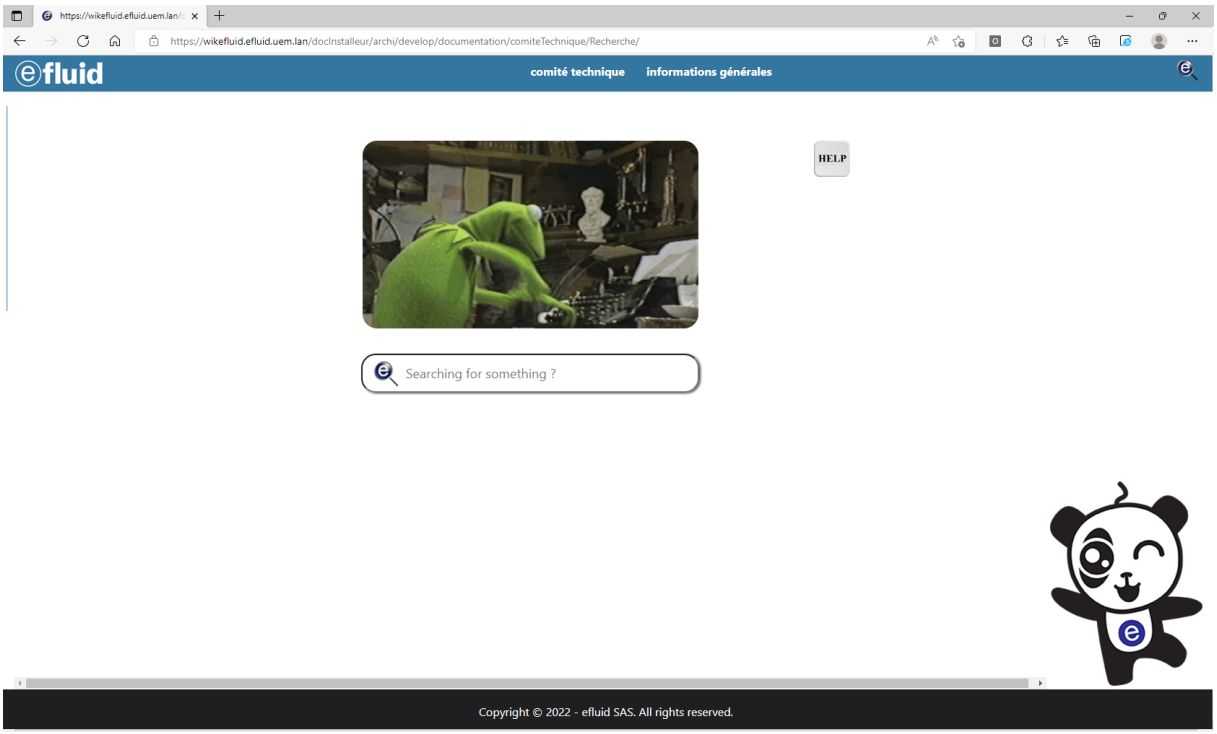


Figure n°14 : page d'accueil d'etroove après intégration

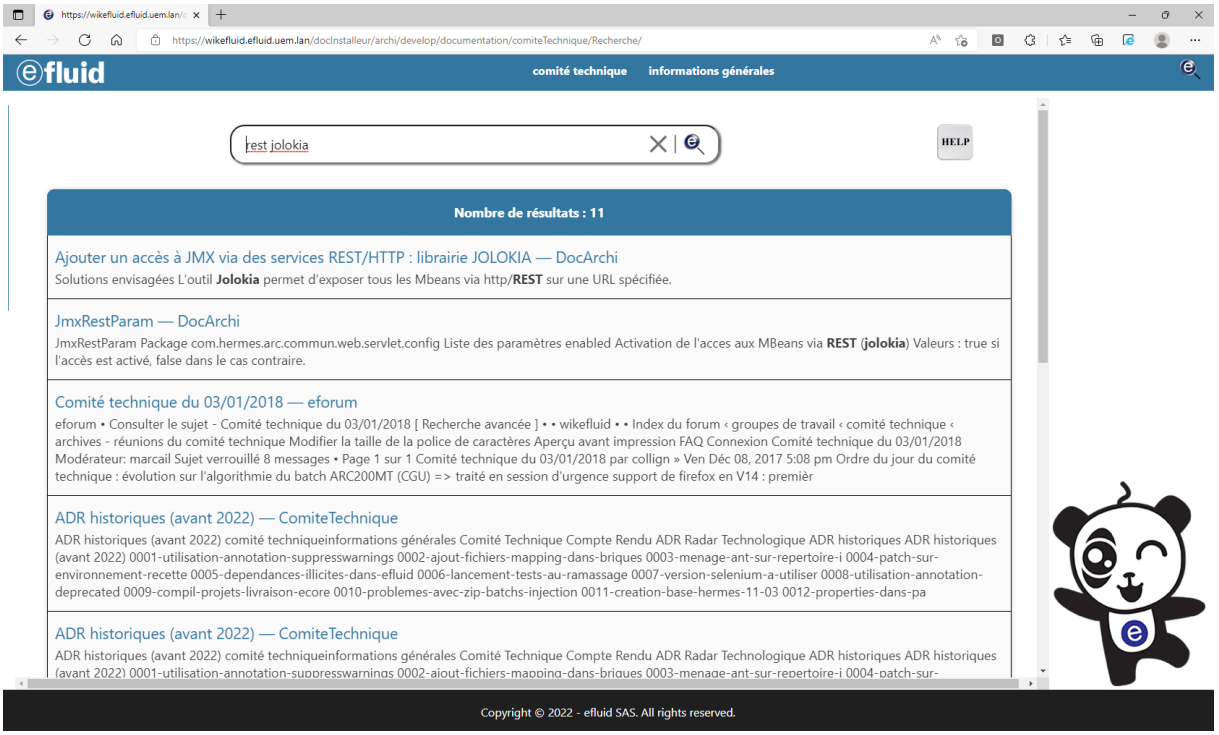


Figure n°15 : page résultats de recherche après intégration

- amélioration constante de l'UX, notamment via l’implémentation de fonctionnalités visant à assister l’utilisateur :
    - temps de réponse moyen du moteur de recherche inférieur à 50ms
    - utilisation du moteur AJAX pour l’affichage real-time des corrections orthographiques et suggestions de mots-clés amenant à un résultat
    - pré-visualisation du nombre de résultats à obtenir
    - navigation au clavier ou à la souris dans les suggestions

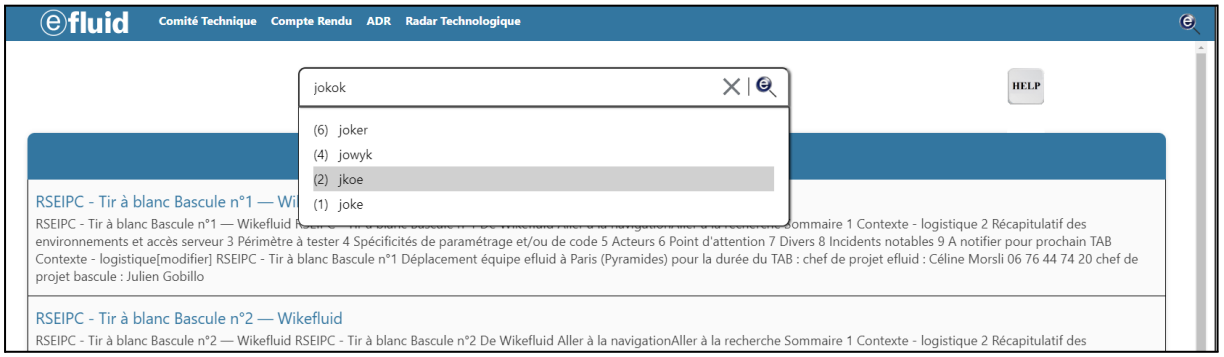


Figure n°16 : suggestions “realtime” lors de la saisie de mots-clés par l’utilisateur et pré-visualisation du nombre de résultats après intégration

    - création d’une page d’aide à l’utilisation d’etroove (cf. Annexe O)
    - création d’un logo etroove en utilisant la police d’écriture du logo du progiciel
    - information de l’utilisateur lors d’un problème au niveau de Solr ; lancement des crawlers et/ou téléchargement du rapport d’audit rendu(s) non disponible(s)

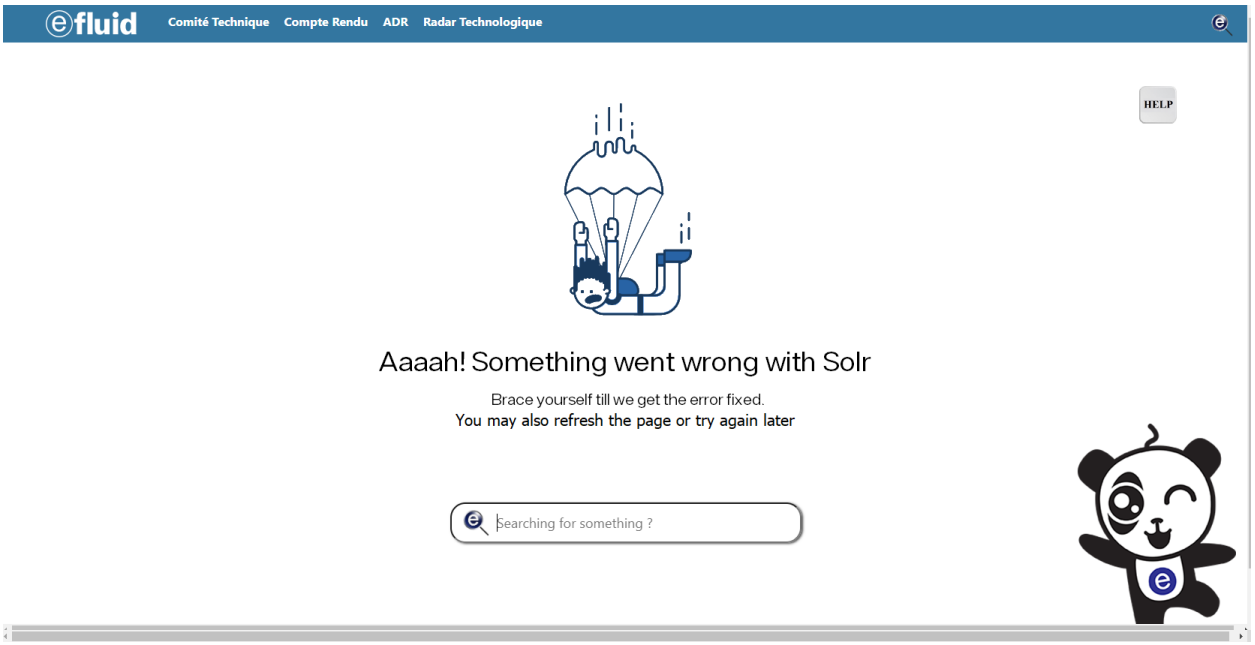


Figure n°17 : information élégante de la non disponibilité du moteur de recherche

- activation HTTPS pour se conformer aux minima de sécurité requis chez efluid
- la récupération d'un rapport d'audit permettant le suivi de l'utilisation du moteur de recherche en vue de son amélioration ou de l'amélioration de la documentation recherchée

Conclusion

L'objectif du stage est atteint puisque le moteur de recherche est déployé et opérationnel. Il est performant et retourne des résultats pertinents et bien ciblés. A l'usage on se rend compte d'un réel gain de temps.

Un premier retour de la part des développeurs m'a permis de résoudre quelques bugs graphiques liés à la résolution supérieure de leurs écrans. Quelques personnes m'ont confié avoir déjà intégré ce nouvel outil lors de leurs recherches d'information.

Il est prévu qu'après mon départ quelqu'un continue à travailler à l'amélioration d'etroove.

Pour ma part, je continuerais son amélioration en incorporant les autres sources d'informations chez efluid SAS pour obtenir un outil vraiment complet. En effet, il paraît intéressant d'intégrer les données contenues dans suivefluid ou encore l'indexation des documents contenus dans eroom.

Travailler pendant deux mois au développement en Java sur un projet Maven multi-modules m'a permis d'apprécier ce langage et l'utilisation de l'IDE IntelliJ. L'expertise du personnel en place a été un précieux complément de formation dans ce domaine. J'ai en effet pu recevoir des conseils qui me seront utiles tout au long de ma carrière. Les astuces pour accélérer le développement ou contrôler un produit en exploitation sont autant d'autres notions que j'ai pu m'approprier pendant ce stage.

Sitographie

« Introduction to Solr :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 20/06/2022, à l'adresse
https://solr.apache.org/guide/solr/latest/getting-started/introduction.html

« Exercise 2 Index Films Data :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 20/06/2022, à l'adresse
https://solr.apache.org/guide/solr/latest/getting-started/tutorial-films.html

« Running Solr :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 20/06/2022, à l'adresse
https://solr.apache.org/guide/6_6/running-solr.html

« Getting Started with Apache Solr ». (s.d.). sematext. Consulté le 20/06/2022, à l'adresse
https://sematext.com/guides/solr/

« Indexing and basic Data Operations :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 21/06/2022, à l'adresse
https://solr.apache.org/guide/6_6/indexing-and-basic-data-operations.html

« Apache Solr Hello World Example ». (18/04/2017). Mkyong.com. Consulté le 21/06/2022, à l'adresse
https://mkyong.com/solr/apache-solr-hello-world-example/

« Nutch Tutorial ». (27/01/2022). cwiki.apache.org. Consulté le 21/06/2022, à l'adresse
https://cwiki.apache.org/confluence/display/nutch/NutchTutorial

zhiqich.github.io. *« Apache Nutch & Solr »*. (05/01/2022). Github. Consulté le 21/06/2022, à l'adresse
https://zhiqich.github.io/2021/01/31/apachenutch/

lobster1234.github.io. *« Apache Nutch - Step by Step »*. (14/08/2017). Github. Consulté le 21/06/2022, à l'adresse
https://lobster1234.github.io/2017/08/14/search-with-nutch-mongodb-solr

« *Hadoop : How to install in 5 Steps in Windows10* ». (27/03/2021). Medium. Consulté le 21/06/2022, à l'adresse
https://medium.com/analytics-vidhya/hadoop-how-to-install-in-5-steps-in-windows-10-61b0e67342f8

« *Hadoop : qu'est-ce que c'est et comment apprendre à l'utiliser* ». (18/03/2021). DataScientest. Consulté le 21/06/2022, à l'adresse
https://datascientest.com/hadoop

« *L'écosystème Hadoop* ». (12/01/2017). illustraDATA.com. Consulté le 21/06/2022, à l'adresse
http://www.illustradata.com/lecosysteme-hadoop/

« *Scalable Web Crawling using StormCrawler and Apache Solr* ». (12/09/2019). Medium. Consulté le 22/06/2022, à l'adresse
https://medium.com/@AliAzG/scalable-web-crawling-using-stormcrawler-and-apache-solr-cbac70926ccc

« AsciiDoc ». (12/09/2019). AsciiDoc. Consulté le 23/06/2022, à l'adresse
https://asciidoc.org/

« Asciidoctor Docs ». (12/09/2019). Asciidoctor Docs. Consulté le 23/06/2022, à l'adresse
https://docs.asciidoctor.org/

« *How to install Maven on Windows* ». (07/09/2018). Mkyong.com. Consulté le 23/06/2022, à l'adresse
https://www.mkyong.com/maven/how-to-install-maven-in-windows/

« *Maven - JaCoCo code coverage example* ». (15/11/2018). Mkyong.com. Consulté le 23/06/2022, à l'adresse
https://mkyong.com/maven/maven-jacoco-code-coverage-example/

« *Maven - PITest mutation testing example* ». (06/11/2018). Mkyong.com. Consulté le 23/06/2022, à l'adresse
https://mkyong.com/maven/maven-pitest-mutation-testing-example/

« *SolrJ :: Apache Solr Reference Guide* ». (s.d.). Apache Software Foundation. Consulté le 23/06/2022, à l'adresse
https://solr.apache.org/guide/solr/latest/deployment-guide/solrj.html

« SolrJ ». (28/06/2019). cwiki.apache.org. Consulté le 23/06/2022, à l'adresse https://cwiki.apache.org/confluence/display/solr/Solrj

« JavaDoc ». (18/05/2022). Wikipedia. Consulté le 24/06/2022, à l'adresse https://fr.wikipedia.org/wiki/Javadoc

« How to write Doc Comments for the Javadoc Tool ». (s.d.). Oracle. Consulté le 24/06/2022, à l'adresse https://www.oracle.com/fr/technical-resources/articles/java/javadoc-tool.html

« Guide To Solr in Java With Apache SolrJ ». (09/10/2021). Baeldung. Consulté le 24/06/2022, à l'adresse https://www.baeldung.com/apache-solrj

« Using SolrJ :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 24/06/2022, à l'adresse https://solr.apache.org/guide/7_4/using-solrj.html#java-object-binding

« SolrClient (Solr 7.4.0 API) ». (s.d.). Apache Software Foundation. Consulté le 24/06/2022, à l'adresse https://solr.apache.org/docs/7_4_0/solr-solrj/org/apache/solr/client/solrj/SolrClient.html

« Defining Fields :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 24/06/2022, à l'adresse https://solr.apache.org/guide/6_6/defining-fields.html

« The Standard Query Parser :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 24/06/2022, à l'adresse https://solr.apache.org/guide/7_3/the-standard-query-parser.html

« The Extended DisMax (eDismax) Query Parser:: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 24/06/2022, à l'adresse https://solr.apache.org/guide/7_3/the-extended-dismax-query-parser.html

« Apache Lucene - Query Parser Syntax ». (s.d.). Apache Software Foundation. Consulté le 27/06/2022, à l'adresse https://lucene.apache.org/core/3_0_3/queryparsersyntax.html

« La recherche Full Text avec Solr ». (s.d.). g-rossolini.developpez-com. Consulté le 27/06/2022, à l'adresse https://g-rossolini.developpez.com/tutoriels/solr/?page=schema

« Documents, Fields, and Schema Design :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 27/06/2022, à l'adresse https://solr.apache.org/guide/6_6/documents-fields-and-schema-design.html

« Understanding Analysers, Tokenizers, and Filters :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 27/06/2022, à l'adresse https://solr.apache.org/guide/6_6/understanding-analyzers-tokenizers-and-filters.html

« Model, ModelMap, and ModelAndView in Spring MVC ». (26/12/2020). Baeldung. Consulté le 27/06/2022, à l'adresse https://www.baeldung.com/spring-mvc-model-model-map-model-view

« mustache - Logic-less templates ». (s.d.). mustache.github.io. Consulté le 27/06/2022, à l'adresse https://mustache.github.io/mustache.5.html

« Guide to Mustache with Spring Boot ». (26/05/2022). Baeldung. Consulté le 27/06/2022, à l'adresse https://www.baeldung.com/spring-boot-mustache

« Cookbook:jsoup Java HTML parser ». (s.d.). jsoup.org. Consulté le 27/06/2022, à l'adresse https://jsoup.org/cookbook/

« Load a Document from a URL:jsoup Java HTML parser ». (s.d.). jsoup.org. Consulté le 27/06/2022, à l'adresse https://jsoup.org/cookbook/input/load-document-from-url

« Use selector-syntax to find elements:jsoup Java HTML parser ». (s.d.). jsoup.org. Consulté le 27/06/2022, à l'adresse https://jsoup.org/cookbook/extracting-data/selector-syntax

« jsoup - Basic web crawler example ». (16/01/2017). jsoup.org. Consulté le 27/06/2022, à l'adresse https://mkyong.com/java/jsoup-basic-web-crawler-example/

« Utiliser Java Jsoup pour analyser HTML ». (s.d.). devstory. Consulté le 27/06/2022, à l'adresse
https://devstory.net/10399/jsoup-java-html-parser

« Jsoup TUtorial with Examples ». (03/08/2021). JavaCodeExamples. Consulté le 27/06/2022, à l'adresse
https://www.javacodeexamples.com/jsoup-tutorial-with-examples/1628

github.com/janl. *« mustache.js - Logic-less {{mustache}} templates with JavaScript »*. (27/04/2021). Github. Consulté le 28/06/2022, à l'adresse
https://github.com/janl/mustache.js#inverted-sections

« Field Type Definitions and Properties :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 28/06/2022, à l'adresse
https://solr.apache.org/guide/6_6/field-type-definitions-and-properties.html#field-type-definitions-and-properties

« WhatTheFont ». (s.d.). myfonts.com Consulté le 29/06/2022, à l'adresse
https://www.myfonts.com/WhatTheFont/

« Request Handlers and Search Components :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 01/07/2022, à l'adresse
https://solr.apache.org/guide/solr/latest/configuration-guide/requesthandlers-searchcomponents.html

« Spell Checking :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 01/07/2022, à l'adresse
https://solr.apache.org/guide/solr/latest/query-guide/spell-checking.html

« How to get Solr Suggester to return spelling suggestions as well ». (11/09/2019). Stack Overflow. Consulté le 01/07/2022, à l'adresse
https://stackoverflow.com/questions/11047119/how-to-get-solr-suggester-to-return-spelling-suggestions-as-well

« Simple Logging Facade for Java (SLF4J) ». (s.d.). SLF4J. Consulté le 01/07/2022, à l'adresse
https://www.slf4j.org/

« Introduction to SLF4J ». (05/07/2022). Baeldung. Consulté le 01/07/2022, à l'adresse
https://www.baeldung.com/slf4j-with-log4j2-logback

« log4j2.xml example ». (26/03/2019). mkyong.com. Consulté le 01/07/2022, à l'adresse https://mkyong.com/logging/log4j2-xml-example/

« Log4j 2 Appenders ». (28/06/2022). Apache Software Foundation. Consulté le 01/07/2022, à l'adresse https://logging.apache.org/log4j/2.x/manual/appenders.html

« Using Log4J 2 with Spring Boot ». (07/04/2016). Spring Framework Guru. Consulté le 01/07/2022, à l'adresse https://springframework.guru/using-log4j-2-spring-boot/#:~:text=In%20a%20Spring%20Boot%20application%2C%20you%20can%20specify,logging%20configuration.%20That%20is%20log4j2-spring.xml%20rather%20than%20log4j2.xml.

« Setting the Log Level in Spring Boot when Testing ». (05/02/2022). Baeldung. Consulté le 01/07/2022, à l'adresse https://www.baeldung.com/spring-boot-testing-log-level

« Spring Boot Banner - Complete Guide ». (30/11/2022). Springhow.com. Consulté le 01/07/2022, à l'adresse https://springhow.com/spring-boot-startup-banner/

« Using Custom Banners in Spring Boot ». (29/05/2022). Baeldung. Consulté le 01/07/2022, à l'adresse https://www.baeldung.com/spring-boot-custom-banners

« Text to ASCII Art Generator (TAAG) ». (s.d.). patorjk.com. Consulté le 01/07/2022, à l'adresse https://patorjk.com/software/taag/#p=display&f=Graffiti&t=etroove

« Azul Mission Control - Azul | Better Java Performance, Superior Java Support ». (s.d.). Azul. Consulté le 01/07/2022, à l'adresse https://www.azul.com/products/components/zulu-mission-control/#block-download

« Suggester :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 01/07/2022, à l'adresse https://solr.apache.org/guide/6_6/suggester.html

« SpellCheckResponse.Suggestion (Solr 7.0.1 API) ». (s.d.). Apache Software Foundation. Consulté le 04/07/2022, à l'adresse https://solr.apache.org/docs/7_0_1/solr-solrj/org/apache/solr/client/solrj/response/SpellCheckResponse.Suggestion.html

« *SpellCheckResponse.Collation (Solr 7.0.1 API)* ». (s.d.). Apache Software Foundation. Consulté le 04/07/2022, à l'adresse https://solr.apache.org/docs/7_0_1/solr-solrj/org/apache/solr/client/solrj/response/SpellCheckResponse.Collation.html

« *Pagination of Results :: Apache Solr Reference Guide* ». (s.d.). Apache Software Foundation. Consulté le 04/07/2022, à l'adresse https://solr.apache.org/guide/solr/latest/query-guide/pagination-of-results.html

« *Sorting, Paging, and Deep Paging in Solr* ». (s.d.). yonik.com. Consulté le 04/07/2022, à l'adresse https://yonik.com/solr/paging-and-deep-paging/

« *QueryResponse (Solr 7.0.1 API)* ». (s.d.). Apache Software Foundation. Consulté le 04/07/2022, à l'adresse https://solr.apache.org/docs/7_0_1/solr-solrj/org/apache/solr/client/solrj/response/QueryResponse.html

« *Spring Data Solr Tutorial: Pagination* ». (20/05/2013). petrikainulainen.com. Consulté le 04/07/2022, à l'adresse https://www.petrikainulainen.net/programming/solr/spring-data-solr-tutorial-pagination/

« *Spring Boot Ajax example* ». (03/02/2017). mkyong.com. Consulté le 05/07/2022, à l'adresse https://mkyong.com/spring-boot/spring-boot-ajax-example/

« *Solr returns only one collation for Suggester Component* ». (22/05/2012). Stack Overflow. Consulté le 06/07/2022, à l'adresse https://stackoverflow.com/questions/10547438/solr-returns-only-one-collation-for-suggester-component

« *Use JSDoc: Index*». (s.d.). *JSDoc* . Consulté le 07/07/2022, à l'adresse https://jsdoc.app/

Google. « Google JavaScript Style Guide ». (s.d.). Github. Consulté le 07/07/2022, à l'adresse https://google.github.io/styleguide/jsguide.html#:~:text=File%20names%20must%20be%20all,js%20.

Google. *« Pagination Best Practices for Google - Google Search Center »*. (s.d.). developers.google.com. Consulté le 07/07/2022, à l'adresse https://developers.google.com/search/docs/advanced/ecommerce/pagination-and-incremental-page-loading

Google. *« Infinite scroll search-friendly recommendations »*. (s.d.). developers.google.com. Consulté le 07/07/2022, à l'adresse https://developers.google.com/search/blog/2014/02/infinite-scroll-search-friendly

« Use video formats for animated content ». (04/10/2019). web.dev. Consulté le 07/07/2022, à l'adresse https://web.dev/efficient-animated-content/?utm_source=lighthouse&utm_medium=devtools

« CharFilterFactories :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 08/07/2022, à l'adresse https://solr.apache.org/guide/7_0/charfilterfactories.html#solr-mappingcharfilterfactory

« The Extended DisMax Query Parser :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 12/07/2022, à l'adresse https://solr.apache.org/guide/6_6/the-extended-dismax-query-parser.html

« Highlighting :: Apache Solr Reference Guide ». (s.d.). Apache Software Foundation. Consulté le 13/07/2022, à l'adresse https://solr.apache.org/guide/solr/latest/query-guide/highlighting.html

Google. *« Syntaxe de requête Lucene dans la Recherche cognitive Azure »*. (27/07/2022). docs.microsoft.com. Consulté le 17/07/2022, à l'adresse https://docs.microsoft.com/fr-fr/azure/search/query-lucene-syntax

« Usine d'électricité de Metz ». (30/03/2022). Wikipedia. Consulté le 17/07/2022, à l'adresse https://fr.wikipedia.org/wiki/Usine_d%27%C3%A9lectricit%C3%A9_de_Metz

« 5 chiffres clés sur l'archivage ». (24/02/2015). Arcalys. Consulté le 17/07/2022, à l'adresse https://www.arcalys.com/archivage/5-chiffres-cles-sur-l-archivage/

« Combien de temps perdu à la recherche d'information? ». (15/05/2017). LinkedIn. Consulté le 17/07/2022, à l'adresse https://www.linkedin.com/pulse/combien-de-temps-perdu-%C3%A0-la-recherche-dinformation-c%C3%A9dric-frickert/?originalSubdomain=fr

« Web scraping ». (20/06/2022). Wikipediaon. Consulté le 18/07/2022, à l'adresse https://fr.wikipedia.org/wiki/Web_scraping

« Robot d'indexation ». (12/05/2022). Wikipedia. Consulté le 18/07/2022, à l'adresse https://fr.wikipedia.org/wiki/Robot_d%27indexation#:~:text=Un%20robot%20d'indexation%20(en,qui%20explore%20automatiquement%20le%20Web.

« regex101: build, test, and debug regex ». (s.d.). Regex101. Consulté le 22/07/2022, à l'adresse https://regex101.com/

« Quick-Start: Regex Cheat Sheet ». (s.d.). Rex Egg. Consulté le 22/07/2022, à l'adresse http://www.rexegg.com/regex-quickstart.html

Upadhyay, M., Marti, R. . *« Single Sign-on Using Kerberos un Java »*. (s.d.). Oracle. Consulté le 25/07/2022, à l'adresse https://docs.oracle.com/javase/8/docs/technotes/guides/security/jgss/single-signon.html

« Part VI: HTTP/SPNEGO Authentication ». (s.d.). Oracle. Consulté le 25/07/2022, à l'adresse https://docs.oracle.com/en/java/javase/11/security/part-vi-http-spnego-authentication.html#GUID-05B34286-D0B6-4C35-B0BF-C98CD9F7E1D2

« SPNEGO ». (06/05/2022). Wikipedia. Consulté le 25/07/2022, à l'adresse https://en.wikipedia.org/wiki/SPNEGO

« Playwright for Java ». (s.d.). Github. Consulté le 27/07/2022, à l'adresse https://github.com/microsoft/playwright-java

Berbel, R. . *« Référence rapide pour les REGEX Lucene »*. (s.d.). oncrawl. Consulté le 03/08/2022, à l'adresse https://help.oncrawl.com/fr/articles/1685982-reference-rapide-pour-les-regex-lucene

« Overview - Eleventy ». (s.d.). 11ty. Consulté le 05/08/2022, à l'adresse https://www.11ty.dev/docs/

« YourKit Documentation: Java Profiler, .NET Profiler, YouMonitor ». (s.d.). YourKit. Consulté le 09/08/2022, à l'adresse https://www.yourkit.com/docs/

« PingRequestHandler(Solr 6.6.1 API) ». (s.d.). Apache Software Foundation. Consulté le 09/08/2022, à l'adresse https://solr.apache.org/docs/6_6_1/solr-core/org/apache/solr/handler/PingRequestHandler.html#:~:text=Ping%20Request%20Handler%20for%20reporting,status%22%20of%20a%20Solr%20server.

« Checking for Keyboard Events in JavaScript with Cross-Browser Support ». (10/02/2018). Medium. Consulté le 11/08/2022, à l'adresse https://devstephen.medium.com/keyboardevent-key-for-cross-browser-key-press-check-61dbad0a067a#:~:text=KeyCode%20was%20deprecated%20because%20in,to%20use%20key%20or%20code%20.

« JavaScript Key Code List & Table ». (s.d.). Toptal. Consulté le 11/08/2022, à l'adresse https://www.toptal.com/developers/keycode/table-of-all-keycodes

« Why doesn't focus() select my container div? ». (25/10/2010). Stack Overflow. Consulté le 11/08/2022, à l'adresse https://stackoverflow.com/questions/4014935/why-doesnt-focus-select-my-container-div

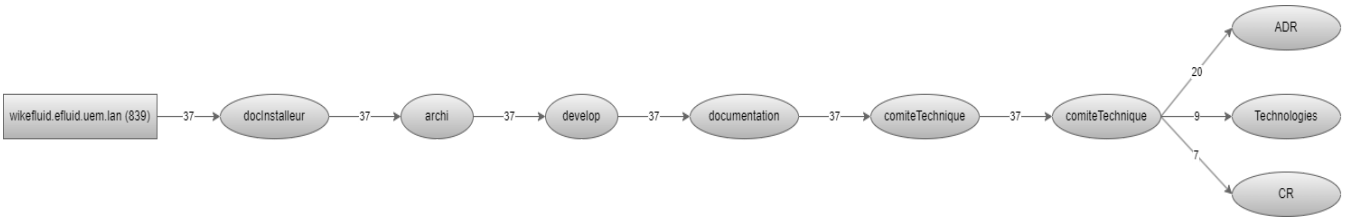
Doudoux, J-M. *« Le framework Executor »*. (s.d.). Développons en Java. Consulté le 11/08/2022, à l'adresse https://www.jmdoudoux.fr/java/dej/chap-executor.htm

« Java - Try with Resources ». (10/03/2022). Baeldung. Consulté le 11/08/2022, à l'adresse https://www.baeldung.com/java-try-with-resources

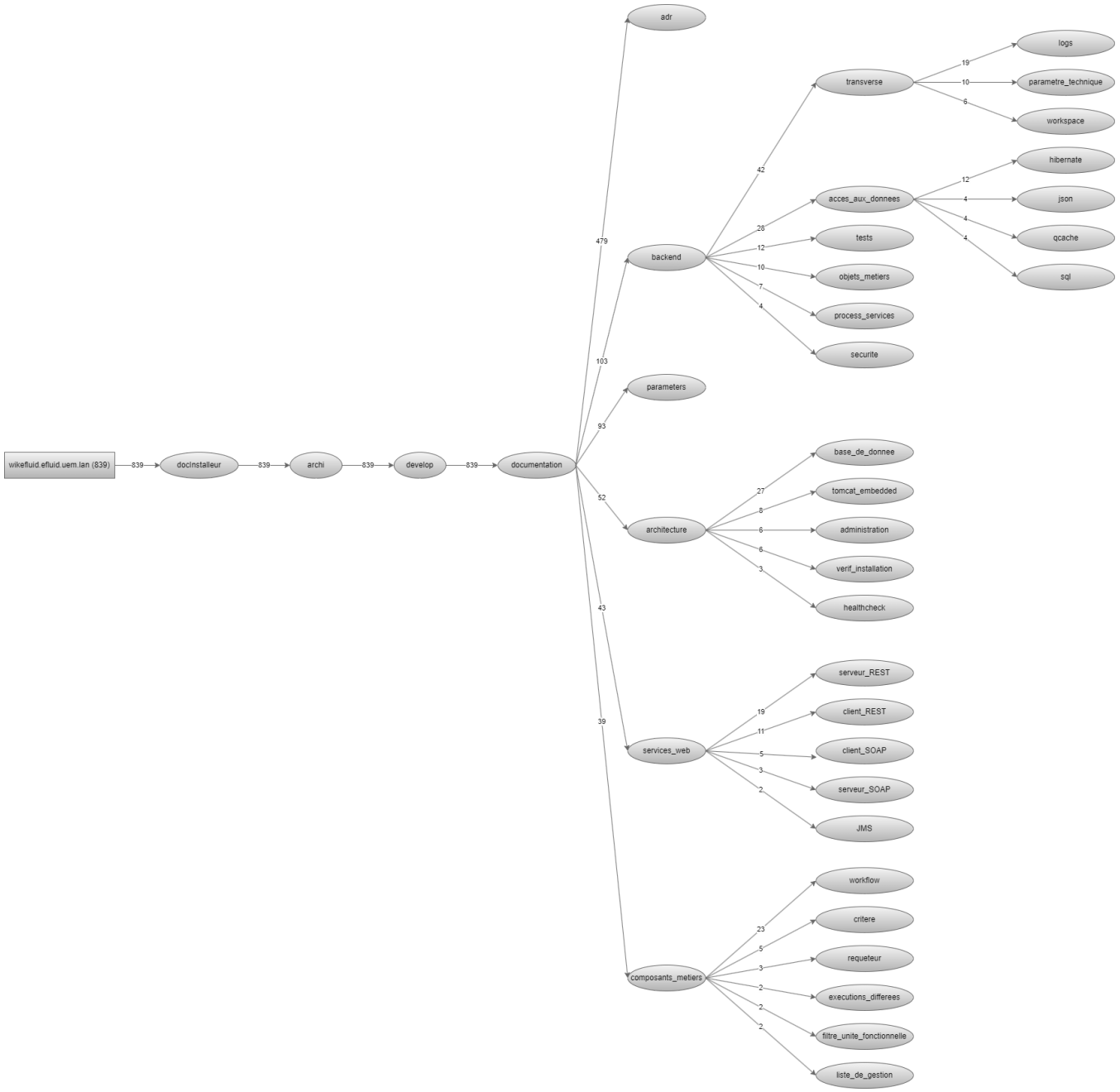
« Les redirections HTTP ». (s.d.). pierre-giraud.com. Consulté le 23/08/2022, à l'adresse https://www.pierre-giraud.com/http-reseau-securite-cours/redirection/

Table des annexes

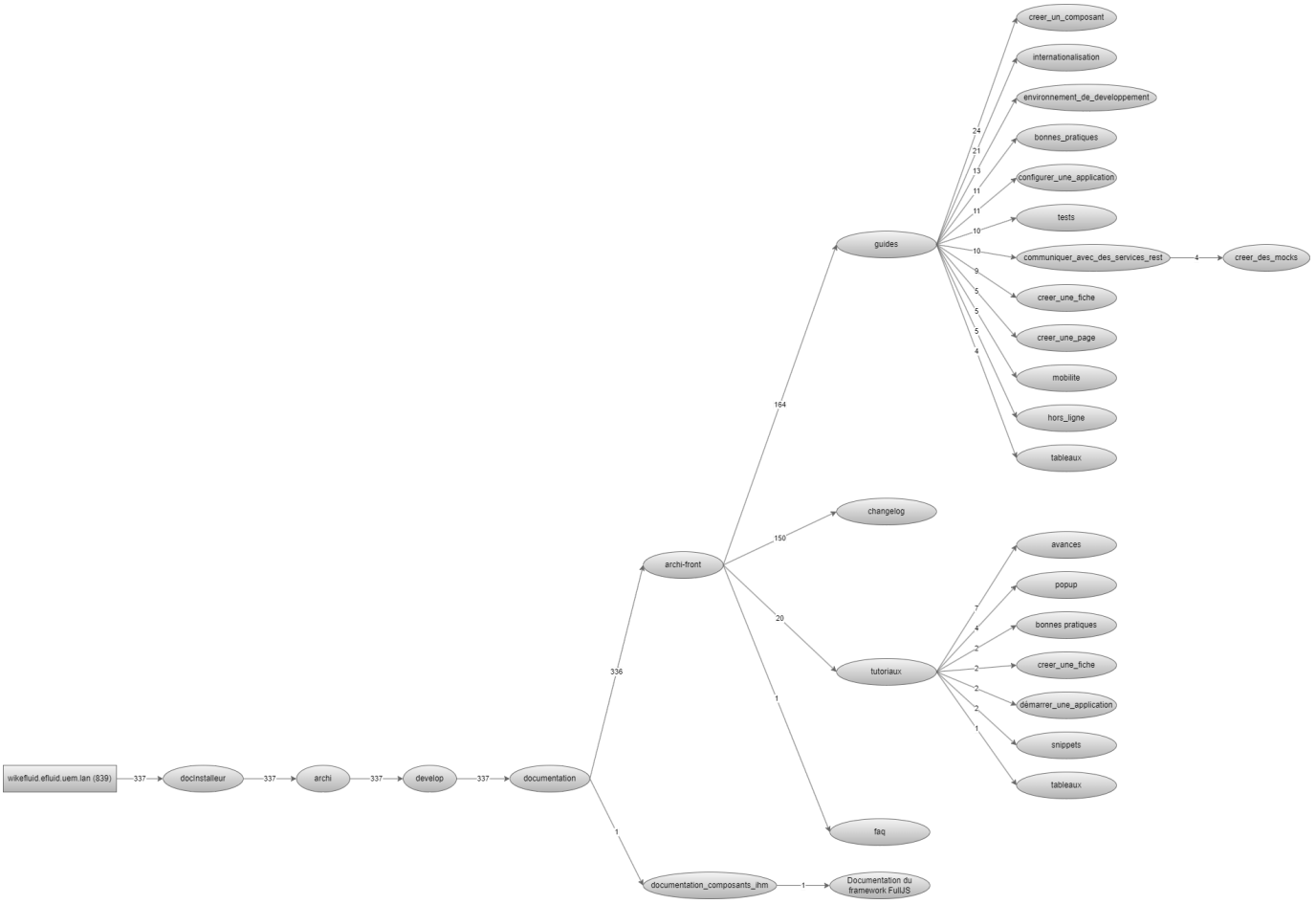
Annexe A : visualisation de l'architecture simplifiée du site "Comité technique"



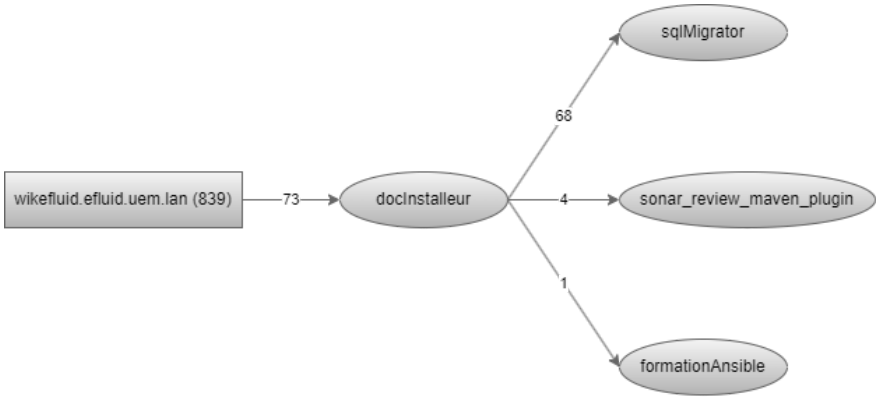
Annexe B : visualisation de l'architecture simplifiée du site "DocArchi"



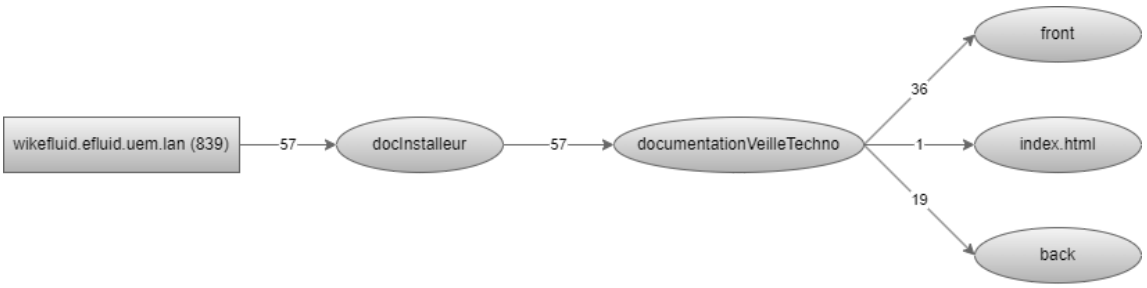
Annexe C : visualisation de l'architecture simplifiée du site "DocArchi-Front"



Annexe D : visualisation de l'architecture simplifiée du site "DocInstalleur"

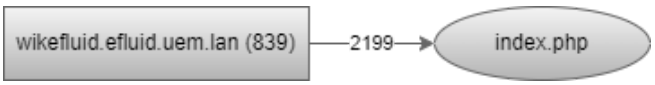


Annexe E : visualisation de l'architecture simplifiée du site "Veille technologique"



Annexe F : visualisation de l'architecture simplifiée du site "eforum"

Annexe G : visualisation de l'architecture simplifiée du site "Wikefluid"



Annexe H : extrait du fichier de configuration solrconfig.xml – searchComponent highlight

D: > Programs > solr-8.11.1 > server > solr > etrouve2 > conf > solrconfig.xml

```
1050    <searchComponent class="solr.HighlightComponent" name="highlight">
1051      <highlighting>
1052        <!-- Configure the standard fragmenter -->
1053        <!-- This could most likely be commented out in the "default" case -->
1054        <fragmenter name="gap"
1055                    default="true"
1056                    class="solr.highlight.GapFragmenter">
1057          <lst name="defaults">
1058            <int name="hl.fragsize">100</int>
1059          </lst>
1060        </fragmenter>
1061
1062        <!-- A regular-expression-based fragmenter
1063             (for sentence extraction)
1064          -->
1065        <fragmenter name="regex"
1066                    class="solr.highlight.RegexFragmenter">
1067          <lst name="defaults">
1068            <!-- slightly smaller fragsizes work better because of slop -->
1069            <int name="hl.fragsize">70</int>
1070            <!-- allow 50% slop on fragment sizes -->
1071            <float name="hl.regex.slop">0.5</float>
1072            <!-- a basic sentence pattern -->
1073            <str name="hl.regex.pattern">[-\w ,/\n\&quot;&apos;]{20,200}</str>
1074          </lst>
1075        </fragmenter>
1076
1077        <!-- Configure the standard formatter -->
1078        <formatter name="html"
1079                   default="true"
1080                   class="solr.highlight.HtmlFormatter">
1081          <lst name="defaults">
1082            <str name="hl.simple.pre"><![CDATA[<em>]]></str>
1083            <str name="hl.simple.post"><![CDATA[</em>]]></str>
1084          </lst>
1085        </formatter>
```

Annexe I : extrait du fichier de configuration solrconfig.xml – requestHandler /select

D: > Programs > solr-8.11.1 > server > solr > etrouve2 > conf > solrconfig.xml

```
777   <requestHandler name="/select" class="solr.SearchHandler">
778     <!-- default values for query parameters can be specified, these
779          will be overridden by parameters in the request
780     -->
781     <lst name="defaults">
782       <str name="echoParams">explicit</str>
783       <!-- Default search field -->
784       <str name="df">all</str>
785       <!-- enables edismax and specifies the weight for each field -->
786       <str name="defType"> edismax </str>
787       <str name="qf">title^50 tags^5 content^1 all^1</str>
788       <str name="pf">title^100 tags^20 content^10 all^1</str>
789       <str name="ps">10</str>
790       <!-- spell check component configuration -->
791       <str name="spellcheck">true</str>
792       <str name="spellcheck.count">5</str>
793       <str name="spellcheck.collate">true</str>
794       <str name="spellcheck.collateExtendedResults">true</str>
795       <str name="spellcheck.maxCollations">10</str>
796       <str name="spellcheck.maxCollationTries">50</str>
797       <!-- highlighting configuration -->
798       <bool name="hl">true</bool>
799       <str name="hl.method">unified</str>
800       <str name="hl.fl">content</str>
801       <str name="hl.requireFieldMatch">true</str>
802       <str name="hl.fragsize">150</str> -->
803       <str name="hl.encoder">html</str>
804       <!-- number of results displayed per SolrResponse -->
805       <int name="rows">10</int>
806
807       <!-- Change from JSON to XML format (the default prior to Solr 7.0)
808            <str name="wt">xml</str>
809       -->
810     </lst>
```

Annexe J : extrait du fichier de configuration solrconfig.xml – searchComponent spellcheck

D: > Programs > solr-8.11.1 > server > solr > etrouve2 > conf > solrconfig.xml

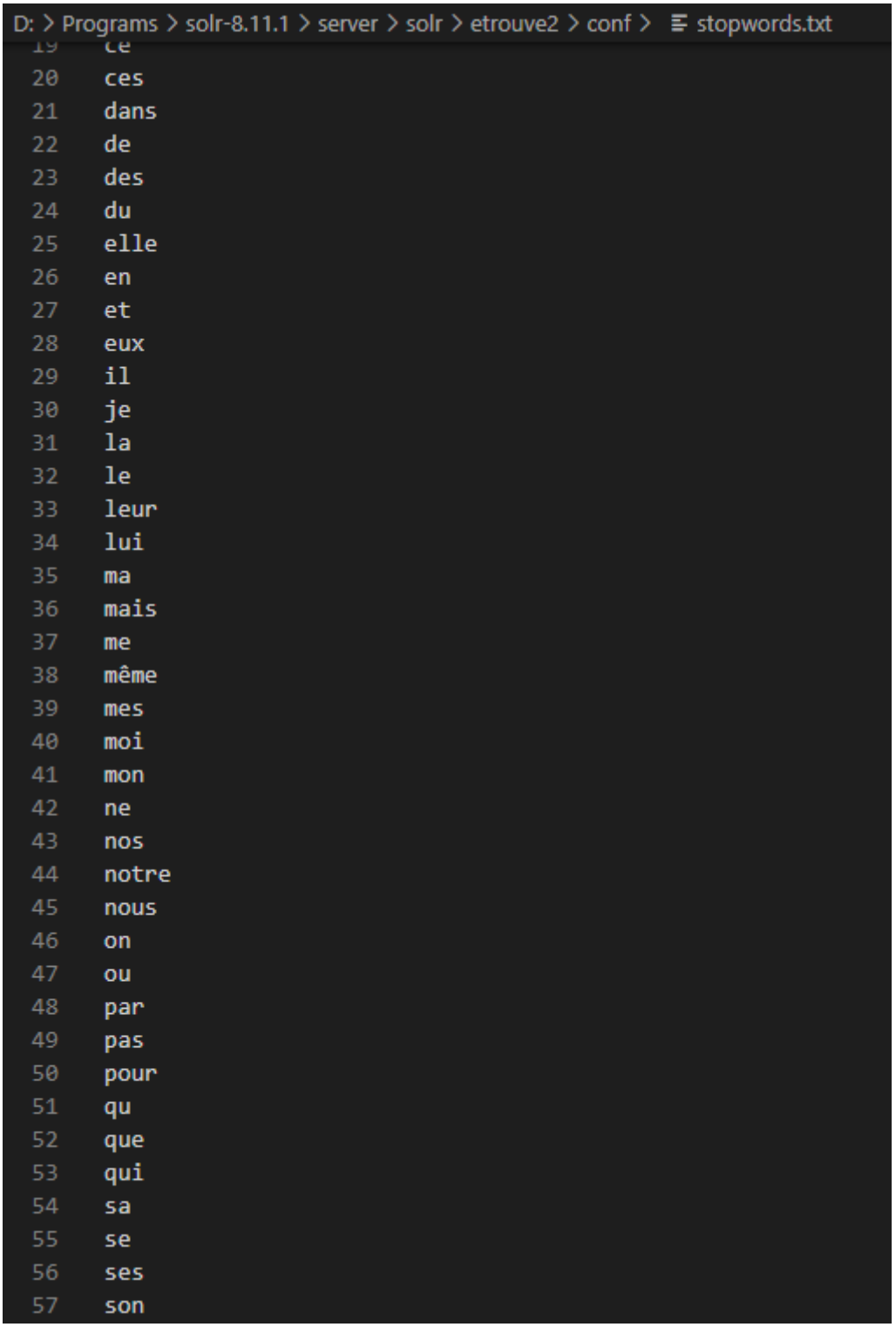
```
940   <searchComponent name="spellcheck" class="solr.SpellCheckComponent">
941
942     <str name="queryAnalyzerFieldType">text_general</str>
943
944     <!-- Multiple "Spell Checkers" can be declared and used by this
945          component
946       -->
947
948     <!-- a spellchecker built from a field of the main index -->
949     <lst name="spellchecker">
950       <!-- <str name="name">default</str> -->
951       <!-- <str name="field">_text_</str> -->
952       <str name="field">all</str>
953       <str name="classname">solr.DirectSolrSpellChecker</str>
954       <!-- the spellcheck distance measure used, the default is the internal levenshtein -->
955       <str name="distanceMeasure">internal</str>
956       <!-- minimum accuracy needed to be considered a valid spellcheck suggestion -->
957       <float name="accuracy">0.5</float>
958       <!-- the maximum #edits we consider when enumerating terms: can be 1 or 2 -->
959       <int name="maxEdits">2</int>
960       <!-- the minimum shared prefix when enumerating terms -->
961       <int name="minPrefix">1</int>
962       <!-- maximum number of inspections per result. -->
963       <int name="maxInspections">5</int>
964       <!-- minimum length of a query term to be considered for correction -->
965       <int name="minQueryLength">2</int>
966       <!-- maximum length of a query term to be considered for correction -->
967       <!-- <int name="maxQueryLength">99999</int> -->
968       <!-- maximum threshold of documents a query term can appear to be considered for correction -->
969       <!-- <float name="maxQueryFrequency">0.01</float> -->
970       <!-- uncomment this to require suggestions to occur in 1% of the documents -->
971       <float name="thresholdTokenFrequency">.0001</float>
972       <!-- buildOnCommit|buildOnOptimize -->
973       <str name="buildOnCommit">true</str>
974       <!-- $solr.solr.home/data/spellchecker-->
975       <str name="spellcheckIndexDir">./spellchecker</str>
976     </lst>
```

Annexe K : extrait du fichier de configuration Solr schema.xml

D: > Programs > solr-8.11.1 > server > solr > etrouve2 > conf > schema.xml

```
284     <fieldType name="text_general" class="solr.TextField" positionIncrementGap="100" multiValued="true">
285       <analyzer type="index">
286         <charFilter class="solr.MappingCharFilterFactory" mapping="mapping-FoldToASCII.txt"/>
287         <tokenizer class="solr.StandardTokenizerFactory"/>
288         <filter class="solr.ElisionFilterFactory" articles="lang/contractions_fr.txt" ignoreCase="true"/>
289         <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
290         <filter class="solr.LowerCaseFilterFactory"/>
291       </analyzer>
292       <analyzer type="query">
293         <charFilter class="solr.MappingCharFilterFactory" mapping="mapping-FoldToASCII.txt"/>
294         <tokenizer class="solr.StandardTokenizerFactory"/>
295         <filter class="solr.ElisionFilterFactory" articles="lang/contractions_fr.txt" ignoreCase="true"/>
296         <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
297         <filter class="solr.SynonymGraphFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
298         <filter class="solr.LowerCaseFilterFactory"/>
299       </analyzer>
300     </fieldType>



462     <fieldType name="text_HTML" class="solr.TextField" positionIncrementGap="100" multiValued="true">
463       <analyzer type="index">
464         <!-- Strips HTML tags (including attributes) -->
465         <!-- See: https://solr.apache.org/guide/6_6/charfilterfactories.html#CharFilterFactories-solr.HTMLStripCharFilterFactory -->
466         <charFilter class="solr.HTMLStripCharFilterFactory"/>
467         <!-- Maps raw query to ASCII = replace special characters and accentuation with simple letters -->
468         <!-- see: https://solr.apache.org/guide/6_6/charfilterfactories.html#CharFilterFactories-solr.MappingCharFilterFactory -->
469         <charFilter class="solr.MappingCharFilterFactory" mapping="mapping-FoldToASCII.txt"/>
470         <tokenizer class="solr.StandardTokenizerFactory"/>
471         <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
472         <filter class="solr.LowerCaseFilterFactory"/>
473       </analyzer>
474       <analyzer type="query">
475         <charFilter class="solr.MappingCharFilterFactory" mapping="mapping-FoldToASCII.txt"/>
476         <tokenizer class="solr.StandardTokenizerFactory"/>
477         <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
478         <filter class="solr.SynonymGraphFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms.txt"/>
479         <filter class="solr.LowerCaseFilterFactory"/>
480       </analyzer>
481     </fieldType>
482
483
484     <uniqueKey>url</uniqueKey>
485
486
487     <field name="_nest_path_" type="_nest_path_"/>
488     <field name="_root_" type="string" docValues="false" indexed="true" stored="false"/>
489     <field name="_text_" type="text_general" multiValued="true" indexed="true" stored="false"/>
490     <field name="_version_" type="plong" indexed="false" stored="false"/>
491     <field name="url" type="string" multiValued="false" indexed="true" required="true" stored="true"/>
492     <field name="title" type="text_general" uninvertible="true" multiValued="false" indexed="true" stored="true"/>
493     <field name="content" type="text_HTML" multiValued="false" indexed="true" stored="true"/>
494     <field name="tags" type="text_general" multiValued="true" indexed="true" stored="false"/>
495     <field name="all" type="text_general" multiValued="true" indexed="true" stored="false"/>



574     <!-- Note: During indexing, Fields are copied BEFORE analysis -->
575     <copyField source="tags" dest="title_str" maxChars="256"/>
576     <copyField source="title" dest="title_str" maxChars="256"/>
577     <copyField source="url" dest="url_str" maxChars="256"/>
578
579     <copyField source="tags" dest="all" maxChars="256"/>
580     <copyField source="title" dest="all" maxChars="256"/>
581     <copyField source="url" dest="all" maxChars="256"/>
582     <copyField source="content" dest="all"/>
583   </schema>
```

Annexe L : extrait du fichier de configuration stopwords.txt



Annexe M : extrait du fichier de configuration synonyms.txt

D: > Programs > solr-8.11.1 > server > solr > etrouve2 > conf > synonyms.txt

```
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

#-----------------------------------------------------------------------
#some test synonym mappings unlikely to appear in real input text
aaafoo => aaabar
bbbfoo => bbbfoo bbbbar
cccfoo => cccbar cccbaz
fooaaa,baraaa,bazaaa

# Some synonym groups specific to this example
GB,gib,gigabyte,gigabytes
MB,mib,megabyte,megabytes
Television, Televisions, TV, TVs
#notice we use "gib" instead of "GiB" so any WordDelimiterGraphFilter coming
#after us won't split it into two words.

# Synonym mappings can be used for spelling correction too
pixima => pixma

#etroove synonyms
sonar,sonarqube
comment,tuto,tutoriel,tutoriaux,tutorial,tutorials
guide,guides
fiche,fiches
junit,junit4,junit5
postgre, postgresql, sql

```

Annexe N : extrait du fichier de configuration mapping-FoldToASCII.txt, utilisé lors du filtrage de la requête utilisateur

D: > Programs > solr-8.11.1 > server > solr > etrouve2 > conf > mapping-FoldToASCII.txt

```
712
713   # è  [LATIN SMALL LETTER E WITH GRAVE]
714   "\u00E8" => "e"
715
716   # é  [LATIN SMALL LETTER E WITH ACUTE]
717   "\u00E9" => "e"
718
719   # ê  [LATIN SMALL LETTER E WITH CIRCUMFLEX]
720   "\u00EA" => "e"
721
722   # ë  [LATIN SMALL LETTER E WITH DIAERESIS]
723   "\u00EB" => "e"
724
725   # ē  [LATIN SMALL LETTER E WITH MACRON]
726   "\u0113" => "e"
727
728   # ĕ  [LATIN SMALL LETTER E WITH BREVE]
729   "\u0115" => "e"
730
731   # ė  [LATIN SMALL LETTER E WITH DOT ABOVE]
732   "\u0117" => "e"
733
734   # ę  [LATIN SMALL LETTER E WITH OGONEK]
735   "\u0119" => "e"
736
737   # ě  [LATIN SMALL LETTER E WITH CARON]
738   "\u011B" => "e"
739
740   # ə  [LATIN SMALL LETTER TURNED E]
741   "\u01DD" => "e"
742
743   # ȅ  [LATIN SMALL LETTER E WITH DOUBLE GRAVE]
744   "\u0205" => "e"
745
746   # ȇ  [LATIN SMALL LETTER E WITH INVERTED BREVE]
747   "\u0207" => "e"
748
749   # ȩ  [LATIN SMALL LETTER E WITH CEDILLA]
750   "\u0229" => "e"
```

Annexe O : page d'aide à l'utilisation d'etroove - utilisation de la syntaxe Lucène

Mentions préalables :

- le moteur de recherche attend par défaut une **orthographe exacte** des mots-clés saisis pour retourner un résultat ➔ utiliser les opérateurs et wildcards pour une recherche plus large
- une correction des mots-clé saisis vous est proposée en autocomplétion lorsqu'ils ne permettent pas directement de récupérer un résultat de recherche
- le moteur de recherche utilise la syntaxe Lucène.
- **opérateur logique par défaut : AND**

  Exemple : **rest jolokia** aura le même comportement que **rest AND jolokia**

Mots-clés :

- Terme unique : entrez un mot-clé tel quel.
- Termes multiples : entrez une série de mots-clés (rappel : AND par défaut !)
- Phrase : les phrases sont des séquences de mots définies dont l'ordre importe ➔ utiliser les guillemets

  Exemple : **"L'outil Jolokia permet d'exposer tous les Mbeans via http/REST"**

Opérateurs booléens :

Par défaut l'opérateur AND est utilisé. Pour modifier le comportement vous pouvez utiliser :

- **AND** : exclue les pages ne contenant pas les termes spécifiés.

  Exemple : **rest AND jolokia** présente les sites contenant ces deux termes, mais pas ceux contenant uniquement l'un des deux

- **OR** : l'opérateur permet de rechercher un terme, ou un autre.
  (Attention : doit être saisi en **majuscule** pour être pris en compte !!!)

- **+** : avec le signe « plus », vous établissez un certain cas du lien « ou ». Si vous placez le caractère directement devant un mot, ce terme doit apparaitre, tandis que l'autre est facultatif.

  Exemple : **+rest OR jolokia**

- **–** : le signe « moins » permet d'exclure un terme (peut être remplacé par l'opérateur **!** ou **NOT**)
  A noter : ne peut être utilisé que pour des recherches comportant plusieurs termes.

  Exemple : **rest -jolokia** ≡ **rest !jolokia** ≡ **rest NOT jolokia**

  Exemple : **rest -jolokia** permet de connaître les pages qui contiennent le mot-clé "rest", mais exclue celles qui contiennent "jolokia".

- **( )** : cet opérateur permet de grouper les mots-clés d'une recherche pour assurer un comportement bien précis.

  Exemple : **rest (GET OR POST)** vous liste des résultats portant sur l'une ou l'autre de ces thématiques.

Opérateurs avancés :

- **source:** permet de spécifier dans quelle source de documentation interne effectuer la recherche.
  A choisir parmis :
  - comitetechnique
  - **docarchi**
  - **docarchifront**
  - **docinstalleur**
  - eforum
  - **veilletechno**
  - **wikefluid**

  Exemple : **source:docarchi rest** retournera l'ensemble des résultats pour le mot-clé "rest" dans la docarchi uniquement.

  Exemple : **source:ef*** retournera l'ensemble des résultats dans eforum.

  Exemple : **source:(docarchi wikefluid) rest** retournera l'ensemble des résultats pour le mot-clé "rest" dans la docarchi ET wikefluid.

- **category:** permet la recherche dans les sous-catégories de chaque source de documentation interne (➔ récupérer l'architecture des sources ci-dessus)

  Exemple : **source:docarchi category:backend** retournera l'ensemble des résultats de la docarchi relatifs au backend.

- **title:** pour une recherche spécifique dans le titre de la documentation.
- **content:** pour une recherche spécifique dans le contenu de la documentation.

Wildcards :

- * : l'astérisque est souvent utilisée pour connaître l'intégralité d'une phrase ou d'une expression.

  Exemple : **Qui vole * vole *** permet de retrouver l'expression "qui vole un œuf vole un bœuf"

  Exemple : **rest jolok*** permet de retrouver l'expression "rest jolokia"

- **?** : l'opérateur permet de remplacer une lettre dans un mot-clé

  Exemple : **r?st** permettra de retrouver "rest"

Types de recherche :

- **~** : l'opérateur tilde **~** permet la recherche approximative (Fuzzy search)

  Exemple : **migre~1** permettra de récupérer des résultats de recherche pour les mots-clés "migre", "migré", migrés, "migrée", "migrer"

- **" "~** : associé aux guillements, le tilde **~** permet également une approximation pour les phrases

  Exemple : **"rest jolokia"~5** recherchera ces deux termes même s'il y a 5 autres mots entre eux

  Exemple : **"rest jolokia"~1** permettra de trouver les documents contenants les phrases **"rest API jolokia"** et **"jolokia rest"** (distance d'édition = 1), mais pas **"rest API via jolokia"** ni **"jolokia API rest"**

- **TO** : l'opérateur de recherche par plage (Range search) permet de rechercher entre deux termes dans une zone spécifique

  Exemple : **migra* TO migre*** recherchera toutes les occurences entre ces deux termes et leurs extensions (ici toutes les variantes de migra* à migre*)

- **/[abc]def/** : Recherche via Regex, pour ce faire, utiliser des slash et des crochets

  Exemple : **/junit[45]/**

Caractères réservés et échappement :

Tous les caractères Unicode peuvent être utilisés dans le modèle, mais certains sont réservés et doivent être échappés. Les caractères réservés de base sont:

. ? + * | { } [ ] ( ) " \

Tous les caractères réservés peuvent être échappés avec un slash inversé : *
Cela comprend également le caractère du slash inversé lui-même : \\

Annexe P : consultation des données de logs du serveur Apache hébergeant les sites Web de documentation via Grafana

Glossaire

Bean
Classe Java définie dans un projet. Contient les attributs et méthodes d'une entité.

Collection
Dans Solr une collection représente un ou plusieurs documents regroupés dans un seul index logique à l'aide d'une configuration et d'un schéma uniques.

Core
Une instance Solr individuelle (représente un index logique).

Crawl
Le Crawl désigne l'exploration d'un site web par le robot d'un moteur de recherche, en cliquant sur les liens proposés, afin de découvrir toutes les pages du site au travers d'une navigation naturelle.

Document
Dans Solr, correspond à un groupe de champs et leurs valeurs. Les documents sont l'unité de données de base d'une collection. Les documents sont attribués aux partitions à l'aide d'un hachage standard ou en attribuant spécifiquement une partition dans l'ID du document. Les documents sont versionnés après chaque opération d'écriture.

Field
Le contenu à indexer/rechercher ainsi que les métadonnées définissant comment le contenu doit être traité par Solr.

Framework
Désigne en programmation informatique un ensemble d'outils et de composants logiciels à la base d'un logiciel ou d'une application.

Frontière
Une frontière de crawl (ou d'exploration) est l'un des composants qui composent l'architecture d'un robot d'indexation Web (ou Web crawler). La frontière de crawl contient la logique et les politiques suivies par un robot lorsqu'il visite des sites Web. Elle est initialisée avec une liste d'URL de départ (cf. seed.txt).

Hadoop
Framework* de programmation basé sur Java permettant de traiter de larges ensembles de données au sein d'un environnement distribué. En fonction des besoins, on peut passer d'un serveur unique à plusieurs milliers de machines grâce à au scaling up*.
À l'origine, Apache Hadoop fut créé pour répondre aux besoins d'entreprises comme Yahoo et Google.

Highlighting
Composant offrant la possibilité de proposer un extrait pertinent d'un document, en relation avec les mots-clés d'une recherche qu'un utilisateur vient d'effectuer. (cf. Annexe H)

Indexation
Phase d'inscription d'un site web dans les différents "index" des moteurs de recherche.

Indexation inversée
Façon de créer un index consultable et qui répertorie chaque mot et les documents qui contiennent ces mots, semblable à un index à la fin d'un livre qui répertorie les mots et les pages sur lesquelles ils peuvent être trouvés. Lors de la recherche de mots-clés, cette méthode est considérée comme plus efficace que l'alternative, qui consisterait à créer une liste de documents associés à chaque mot utilisé dans chaque document.

Javadoc
Il s'agit d'un outil développé par Oracle, permettant de créer une documentation d'API en format HTML depuis les commentaires présents dans un code source en Java.

Métadonnées
Littéralement il s'agit de données à propos de données. Les métadonnées sont des informations relatives à un document, comme son titre, son auteur, ou encore sa catégorie.

PoC
Un Proof of Concept, ou preuve de concept en français, est une méthode qui permet d'évaluer la faisabilité d'un projet. Il fait partie du cadrage du projet et est notamment très utilisé dans les méthodes Agile.

Profondeur de page
Appelée également "profondeur de clic" ou "crawl depth", il s'agit du nombre de clics nécessaires pour atteindre une page à partir de la page d'accueil d'un site Web, en utilisant le chemin le plus court.

Progiciel
Le terme progiciel est issu de la contraction entre les mots professionnel et logiciel. Il est aussi connu sous la dénomination ERP en anglais (Enterprise Resource Planning) ou PGI en français (Progiciel de gestion intégré). Un progiciel est un logiciel d'application fourni par un éditeur qui propose des fonctions polyvalentes pour les professionnels. Il est conçu pour répondre aux usages les plus courants et aux différents besoins d'une entreprise. Il permet donc d'effectuer plusieurs tâches très différentes mais liées entre elles par un domaine d'activités. Un progiciel peut par conséquent être composé de plusieurs logiciels.

Query Parser
Analyseur de requête en français, il traite les termes saisis par un utilisateur.

Regex
En informatique, une expression régulière (formé depuis l'anglais « regular expression ») ou motif est une chaîne de caractères qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles.

RequestHandler
Paramètres de logique et de configuration (cf. Annexe I) qui indiquent à Solr comment gérer les requêtes entrantes, que les demandes visent à renvoyer des résultats de recherche, à indexer des documents ou à gérer d'autres situations personnalisées.

Robot d'indexation
cf. Web crawler

Scaling up
Augmentation des performances par amélioration des équipements

SearchComponent
Paramètres de logique et de configuration (cf. Annexe J) utilisés par les requestHandlers pour traiter les requêtes visant à récupérer un résultat. Des exemples de searchComponent : proposition de correction de l'orthographe, proposition d'extraits pertinents d'un document en fonction de la recherche de l'utilisateur.

Seed.txt
Fichier contenant les URL des pages de départ à partir desquelles l'on souhaite extraire les informations. Une fois la frontière initialisée avec ce fichier, le crawler lui demande quelles pages doivent être visitées ensuite. Au fur et à mesure que le robot commence à visiter les pages et obtient des résultats, il informe la frontière de chaque réponse de page ainsi que des hyperliens extraits contenus dans la page. Ces liens sont ajoutés par la frontière en tant que nouvelles demandes de visite conformément aux politiques définies.

Schéma Solr (managed-schema et schema.xml)
Le schéma d'index Solr (voir Annexe K) définit les champs à indexer et le type du champ (texte, nombres entiers, etc.). Par défaut, les données du schéma sont conservées dans un fichier nommé managed-schema que Solr modifie selon les besoins. Cependant une collection peut être configurée pour utiliser un schéma statique, qui n'est chargé qu'au démarrage à partir d'un fichier de configuration modifié par l'homme - généralement nommé schema.xml.

Sélecteur CSS
Expression qui indique au navigateur à quelle entité HTML s'applique la règle CSS correspondante. Dans le cadre du Web scraping, les sélecteurs sont utilisés pour cibler les éléments HTML de manière très fine.

Slug
Le terme “slug” désigne les parties d’une URL composées d’un ou plusieurs mots et qui peuvent à la fois être lues par les moteurs de recherche et comprises par les utilisateurs. En règle générale, les slugs se trouvent à la fin d’une URL et font clairement référence à une ressource numérique en les mentionnant à l’aide d’une structure d’URL particulière.

solrconfig.xml
Fichier de configuration d'Apache Solr. Il définit les options d'indexation, les RequestHandlers, la proposition d’extrait (Highlighting), la vérification orthographique (Spellchecker) et diverses autres configurations. Le fichier, solrconfig.xml se trouve dans le répertoire /home/conf de Solr.

S**pellchecker**
Composant offrant la possibilité de suggérer d'autres orthographes de termes de recherche à un utilisateur, afin de vérifier les fautes d'orthographe entraînant peu ou pas de résultats.

Stemming
Procédé de réduction des mots à leur forme racine. Utile dans la recherche car les utilisateurs recherchent souvent la racine d’un mot et non le verbe, sa conjugaison ou encore toutes les dérivations possibles (forme plurielle, …).

stopwords.txt
Fichier de configuration (cf. Annexe L) qui regroupe des mots qui ont peu de sens ou de réel intérêt pour la recherche d’un utilisateur, mais qui peuvent avoir été saisis dans le cadre d’une requête en langage naturel. Ils correspondent généralement à des pronoms, conjonctions, prépositions (*e.g.* : « le », « avec », « et »).

synonyms.txt
Fichier de configuration (cf. Annexe M) qui regroupe des termes dont le sens est proche et qui peuvent se substituer les uns aux autres. Dans une implémentation de moteur de recherche, les synonymes peuvent être des abréviations ainsi que des mots ou des termes qui ne sont pas systématiquement coupés. Des exemples de synonymes dans ce contexte seraient « postgre » « postgresql » et « sql », ou encore « sonar » et « sonarqube ».

Syntaxe Lucène
Il s'agit de mot-clés et d'opérateurs que l'on peut utiliser lors de la saisie d'une requête pour définir un comportement spécifique (voir Annexe O).

UniqueKey
Le champ UniqueKey permet l'identification d'un document au sein d'un index. Dans le jargon des bases de données il s'agit de la clé primaire.

UX
Le terme UX provient d'User eXperience ou expérience utilisateur. Le travail de l'UX designer consiste à concevoir une interface accessible et facile à prendre en main.

Web crawler
Un robot d'indexation (en anglais Web crawler ou Web spider, littéralement araignée du Web) est un logiciel qui explore automatiquement le Web. Il est généralement conçu pour collecter les ressources (pages Web, images, vidéos, documents Word, PDF ou PostScript, etc.), afin de permettre à un moteur de recherche de les indexer.

Web scraping
Technique d'extraction du contenu de sites Web dans le but de le transformer pour permettre son utilisation dans un autre contexte, par exemple le référencement.

Wildcard
Une wildcard (ou caractère générique en français) permet de remplacer une ou plusieurs lettres d'un mot pour tenir compte d'éventuelles variations d'orthographe ou de temps.

Table des matières

Résumé

Lorsque l’accès à une ressource d'information est compliquée par la multiplication des canaux de diffusion, associée à la forte augmentation du volume de données, le développement d’un moteur de recherche est une solution qui répond parfaitement à cette problématique. Un tel projet s’articule autour de trois axes.

Le premier avec la création d’un robot d’indexation, ou Web crawler en anglais, dont l’objectif est la recherche des ressources à indexer. La mise en place de plusieurs Web crawlers devient nécessaire pour faire face aux spécifications de chacune des sources d’informations. Le contrôle de leur lancement en multi-threading garantit la rapidité et la robustesse de l’application.

La seconde étape est l’indexation du contenu. En faisant le choix d’utiliser Apache Solr on utilise un outil stable et performant permettant d’obtenir un comportement personnalisé. La configuration d’un analyseur de requête permettant d’utiliser toute la puissance de la syntaxe Lucène, ou encore la mise en place de fonctionnalités à vocation d’assister l’utilisateur (correction orthographique et suggestion de mots-clés real-time, proposition d’extraits pertinents, …) sont autant de paramètres à prendre en considération avant le déploiement de Solr.

Enfin, la mise en place d’une interface utilisateur simple et élégante offre une présentation efficace des résultats de recherche. Une page d’administration permet le contrôle à distance du lancement des crawlers, de connaître leur date de dernier lancement réussi, ou encore le téléchargement d’un rapport d’audit permettant de récupérer des informations sur l’utilisation de l’outil par les utilisateurs.