Active

Auteurs
NDJANDA MBIADA Jacques Charles
(chef de projet)
LEBEAU Christophe
JOOSSEN Matthieu

Enseignants
ZACCHIROLI Stefano
KESNER Delia
SIGHIREANU Mihaela

Préface

Document

Ce document a pour but de décrire les besoins, les exigences, les contraintes et les limites que nous allons devoir respecter durant la réalisation de notre projet.

Il résume les tâches que le programme doit pouvoir effectuer.

Il est composé de d'une première partie ouverte à tout lecteur et d'une partie technique qui explique les spécificités du produit et est plutôt destinée à un public averti.

Il sert également de planning pour délimiter le temps nécessaire à la réalisation de chaque étape du projet et ce dans le but de le terminer à temps.

Versions

Nous souhaitons présenter ci-dessous l'intérêt de chaque version de ce présent document. Les modifications et ajouts importantes.

Version 2

Finalisation de la spécification en vu du début de développement.

Définition du langage de requête et liste des charges.

Version 2 : Beta ← (HEAD)

Définition du langage de requête et support de définition. Choix final de la structure global

Des langages et technologies utilisés.

Version 2 : Alpha

Restructuration du document. Avec respect du modèle donné en cours

Version 1

Brouillon complet avec éléments de spécification et de la structure global du projet

Glossaire

A

arborescence ensemble de fichiers contenus dans un répertoire donné.

B

BI base d'indexation.

C

console logiciel qui émule le fonctionnement d'un terminal informatique.

D

daemon désigne un type de programme informatique, un processus ou un ensemble de processus qui s'exécute en arrière-plan plutôt que sous le contrôle direct d'un utilisateur.

DTD Document Type Definition (littéralement, « Définition de Type de Document »), est un document permettant de décrire un modèle de document SGML ou XML. Le modèle est décrit comme une grammaire de classe de documents : grammaire parce qu'il décrit la position des termes les uns par rapport aux autres, classe parce qu'il forme une généralisation d'un domaine particulier, et document parce qu'on peut former avec un texte complet.

E

expression régulière chaîne de caractère.

F

fichier fichier régulier ou répertoire.

M

métadonnée est une donnée servant à définir ou décrire une autre donnée quel que soit son support.

MI moteur d'indexation.

MR moteur de recherche.

P

parser consiste à analyser un texte ainsi que sa structure syntaxique.

S

socket il s'agit d'un modèle permettant la communication inter processus afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP.

T

TCP Transmission Control Protocol (littéralement, « Protocole de Contrôle de Transmissions »), est un protocole de transport fiable, en mode connecté. Dans le modèle Internet, aussi appelé modèle TCP/IP, TCP est situé au niveau de la couche transport.

X

XML Extensible Markup Language (littéralement, « langage de balisage extensible ») est un langage informatique utilisant des balises (« > » et « < »).

Table des matières

1 Introduction

1.1 Les besoins

Mot(s) du glossaire utilisés : fichier.

Le progrès technologique est indéniable à notre époque, notamment en termes de capacité de stockage des ordinateurs personnels. L'un des enjeux des systèmes pour PC est la gestion des données utilisateur, à savoir permettre à ces derniers d'accéder rapidement à une information datant de plusieurs semaines, mois ou années sans y perdre des heures de recherche.

La capacité de nos disques durs peut atteindre aujourd'hui des milliers de gigaoctets (Go) et nous pouvons donc stocker des milliards de fichiers sur un ordinateur.

Un gestionnaire de système de fichiers est un programme permettant d'accéder à ces données, d'y naviguer et de visualiser leur relation (quel fichiers appartient à quel dossier par exemple). Ce programme n'a pas pour rôle de trier mais de rechercher les fichiers et c'est pourquoi, en l'état, l'utilisateur devra trier son disque dur à la main, si c'est ce qu'il souhaite. Bien que la plupart des systèmes d'exploitation répandus disposent, en plus du gestionnaire de système de fichiers, d'un programme d'indexation, nous souhaitons écrire un programme alternatif au programme d'indexation natif de Linux / Unix (nous n'envisageons pas, du moins dans un premier temps, de développer notre produit pour les systèmes d'exploitation Windows) et pourquoi ne pas le surpasser sur le plan technique et esthétique.

La réalisation du projet ne doit nécessiter aucun fond d'investissement, user uniquement de technologies open sources ou libres d'utilisation sans contraintes particulières sur le produit final. Pour palier à tout dépense supplémentaire (notamment sur le matériel de travail), nous avons à disposition des machines sous différentes plates-formes (NT, Linux, Unix) connectées à internet. Tout module permettant la réalisation du projet est inexistant et devra alors être développé par l'équipe en charge du projet. Le but est bien entendu de rendre le programme terminé, mais également de parvenir à produire ce dernier par une démarche professionnelle et de manière coopérative sous la direction d'un chef de projet.

1.2 Brève description

Mot(s) du glossaire utilisés : arborescence, fichier.

On peut distinguer facilement trois modules dans le projet :

- un moteur de recherche : ce module est en charge notamment de la communication avec l'utilisateur
- une base d'indexation : ce module est en charge de la sauvegarde des données
- un moteur d'indexation : ce module est en charge de la surveillance des fichiers sur une arborescence donnée.

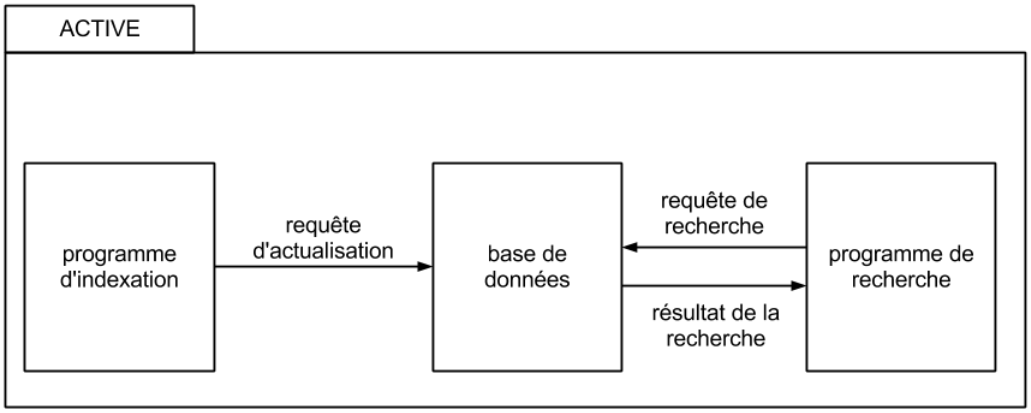

FIGURE 1.1 – Fonctionnement global d'Active

1.3 Objectifs du produit

Mot(s) du glossaire utilisés : arborescence, fichier.

Le produit que nous souhaitons développer devra répondre à certaines normes de communications définies avec d'autres projets afin d'établir une inter-opérabilité au niveau des trois modules principaux, définis brièvement dans la section précédente.

Il devra être opérationnel sur plusieurs architectures matériels afin qu'un utilisateur puisse l'utiliser sur plusieurs systèmes d'exploitation, sans avoir à changer ses habitudes.

Le module « moteur d'indexation » sera chargé d'analyser les modifications sur l'arborescence surveillée et d'en informer le module « base d'indexation » afin que celui-ci mette à jour les informations dans la base de données, qui stocke les informations sur tous les fichiers sauvegardés.

L'arborescence surveillée doit être modifiable par l'utilisateur, ce qui entraîne, à chaque changement, une nouvelle indexation de la nouvelle arborescence à surveiller.

Notre produit devra permettre à l'utilisateur de relancer une indexation complète à tout moment afin de s'assurer du bon contenu de la base de données. Bien évidement, si le produit est utilisé sur un appareil portable, lors d'une telle demande le moteur d'indexation devra prendre en compte l'autonomie restante afin de ne pas empêcher l'utilisateur de pouvoir utiliser son appareil. Dans ce cas un message de confirmation d'indexation immédiate devra être donné à l'utilisateur.

Nous souhaitons que l'utilisateur puisse faire une recherche avec un maximum d'arguments afin de limiter au maximum les résultats, ce qui permet de retrouver plus facilement un fichier.

L'affichage des résultats devra offrir la possibilité d'être triés.

2 Charges d'utilisation

2.1 Généralité

2.1.1 Public visé

Mot(s) du glossaire utilisés : fichier.

Nous visons toute personne utilisant un ordinateur sur lequel notre programme est compatible. Notre programme a pour but de permettre la recherche d'un fichier à l'aide de mots clés. Des options de recherche sont également disponible tels qu'une fourchette de date, le type de fichier. Les mots clé peuvent concerné le nom de fichier et/ou le contenu. Ainsi, si l'utilisateur ne se souvient que de quelques informations sur un de ses fichiers, dont un minimum de mots il peut utiliser notre programme afin de tenter de retrouver son fichier. Dans le cas où notre programme trouve un résultat (un ou plusieurs fichiers) ceux-ci seront affichés à l'utilisateur.

2.1.2 Pré-requis

Mot(s) du glossaire utilisés : console.

Au niveau système, il sera demandé à l'utilisateur d'avoir une machine tournant sous Linux, GNU ou Unix, d'avoir installer au préalable les librairies pour « sqlite3 » ainsi que celles de « tinyxml » afin de pouvoir utiliser notre programme correctement.

Une interface fonctionnant via console sera proposée. Cette interface nécessite un minimum de notion d'utilisation des lignes de commandes. Toutefois, pour les autres utilisateurs, l'utilisation avec interface graphique est équivalente en terme de recherche.

2.2 Spécifications système des utilisateurs

2.2.1 Cas d'utilisation en recherche

Mot(s) du glossaire utilisés : DTD, fichier, XML.

Lorsque l'utilisateur lance le programme, il lui est demandé de donner un ou des mots clés et, si il le désire, de remplir des informations complémentaires (intervalles de temps pour la création et la modification, type de fichier).

La requête utilisateur est alors analysée afin de vérifier si elle est correcte. Si c'est le cas, elle est alors transformée en une ou plusieurs requête XML (conformément à la DTD, p. 21) en fonction de la recherche demandée. Par exemple, la recherche sur un mot contenu dans le fichier, régulier, et sur un autre correspondant au nom du fichier impliquera deux requêtes XML. Il en va de même en cas de conjonction (« et ») et de disjonction (« ou »).

Les requêtes XML, contenant chacune un identifiant différent, sont alors envoyées à la base d'indexation qui va les traiter et envoyer , pour chacune d'elle, une réponse sous forme XML (conformément à la DTD, p. 22) contenant l'identifiant correspondant à la requête de recherche.

Ces résultats vont être alors analysés et concaténés en fonction de la recherche initiale (par exemple, pour les conjonctions une intersection des résultats sera faite) puis affichés à l'utilisateur.

Voir cas d'utilisation p. 23 et diagramme de séquence p. 24.

2.2.2 Cas d'utilisation en indexation

Mot(s) du glossaire utilisés : daemon, fichier, socket.

L'utilisateur peut à tout moment interagir avec le moteur d'indexation, via une communication utilisant des sockets. Les opérations permises sont :

- démarrer le daemon (sur un répertoire précis ou non)
- arrêter et redémarrer le daemon (sur un répertoire précis ou non)
- lister les fichiers surveillés
- supprimer la surveillance sur un fichier (récursivement ou non)
- tuer le daemon

Voir le diagramme de décision p. 28.

3 Architecture du système

3.1 Vue d'ensemble

Mot(s) du glossaire utilisés : fichier.

Le projet que nous développons a pour but d'être compatible avec des machines équipée d'un système d'exploitation Linux, GNU ou Unix sur lequel les librairies « sqlite3 » et « tinyxml » auront été installée au préalable (car nécessaires pour le bon fonctionnement du programme).

Les exécutables que nous produisons est issus de code C++ et Java. Pour son fonctionnement, notre programme crée un base de données et vérifie l'existence de celle-ci à chaque lancement. Cette base de données sert à stocker les informations sur les fichiers surveillés.

3.2 Modularité

Le programme regroupe trois principaux modules, à savoir la base d'indexation, le moteur de recherche et le moteur d'indexation. Ces trois modules peuvent être remplacés par d'autres modules fonctionnant avec le même mode de fonctionnement (voir l'inter-opérabilité p. 14). Un quatrième module vient se rajouter à ces derniers, le module « common » comportant les outils nécessaires à plusieurs des trois modules précédemment cités (voir les diagrammes de classes p. 31).

3.2.1 Moteur de recherche

Mot(s) du glossaire utilisés : console, DTD, expressions régulières, fichier, XML.

Ce module est celui qui est lancé par l'utilisateur que ce soit avec un interaction avec la console ou via l'interface graphique que nous proposons.

Ce module est celui qui s'occupe de la partie recherche. Il gère l'interaction avec l'utilisateur.

Pour un requête utilisateur plusieurs requêtes XML peuvent être crées. En effet, la DTD ne prend en compte ni la recherche simultanée d'un contenu et d'un

nom de fichier pour les mots clés ni les opérations sur les expressions (tel que les conjonctions, disjonction et plus généralement les expressions régulières). Chacune des requêtes envoyées à la base d'indexation le sera avec un identifiant qui sera réutilisé pour la réponse afin de pour analyser correctement les résultats.

En fonction de la demande initiale de l'utilisateur les requêtes seront alors combinées afin que le résultat corresponde à la requête de l'utilisateur. Dans la fenêtre d'affichage des résultats, les données pourront être triées en fonction d'un critère (nom du fichier, date de création, date de dernière modification, type de fichier, propriétaire, groupe).

3.2.2 Moteur d'indexation

Mot(s) du glossaire utilisés : arborescence, daemon, DTD, fichier, XML.

Il s'agit là d'un programme qui doit être lancé lors du démarrage de la session de l'utilisateur, autrement dit un daemon. Une connexion avec la base d'indexation, qui doit avoir été lancée au préalable, va alors être ouverte afin de permettre l'envoie de message.

Via des outils de surveillance du système, il va être attentif à toutes les modifications intervenant sur l'arborescence surveillée et en informer la base d'indexation. Ces modifications peuvent être une création, une suppression ou encore la modification d'un fichier (voir diagramme de décision p. 27).

Pour chaque modification, un flux XMLX (respectant la DTD correspondante, voir p. 19) va alors être envoyé à la base d'indexation.

L'utilisateur peut, à tout moment, entrer en contact avec le moteur d'indexation pour lui demander de changer l'arborescence à surveiller, lancer une indexation sur l'arborescence actuellement sous surveillance.

En cas de changement d'arborescence, une demande de suppression (de la base de données) de tous les fichier actuellement surveillés est alors demandée. Ce module doit prendre en compte l'autonomie restante lors d'une utilisation sur un ordinateur n'étant pas branché sur le secteur afin de ne pas vider l'autonomie de celui-ci. Le programme doit pouvoir alors se mettre en veille et signaler à l'utilisateur de relancer une indexation lorsque l'autonomie aura atteint un seuil raisonnable d'utilisation.

3.2.3 Base d'indexation

Mot(s) du glossaire utilisés : daemon, DTD, fichier, parser, XML.

La base d'indexation doit être lancée comme serveur de manière à pouvoir accepter deux clients : un moteur de recherche et un moteur d'indexation. Ces communications doivent s'établir de la façon définie par l'inter-opérabilité (voir p. 14). De ce fait, la base d'indexation doit être lancée au démarrage de la session afin que les clients puissent se connecter. La base d'indexation peut donc être considérée comme un daemon

La base d'indexation doit être capable de parser le XML de la DTD, définie p. 19 et de générer les réponses de requêtes utilisateur en respectant scrupuleusement la DTD.

Nous avons décidé de pouvoir insérer dans cette base de données le maximum d'informations sur les fichiers afin que la recherche utilisateur puisse être la plus sélective possible. Un schéma de la base de donnée est disponible en annexe p. 30, dans lequel on peut voir les informations stockées.

Là encore, il faut que les informations reçues respectent la DTD.

En revanche, la DTD ne prévoit pas de réponse du moteur de recherche lors de la communication avec le moteur d'indexation.

3.2.4 Inter-opérabilité

Mot(s) du glossaire utilisés : DTD, TCP.

Notre projet a pour but d'être inter-opérable avec tous les autres projets ayant le même but et utilisant le même protocole de communication entre les modules et la même DTD (voir annexe p. 19) pour le contenu de la communication.

Pour chacune des deux communication la base d'indexation sert de serveur et les moteurs de recherche et d'indexation sont les clients.

Pour chacune des communications trois ports ont été choisis dans l'idée que si le premier est utilisé on passe au deuxième et si celui-ci aussi est occupé, on se connecte au troisième. Si les trois ports sont occupés, la communication ne pourra alors pas s'établir. Les ports choisis pour la communication sont :

– 40000, 40001 et 40002 pour la communication entre la base d'indexation et le moteur d'indexation

- 30000, 30001 et 30002 pour la communication entre la base d'indexation et le moteur de recherche

Dans les deux cas les communications se font en TCP. Cette inter-opérabilité a pour avantage de pouvoir faire évoluer les trois modules de manière indépendante tout en gardant l'intégrité du produit à partir du moment où les normes sont respectées.

4 Spécifications du système

4.1 Au niveau du moteur d'indexation

Mot(s) du glossaire utilisés : arborescence, DTD, fichier.

Il doit pouvoir scanner à tout moment le système (ou du moins la partie de l'arborescence surveillée) et ce de manière intelligente pour faire une mise à jour de la base de données.
Il doit pouvoir afficher les fichiers surveillés ainsi que sa capacité maximale de surveillance, mais également ajouter ou enlever des fichier de la surveillance.
En attente d'événement (modification, création, suppression) il doit attendre sans bloquer. Il doit stocker tous les événement en attendant de les stocker et les supprimer de sa liste une fois la requête envoyée à la base d'indexation (afin de ne pas répéter l'événement).
Il doit accepter qu'un utilisateur se connecte pour communiquer avec lui.
Dans le cas de création, modification de fichier texte ou renommage vers un format textuel le moteur d'indexation doit lancer l'indexation sur ce fichier. Il s'agit là de lire tous les mots contenus dans le fichiers avec leur occurrence afin de calculer leur taux d'apparition dans le fichier. Les mots apparaissant avec un taux d'au moins 5% seront alors indexés dans la base de données. A cette règles échappent les mots usuels (voir les *stop list* française et anglaise p. 33.

Afin de fonctionner correctement sous toutes les architectures que nous visons, nous le munissons d'un sous module « ANotify » qui s'occupe d'utiliser les fonctions système de surveillance sur les fichiers propre au système d'exploitation.
Il doit pouvoir gérer les cas de déconnexion (brutales ou non), en se remémorant de toutes les informations à envoyer.
Il doit pouvoir gérer les mauvaises requêtes utilisateur (non conforme à la DTD). Toutes les requêtes crées par le moteur d'indexation doivent être conforment à la DTD.

Un diagramme de classes du moteur d'indexation est fourni p. 29.

4.2 Au niveau du moteur de recherche

Mot(s) du glossaire utilisés : console, DTD, parser, XML.

Le moteur de recherche ne doit pas s'arrêter brutalement. Toutes les composants de l'interface graphique doivent s'afficher correctement et cela à tout moment (notamment lors de l'affichage des résultats).
Il doit se connecter à la base d'indexation à chaque nouvelle requête et fermer cette connexion lors de la réception des résultats. Il doit transformer la requête utilisateur en flux XML, ce qui peut correspondre à plusieurs flux, afin de communiquer correctement avec la base d'indexation et doit « parser » le flux XML donné en retour par la base d'indexation.
Il doit pouvoir gérer les mauvaises requêtes utilisateur et les réponses des la base d'indexation mal formées (non conforme à la DTD). Toutes les requêtes crées par le moteur de recherche doivent être conforment à la DTD.
Le client doit pouvoir se connecter directement avec la moteur de recherche via console ou via l'interface graphique fournie. L'interface graphique doit être au maximum indépendante du système.
L'utilisation de l'interface graphique fait appel au même programme que l'utilisateur doit utiliser afin de faire une requête. L'interface graphique doit donc pouvoir exécuter un programme externe et récupérer la sortie de celui-ci. L'interface graphique et l'utilisateur doivent respecter le langage définit p. 25 afin que la requête soit valide.

Les coupures de connexions avec la base d'indexation doivent être gérées de manière correcte. L'utilisateur doit être averti de la coupure de connexion et le choix doit lui être laissé d'arrêter ou non la recherche (qui reste en attente de connexion jusqu'à nouvelle connexion).
En cas de problème de connexion il faut empêcher l'utilisateur de faire une nouvelle recherche afin de ne pas bloquer toute l'application et à prendre de l'espace mémoire inutilement.

Un diagramme de classes du moteur de recherche est fourni p. 26.

4.3 Au niveau de la base d'indexation

Mot(s) du glossaire utilisés : DTD, fichier.

La base d'indexation ne doit pas recréer la base de données à chaque lancement. La base d'indexation ne doit en aucun cas se déconnecter d'elle même, qui plus est sans avoir répondu à toutes les requêtes qu'elle a reçu.
Elle doit pouvoir gérer les requêtes des moteur de recherche et d'indexation ne respectant pas la DTD. Toutes les flux crées par la base d'indexation doivent être conforment à la DTD.
A la réception d'une requête utilisateur, l'opération à effectuer sur la base de données est un `SELECT`, en fonction des arguments reçus.
Les opérations possibles a effectuer lors de la réception d'un requête venant du moteur d'indexation sont :

- nouvelle entrée lors de la création d'un fichier, avec toutes les dépendances nécessaires, à l'aide de `INSERT`
- mise à jours d'un champ pour une modification, en utilisant `UPDATE`
- suppression d'un champ pour une suppression, avec `DELETE`

Pour ces trois cas, la modification sur la base de donnée doit se faire de manière complète, c'est à dire sur toutes les entrées correspondantes dans chacune des bases concernées. Autrement dit lors de la suppression d'un fichier il faut parcourir la table des mots pour supprimer toutes les apparitions de ce fichier. Pour la création il faut regarder pour chaque mot à ajouter si il est déjà existant dans la base. Si c'est le cas alors il faut juste rajouter l'identifiant du fichier à la suite des identifiants de fichier, sinon créer une nouvelle entrée pour ce mot avec comme valeur associée l'identifiant du fichier. Dans le cas d'une modification, si le fichier texte de départ était un fichier texte alors il faut faire l'étape de suppression. Si le format d'arrivée est un format texte alors il faut faire l'étape de création.

En cas de coupure de connexion avec un (ou plusieurs) des moteurs, toutes les réponses à envoyer doivent être stockées pendant une certaine durée (10 minutes) dans l'attente que la connexion se refasse.

Un diagramme de classes de la base d'indexation est fourni p. 30.

5 Annexes

5.1 DTD

5.1.1 Base d'indexation ↔ Moteur d'indexation

```
<!-- REGLE GENERALE Chaque fichier Xml doit contenir la
    ligne suivante :
      <?xml version="1.0" encoding="UTF-8"?>
-->

<!ELEMENT INDEXATION (RENOMMAGES?, MODIFICATIONS?,
   SUPPRESSIONS?, CREATIONS?)>
<!-- comme convenue toutes les balises de l'indexation sont
   facultatives -->

<!ELEMENT RENOMMAGES (FICHIERRENOMME)*>
<!ELEMENT MODIFICATIONS (FICHIERMODIFIE)*>
<!ELEMENT SUPPRESSIONS (FICHIERSUPPRIME)*>
<!ELEMENT CREATIONS (FICHIERCREE)*>

<!-- un fichier renomme n'est pas necessairement un fichier
     modifie -->
<!ELEMENT FICHIERRENOMME (PATH, NEWPATH)>
<!-- un fichier modife necessite une re-indexation -->
<!ELEMENT FICHIERMODIFIE (PATH, DATEMODIFICATION, TAILLE,
   PROPRIETAIRE, GROUPE, PERMISSIONS, INDEXAGE, NEWPATH?)>
<!ELEMENT FICHIERSUPPRIME (PATH)>
<!ELEMENT FICHIERCREE (PATH, format, DATECREATION, TAILLE,
   PROPRIETAIRE, GROUPE, PERMISSIONS, INDEXAGE)>

<!-- les meta-donnees -->
```

```
<!ELEMENT PATH (#PCDATA)>
<!ELEMENT format (#PCDATA)>
<!ELEMENT DATECREATION (#PCDATA)>
<!ELEMENT DATEMODIFICATION (#PCDATA)>
<!ELEMENT TAILLE (#PCDATA)>
<!ELEMENT PROPRIETAIRE (#PCDATA)>
<!ELEMENT GROUPE (#PCDATA)>
<!ELEMENT PERMISSIONS (#PCDATA)>
<!ELEMENT INDEXAGE (MOT*)>
<!ELEMENT MOT (#PCDATA)>
        <!ATTLIST MOT frequence CDATA #REQUIRED>
<!ELEMENT NEWPATH (#PCDATA)>

<!-- les id_s seront utlises pour la tracabilitee et la
    detection d_eventuel erreurs, elle sont tout de meme
    facultatives -->

<!ATTLIST RENOMMAGES id CDATA #IMPLIED>
<!ATTLIST MODIFICATIONS id CDATA #IMPLIED>
<!ATTLIST SUPPRESSIONS id CDATA #IMPLIED>
<!ATTLIST CREATIONS id CDATA #IMPLIED>
```

5.1.2 Base d'indexation ↔ Moteur de recherche

Recherche

```
<!-- Description de la requete de recherche -->
<!ELEMENT SEARCH (WORD, CONTENT?, PATHDIR?, PERM?,
   EXTENSION?, TIMESLOT?)>
<!ATTLIST SEARCH id CDATA #REQUIRED>
<!-- Le mot a chercher -->
<!ELEMENT WORD (#PCDATA)>
<!-- Un boolean qui dit si l_on fait une recherche de
   contenu (tru) ou une recherche sur les noms de fichier (
   false) -->
<!ELEMENT CONTENT #PCDATA)>
<!-- Le nom du fichier a partir duquel on recherche -->
<!ELEMENT PATHDIR (#PATHDATA)>
<!-- Les permissions du fichier a chercher -->
<!ELEMENT PERM (#PATHDIR)>
<!-- L_extension des fichiers a chercher -->
<!ELEMENT EXTENSION (#PCDATA)>
<!-- Intervalle de temps -->
<!ELEMENT TIMESLOT (BEGIN, END)>
<!-- Debut de l_intervalle -->
<!ELEMENT BEGIN (DAY, MONTH, YEAR)>
<!-- Fin de l_intervalle -->
<!ELEMENT END (DAY, MONTH, YEAR)>
<!-- Le jour -->
<!ELEMENT DAY (#PCDATA)>
<!-- Le mois -->
<!ELEMENT MONTH (#PCDATA)>
<!-- L_annee -->
<!ELEMENT YEAR (#PCDATA)>
```

Résultat

```
<!-- Balise contenant les resultats de la requete search --
   >
<!ELEMENT RESULT (FILE*)>
        <!ATTLIST RESULT id CDATA #REQUIRED>
<!-- Balise file correspond a un fichier resultat donc n
   balises files = n resultats -->
<!ELEMENT FILE (NAME, PATH, PERM, SIZE, LASTMODIF?, PROPRIO
   ?)>
<!-- Le nom du fichier -->
<!ELEMENT NAME (#PCDATA)>
<!-- Le chemin complet du fichier -->
<!ELEMENT PATH (#PCDATA)>
<!-- Les permissions du fichier -->
<!ELEMENT PERM (#PCDATA)>
<!-- La taille du fichier -->
<!ELEMENT SIZE (#PCDATA)>
<!-- La date de derniere modification du fichier -->
<!ELEMENT LASTMODIF (#PCDATA)>
<!-- Le proprietaire du fichier -->
<!ELEMENT PROPRIO (#PCDATA)>
```

5.2 Moteur de recherche

5.2.1 Cas d'utilisation

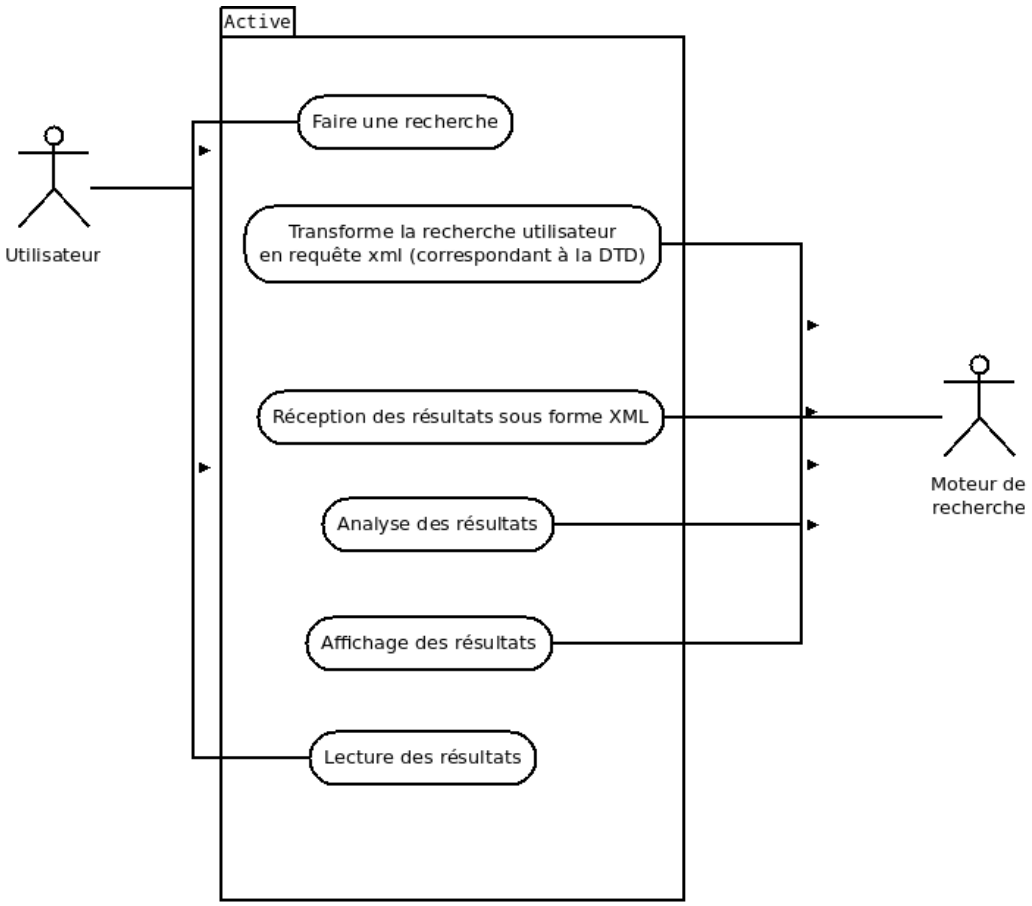

FIGURE 5.1 – Cas d'utilisation en recherche

5.2.2 Diagramme de séquence

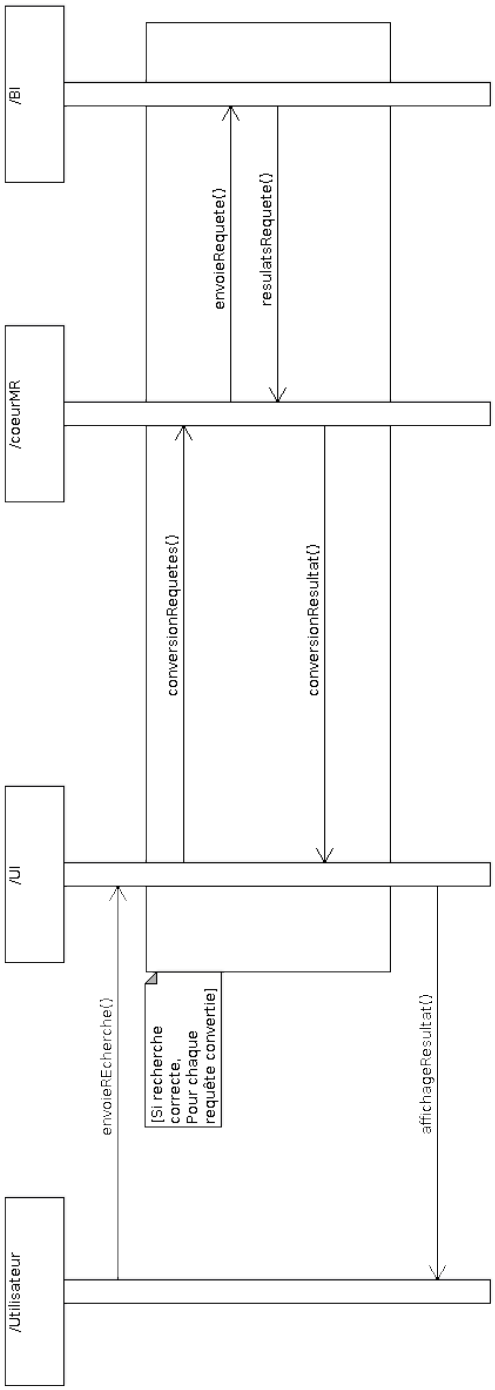

FIGURE 5.2 – Diagramme de séquence en recherche

5.2.3 Langage de communication avec le moteur de recherche

Mis à part `word` aucune des arguments n'est obligatoire. Les arguments peuvent être mis dans un ordre quelconque.
`word` prend en valeur une suite de lettre. Si la recherche doit être faite sur plusieurs mots il faut alors placer entre ces mots soit `|` (en cas de disjonction) ou `&` (en cas de conjonction). Pour `extension` la règle est la même si ce n'est que les chiffres sont autorisés.
`content` doit avoir comme valeur soit `true` soit `false`. La valeur par défaut est `false`.
`permission` doit être une suite de trois chiffres compris entre 0 et 7.
`pathdir` correspond à l'arborescence à partir de laquelle faire la recherche. La valeur de cette option commence soit par une lettre, soit par un chiffre soit par le caractère « . » soit par un underscore. Ce premier caractère peut être suivi par une suite des caractères du même ensemble auquel on ajoute l'espace.
`begin` et `end` sont une suite de 8 chiffres. L'analyse vérifiera que la date est une date existante. Elle est a fournir sous le format « jjmmaaaa ».

```
content=true|false
permission=[0-7]{3}
path=[a-zA-Z0-9\._][a-zA-Z0-9\.\\_ ]*
extension=[a-zA-Z0-9]+([\||&][a-zA-Z0-9]+)*
begin=[0-9]{8}
end=[0-9]{8}
word=[a-zA-Z]+([\||&][a-zA-Z]+)*
```

5.2.4 Diagramme de classes

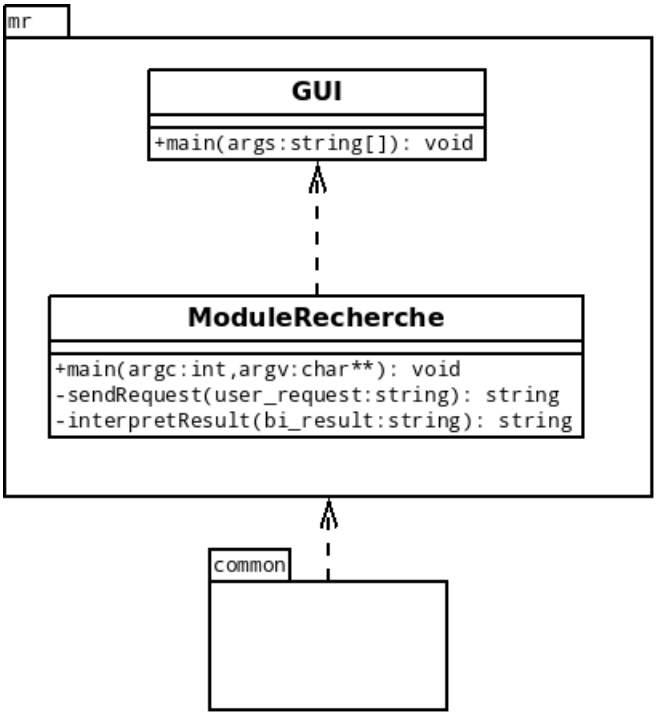

FIGURE 5.3 – Diagramme de classes du moteur de recherche

5.3 Moteur d'indexation

5.3.1 Côté daemon

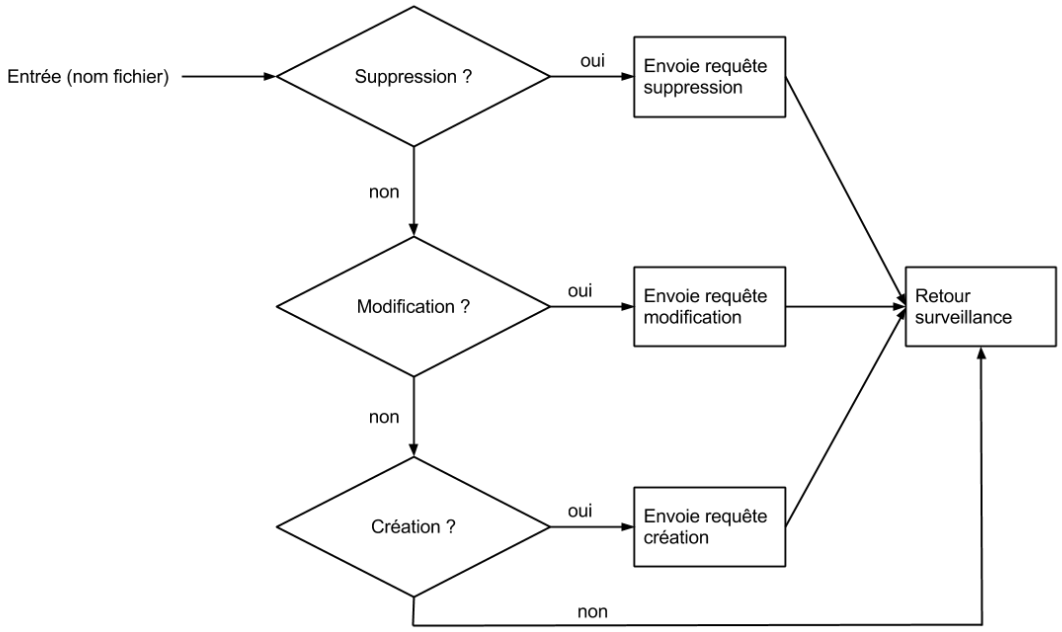

FIGURE 5.4 – Diagramme de décision du moteur d'indexation au niveau du daemon

5.3.2 Côté utilisateur

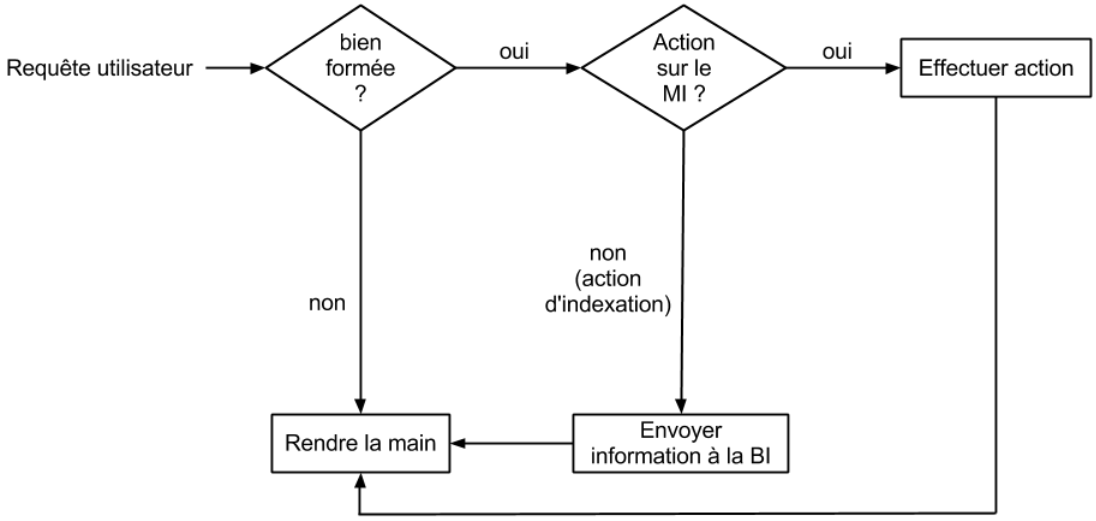

FIGURE 5.5 – Diagramme de décision du moteur d'indexation en interaction avec l'utilisateur

5.3.3 Diagramme de classes

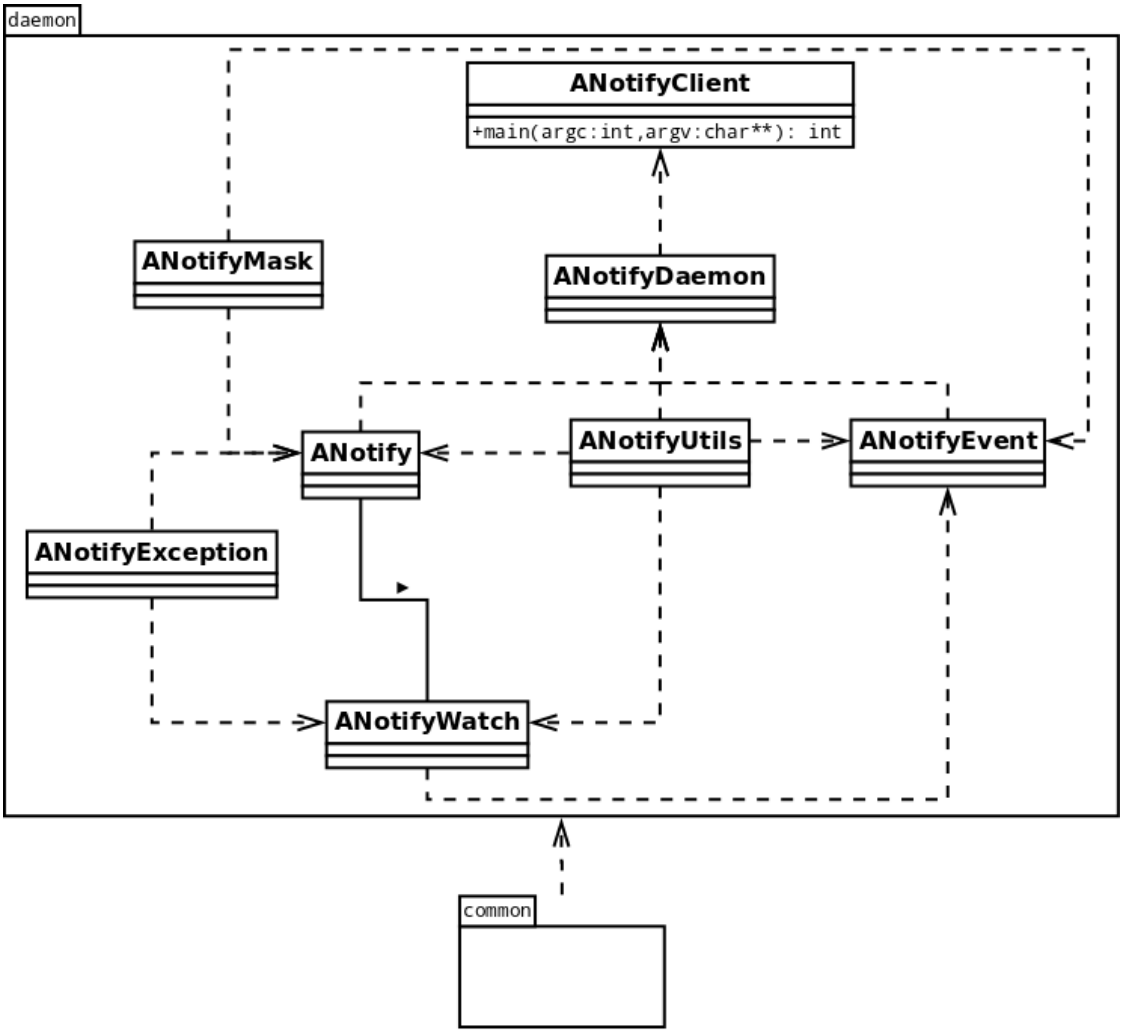

FIGURE 5.6 – Diagramme de classes du moteur d'indexation

5.4 Base d'indexation

5.4.1 Modélisation de la base de données

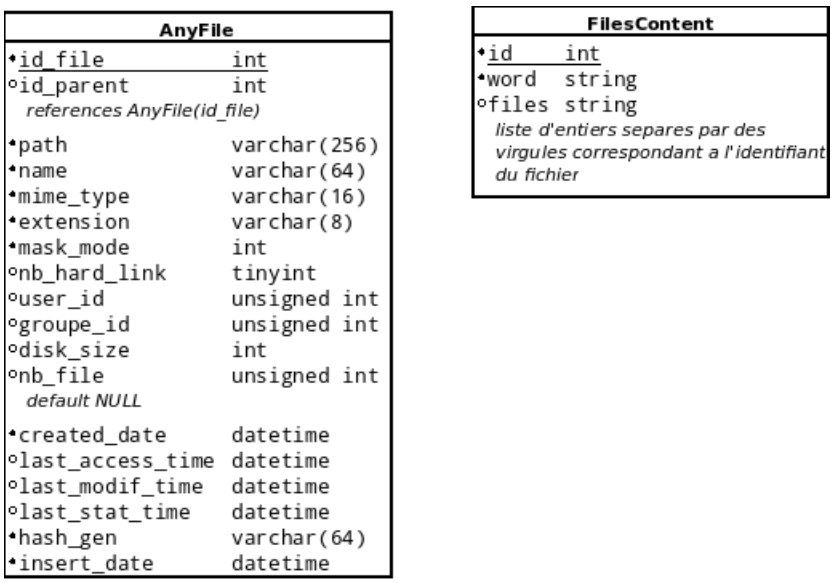

FIGURE 5.7 – Modélisation de la base de données

5.4.2 Diagramme de classes

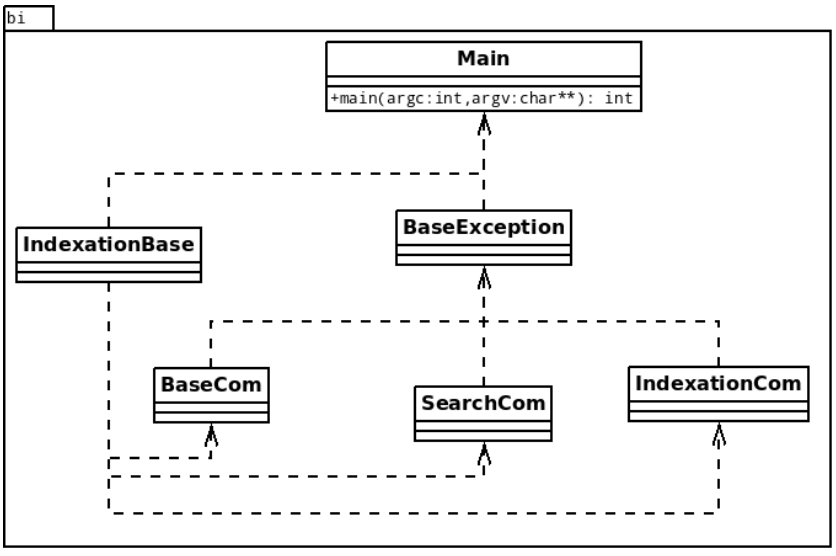

FIGURE 5.8 – Diagramme de classes de la base d'indexation

5.5 Common

5.5.1 Diagramme de classes pour « AQuery »

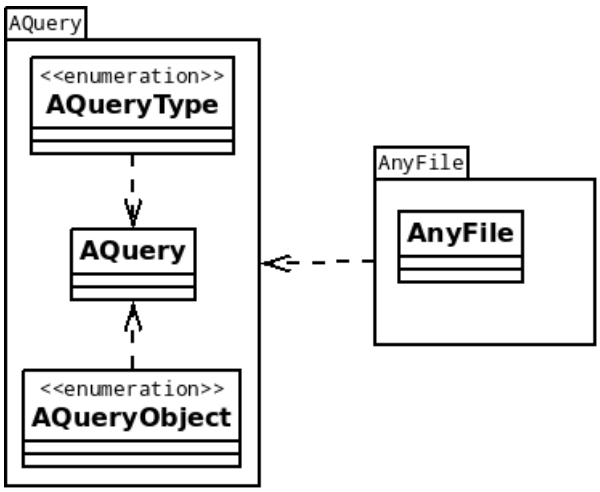

FIGURE 5.9 – Diagramme de classes de la partie relative à « AQuery »

5.5.2 Diagramme de classes pour la génération du XML

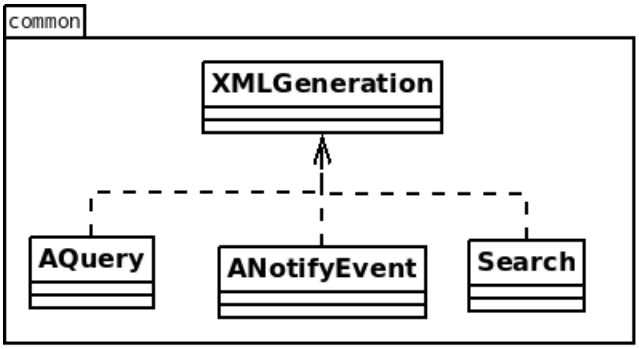

FIGURE 5.10 – Diagramme de classes de la partie relative à la génération XML

5.5.3 Diagramme de classes pour le parser XML

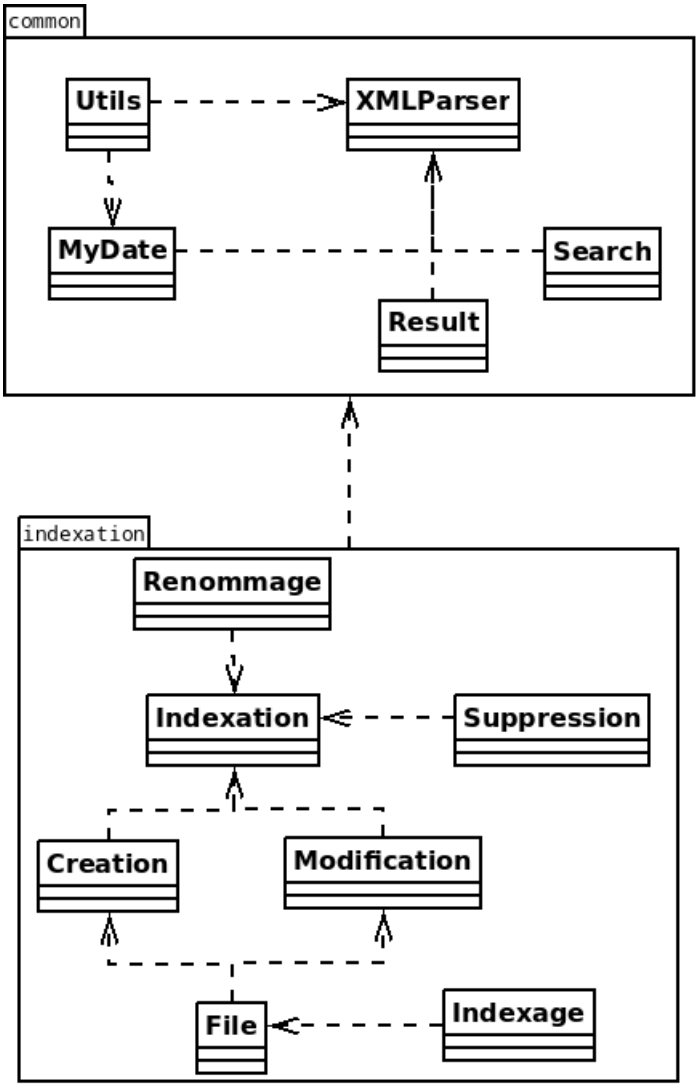

FIGURE 5.11 – Diagramme de classes de la partie relative au parser XML

5.6 Stop list

5.6.1 Stop list française

A : alors, au, aucuns, aussi, autre, avant, avec, avoir

B : bon

C : car, ce, cela, ces, ceux, chaque, ci, comme, comment, ça

D : dans, des, du, dedans, dehors, depuis, deux, devrait, doit, donc, dos, droite, début

E : elle, elles, en, encore, essai, est, et, eu, étaient, état, étions, été, être

F : fait, faites, fois, font, force

H : haut, hors

I : ici, il, ils

J : je, juste

L : la, le, les, leur, là

M : ma, maintenant, mais, mes, mine, moins, mon, mot, même

N : ni, nommés, notre, nous, nouveaux

O : ou, où

P : par, parce, parole, pas, personnes, peut, peu, pièce, plupart, pour, pourquoi

Q : quand, que, quel, quelle, quelles, quels, qui

S : sa, sans, ses, seulement, si, sien, son, sont, sous, soyez, sujet, sur

T : ta, tandis, tellement, tels, tes, ton, tous, tout, trop, très, tu

V : valeur, voie, voient, vont, votre, vous, vu

5.6.2 Stop list anglaise

A : a, about, above, after, again, against, all, am, an, and, any, are, aren't, as, at

B : be, because, been, before, being, below, between, both, but, by

C : can't, cannot, could, couldn't

D : did, didn't, do, does, doesn't, doing, don't, down, during

E : each

F : few, for, from, further

H : had, hadn't, has, hasn't, have, haven't, having, he, he'd, he'll, he's, her, here, here's, hers, herself, him, himself, his, how, how's

I : i, i'd, i'll, i'm, i've, if, in, into, is, isn't, it, it's, its, itself

L : let's

M : me, more, most, mustn't, my, myself

N : no, nor, not

P : of, off, on, once, only, or, other, ought, our, ours, ourselves, out, over, own

S : same, shan't, she, she'd, she'll, she's, should, shouldn't, so, some, such

T : than, that, that's, the, their, theirs, them, themselves, then, there, there's, these, they, they'd, they'll, they're, they've, this, those, through, to, too

U : under, until, up

V : very

W : was, wasn't, we, we'd, we'll, we're, we've, were, weren't, what, what's, when, when's, where, where's, which, while, who, who's, whom, why, why's, with, won't, would, wouldn't

Y : you, you'd, you'll, you're, you've, your, yours, yourself, yourselves

Index