Développement d'un système d'indexation par les compétences avec les technologies du Web sémantique pour Sésamath

Dominique Gentaz

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS
CENTRE RÉGIONAL RHÔNE-ALPES
CENTRE D'ENSEIGNEMENT DE GRENOBLE

MÉMOIRE

présenté par **Dominique Gentaz**

en vue d'obtenir

LE DIPLÔME D'INGÉNIEUR C.N.A.M.

en INFORMATIQUE

Développement d'un système d'indexation par les compétences avec les technologies du Web sémantique pour Sésamath

Soutenu le 27 mai 2013

JURY

Président :	M. Éric Gressier-Soudan (CNAM Paris)
Membres :	M. Claude Genier (CNAM Lyon)
	M. Jean-Pierre Giraudin (Université de Grenoble)
	M. Cyrille Desmoulins (maître de stage)
	M. Frédéric Bianchi (CNAMTS)

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS
CENTRE RÉGIONAL RHÔNE-ALPES
CENTRE D'ENSEIGNEMENT DE GRENOBLE

MÉMOIRE

présenté par **Dominique Gentaz**

en vue d'obtenir

LE DIPLÔME D'INGÉNIEUR C.N.A.M.

en INFORMATIQUE

Développement d'un système d'indexation par les compétences avec les technologies du Web sémantique pour Sésamath

Soutenu le 27 mai 2013

Les travaux relatifs à ce mémoire ont été effectués au Laboratoire Informatique de Grenoble, au sein de l'équipe MeTAH, sous la direction de Cyrille Desmoulins, tuteur de stage.

Avant-propos

Remerciements

J'exprime tout mon respect et mes remerciements à l'ensemble des membres du jury qui est présent pour ma soutenance. Je sais que le temps consacré à ce type d'événement est important et il ne faut pas l'oublier.

Je remercie particulièrement mon tuteur, M. Cyrille Desmoulins, pour ses conseils, son soutien et sa disponibilité tout au long du stage.

Je salue les membres de l'équipe MeTAH du LIG. Ceux-ci m'ont aimablement accueilli dans leur environnement et m'ont ainsi permis de découvrir le contexte et l'ambiance d'un laboratoire de recherche.

Un grand merci à Paul Libbrecht pour son aide précieuse lors de l'étude et de la mise en place de la plate-forme d'Intergeo. Que d'heures passées sur Skype afin de compiler et réinstaller l'ensemble des composants du site. Merci pour sa patience et sa gentillesse.

Je remercie Daniel Caillibaud, membre de l'association Sésamath, qui nous a si bien reçu dans son logement parisien lors du démarrage du projet au mois d'avril 2012. Son aide fut aussi précieuse, pour appréhender le cadriciel Symfony2, lors de l'intégration du projet au niveau de la « bibliothèque » Sésamath. Ses conseils et recommandations en terme de développement m'ont permis de progresser dans ce domaine qui m'était peu familier. Je remercie aussi Sébastien Hache, membre de l'association Sésamath et maîtrise d'ouvrage sur le projet.

Je salue Alban Chazot du LIG et le remercie de son implication lors de la mise en place de la machine virtuelle de démonstration du projet.

Je remercie les membres de l'association AICNAM-PST pour leur précieuse aide quant à la relecture de ce mémoire et à la préparation de la soutenance. Une pensée particulière pour mon parrain Philippe Bollard qui m'a encouragé au cours de cette période.

Je salue mes collègues et responsables de la CNAMTS et les remercie pour leurs témoignages d'amitié tout au long de mon exil universitaire.

Enfin, je dédie ce mémoire à ma compagne et la remercie pour sa patience et d'avoir assumé la gestion du quotidien lorsque le temps me manquait.

A ma fille Nausicaa :
J'espère maintenant pouvoir être encore plus présent et profiter de te voir grandir.

Liste des acronymes

AJAX :	Asynchronous JavaScript and XML
API :	Application Programming Interface
BASH :	Bourne-Again SHell
CNAMTS :	Caisse Nationale d'Assurances Maladie des Travailleurs Salariés
CNIL :	Commission Nationale de l'Informatique et des Libertés
CRIM :	Centre de Recherche Informatique de Montréal
CNRS :	Centre National de la Recherche Scientifique
CSS	Cascading Style Sheets
CSV :	Comma-Separated Values
DAO :	Data Access Object
DOM :	Document Object Model
EIAH :	Environnement Informatiques pour l'Apprentissage Humain
GUI :	Graphical User Interface
GWT :	Google Web Toolkit
HTML :	Hypertext Markup Language
HTTP:	HyperText Transfer Protocol
IDF :	Inverse Document Frequency
IHM :	Interface Homme Machine
INPG :	Institut National Polytechnique de Grenoble
INRIA :	Institut National de Recherche en Informatique et en Automatique
IP :	Internet Protocol
JAR :	Java ARchive
JAVA2 EE :	Java Enterprise Edition
JDBC :	Java DataBase Connectivity
JDOM :	Java Document Object Model
JSON :	JavaScript Object Notation
JSP :	JavaServer Page

JVM :	Java Virtual Machine
LDAP:	Lightweight Directory Access Protocol
LIG :	Laboratoire d'Informatique de Grenoble
METAH :	Modèles et Technologies pour l'Apprentissage Humain
MVC :	Modèle Vue Contrôleur
ORM	Object-Relational Mapping
OWL :	Ontology Web Language
PDF :	Portable Document Format
PHP-FPM :	Hypetext Preprocessor FastCGI Process Manager
POM :	Project Object Mode
PPT :	Microsoft PowerPoint
QCM :	Questionnaire à Choix Multiple
RAM :	Random Access Memory
REST :	REpresentational State Transfer
RTF :	Rich Text Format
SGBD :	Système de Gestion de Base de Données
SNMP :	Simple Network Management Protocol
SOAP :	Simple Object Access Protocol
SSO	Single Sign-On
SSL :	Secure Sockets Layer
SVN :	SubVersioN
SQL :	Structured Query Language
TF :	Term Frequency
TICE :	Technologies de l'Information et de la Communication pour l'Enseignement
URI :	Uniform Resource Identifier
URL :	Uniform Resource Locator
XML :	eXtensible Markup Language
W3C :	World Wide Web Consortium
WEBDAV :	Web-based Distributed Authoring and Versioning
XMPP :	Extensible Messaging and Presence Protocol

Sommaire

Table des figures

Liste des tableaux

Introduction et contexte du stage

Le Laboratoire d'Informatique de Grenoble

Le Laboratoire d'Informatique de Grenoble (LIG) est réparti sur le site de Montbonnot et sur le domaine universitaire de Grenoble. Le LIG a pour principaux partenaires académiques le CNRS, Grenoble INP, INRIA Grenoble Rhône-Alpes, l'Université Joseph Fourier, l'Université Pierre-Mendès-France et l'Université Stendhal. Il rassemble près de 500 chercheurs, enseignants-chercheurs, doctorants et personnels en support à la recherche, regroupés en 23 équipes structurées autour de 5 axes de recherche :

- Génie des Logiciels et des Systèmes d'Information ;
- Méthodes Formelles, Modèles et Langages ;
- Systèmes Répartis, Calcul Parallèle et Réseaux ;
- Traitement de Données et de Connaissances à Grande Échelle ;
- Systèmes Interactifs et Cognitif. [LIG 2012].

Parmi ces 23 équipes, l'équipe MeTAH (Modèles et Technologies pour l'Apprentissage Humain) rassemble informaticiens et didacticiens autour de la question de la conception, du développement et des usages des Environnements Informatiques pour l'Apprentissage Humain (EIAH). Elle se compose d'une trentaine de personnes dont une vingtaine de permanents. MeTAH se donne pour objectif de comprendre comment les dimensions éducatives (didactiques ou pédagogiques) et les usages peuvent être pris en compte dans :

- la conception d'artefacts informatiques techniques (micro mondes, simulations, tuteurs intelligents, jeux pour l'apprentissage, environnements collaboratifs) ;
- la conception de descriptions calculables de leur utilisation (scénarios d'apprentissage, d'encadrement) ;
- la conception de modèles computationnels des connaissances épistémiques et didactiques et de fonctionnalités associées (mécanismes de rétroaction, supervision) [METAH 2012].

Certains membres de l'équipe MeTAH ont notamment participé au projet européen « Intergeo » en partenariat avec d'autres centres de recherche, universités européennes et des membres de l'association Sésamath. Le projet dans lequel se situe mon stage est une déclinaison de ce qui a été réalisé dans le cadre d'Intergeo afin de mettre en œuvre un système d'indexation des ressources Sésamath, par les compétences (capacités et savoirs) avec les technologies du Web Sémantique. Il vise à développer un tel système d'indexation basé sur une ontologie. Il doit permettre à un enseignant de retrouver des ressources correspondant à l'objectif ou l'activité pédagogique qu'il envisage avec un des outils Sésamath. De façon plus ambitieuse, un tel système peut aussi servir de base pour assister l'enseignant dans l'évaluation des élèves ou dans la construction de séances ou de parcours pédagogiques.

Le Projet Intergeo

Le projet Intergeo est un projet subventionné en partie par la Communauté Européenne sous le programme eContentPlus dont l'objectif principal est la mise à disposition large et accessible de contenus digitaux. Ce projet s'attaque à trois freins à l'adoption de la géométrie dynamique par les enseignants et à l'utilisation des ressources existantes :

- **le manque d'outils de partage de ressources :** les ressources utilisant la géométrie dynamique sont très nombreuses mais sont dispersées sur de multiples serveurs ; il est difficile et souvent coûteux pour un enseignant de trouver la ressource adaptée à ses besoins et son public. Le projet rassemble un grand nombre de ressources existantes (plus de 3 300) sur un serveur où elles sont enrichies avec des métadonnées pédagogiques multilingues qui simplifient la recherche d'exemples adaptés à un contexte pédagogique donné. De plus, les questions relatives à la propriété intellectuelle sont clarifiées et les enseignants peuvent être certains d'avoir le droit d'utiliser les ressources ;
- **le problème de l'interopérabilité :** différents logiciels de géométrie dynamique existent et les ressources sont le plus souvent développées que pour un logiciel en particulier. Un format de fichier commun permet aux enseignants d'utiliser les ressources trouvées avec le logiciel de leur choix ;
- **l'absence d'évaluation de la qualité :** les ressources disponibles ne sont pas toujours pertinentes pour une utilisation en classe et lorsqu'il y a plusieurs ressources aucun élément ne permet de guider le choix de l'enseignant. Les ressources déposées sur le serveur Intergeo sont largement testées en classe et des rapports d'utilisation, disponibles en ligne permettent d'évaluer leur qualité. Des experts en didactique et des groupes d'enseignants analysent ces résultats et les données ainsi obtenues permettent aux enseignants de choisir en toute confiance une situation pédagogique donnée.

La figure 1 montre l'interface d'accueil du site accessible par l'adresse http://i2geo.net.

Figure 1 - Site i2geo.net

Après la durée officielle de trois ans du projet, l'infrastructure a été transférée à la communauté et le code source de l'ensemble de la plate-forme est sous licence libre depuis fin septembre 2010.

L'équipe MeTAH s'est impliquée dans le projet Intergeo d'une part en développant une ontologie des capacités et notions des programmes de mathématiques à travers l'Europe permettant la description, la classification automatique et la recherche de ressources de mathématiques, et d'autre part en développant un système d'évaluation des ressources du portail basé sur des dimensions didactiques. L'association Sésamath était partenaire associé au projet.

L'association Sésamath

L'association Sésamath a pour vocation essentielle de mettre à disposition de tous, gratuitement, des ressources pédagogiques et des outils professionnels libres utilisés pour l'enseignement des mathématiques via Internet [SESAMATH 2012]. Elle offre de nombreux outils, accessibles depuis Internet, aux enseignants ainsi qu'aux élèves du secondaire, parmi lesquels :

- Mathenpoche, dédié aux élèves, pour revoir les cours et s'exercer sur les ressources Sésamath. La figure 2 illustre l'interface du site accessible par l'adresse http://mathenpoche.sesamath.net ;

Figure 2 - Site Mathenpoche

- LaboMep, dédié aux enseignants, pour définir des séances d'activité en ligne avec une classe, sur les ressources de Sésamath. La figure 3 illustre l'interface du site accessible par l'adresse http://www.labomep.net ;

Figure 3 - Page d'accueil de LaboMep

- les Manuels Sésamath, la version en ligne et libre des manuels papiers édités chez « Génération 5 » par Sésamath ;
- MutuaMath, forge documentaire pour les enseignants de mathématiques.

Au total, 21 outils en ligne sont disponibles, la plupart centrés sur les ressources des Manuels Sésamath, et un total d'environ 8 000 ressources. Chaque ressource est élaborée de façon collaborative et bénévole par certains des 15 600 enseignants de la communauté Sésamath. Elle est ensuite référencée par son chapitre et sa section.

Avec la possibilité récente qu'un enseignant ajoute de nouvelles ressources dans LaboMep ou dans MutuaMath, un système d'indexation est nécessaire, comme c'est le cas dans Intergeo.

Problématique et plan du mémoire

Dans le cadre de l'épreuve TEST [GENTAZ 2011], une étude préliminaire avait été réalisée afin de déterminer si l'utilisation d'une ontologie peut être une solution envisageable pour le système de représentation des compétences de Sésamath. Actuellement les ressources ont été construites avec une approche documentaire, par chapitre et section du manuel. Chaque ressource est ainsi repérée par la section qui la comprend et indique sur quoi elle porte. Les ressources qu'un enseignant peut ajouter dans LaboMep ou dans MutuaMath ne sont attachées à aucune section/chapitre et n'ont de ce fait aucun descriptif lié à l'enseignement des mathématiques. Il manque un système d'indexation des ressources par les notions et capacités pour l'enseignement des mathématiques.

Le premier chapitre de ce mémoire présente plus en détail le fonctionnement, la création et la gestion de ressources au sein de l'association Sésamath.

Un second chapitre aborde les réalisations du stage. Il est en effet consacré à l'étude du système d'indexation, de ressources pédagogiques par les notions et capacités, instancié dans le cadre du projet européen Intergeo. Ce chapitre analyse les composants de la plate-forme, les principes mis en œuvre et permet de déterminer la possibilité de reprendre le code d'Intergeo pour le porter dans l'infrastructure Sésamath. Une présentation détaillée de l'API Lucene est aussi exposée.

Avec le troisième chapitre, nous présentons un rapide état de l'art des moteurs d'indexation dont notamment « Apache Solr » et « ElasticSearch » basés sur l'API Lucene. Ce chapitre reprend, principalement, une étude réalisée en mars 2012 par le Centre de Recherche Informatique de Montréal.

Le chapitre suivant détaille le fonctionnement générique du moteur d'indexation Solr ainsi que son adaptation en vue de réaliser le système d'indexation des ressources Sésamath par les capacités et notions. Les prototypes développés pour ce faire sont aussi présentés dans cette partie.

Le dernier chapitre évoque l'intégration des prototypes dans la bibliothèque de Sésamath en cours de développement. Nous traitons des fonctionnalités développées afin de permettre une recherche de ressources par les notions et capacités, en même temps que d'autres.

Enfin, nous terminons ce mémoire par une synthèse du travail réalisé, des perspectives et évolutions possibles pour le futur. Un bilan personnel faisant office de retour d'expérience clôture ce rapport.

1. Sésamath – État des lieux général

L'association Sésamath a été créée le 31 octobre 2001. Elle s'adresse aux professeurs et à leurs élèves. Elle a pour but de promouvoir :

- l'utilisation des TICE[1] dans l'enseignement des mathématiques ;
- le travail coopératif et la co-formation des enseignants ;
- une philosophie de Service Public ;
- des services d'accompagnement des élèves dans leur apprentissage ;
- et plus généralement, toute activité pouvant se rattacher directement ou indirectement à l'un des objets spécifiés, ou à tout autre objet similaire ou connexe, de nature à favoriser, directement ou indirectement les buts poursuivis par l'association, son extension et son développement.

L'objectif de ce chapitre est de présenter l'association, son fonctionnement ainsi que les différents sites ouverts aux enseignants et aux élèves. Une analyse détaillée de la gestion des ressources permettra de comprendre la problématique et les solutions pouvant être mises en œuvre.

1.1 Présentation de Sésamath

L'association Sésamath est le fruit de plusieurs fusions et regroupement de sites Internet issus d'actions individuelles menées par des enseignants de mathématiques au début des années 2000. Le projet s'est développé au fil des ans pour atteindre au premier décembre 2012 :

- 18 067 professeurs inscrits à Sésaprof ;
- 14 705 825 visites sur les différents sites de Sésamath en 2011 ;
- 1 011 792 élèves inscrits à LaboMep depuis le 1er septembre 2012 ;
- 1 600 exercices interactifs et 1 144 animations Instrumenpoche pour les corrigés d'exercices ;
- plus de 400 articles dans la revue Mathematice.

Grâce aux TICE, Internet en particulier, Sésamath favorise les échanges entre les professionnels de l'enseignement auxquels elle s'adresse initialement. Ces échanges, qui constituent un puissant moteur de co-formation, donnent souvent lieu à la création de ressources pédagogiques que l'association diffuse alors gratuitement. Mais Sésamath s'adresse aussi, et de plus en plus, à d'autres publics, en particulier les élèves et leurs parents, en créant des espaces dédiés qui répondent à une demande croissante d'accompagnement.

L'association se reconnaît dans les valeurs d'ouverture, d'échange et de partage véhiculées par les logiciels libres. C'est pourquoi Sésamath utilise dans la mesure du possible des outils libres et des formats ouverts pour les contenus qu'elle produit. Ces outils libres et ces formats ouverts accessibles à tous favorisent ainsi la mise en œuvre d'un large travail collaboratif.

Les ressources proposées sont diverses, complémentaires et libres. Elles peuvent être traduites et adaptées à d'autres pays. Sésamath a la volonté de favoriser ce mouvement en nouant des partenariats internationaux. Sésamath encourage également des travaux en commun avec des équipes de recherche afin d'améliorer constamment la qualité des ressources produites.

1 Les TICE recouvrent les outils et produits numériques pouvant être utilisés dans le cadre de l'éducation et de l'enseignement.

1.2 Infrastructure matérielle et logicielle de Sésamath

1.2.1 Infrastructure matérielle

L'infrastructure Sésamath est hébergée chez OVH qui est un hébergeur de sites Web français. OVH propose notamment des serveurs dédiés, des serveurs privés, de l'hébergement mutualisé, etc.

L'association Sésamath loue des serveurs physiques faisant office de serveurs hôtes afin d'héberger plusieurs machines virtuelles. Le serveur hôte est équipé d'un processeur « Intel Core i5-2400 » de fréquence 4x3.1 GHz et de 16 Go de mémoire vive. Son système d'exploitation est un GNU/Linux Debian Squeeze.

Les machines virtuelles hébergées sont réalisées par le système d'isolation OpenVZ. Ce système permet la virtualisation au niveau noyau du serveur hôte. Il s'agit d'un partitionnement logique au niveau des ressources systèmes : processus, mémoire, réseau et système de fichiers. Ce procédé offre l'avantage de n'apporter aucune charge supplémentaire contrairement à une émulation logicielle du type Vmware, VitualBox ou autre. OpenVZ est donc plus performant mais moins flexible que les autres systèmes car il ne permet de virtualiser que des machines ayant un noyau Linux [OPENVZ 2009].

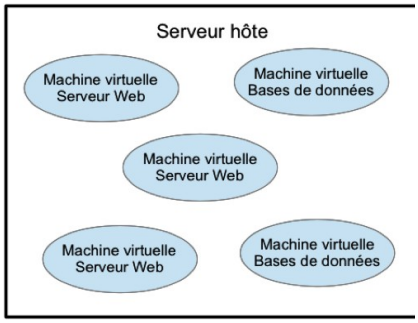


Figure 4 - Architecture Sésamath

Actuellement toute l'architecture Sésamath, illustrée par la figure 4, repose principalement sur 2 serveurs hôtes avec 5 machines virtuelles pouvant être distinguées en 2 grandes catégories de serveur :

- Serveur Web ;
- Serveur bases de données.

Les serveurs Web sont partagés en 3 machines virtuelles hébergeant une vingtaine de sites gérés par Sésamath. Les serveurs bases de données sont répartis sur 2 machines virtuelles pour 3 serveurs MySQL gérant en tout une cinquantaine de bases de données.

1.2.2 Infrastructure logicielle

Il n'y a pas de licences car toute l'infrastructure appartient à Sésamath et s'appuie sur des logiciels Open Source.

Les serveurs Web frontaux utilisent un serveur HTTP Nginx et PHP-FPM version 5.3. Sésamath a choisi la solution Nginx pour les serveurs Web plus rapide pour fournir des fichiers statiques et plus performant qu'un serveur Apache en terme de consommation mémoire pour des connexions simultanées. PHP-FPM est quant à lui un gestionnaire de processus FastCGI pour PHP. Il permet de remplacer le fonctionnement standard de PHP, par un ensemble de processus PHP chargés en mémoire répondant au fur et à mesure aux requêtes transmises par Nginx. L'intérêt est d'éviter une

surconsommation de mémoire et d'entrées-sorties provoquée par le lancement d'un processus PHP à chaque requête. Le choix de ces technologies répond à un besoin d'optimisation des ressources machines du serveur physique hébergeant l'ensemble de l'infrastructure de Sésamath.

Les serveurs de bases de données utilisent MySQL 5.1 et son système de réplication en « maître-maître ». Des scripts « Sésamath » permettent un système de haute disponibilité en utilisant l'API d'OVH pour router une IP sur un serveur physique ou un autre.

Les environnements de pré-production et production sont répartis dans différents hôtes virtuels du même serveur Web dont chacun possède sa base de données dédiée sur le même serveur MySQL.

En fonctionnement « normal », chaque machine virtuelle Web interroge la machine virtuelle MySQL située sur le même hôte afin de gagner en débit et temps de latence. Pour le moment un seul serveur hôte pourrait faire tourner l'ensemble de l'infrastructure sans problème, mais en temps normal la charge est répartie sur 2 hôtes. En cas de problème sur un serveur, un script bascule les machines virtuelles vers l'autre serveur hôte. Il y a un décalage de quelques millisecondes pour la réplication MySQL. La synchronisation des fichiers périodiques est actuellement réalisée toutes les heures. Il n'y a aucune contrainte en terme de sauvegardes qui sont réalisées quotidiennement à chaud sans interruption de service.

La seule contrainte actuelle est que l'ensemble de l'infrastructure puisse continuer à fonctionner sur un serveur hôte unique pour que le système de bascule sur le serveur de réserve fonctionne.

Des tests sur quelques sessions types ont montré que l'infrastructure devrait supporter 10 000 utilisateurs simultanés, ce que l'utilisation réelle semble confirmer, même si l'on en est loin actuellement. Avec le serveur physique actuel, 2 000 utilisateurs simultanés ne représentent que 5% de charge.

1.3 Principaux sites de Sésamath

L'association Sésamath gère plusieurs sites Internet et est aussi associée avec d'autres. Cette partie ne recense que les sites sous la responsabilité de Sésamath.

1.3.1 Pour les élèves : Mathenpoche

Avec le site Mathenpoche, Sésamath a pour ambition de proposer aux élèves un maximum de ressources de tout type : cours, exercices, aides animées, QCM et devoirs pour s'entraîner mais aussi de l'entraînement au calcul mental, des jeux logique, etc.

Ce site dédié aux élèves permet l'accès aux ressources Sésamath. Les ressources sont accessibles par niveau puis chapitre, sous-chapitre, etc. Comme le montre la figure 5, un élève souhaitant travailler une compétence précise, doit donc parcourir l'arborescence et connaître le programme scolaire pour pouvoir trouver les cours et exercices traitant le sujet.

Figure 5 - Interface Mathenpoche

Ce site permet un rappel des notions et capacités que l'élève doit maîtriser pour réaliser l'exercice (section : « je me souviens »). Un cours, section « j'apprends et j'applique », est disponible pour améliorer les connaissances liées au domaine. Enfin, l'élève peut s'exercer en utilisant les ressources Sésamath accessibles par la section « je m'évalue ».

1.3.2 Pour les professeurs

1.3.2.1 Sésaprof

Le site Sésaprof est le portail d'accès aux différents sites Sésamath dédiés aux enseignants. Il permet aux enseignants :

- d'accéder à l'interface professeur de LaboMep ;
- d'accéder à des ressources dédiées, comme par exemple les livres du professeur des manuels Sésamath ;
- d'accéder à des ressources réservées, comme par exemple les corrections des cahiers Sésamath ;
- de tester des avant-premières de logiciels ou de publications, comme par exemple Instrumenpoche ;
- de rejoindre des communautés d'enseignants sur des thèmes donnés ;
- de bénéficier d'informations relatives au système éducatif et aux mathématiques tel que les programmes, actualités, etc. ;
- de communiquer avec des collègues ou l'association par l'intermédiaire de forum, formulaires de contact, annonces ;
- de partager leur expérience, proposer des ressources.

Sésaprof compte, au 29 mars 2013, 20 161 professeurs inscrits [SESAPROF 2012]. Ce site est accessible par l'adresse http://sesaprof.sesamath.net.

1.3.2.2 MutuaMath

Le principe de ce site est le partage de documents pédagogiques, et l'amélioration collective de ceux-ci. Tout enseignant inscrit à Sésaprof peut ajouter ses propres contributions en acceptant que ces ressources soient soumises à la licence CREATIVE COMMONS BY-SA. Pour ajouter une ressource à partager, il est demandé d'utiliser de préférence un modèle spécifié par Sésamath afin d'assurer une certaine cohérence entre les différentes ressources. La figure 6 représente le cycle de vie générique d'une ressource au sein de MutuaMath.

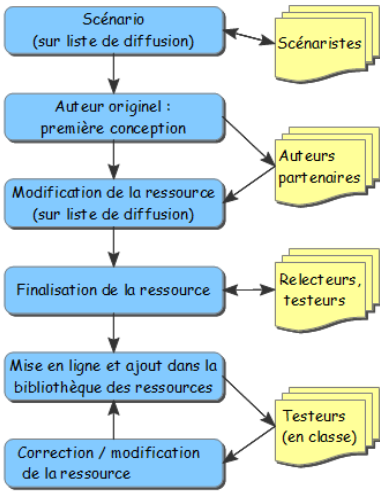


Figure 6 - Cycle de vie d'une ressource Sésamath

Les enseignants peuvent donc également modifier des ressources existantes, même si ils n'en sont pas l'auteur. Ce site est accessible par l'adresse http://mutuamath.sesamath.net.

1.3.3 Outils et logiciels

Sésamath met aussi à disposition de la communauté enseignante divers outils et logiciels permettant la conception de ressources interactives. Les descriptions de ces différents outils sont issues des sites de Sésamath.

1.3.3.1 Instrumenpoche

Instrumenpoche est une interface qui permet à la fois de créer et visualiser des constructions géométriques animées [INSTRUMENPOCHE 2012].

Instrumenpoche propose tous les instruments de géométrie usuels : crayon, compas, règle, équerre, rapporteur, etc. ainsi que la possibilité de placer des points nommés, créer des textes mathématiques, un repère, des courbes de fonctions, des codages de longueur ou d'angle droit.

Toute construction géométrique créée avec Instrumenpoche peut être visualisée pas à pas, comme un film. La bibliothèque du site Instrumenpoche.net contient de nombreuses constructions de base ou plus sophistiquées. Le site des manuels Sésamath propose la correction de nombreux exercices réalisés avec Instrumenpoche.

Pour une utilisation plus avancée, Instrumenpoche est entièrement paramétrable. Il est ainsi possible de créer des exercices interactifs en limitant les instruments mis à disposition de l'élève

comme par exemple, construire un angle droit en n'utilisant que le compas, la règle et le crayon. De plus, Instrumenpoche peut être intégré dans des pages HTML (et piloté par JavaScript) ou d'autres applications Flash© Adobe.

1.3.3.2 TracenPoche

TracenPoche est un logiciel de géométrie dynamique utilisable sur Internet ou en local grâce à la technologie Flash© Adobe. C'est un projet de Sésamath et un module de l'ensemble Mathenpoche.

Ce logiciel est prévu pour être utilisé par tout curieux de géométrie qui, en faisant bouger les figures géométriques, voit apparaître leurs propriétés. TracenPoche participe au projet européen Intergeo I2G qui vise à définir un format de fichier commun aux différents logiciels de géométrie dynamique afin de rendre interopérables leurs ressources et de les mettre à disposition de tout un chacun [TRACENPOCHE 2012].

1.3.3.3 SACoche

Les trois premières lettres de SACoche signifient « Suivi d'Acquisition de Compétences ». L'application SACOCHE permet :

- d'évaluer les élèves par compétences ;
- de partager des référentiels de compétences ;
- de conserver un historique de leur parcours ;
- de déterminer un état d'acquisition de chaque compétence ;
- de les collecter pour assister la validation du socle commun [SACOCHE 2012].

SACoche a pour objectif premier d'évaluer les élèves sur des items personnalisés, en mémoriser l'historique, et proposer des outils pédagogiques associés.

1.3.4 Pour la classe

Sésamath fourni aussi des applications spécifiques pouvant être utilisées directement en classe tant par les enseignants que par les élèves.

1.3.4.1 Les manuels Sésamath

L'une des activités essentielles de Sésamath est l'élaboration de manuels et de cahiers d'exercices et corrigés principalement pour la sixième à la troisième. Ces manuels et cahiers sont à télécharger gratuitement depuis le site http://manuel.sesamath.net/index.php.

Une version papier, réalisée en collaboration avec les éditeurs Génération 5 et Magnard, est aussi vendue en librairie. Sur une quinzaine d'éditeurs, Sésamath détient environ 20% de part de marché. Ce qui représente 80 000 manuels 100 000 cahiers d'exercices vendus chaque année et constitue la ressource principale de Sésamath.

Tous les ouvrages sont élaborés par plusieurs dizaines d'enseignants en activité, expérimentés, corrigés et testés à partir des besoins réels d'utilisation en classe. Leur conception résulte de nombreux échanges. Le travail des auteurs est complètement bénévole pour chacun des ouvrages. Les auteurs sont des membres de Sésamath ou des contributeurs occasionnels qui ont simplement répondu à l'appel lancé par Sésamath dans le but de créer des ouvrages sous licence libre. Aucune rémunération n'est sollicitée.

Les cours et exercices des manuels et cahiers de Sésamath sont aussi découpés en ressources et se retrouvent dans les sites Sésamath tels que Mathenpoche et LaboMep.

1.3.4.2 LaboMep

LaboMep permet aux professeurs de créer des séances personnalisées à partir des 1 600 exercices interactifs auxquels ont été rajoutées l'ensemble des ressources de Sésamath : manuels, cahiers, animations, corrections, etc. De plus, des outils de calcul mental, de géométrie dynamique, de QCM permettent aux professeurs de créer facilement leurs propres exercices. Une fois créé dans la base, par son enseignant, l'élève se connecte via Internet pour faire sa séance et le professeur récupère ensuite son travail :

- figures animées ;
- réponses aux QCM ;
- réponses aux exercices ;
- textes écrits par l'élève ;
- etc.

L'enseignant peut ainsi évaluer et voir les faiblesses de l'élève dans tel ou tel domaine [LABOMEP 2012].

1.4 Analyse détaillée de LaboMep

LaboMep se décline d'abord par établissement. Dès qu'un enseignant est connecté, il est connecté en tant qu'enseignant d'un établissement donné (s'il exerce dans plusieurs établissements, il peut facilement passer de l'un à l'autre mais les deux espaces sont distincts). Par ailleurs, tous les enseignants d'un établissement se « partagent » l'ensemble des élèves et il est donc possible à un enseignant de donner du travail à tout élève d'un établissement où il exerce. Cela facilite les types d'organisation pédagogique qui s'affranchissent parfois du groupe classe.

L'inscription à LaboMep implique d'être un enseignant en mathématiques rattaché à un établissement scolaire référencé par Sésamath à partir de la base officielle de l'éducation nationale. La figure 7 montre cette nécessité.

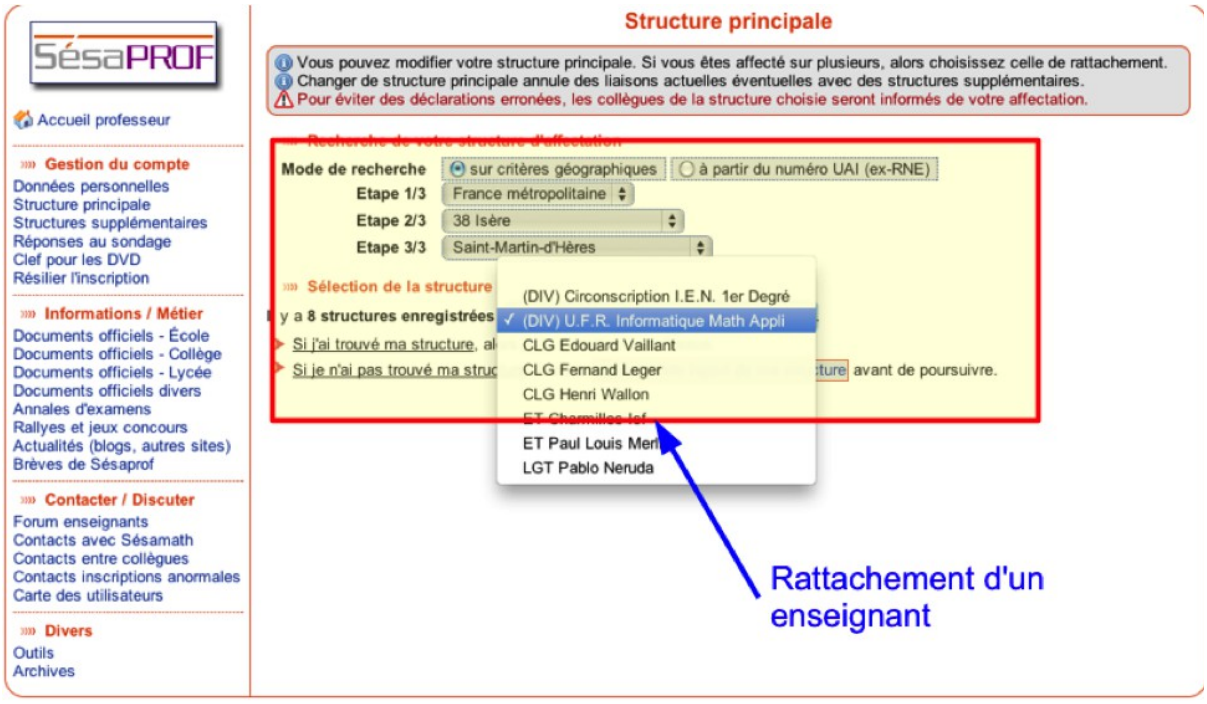


Figure 7 - Espace enseignant dans Sésaprof permettant l'utilisation de LaboMep

L'enseignant peut ensuite se connecter à LaboMep depuis le site Sésaprof. Les élèves se connectent directement via l'URL de connexion : http://www.labomep.net/identification/.

1.4.1 Côté enseignant

LaboMep est une application Web développée en PHP et JavaScript avec les ressources Sésamath en Flash© Adobe dont l'interface se décomposant en différentes zones interactives, illustrée par la figure 8.

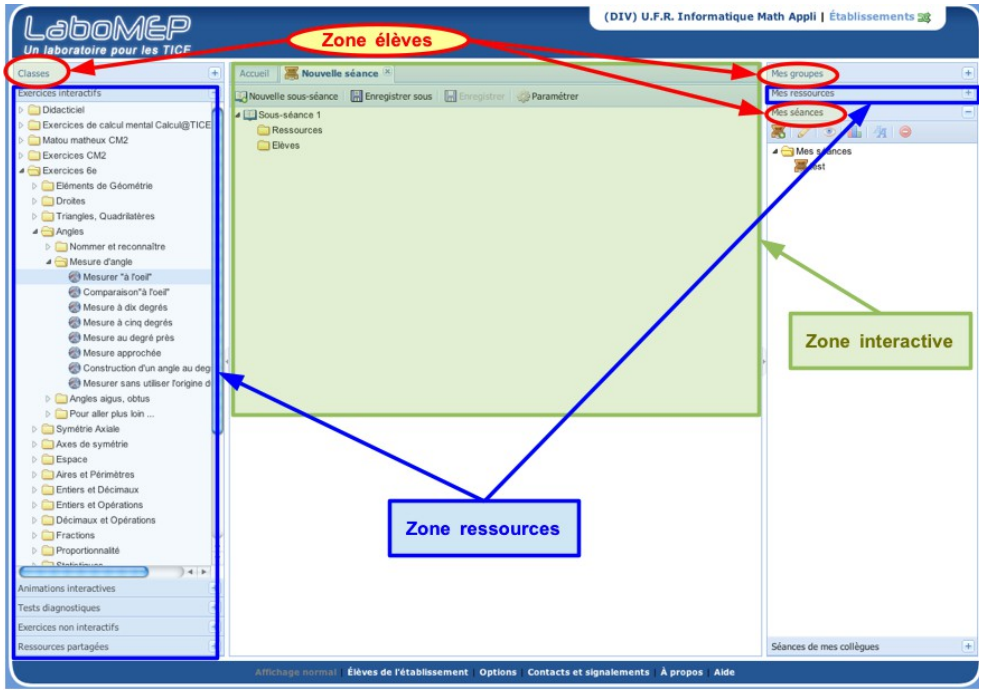


Figure 8 - Interface d'un enseignant connecté à LaboMep

La partie gauche est un espace commun à tous les enseignants pour la « zone ressources ». L'onglet classe est quant à lui plus spécifique à l'établissement. La partie centrale permettra de réaliser diverses opérations et peut être considérée comme l'espace interactif de travail. Toutes les opérations de « glisser déposer » seront notamment réalisées dans cette zone. Enfin, la partie droite correspond à l'espace personnel de l'enseignant. Il pourra y gérer ses groupes de travail, ses ressources ainsi que les séances qu'il souhaite soumettre aux élèves de son établissement.

1.4.1.1 Zone élèves

La zone « élèves » permet à un enseignant de gérer les élèves des classes de son établissement. Cette zone offre différentes actions possibles au professeur. Il peut lister les différentes classes de son établissement ou en créer si certaines sont manquantes. Une fois la classe créée, l'enseignant peut ajouter des élèves à cette classe, soit manuellement (élève par élève), soit par l'importation d'un fichier XML ou issu d'un tableur. Il peut ensuite constituer des groupes de travail en y affectant des élèves dans le but de les faire participer à des séances d'exercices, auxquelles il aura ajouté des ressources.

Il pourra ainsi suivre l'évolution de chaque élève grâce à un bilan des activités et de leur réussite au terme de la séance.

1.4.1.2 Zone ressources

La zone « ressources » permet de rechercher et d'utiliser les exercices mis à la disposition des enseignants par Sésamath. Les ressources dans LaboMep rassemblent les différentes façons de donner du travail aux élèves. Cette zone permet à l'enseignant de rechercher et utiliser différents types d'exercices pour ses séances.

LaboMep offre 5 catégories de ressources :

- exercices interactifs réalisables en ligne ;
- animations interactives qui reprennent les aides des exercices interactifs ;
- tests diagnostiques. Ils permettent d'évaluer les connaissances des élèves sur des sujets particuliers. Pour l'instant les tests concernent l'algèbre en 3ème et concernent le calcul, la traduction, et la résolution de problèmes ;
- exercices non interactifs issus des manuels Sésamath, des cahiers Sésamath et des cahiers Mathenpoche ;
- ressources partagées par les autres enseignants de l'établissement.

La classification des ressources au sein de ces différentes catégories suit une organisation en chapitres de cours : par classe puis par différents sous-chapitres et enfin par ressource comme le montre la figure 9.

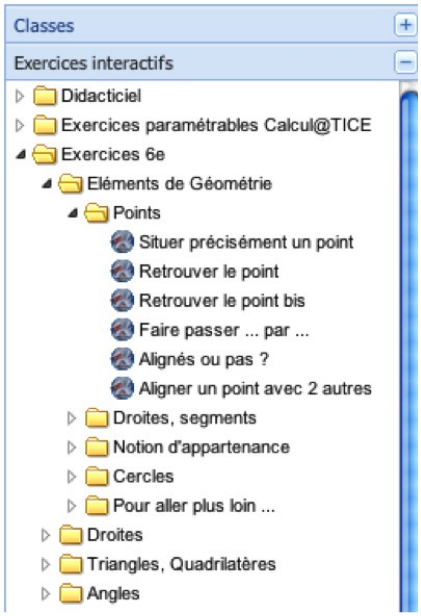


Figure 9 - Recherche d'exercices dans LaboMep

Ce mode de classification demande un minimum de connaissance de l'organisation des programmes pour pouvoir trouver une ressource particulière. LaboMep n'offre pas de recherche d'une ressource par mots clés ou par un autre moyen.

L'enseignant peut aussi créer ses propres ressources à partir de cette zone. Pour ce faire, il peut utiliser les outils en ligne, intégrés dans LaboMep, permettant la création de ressources tels que TracenPoche, GeoGebra, etc. Il peut aussi créer une ressource à partir d'un lien vers un autre site Internet ou de poser tout simplement une question écrite qu'il soumettra à l'élève comme le montre la figure 10.

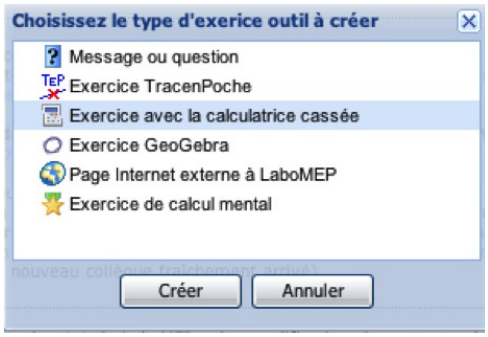


Figure 10 - Types d'exercices à créer

Ces ressources créées directement par les enseignants ne sont pas référencées parmi les ressources que Sésamath propose et ne sont donc pas accessibles à l'ensemble de la communauté LaboMep. Néanmoins, l'enseignant créant l'exercice a la possibilité de les partager avec les autres professeurs de son établissement.

1.4.1.3 Zone interactive

C'est dans cette zone, présentée par la figure 11, que s'affichent les différentes actions réalisées par le professeur. Comme vu précédemment, elle offre une fenêtre pour créer des séances par « glisser déposer » des élèves et des ressources. Le mode de fonctionnement reste le même pour la création de groupe ou de ressources.

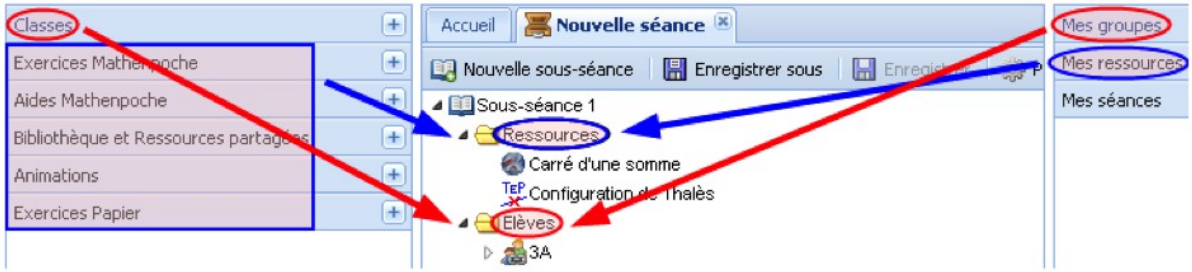


Figure 11 - Création d'une séance d'exercices

Une fois la séance enregistrée, les élèves peuvent la réaliser soit en salle de cours, soit en devoir à la maison.

1.4.2 Focus sur la création et modification de ressources par un enseignant

La création et la modification de ressources sont le point essentiel à analyser en prévision de l'indexation par les notions et compétences. En effet, le moteur de recherche des ressources par les notions et capacités pourrait s'intégrer dans l'interface de LaboMep.

La création d'une nouvelle ressource s'effectue depuis la zone « Mes ressources » de l'enseignant, présentée par la figure 12.

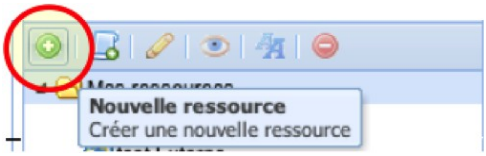


Figure 12 - Icône de création de ressource

L'enseignant a ensuite la possibilité de créer un exercice à partir de 8 types d'exercices outils :

- **Message ou question**, permettant comme son intitulé l'indique au système d'envoyer un message ou de poser une question aux élèves ;
- **Exercice TracenPoche**, permettant de réaliser des exercices de géométrie dynamique ;
- **Exercice avec la calculatrice cassée**. Le professeur décide des touches qui sont disponibles ou « cassées » sur une calculatrice virtuelle, pour afficher un résultat donné, ce qui oblige l'élève à développer une stratégie de calcul ;
- **Exercice d'opération posée**, permettant d'effectuer des opérations d'addition, soustraction, multiplication et division ;
- **Exercice GeoGebra**. GeoGebra est un logiciel libre de géométrie dynamique en 2D, c'est-à-dire qu'il permet de manipuler des objets géométriques du plan (cercle, droite et angle, par exemple) et de voir immédiatement le résultat ;
- **Page internet externe à LaboMep**. Ce type d'exercice outil permet d'intégrer facilement dans LaboMep d'autres ressources du Web, de différents types ;
- **Exercice de calcul mental**. L'outil « Exercice de calcul mental » permet de proposer aux élèves des exercices de calcul mental en paramétrant très finement un certain nombre de calculs. En particulier, il est possible de fixer certains nombres, de les prendre dans des intervalles, de fixer le nombre de chiffres après la virgule, etc. et régler le temps d'affichage de chaque partie. Le temps de réponse pour l'élève est aussi paramétrable ;
- **Exercice LaboMep**. Le premier objectif de cet outil est de pouvoir créer des exercices type QCM. Pour cela, il est possible de taper facilement des formules mathématiques, d'insérer des images mais aussi des figures dynamiques. Les réponses des élèves sont enregistrées automatiquement dans le bilan des séances contenant ce type d'exercices.

L'annexe 1A présente les processus de création de ressources à partir de ces différents types d'exercices outils.

Toutes les ressources personnelles qu'un enseignant peut créer doivent impérativement avoir un titre et facultativement une description. Le choix du type d'exercices nécessite plus ou moins de paramétrage.

Une ressource créée par un enseignant n'est pas référencée parmi les ressources de Sésamath et est donc invisible à l'ensemble des enseignants des autres établissements. Seul l'enseignant créateur peut la voir et éventuellement, s'il le souhaite, la partager avec les enseignants des établissements auxquels il est rattaché.

1.4.3 Côté élève

Un élève peut se connecter à l'interface de LaboMep, illustrée par la figure 13, depuis son établissement scolaire ou son domicile pour effectuer des séances d'exercices. Un élève ne peut pas s'inscrire lui-même à LaboMep, c'est un enseignant de son établissement scolaire qui lui crée un compte et lui fournit des identifiants de connexion.

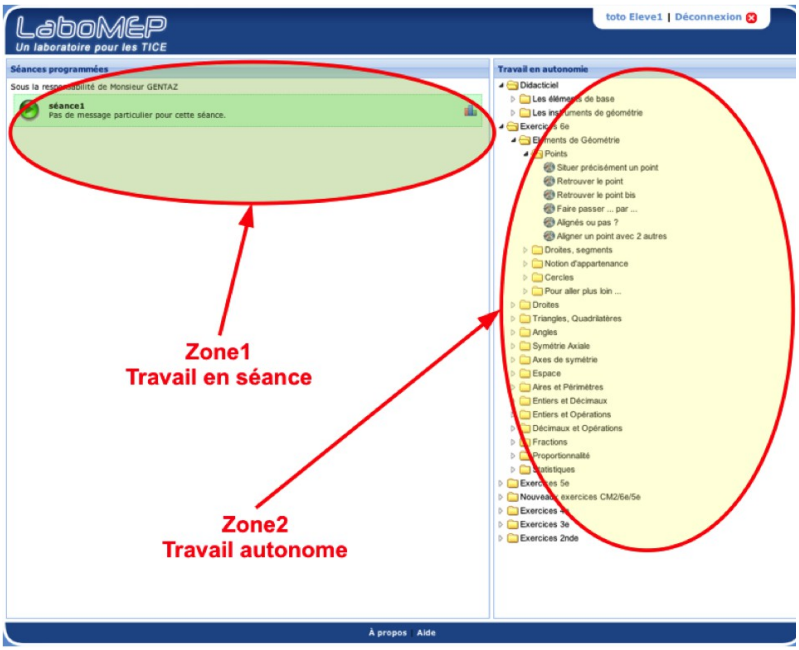


Figure 13 - Interface d'accueil d'un élève à LaboMep

L'interface LaboMep pour les élèves est beaucoup plus épurée et ne permet que 2 actions :

- réaliser une séance de travail assignée par un enseignant ;
- travailler de manière autonome sur les exercices proposés par Sésamath.

La zone permettant de travailler de manière autonome ne dispose pas de moteur de recherche de ressource. L'élève doit donc parcourir une arborescence, de type chapitre, sous-chapitre, pour trouver une compétence qu'il souhaite travailler.

1.4.4 Bilan élève

La fonction de bilan à l'enseignant, permet une fois la séance terminée par les élèves (exercices assignés terminés), de consulter le bilan lié à chaque élève pour chaque exercice, et ainsi évaluer les progrès et les points faibles de chacun. Il peut éventuellement faire retravailler les points faibles lors de prochaines séances LaboMep.

Le bilan d'une séance réalisée par un élève se présente sous la forme d'un récapitulatif des activités proposées, comme le montre la figure 14.

Figure 14 - Bilan d'un élève

Un score indique le nombre de questions traitées correctement sur l'ensemble de l'exercice. Des codes couleurs permettent aussi de visualiser rapidement les travaux réalisés par l'élève pour chaque exercice. La figure 15 détaille la signification des codes couleurs des bilans.

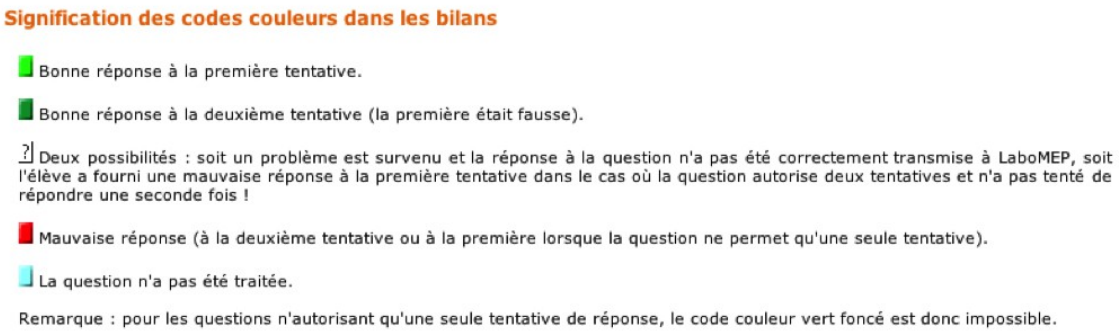


Figure 15 - Définition des codes couleurs dans les bilans

En fin d'année scolaire, les comptes des élèves et leurs historiques sont supprimés automatiquement pour répondre aux règles de la CNIL. Par contre les ressources créées par les professeurs ne sont pas effacées sauf si le professeur ne s'est pas connecté pendant plus de 18 mois.

1.5 Bilan

Cette présentation de l'association Sésamath et l'étude de son fonctionnement mettent en évidence la problématique de gestion des ressources mises à disposition des utilisateurs. L'absence de moteur de recherche implique un minimum de connaissance des programmes scolaires pour retrouver des exercices, cours, etc. spécifiques à une compétence que l'on souhaite améliorer. La multitude de sites mis en œuvre et l'hétérogénéité des données complexifient d'autant plus la gestion des ressources. Ces dernières ne sont pas toutes centralisées et ne sont pas référencées par un point d'entrée unique.

De plus Sésamath possède une grosse communauté où chaque participant peut potentiellement apporter ses propres contributions. La perspective d'ouverture sur les programmes de mathématiques d'autres pays européens implique un nombre croissant de ressources à gérer par les membres bénévoles et salariés de l'association. Il devient impératif de mettre en œuvre un système permettant de centraliser et retrouver de manière rapide et pertinente une ressource parmi la multitude.

Le système doit prendre en compte les contraintes matérielles de l'association puisque un seul serveur physique doit pouvoir héberger l'ensemble des sites Sésamath. Un système « léger » peu demandeur de ressources machine, pour que l'ensemble de l'infrastructure Sésamath puisse continuer à fonctionner, est un pré-requis au projet Compmep.

Il convient donc d'analyser le système mis en œuvre dans le cadre du projet Intergeo afin d'examiner si les principes implémentés pourraient apporter une solution à la problématique rencontrée par Sésamath. Cette étude permettra aussi de voir si les composants peuvent être portés sur l'infrastructure de Sésamath.

2. Le système i2geo.net – État des lieux général

Le projet Intergeo a pour objectif principal la gestion d'un grand nombre de ressources pédagogiques de mathématiques libres. Afin de préparer au mieux le projet Compmep, il convient d'analyser l'architecture matérielle et logicielle ainsi que les différents modules de la plate-forme Intergeo représentés dans la figure 16. L'installation de ces différents modules permet ensuite de mettre en place une plate-forme de démonstration à la maîtrise d'ouvrage de Sésamath et de valider si la solution est réutilisable. Une présentation générique des termes et principes spécifiques à l'API Lucene permet d'introduire l'analyse du fonctionnement de l'indexation et de la recherche mis en œuvre dans le cadre du projet. Des bilans ponctuent chaque étape de cette analyse. Enfin un bilan général permet de définir si la solution Intergeo peut être réutilisée totalement ou partiellement dans le cadre du projet Compmep.

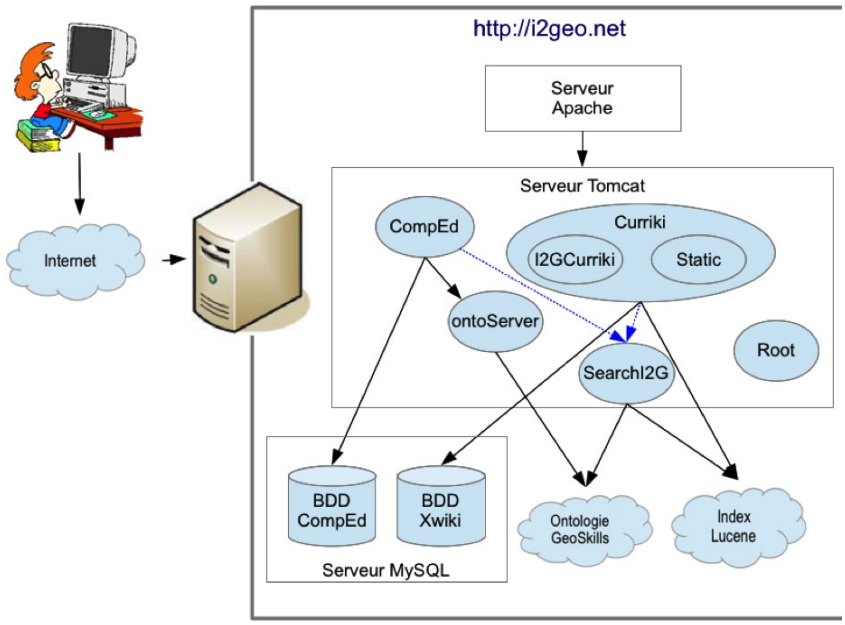


Figure 16 - Architecture générale de la plate-forme Intergeo

Cette solution s'appuie sur différentes applications Web développées séparément mais communiquant les unes avec les autres. Cette architecture de type client-serveur repose sur un serveur physique sur lequel sont installés différents serveurs logiciels.

2.1 Infrastructure matérielle

Le système Intergeo est actuellement hébergé sur un seul serveur de l'université de Karlsruhe. Ce serveur est un Mac Mini équipé d'un processeur « Intel Dual i7 » de 2.0GHz et doté de 4Go de mémoire vive.

Son système d'exploitation est un « MacOSX 10.7 server » sur lequel est installé une infrastructure serveur logiciel Open Source.

Il n'y a pas eu d'étude spécifique de charge pour valider la robustesse de cette infrastructure. Cependant un test avec une trentaine d'utilisateur a simplement été réalisé et le système s'est bien comporté.

2.2 Infrastructure logicielle

Le système Intergeo nécessite l'utilisation d'un serveur Web Apache utilisé comme « proxy inverse » devant un serveur d'application Apache Tomcat où sont déployées les différentes applications Web de la plate-forme. Un serveur MySQL 5 est utilisé pour la partie base de données [I2GEO 2012]. La figure 17 illustre la chaîne de liaison de cette infrastructure.

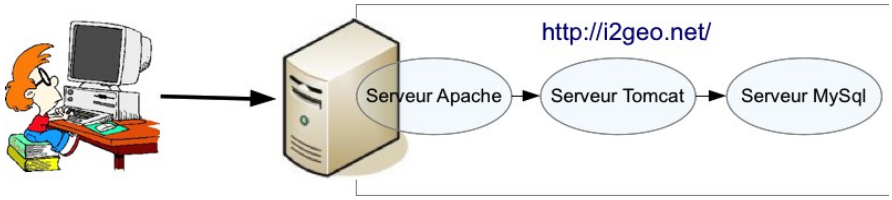


Figure 17 - Architecture N-tiers du site i2geo.net

L'installation des modules Java sur la plate-forme nécessite en pré-requis une machine virtuelle Java ainsi que le logiciel « Apache Maven » pour la gestion et l'automatisation de production du projet Intergeo.

Maven permet de produire un logiciel à partir de ses sources, en optimisant les tâches et en garantissant le bon ordre de fabrication. Maven utilise notamment un Project Object Model (POM) afin de décrire principalement un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour la compilation des différents modules du projet [MAVEN 2009].

Maven offre notamment des fonctionnalités de :

- construction, compilation ;
- documentation ;
- rapport ;
- gestion des dépendances ;
- gestion des sources ;
- mise à jour de projet ;
- déploiement.

Toutes les applications Web Java développées dans le cadre du projet Intergeo utilisent Maven pour leur compilation.

2.3 Architecture de la plate-forme I2geo

La plate-forme Intergeo est composée de différents modules Open Source et interdépendants les uns des autres. Ces modules sont disponibles sur le dépôt SVN d'activemath.org (http://svn.activemath.org/Intergeo/). Ils se refabriquent par l'intermédiaire de commandes Maven et se déploient sur le serveur d'application Tomcat.

Il y a en tout 11 modules nécessaires à la mise en œuvre de la plate-forme (cf. figure 16). Ces modules, développés en Java, peuvent être décomposés en 2 catégories d'utilisation.

Les composants utilisés pour la compilation des applications Web sont :

- CompEd-maven-plugin. Ce module est nécessaire à la compilation de l'application CompEd ;
- i2GCurriki. Ce module est nécessaire pour la personnalisation de Curriki et son adaptation aux différents programmes scolaire européens ;
- ServletUtils. Ce module est nécessaire à la compilation de l'application RootWebAPP ;
- WIRISHTMLconversion. Ce module est nécessaire à la compilation de l'application Curriki.

Les applications Web déployées dans Tomcat :

- Curriki. Ce module est déployé sous le nom de « Xwiki » ;
- CompEd ;
- Ontologies. Ce module est en fait un répertoire permettant de stocker les ontologies au format fichier « .owl » ;
- OntoServer. Ce module est déployé sous le nom de « ontoUpdate » ;
- RootWebAPP. Ce module est déployé sous le nom de «Root » ;
- SearchI2G ;
- Static.

2.3.1 Module Root

Le module Root est une application Web. Cette application permet la redirection de requêtes HTTP contenant de courtes ou ancienne URL du projet. Root s'appuie sur une table de redirection stockée dans un fichier XML. La seconde fonction de Root est d'offrir un service de fichiers statiques de qualité pouvant être utilisés dans l'infrastructure Apache et Tomcat.

Root permet d'effectuer des redirections vers les 2 principales applications Web du système, accessibles depuis Internet, que sont CompEd et Curriki.

2.3.2 Module Curriki

Curriki est un Wiki spécialisé, permettant de gérer des programmes scolaires, ayant pour fonction d'offrir un espace coopératif de travail et de partage à la communauté enseignant. L'application Web Curriki est basée sur Xwiki dans le but de créer collaborativement des ressources pédagogiques. Cette application permet d'ajouter et d'organiser des contenus éducatifs dans de très nombreux formats. Curriki intègre un moteur d'indexation et de recherche utilisant l'API Apache Lucene permettant de retrouver des ressources éducatives grâce à des recherches multicritères, afin de les visualiser et de les enrichir grâce au réseau social et aux groupes de travail. Les fonctionnalités de l'API Lucene sont étudiées de manière détaillée dans le chapitre 2.5 intitulé « I2Geo – Indexation et recherche » de ce document.

Curriki utilise aussi une base de données MySQL pour stocker des informations sur les utilisateurs, les groupes, les ressources, etc. Cette solution est utilisée dans Intergeo comme « portail Web » apportant l'interface utilisateur, permettant l'ajout de ressources pédagogiques et leur stockage de manière centralisée comme le montre la figure 18.

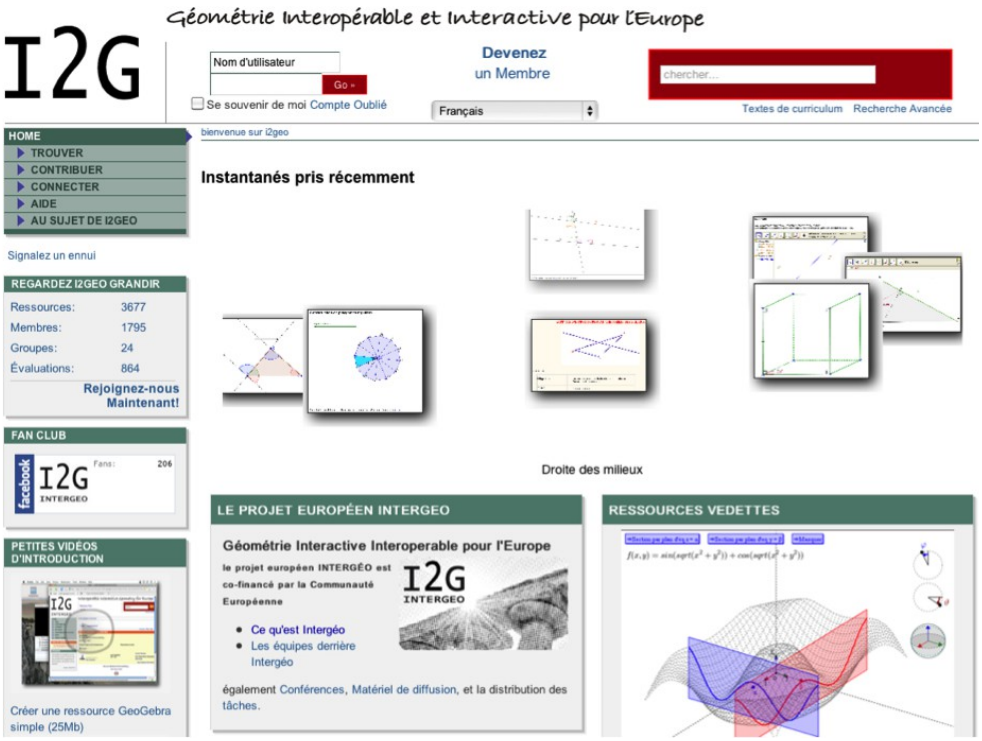


Figure 18 - Le portail d'accès i2geo.net basé sur Curriki

Toutes les fonctionnalités de Curriki sont utilisées dans Intergeo à l'exception de quelques propriétés de métadonnées telles que le sujet et les niveaux d'éducation ainsi que l'outil de recherche.

Le module Curriki est développé en Java mais s'appuie aussi sur le moteur de substitution « Apache Velocity ». Velocity sépare le code Java des pages Web afin de rendre le site plus facile à maintenir tout en utilisant une architecture de type MVC (Modèle-Vue-Contrôleur). Velocity est utilisé, dans le module Curriki, pour générer des pages Web à partir de gabarit. Le langage de programmation « Groovy » est aussi utilisé par le module Curriki. Ce langage permet de mettre du code comme Java dans les pages Wiki et de l'exécuter, par exemple depuis Velocity [VELOCITY 2012][GROOVY 2012].

La figure 19 représente l'ensemble des composants nécessaires à la mise en place du portail Curriki.

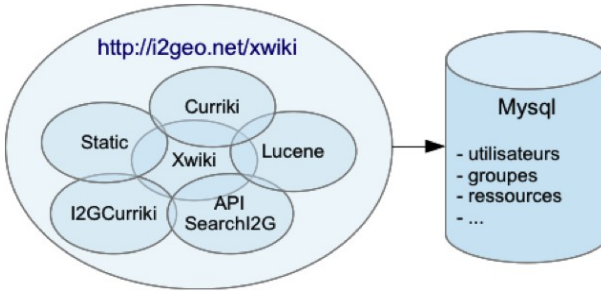


Figure 19 - Composants de Curriki

La construction de la plate-forme nécessite aussi les modules i2GCurriki, l'API de SearchI2G et Static développés spécialement par le projet Intergeo. Le module I2GCurriki permet la customisation de Curriki, notamment, en le rendant multilingue grâce à différents fichiers de traduction et en

permettant l'adaptation du site aux différents programmes scolaires européens. Le module Static permet l'ajout de pages statiques au niveau du Wiki tout en prenant en compte l'aspect multilingue de la plate-forme. Les modules Static et i2GCurriki apportent aussi des macros Velocity spécifiques à « Curriki Intergeo ».

Enfin l'API SearchI2G permet d'intégrer le moteur de recherche, développé dans le cadre du projet, au composant Curriki afin de pouvoir effectuer des recherches de ressources par les capacités et notions via des mots clés. Le module SearchI2G se substitue ainsi au moteur de recherche embarqué par Curriki d'origine.

La construction du site s'effectue par l'intermédiaire de plusieurs commandes de compilation Maven. Après déploiement de l'application au niveau du serveur Tomcat, ce dernier alimente la base de données de Xwiki lors d'un redémarrage. Toutes les tables et données nécessaires au bon fonctionnement du site sont ainsi créées.

2.3.3 Module Static

Ce module est nécessaire pour la customisation de Curriki dont principalement l'internationalisation en différentes langues de la plate-forme. Ce module apporte aussi plusieurs pages HTML statiques, images et JavaScript accessibles depuis Xwiki.

L'installation du module Static ne nécessite pas de compilation et se déploie au niveau des applications Web du serveur Tomcat.

2.3.4 Module SearchI2G

Le module SearchI2G est un service Web composé de différentes fonctionnalités permettant d'effectuer :

- l'analyse de l'ontologie GeoSkills.owl ;
- le raisonnement sur l'ontologie ;
- l'indexation des notions et capacités de l'ontologie ;
- la recherche de notions et capacités indexées.

Le module SearchI2G est développé en Java et utilise les outils Maven pour la gestion des dépendances et la compilation. Il utilise différentes bibliothèques Java pour réaliser chacune des fonctionnalités citées. La partie manipulation de l'ontologie, au format fichier « .owl », utilise l'API OWLAPI, développée par l'université de Manchester, pour l'analyse. Le raisonneur Pellet permet d'effectuer les déductions sur l'ontologie [OWLAPI 2012]. L'API Xstream permet de faciliter la conversion du langage Java vers le langage XML et inversement.

Ce module est basé sur l'API libre Apache Lucene en version 2.9.3. Il permet l'indexation des notions et capacités, spécifiées dans l'ontologie GeoSkills.owl. Le moteur de recherche, nommé « SkillsTextBox », intégré dans ce module permet la recherche par ces mêmes capacités et notions via des mots clés. Ce moteur de recherche s'intègre dans le module Curriki pour l'indexation et la recherche de ressources par les notions et les capacités. On le retrouve aussi au niveau du site CompEd, présenté dans le chapitre 2.3.6, dans le cadre de la création et la modification de notions et de capacités.

SkillsTextBox utilise aussi les outils de l'API GWT afin de créer des pages Web dynamiques et constituer les fichiers de cache contenant les objets de l'ontologie GeoSkills tout en prenant en compte la problématique de compatibilité d'affichage entre les différents navigateurs Web [GWT 2012]. Le code JavaScript ainsi généré utilise des techniques d'HTML dynamique et de manipulation du DOM (Document Object Model) pour les aspects dynamiques de l'interface. La figure 20 illustre une recherche de notions et de capacités à partir d'une partie du terme angle.

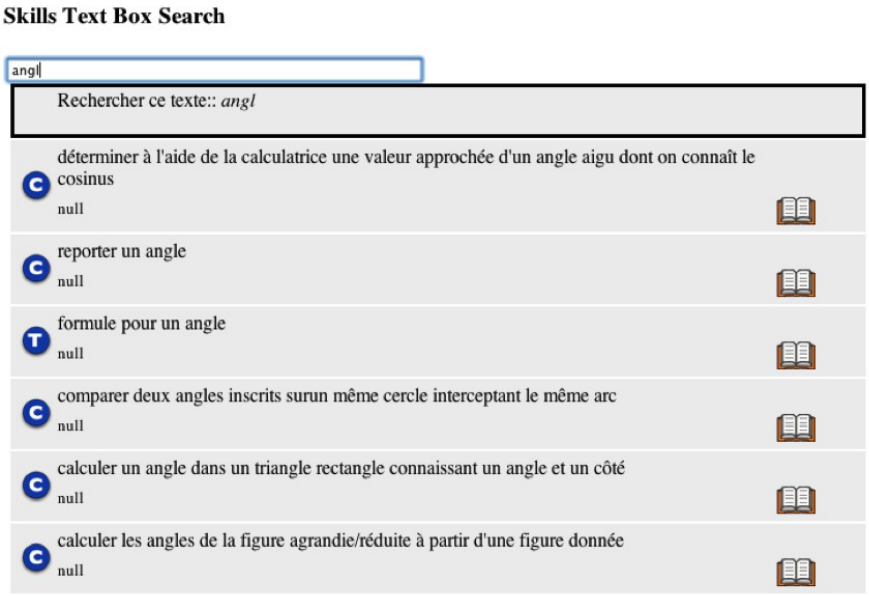


Figure 20 - Exemple d'une recherche depuis une JSP de test du module SearchI2G

Le chapitre 2.5 intitulé « I2Geo – Indexation et recherche » de ce document présente plus en détail le fonctionnement de ce composant.

2.3.5 Module ontologie

Ce module est un répertoire permettant de stocker les versions des ontologies au format fichier « .owl » ainsi que la documentation OWLDoc de celles-ci au format HTML. L'ontologie GeoSkills.owl est donc accessible depuis ce répertoire ainsi que l'ontologie Subjects.owl qui permet de déduire les différentes relations de l'ontologie GeoSkills.

L'ontologie GeoSkills.owl a été générée à partir du langage de description logique OWL, en version 1.1. Le format OWL est un standard promulgué par le W3C. Cette ontologie est organisée autour d'une première hiérarchie de classes centrales :

- Competency ;
- Topic ;
- EducationalProgram ;
- EducationalLevel ;
- EducationalPathway ;
- EducationalRegion.

La classe « Competency » représente les capacités, c'est-à-dire les savoir-faire. Une classe de Competency est définie par un verbe et est reliée par la propriété « hasTopic » à au moins une notion, représentée dans la hiérarchie Topic, qui constitue la seconde grande hiérarchie de classe de l'ontologie. Elle est rattachée à un ou plusieurs curriculums par la propriété « belongsToCurriculum ». Ces curriculums sont représentés dans les hiérarchies « EducationalProgram » et « EducationalLevel ». La figure 21 illustre ces différentes relations ainsi que la classification amenée avec les principes d'héritage entre les classes.

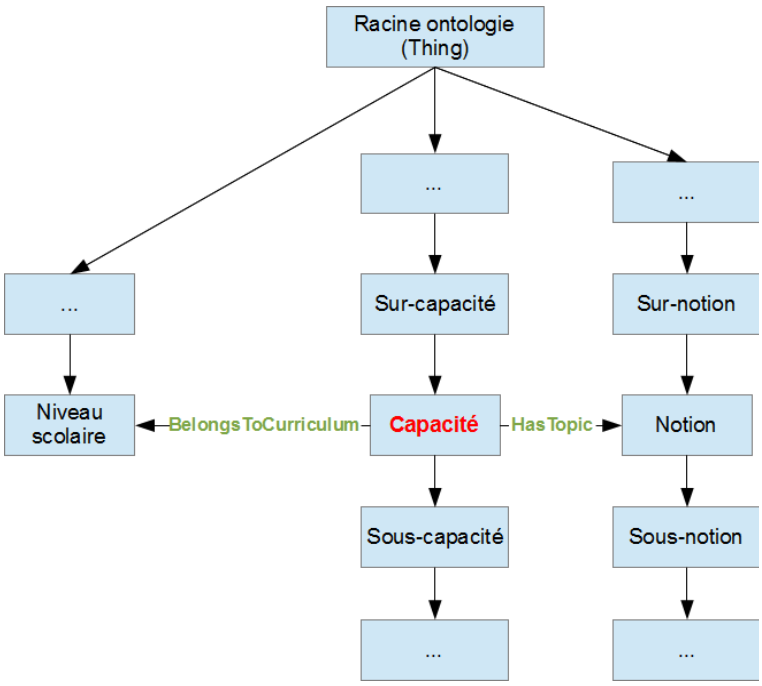


Figure 21 - Modélisation de l'ontologie GeoSkills

Un élément de l'ontologie possède donc 1 ou plusieurs ancêtres si il ne s'agit pas de la racine. L'élément est aussi en relation avec plusieurs autres éléments. Des raisonnements peuvent ensuite être réalisés en fonction de ces différents liens.

La hiérarchie « Topic » représente les notions (savoirs) des curriculums, appelées « connaissances » dans les programmes scolaires français. La figure 22 présente un extrait de cette hiérarchie.

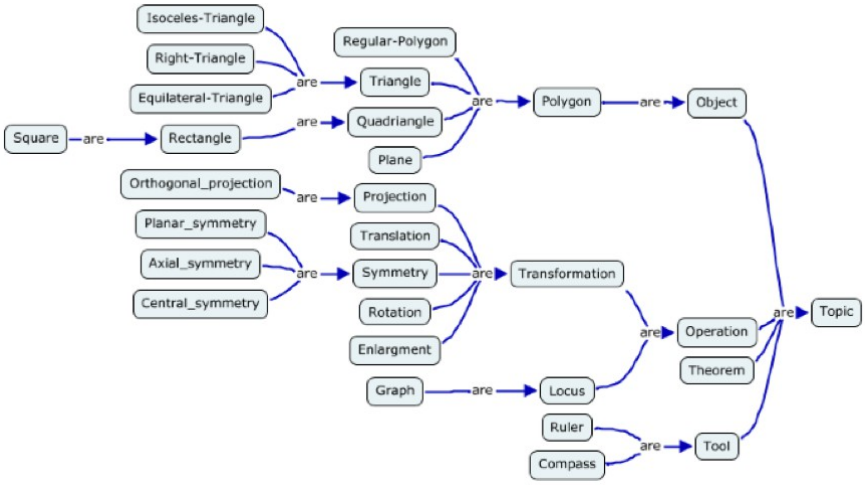


Figure 22 - Extrait de la hiérarchie des Topics de GeoSkills [COMPED 2009]

La classe « EducationalProgram » représente les programmes scolaires pour une matière et un niveau scolaire dans une région donnée. La classe « EducationalLevel » représente les niveaux scolaires d'un cursus (primaire, collège, etc.). La classe « EducationalPathway » représente le parcours scolaire, par exemple Collège en France et la classe « EducationalRegion » représente les régions éducatives (une seule en France).

L'édition de l'ontologie peut être réalisée à l'aide d'un éditeur ontologique tel que Protégé ou par l'intermédiaire des applications Web CompEd et ontoUpdate.

2.3.6 Module CompEd

Le module CompEd est développé en Java et utilise les outils Maven pour la gestion des dépendances et la compilation. Dans le cadre du projet Intergeo, CompEd a été développé afin de faciliter l'accès pour l'édition collaborative de l'ontologie GeoSkills. La figure 23 montre les différents processus de ce composant.

Figure 23 - Architecture détaillée de CompEd [COMPED 2009]

CompEd est une application Web basée sur le cadriciel Appfuse permettant de développer des applications Web Java 2EE [APPFUSE 2012]. Ce cadriciel Java permet d'utiliser notamment des technologies Open Source tels que Struts, Spring, Hibernate, le patron DAO, etc.

CompEd accède à une base de données MySQL. C'est dans cette base de données que sont stockés les notions et les capacités de l'ontologie GeoSkills.owl ainsi que les utilisateurs de l'application. La modélisation du schéma de la base CompEd est consultable en annexe 2A. Cet outil permet l'affichage, en arbre ou en liste, des capacités et des notions de l'ontologie GeoSkills chargées dans une base de données, comme le montre la figure 24.

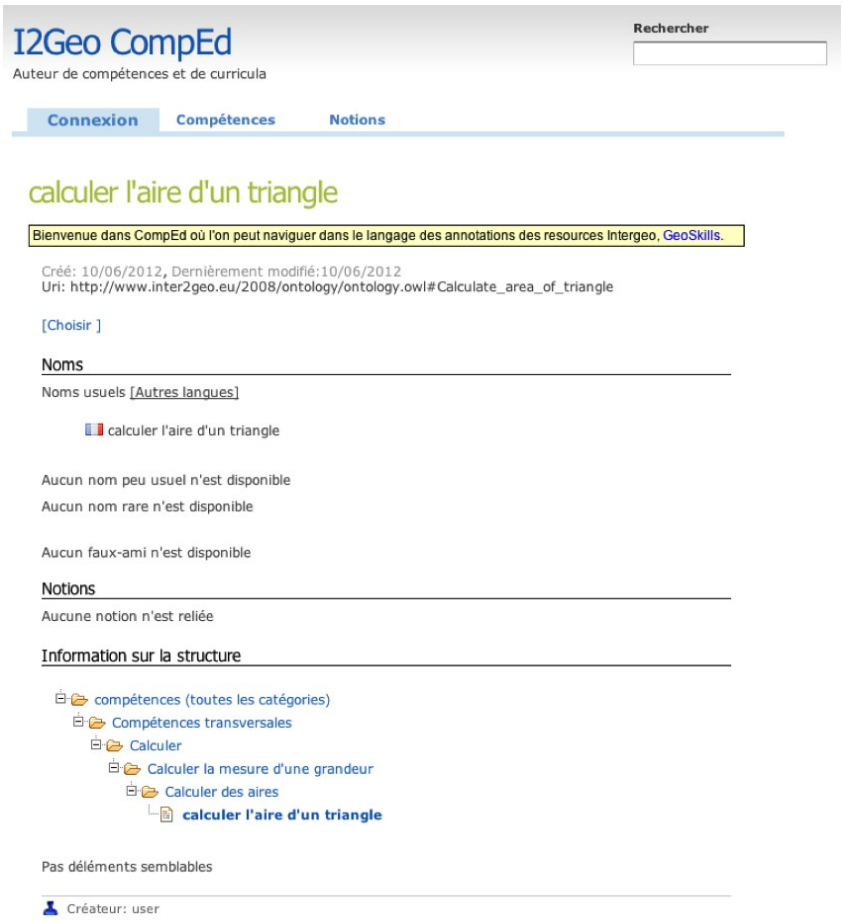


Figure 24 - Présentation d'une capacité dans CompEd

Tout changement au niveau de l'ontologie fichier implique un repeuplement de la base de données par un vidage de la base, pour les tables concernant les notions et les capacités, puis de rechargement. Cette opération s'effectue par une nouvelle compilation de l'application avec Maven et nécessite, au préalable, l'arrêt de l'application CompEd au niveau du serveur d'applications Tomcat. CompEd offre aussi la possibilité d'éditer et de modifier l'ontologie stockée en base de données, avec différents profils.

Chaque profil a été répertorié en lui associant un rôle avec des permissions prédéfinies quant aux actions pouvant être réalisées sur l'ontologie chargées dans la base de données. Le tableau 1 liste ces différents profils.

Profil	Rôle	Accès à l'ontologie
Annoteur	Acteur ajoutant une ressource au niveau d'Intergeo et l'annotant avec des capacités ou des notions.	Lecture seule
Chercheur	Intervenant recherchant des ressources par notions ou capacités	Lecture seule
Développeur de Curriculum	Acteur créant les capacités et les notions de l'ontologie.	Écriture au niveau des classes
Traducteur de compétences	Acteur ajoutant ou modifiant les noms de capacités, des notions ou des descriptions dans sa propre langue.	Écriture au niveau des dénominations ontologique
Ingénieur ontologue	Acteur pouvant réaliser des modifications à tous les niveaux de l'ontologie, principalement sur les axiomes ou les niveaux d'enseignement.	Écriture sur l'ensemble de la structure de l'ontologie

Tableau 1 - Profils CompEd

2.3.7 Module ontoUpdate

Le module ontoUpdate est développé en Java et utilise les outils Maven pour la gestion des dépendances et la compilation. Comme le module SearchI2G, ce service Web manipule l'ontologie au format fichier « .owl ». Il permet la lecture et la mise à jour des capacités et notions. Il utilise OWLAPI et Pellet.

La lecture de l'ontologie intervient au moment de la compilation de l'application CompEd afin d'extraire les notions et les capacités pour les charger dans le SGBD de CompEd. Le module ontoUpdate utilise aussi les librairies Java du raisonneur Pellet pour la classification et la vérification de la cohérence de l'ontologie GeoSkills.owl.

Inversement, cette application Web permet d'écrire dans l'ontologie au format fichier afin de reporter des modifications (ajout, suppression ou mise à jour de notions ou capacités) effectuées depuis CompEd sur les éléments de l'ontologie stockées dans la base de données MySQL. La figure 25 détaille ce processus.

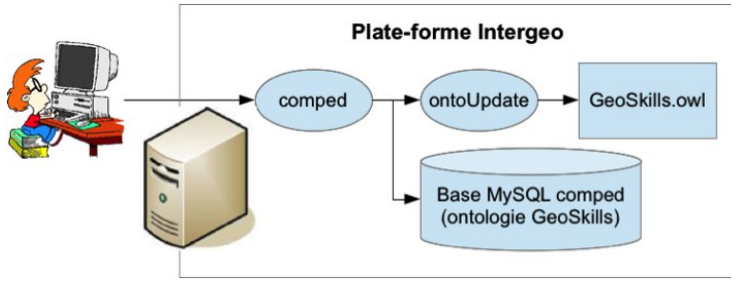


Figure 25 - Mise à jour de l'ontologie GeoSkills.owl

2.4 Installation de la plate-forme

Les objectifs de cette étape sont d'installer les différents composants listés précédemment afin de mettre en place une plate-forme de démonstration pour valider avec Sésamath que cette solution correspond à leurs attentes. Cette installation permet aussi d'étudier la possibilité de réutilisation des composants pour le projet Compmep.

2.4.1 Installation des composants

L'installation de ces différents modules est réalisée en se basant sur le manuel d'administration d'Intergeo intitulé « Platform's Administration Manual » [I2GEO 2009]. Ce document fait référence, en premier lieu, à un script BASH permettant une installation automatisée de la partie Curriki et I2GCurriki. Malheureusement, ce script s'avère non fonctionnel en l'état et l'enchaînement manuel des commandes permet de relever un certain nombre de problèmes.

L'installation de l'ensemble des composants de la plate-forme est rendue très complexe suite à de nombreux problèmes de compilation. Ces problèmes sont liés notamment à l'évolution du code source, des bugs connus et non corrigés par le projet dans le dépôt SVN de gestion de version d'activemath.org, mais aussi à une problématique de gestion des dépendances vers des librairies externes au projet ayant évoluées. Certains modules nécessitant au préalable la compilation d'autres composants pour pouvoir se refabriquer sont aussi un frein dans le bon déroulement de l'installation.

La procédure d'installation peu explicite notamment sur les liens, le paramétrage et le fonctionnement entre les applications demande un important temps d'appropriation de l'architecture de la plate-forme.

Toute la procédure d'installation a donc été revue en apportant un certain nombre de patchs correctifs et autres actions non documentées. Ce travail a permis d'enrichir le script initial d'installation automatique en prenant aussi en compte les autres modules cités précédemment. Le déroulement des différentes étapes du script de l'installation automatisée est consultable en annexe 2B.

Il n'y a pas de fichiers de paramétrage pour les liens de la plate-forme. Tous les liens des chemins vers les répertoires et des URL sont renseignés en « dur » dans le code source des différents modules. Il est donc nécessaire d'effectuer une recherche automatisée dans le code source de l'ensemble des fichiers afin de modifier les URL et chemins présents pour chaque module. Le nouveau script automatique d'installation de la plate-forme fait donc appel à un script, spécifique à cette opération, présenté par la figure 26.

```
#!/bin/bash

#Cleaning directories .svn and .git
find $1 -name .svn | xargs rm -rf
find $1 -name .git | xargs rm -rf

#Mofifying URL COMPED
URL1="http://i2geo.net/comped/show.html"
URL2="http://Compmep.liglab.fr/comped/show.html"
find $1 -name "*" -type f -exec sed -i -e 's|'$URL1'|'$URL2'|g' {} \;

#Mofifying URL i2geo.net and www.i2geo.net
URL3="http://i2geo.net"
URL4="http://www.i2geo.net"
URL5="http://Compmep.liglab.fr"
find $1 -name "*" -type f -exec sed -i -e 's|'$URL3'|'$URL5'|g' {} \;
find $1 -name "*" -type f -exec sed -i -e 's|'$URL4'|'$URL5'|g' {} \;

#Mofifying path to GeoSkills.owl
URL6="file:///Users/paul/projects/Intergeo/ontologies/GeoSkills.owl"
URL7="file:///usr/share/tomcat6//data/dev/GeoSkills.owl"
find $1 -name "*" -type f -exec sed -i -e 's|'$URL6'|'$URL7'|g' {} \;

#Mofifying URI
URL8="http://www.inter2geo.eu/2008/ontology/GeoSkills"
URL9="http://www.inter2geo.eu/2008/ontology/ontology.owl"
find $1 -name "*" -type f -exec sed -i -e 's|'$URL8'|'$URL9'|g' {} \;

#Mofifying URL draft.i2geo.net
URL10="http://draft.i2geo.net"
URL11="http://draft.Compmep.liglab.fr"
find $1 -name "*" -type f -exec sed -i -e 's|'$URL10'|'$URL11'|g' {} \;
```

Figure 26 - Script de modification des URL dans le code source

Ce script d'installation est devenu un composant à part entière car il apporte différents binaires et fichiers de configuration nécessaires au bon déroulement de l'installation suite aux problèmes rencontrés.

Ce composant intitulé « install-I2geo » se compose de scripts BASH réalisant l'installation automatique, de correctifs pour les binaires et les fichiers de configuration erronés du dépôt SVN de gestion de version d'activemath.org, d'une documentation évoquant les pré-requis et le déroulement de l'installation ainsi qu'un répertoire « logs » destiné à recevoir les traces d'exécution de l'installation

de chaque module. Un fichier de paramétrage apporte aussi de la souplesse à l'installation par l'utilisation de différentes variables d'environnement notamment au niveau des chemins d'installation et assure une réinstallation possible dans des contextes différents.

Toutes les modules et dépendances Maven sont aussi présents dans ce composant afin de rendre possible une installation de la plate-forme sans utiliser les sources des dépôts externes.

2.4.2 Bilan de l'installation

Tous les composants de la plate-forme Intergeo sont réinstallés sur la machine de démonstration. Plus d'un mois a été nécessaire à la prise en main et pour que tout soit fonctionnel. L'utilisation de Maven et la construction du projet Intergeo montre une problématique de gestion des dépendances externes non maîtrisable. En effet si un site hébergeant l'une de ces dépendances rencontre des problèmes réseaux cela peut bloquer l'installation de la plate-forme. Il n'est donc pas possible de garantir la pérennité et de pouvoir réutiliser cette procédure d'installation dans le temps.

Les temps d'installation sont tributaires des performances réseau et des ressources machines pour la compilation des modules.

Quelques problèmes mineurs de liens, d'affichage et de pages non trouvées au niveau de la plate-forme Xwiki sont potentiellement à corriger. Ces problèmes sont listés en annexe 2C.

Tous les liens, « i2geo.net » et autres, en dur dans le code source ne sont potentiellement pas tous corrigés ce qui peut expliquer certains dysfonctionnements sur la plate-forme. L'ajout de ressources et leur indexation par les notions et capacités ainsi que la recherche simple de ressource par les notions et les capacités sont opérationnels. Toutefois, le module de recherche avancée de ressource par les notions et capacités n'est pas fonctionnel.

Concernant la partie CompEd, la structure de l'ontologie ainsi que le code a évolué depuis 2009. Le module CompEd ne fonctionne donc que partiellement avec l'ontologie de 2009 contenant l'ensemble des programmes de mathématiques des collèges français. Les notions et capacités sont bien chargées dans la base de données CompEd par contre la construction des liens pour la navigation n'est correcte. Le chargement d'une ontologie de 2012 ne fonctionne pas et n'est pas envisageable du fait que les programmes de collège ont été supprimés. L'installation de plate-forme est donc réalisée à partir de l'ontologie et du code source de 2009, référencée dans le SVN ligForge.

Le fonctionnement de la partie CompEd et OntoServer, permettant la mise à jour de l'ontologie fichier suite à modification de l'ontologie stockée dans la base CompEd, n'a pas été analysé en détail. Une tentative de modification de l'ontologie dans la base CompEd a permis toutefois de constater que la mise à jour de l'ontologie au format fichier « .owl » n'est pas fonctionnelle. CompEd semble envoyer un fichier XML, contenant les modifications à apporter sur l'ontologie fichier, erroné au module OntoServer. Une analyse et correction du code source de ces 2 modules est donc à prévoir pour réutiliser la solution.

Il convient maintenant d'étudier plus en détail les processus d'indexation et de recherche de la plate-forme afin de déterminer si cette solution peut être réutilisée ou non dans le cadre du projet Compmep.

2.5 I2Geo – Indexation et recherche

Les différents modules étant installés sur la plate-forme gérée par le LIG, il convient d'étudier plus en détail les principes et le fonctionnement du moteur d'indexation et de recherche mis en œuvre. Des

bilans intermédiaires sur les différents processus implémentés permettent de récapituler les principes et de lister les éventuels problèmes constatés.

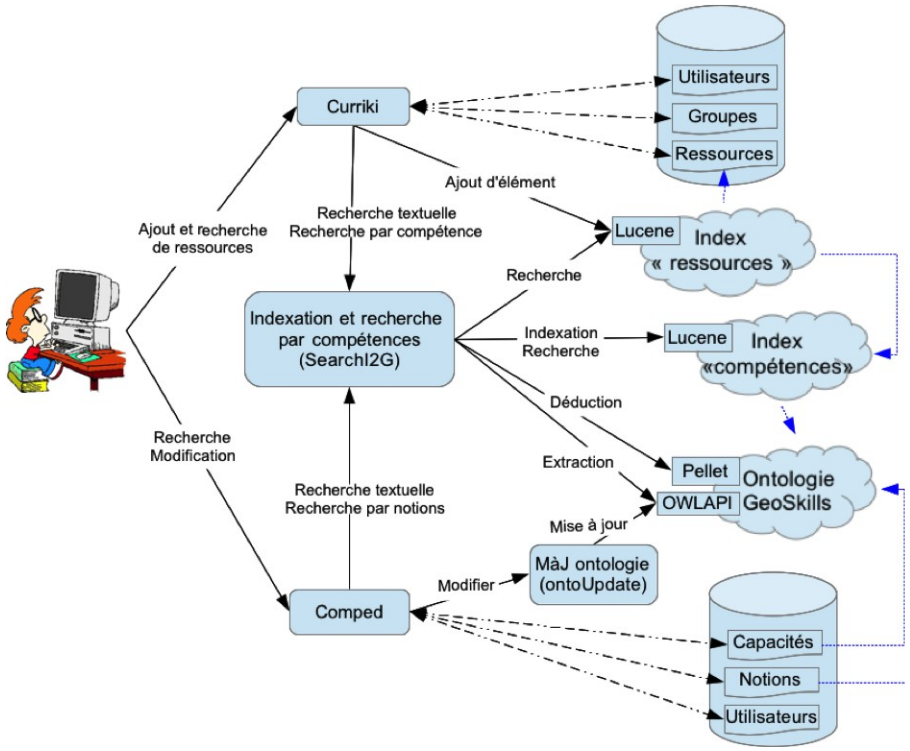


Figure 27 - Processus d'indexation et de recherche dans I2Geo

La figure 27 montre que le cœur du système d'indexation d'Intergeo est le composant SearchI2G utilisant l'API d'Apache Lucene. Il est donc nécessaire de présenter cette API et ses fonctionnalités afin de mieux appréhender les principes d'indexation et de recherche mis en place.

2.5.1 Apache Lucene

Lucene est une API, libre et écrite en Java, permettant de créer un moteur de recherche afin d'indexer et de rechercher du texte et des documents [LUCENE 2012]. Cette API permet d'effectuer des recherches « plein texte ». Elle est entièrement écrite en Java, mais a aussi été portée dans différents autres langages tels que C/C++, Ruby, .NET, Python ou PHP.

2.5.1.1 Présentation de Lucene

Lucene agit en quelque sorte comme une couche intermédiaire entre les données à indexer et une application. Pour ce faire, il indexe des objets appelés des « documents ». Par document, on ne parle pas de fichiers de type Excel, Word, PDF ou HTML, mais d'une structure de données constituée de champs. Un champ est une donnée possédant un nom (titre, auteur, date de publication, contenu, etc.) et à laquelle est associée une valeur et plus généralement du texte. C'est ce texte qui est indexé, recherché et affiché. Les documents indexés sont regroupés au sein d'une collection de documents appelée « index ».

À partir de l'index, Lucene permet ensuite une recherche rapide et efficace dans ces documents. L'utilisation de Lucene implique nécessairement une couche de programmation afin de pouvoir mettre en œuvre une indexation, une IHM de moteur de recherche, la pondération de priorité entre les documents indexés, etc.

2.5.1.2 Fonctionnalités apportées par Lucene

La librairie lucene-core.jar est le cœur de l'API permettant l'indexation et la recherche. Elle permet principalement de réaliser les opérations suivantes :

- indexation incrémentale aussi rapide que l'indexation par lots ;
- classement de recherche, les meilleurs résultats sont retournés en premier ;
- recherche par champ sur les documents ;
- de nombreux autres types de requêtes : requêtes par phrases, recherche joker, de proximité, par intervalle, etc. ;
- tri par champ ;
- recherche sur plusieurs index avec des résultats fusionnés ;
- recherche et mise à jour simultanée.

Il n'est pas fourni d'outils permettant l'indexation de données en quelques clics de souris et quelques paramétrages. Il faut passer par une phase de programmation Java afin de mettre en place une solution sur mesure de recherche plein texte.

2.5.1.3 Principes d'indexation

L'indexation de données met en œuvre 5 classes Lucene :

- « document » : représente un rassemblement de champs (f*ield*). Les champs d'un document représentent le document et ses métadonnées ;
- « field » : composant d'un document, décrivant un attribut du document pour l'indexation. Par exemple le nom, l'auteur ou la date du document. La classe field contient un nom de champ et sa valeur. Lucene fournit des options pour spécifier si un champ doit être indexé ou analysé et si sa valeur doit être stockée. Ces options peuvent être passées à la création d'une instance de champ ;
- « directory » : représente l'emplacement de l'index de Lucene ;
- « indexWriter » : composant central du processus d'indexation. Cette classe permet la création d'un nouvel index, d'ajouter des documents à un index existant, etc. ;
- « analyzer » : ensemble de classes ayant pour but le découpage du texte (contenu dans la valeur d'un « field ») en « token » (mot) et la normalisation du texte à indexer.

Les principaux analyseurs fournis sont :

- « simpleAnalyzer » : découpe le texte en mot et le convertit en minuscule ;
- « stopAnalyzer » : converti le texte en minuscule et supprime les mots vides (mots sans intérêt dans le processus de recherche : le, la, de, un …) ;
- « standardAnalyzer » : combine les deux analyseurs précédents.

La figure 28 schématise les appels aux différentes classes lors d'une indexation.

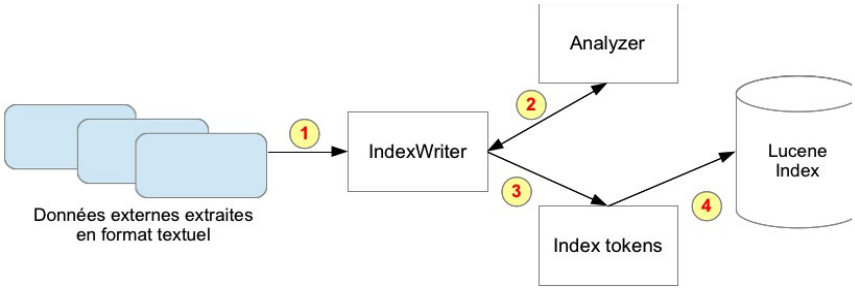


Figure 28 - Processus d'indexation Lucene

Le choix de l'analyseur et donc la façon d'indexer a ensuite une très forte influence sur la recherche et le retour des résultats.

Lors de l'indexation, il faut appliquer les paramètres suivants sur chacun des champs :

- STORE (true ou false) : ce paramètre indique à l'API Lucene si la valeur contenue dans le champ doit être stockée dans l'index ;
- INDEX (YES ou NO, TOKENIZED ou UN_TOKENIZED) : ce paramètre définit si la valeur du champ doit être indexée ou non. Si elle est indexée, il faut préciser si elle est découpée (TOKENIZED) en terme ou considérée comme une chaîne de caractères à part entière (UN_TOKENIZED).

La valeur d'un champ uniquement STORE peut simplement être affichée. Il n'est possible d'effectuer que des recherches sur un champ uniquement INDEX. La valeur ne peut être affichée. Pour un champ STORE et INDEX, la valeur peut être affichée et recherchée.

2.5.1.4 Principes de recherche

La recherche met en œuvre 7 classes Lucene :

- « indexSearcher » : classe donnant accès aux index en recherche ;
- « analyzer » : tout comme pour l'indexation les analyseurs font partie du processus de recherche afin de normaliser les critères de recherche ;
- « queryParser » : un analyseur de requête ;
- « query » : un objet qui représente la requête de l'utilisateur et utilisé par un indexSearcher. Il existe différentes implémentations de requêtes afin de réaliser une recherche, dont les principales sont :
  - « TermQuery » : le type de requête le plus basique pour chercher dans un index. La requête peut être construite en utilisant un ou plusieurs termes ;
  - « PrefixQuery » : recherche sur des termes sur un préfixe donné lors de la construction de la requête. La réponse contiendra les documents contenant des termes qui commencent par ce préfixe ;
  - « PhraseQuery » : on peut chercher une expression, c'est-à-dire une séquence particulière de termes ;
  - « WildcardQuery » : implémente une recherche avec des jokers, qui permet de faire des recherches telles que arch* (qui permet de trouver les documents contenant architecte, architecture, etc.) ;
  - « FuzzyQuery » : rechercher sur des termes similaires. La mesure de similarité est basée sur l'algorithme de Levenshtein[2].
- « hits » : une collection d'éléments résultats de la recherche ;
- « hit » : un élément de la collection des résultats ;
- « document » : un document retrouvé et tel qu'il était lors de son ajout dans l'index (constitué des mêmes champs).

La figure 29 schématise les appels aux différentes classes lors d'une recherche.

2 La distance de Levenshtein mesure le degré de similarité entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre. C'est une distance au sens mathématique du terme, donc en particulier c'est un nombre positif ou nul, et deux chaînes sont identiques si et seulement si leur distance est nulle.

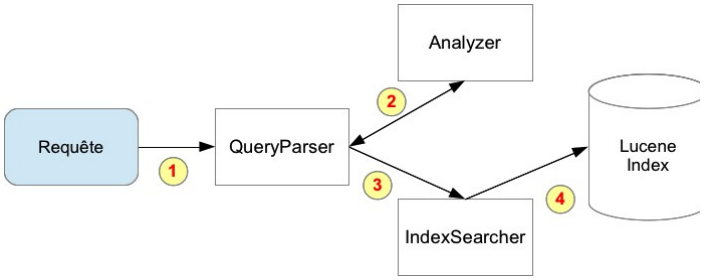


Figure 29 - Processus de recherche Lucene

2.5.1.5 Autres fonctionnalités Lucene

En plus de la librairie lucene-core.jar, Lucene apporte plusieurs autres librairies permettant la personnalisation et l'enrichissement du moteur de recherche. On retrouve entre autres fonctionnalités :

- « lucene-analyzers.jar » : ajout d'analyseurs complexes permettant l'analyse sémantique permettant la stemmatisation[3] et la lemmatisation[4] des termes indexés en fonction de la langue ;
- « lucene-highlighter.jar » : permet de fournir des mots clés dans le contexte. Ces fonctions sont généralement utilisées pour mettre en évidence les termes de recherche dans le texte des pages de résultats ;
- « lucene-memory.jar » : permet de hautes performances lors de recherche plein texte dans l'index ;
- « lucene-xml-query-parser.jar » : analyseur XML pour des requêtes réalisées en XML ;
- « lucene-spellchecker.jar » : vérifie l'orthographe des mots et propose des corrections ;
- etc.

2.5.1.6 Pondération des documents

Le classement Lucene utilise une combinaison du modèle de l'espace vectoriel[5] (VSM) de recherche d'information et le modèle booléen[6] pour déterminer la pertinence d'un document donné par rapport à une requête donnée en fonction des autres documents composant l'index.

L'évaluation du score de chaque document par Lucene dépend fortement de l'indexation. Des poids sont attachés à chaque document en fonction du nombre de termes, de leur présence plus ou moins importante, etc. L'indexation attribue un score à chaque champ du document et son score total est la combinaison de ces scores. Deux documents ayant le même contenu mais indexé différemment, par exemple les mots seront découpés pour l'un et considéré comme des chaînes pour l'autre, n'auront pas le même score sur une même requête.

3 La stemmatisation ou racinisation est le nom donné au procédé qui vise à transformer les flexions en leur radical ou stemme. La racine d'un mot correspond à la partie du mot restante une fois que l'on a supprimé son préfixe et son suffixe, à savoir son radical. Elle est aussi parfois connu sous le nom de stemme d'un mot. Contrairement au lemme qui correspond à un mot réel de la langue, la racine ou stemme ne correspond généralement pas à un mot réel.

4 La lemmatisation est le nom du procédé en traitement automatisé de la langue qui consiste à transformer les flexions en leur lemme. Les flexions sont les différentes formes fléchies d'un même mot. Les formes fléchies correspondent aux formes « conjuguées » ou « accordées » d'un mot de base non conjugué et non accordé : le lemme.

5 Le modèle de l'espace vectoriel sert de base à la représentation des données textuelles par des vecteurs dans l'espace euclidien. Selon ce modèle, l'élément sémantique de chaque document est le terme. Un terme peut être un mot simple ou un mot composé (un groupe de mots). À partir de cette caractéristique, chaque document est représenté par un vecteur des termes.

6 Un modèle booléen est une méthode ensembliste de représentation du contenu d'un document. C'est l'un des premiers modèles utilisés en recherche d'information, permettant de fouiller automatiquement les grand corpus de bibliothèques.

Le score d'un document dans Lucene est un flottant compris dans un intervalle de 0 à 1, qui indique son poids par rapport aux autres documents de l'index.

La figure 30 représente la formule de calcul du score d'un document.

Figure 30 - Formule de calcul de score par Lucene

Le score d'indexation pour un terme de l'index est généré par Lucene par rapport à l'espace des documents. Les facteurs impliqués dans calcul de score d'indexation sont les suivants :

- TF(T,D) (Term Frequency) mesure la fréquence du terme T dans le document D. Les documents qui contiennent le plus d'occurrences d'un terme sont plus valorisés pour ce terme. TF est une sorte de table « document x terme » contenant ces fréquences. Cette valeur est calculée par Lucene lors de l'indexation ;
- IDF(T) (Inverse Document Frequency) mesure la fréquence d'un terme T dans l'espace de document. Les termes communs (apparaissant dans beaucoup de fiches) sont moins valorisés que les termes rares (apparaissant dans peu de fiches). IDF est une table des termes avec leur fréquence globale. Cette valeur est calculée par Lucene lors de l'indexation ;
- lengthNorm(T,C) mesure l'importance du terme T en fonction du nombre total de termes dans le champ C. Moins un champ a de termes, plus il est valorisé. Cette valeur est calculée par Lucene lors de l'indexation ;
- Boost (index) augmente ou diminue le score d'indexation d'un champ, par le biais d'un paramètre donné à Lucene par la méthode setBoost(). La méthode getBoost() permet de collecter les valeurs appliquées. Cette valeur est appliquée par le programmeur et s'applique lors de l'indexation.

Le poids des documents s'obtient, comme le montre la figure 30, par la multiplication de ces valeurs entre elles auxquelles se rajoute les facteurs coord et queryNorm, évoqués pour classer l'ordre d'affichage des documents suite à une recherche sur un ou plusieurs termes :

- coord est le nombre de termes du texte recherché présents dans un document. Cette valeur est calculée par Lucene lors de la recherche ;
- queryNorm est un facteur de normalisation utilisé pour rendre les scores entre des requêtes successives comparables. Ce facteur n'a aucune incidence sur le classement des documents d'une requête donnée. Cette valeur est calculée par Lucene lors des recherches ;
- Boost augmente ou diminue le score sur la recherche d'un champ donné. Cette valeur est définie par le programmeur et est passée en paramètre lors de la recherche afin de donner plus de poids sur un ou plusieurs termes de la recherche.

Globalement, ce score de recherche indique l'importance des termes de la requête dans le document en prenant en compte l'importance de ces termes dans l'espace des documents.

2.5.1.7 Outil d'analyse d'un index Lucene

Luke est un outil Java, libre sous licence Apache 2.0, permettant notamment le monitoring et la consultation des index. Luke rassemble plusieurs classes spécifiques de Lucene pour la recherche et l'analyse. Il permet principalement :

- d'attribuer un classement des termes les plus fréquents ;
- d'exécuter une recherche (par terme, phrase, wildcards, booléenne, etc.) ;
- de supprimer des documents de l'index ou en rajouter ;
- d'optimiser l'index ;

- de calculer le score d'une requête avec le TF-IDF.

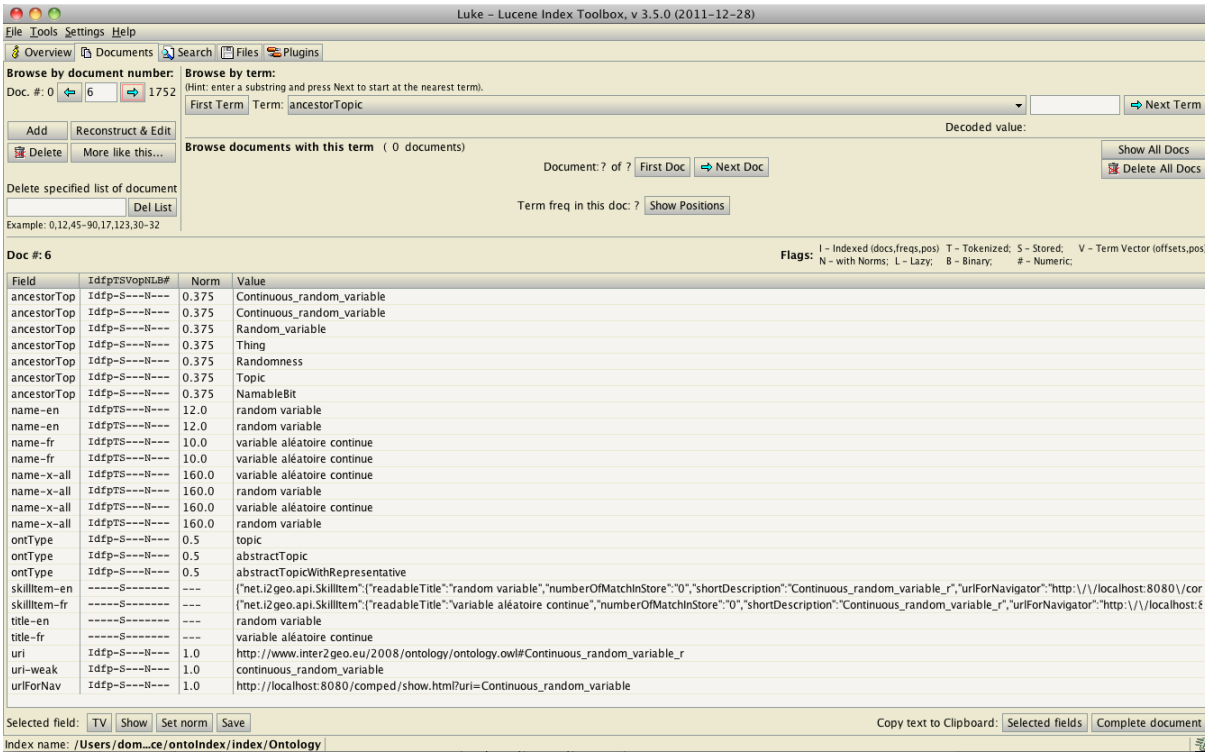


Figure 31 - Vue d'un document Lucene depuis Luke

La figure 31 illustre un document de l'index des capacités et notions d'I2geo.net. Les premières et dernières colonnes montrent les différents champs et valeurs caractérisant la notion « variable aléatoire continue ». La seconde colonne donne des informations sur l'indexation des différents champs et la colonne « Norm » montre les scores des différents champs du document.

Luke est téléchargeable sur le site http://code.google.com/p/luke/.

2.5.2 Indexation des notions et capacités par Intergeo

L'indexation sur i2geo.net est réalisée en deux temps :

1. indexation des notions et capacités de l'ontologie prise en charge par le module Searchi2G ;
2. indexation des ressources prise en charge par le module Curriki.

L'indexation des notions et des capacités de l'ontologie GeoSkills est réalisée par le module SearchI2G basé sur l'API Lucene. L'application Searchi2G est composée de différents paquetages :

- net.i2geo.api : paquetage faisant office d'API dont les fonctions sont appelées par les autres paquetages ;
- skillstextbox : paquetage utilisé pour la partie recherche ;
- net.i2geo.index : paquetage permettant d'effectuer l'indexation et utilisé pour la recherche ;
- net.i2geo.onto : paquetage prenant en charge la manipulation de l'ontologie ;
- net.i2geo.web : paquetage utilisé pour la partie recherche ;
- net.i2geo.xwiki : paquetage permettant la communication avec le composant Curriki.

Dans cette partie nous allons étudier les différentes API utilisées dans lors processus d'indexation ainsi que les classes implémentées dans SearchI2G.

2.5.2.1 Processus d'indexation des notions et capacités

Pour l'indexation, les principaux paquetages utilisés sont net.i2geo.onto et net.i2geo.index. La classe GeoSkillsIndexer, du paquetage net.i2geo.index, effectue l'indexation. Cette classe s'appuie sur différentes classes du module SearchI2G dont principalement IndexHome afin de configurer et utiliser l'index Lucene, GeoSkillsAccess du paquetage net.i2geo.onto pour accéder à l'ontologie et effectuer des raisonnements et GSIUtil pour, entre autre, la pondération des champs.

La figure 32 illustre le processus d'indexation mis en place pour le module SearchI2G.

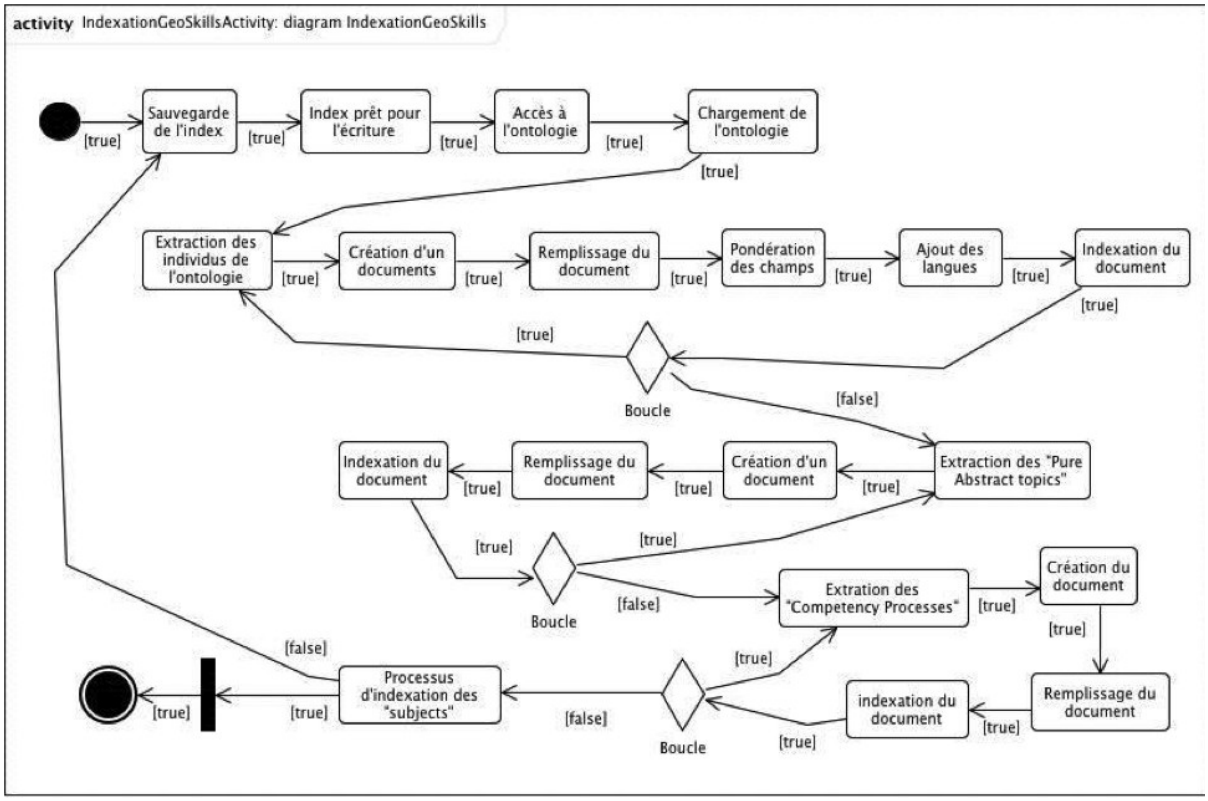


Figure 32 - Processus d'indexation Intergeo

Le processus d'indexation de l'ontologie des « Subjects » reprend le même cheminement que l'indexation GeoSkills.

L'analyse de l'ontologie est réalisée avec l'API OWLAPI permettant de récupérer les individus (ici les capacités, notions et niveaux scolaires), les classes et autres propriétés nécessaires pour la constitution des documents à indexer.

Il y a donc 3 types de documents différents créés lors de l'indexation des notions et capacités de l'ontologie GeoSkills :

- un document concernant les notions, capacités et niveau scolaire ;
- un document spécifique « pure abstract topics » pour les classes des notions abstraites n'ayant pas d'instances ;
- un document spécifique « competency processes » pour les classes de capacités.

2.5.2.2 Processus de constitution des documents

La constitution de ces documents est réalisée à partir des classes GeoSkillsIndexer et GSIUtil. Un document se compose d'un champ auquel correspond une valeur et des paramètres d'indexation.

La classe GeoSkillsIndexer analyse chaque individu de l'ontologie par une boucle. Cette boucle permet d'extraire de l'ontologie et d'ajouter au document Lucene, représentant une notion, capacité ou niveau, les éléments suivants :

- l'URI de l'individu ;
- l'URI (uri-weak) découpée pour obtenir l'identifiant de l'élément dans l'ontologie ;
- le type (notion, capacité ou niveau scolaire) de l'individu. Pour les notions il y a une seconde analyse pour différencier les « CONCRETE_TOPIC » (notions spécifiques aux théorèmes), des « ABSTRACTTOPIC » et des « ABSTRACTTOPIC_WITH_REPRESENTATIVE » ;
- l'URL d'accès à la notion ou compétence par CompEd ou l'URL d'accès à l'OWLDoc pour les niveaux ;
- les ancêtres (les super-classes de l'individu illustrées par la figure 33) de la notion, capacité ou niveau. Pour les capacités, les notions rattachées sont aussi recherchées. Les ancêtres sont déduits par raisonnement grâce à l'API OWLAPI apportant le raisonneur « Pellet » ;

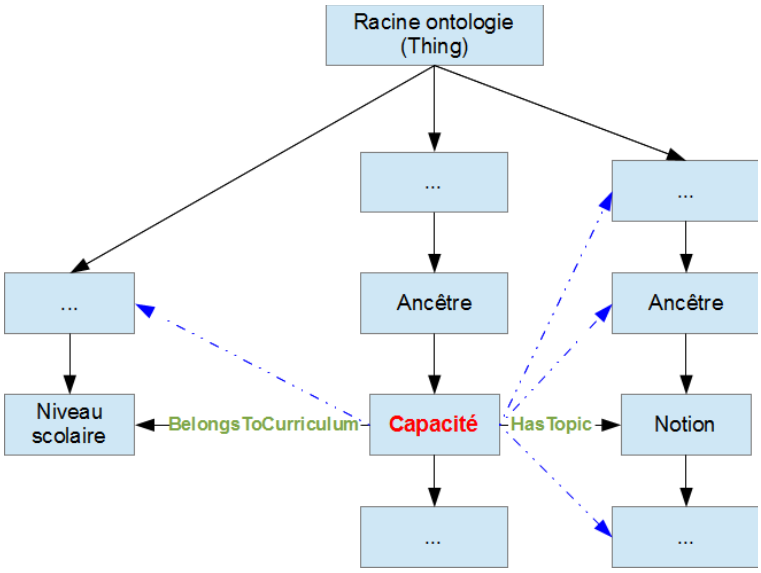


Figure 33 - Relation et héritage entre les éléments de l'ontologie

- les différents noms de l'individu dans les différentes langues des pays participants au projet. Les 5 types de noms potentiels de l'individu sont collectés :
  - defaultCommonName : nom par défaut de l'individu. Ce nom est unique pour chaque individu dans chaqu3e langue ;
  - commonName : nom commun de l'individu. Il est possible de trouver plusieurs noms communs pour chaque langage ;
  - unCommonName : nom peu commun de l'individu. Il est possible de trouver plusieurs noms peu communs pour chaque langage ;
  - rareName : nom rare de l'individu. Il est possible de trouver plusieurs noms rares pour chaque langage ;
  - falseFriendName : nom faux ami. Il est possible de trouver plusieurs noms faux amis pour chaque langage.

La classe GSIUtil applique une pondération au moment de l'indexation sur les champs correspondant :

- BOOST_DEFCOMMONNAME = 7 ;
- BOOST_COMMONNAME = 5 ;
- BOOST_UNCOMMONNAME = 4 ;
- BOOST_RARENAME = 3 ;
- BOOST_FALSEFRIENDNAME = 0.5f.

Certains de ces éléments sont ajoutés à un tableau JSON stocké dans le champ « skillItem » qui fait aussi partie du document. Dans ce tableau, on retrouve notamment le type, le nom par défaut, l'URI, l'URL de l'individu.

Deux autres boucles permettent de récupérer ces informations afin de constituer les documents des « pure abstract topics » et des « competency processes ». En fin de programme, un quatrième type de document est rajouté à l'index. Il est constitué uniquement de la date de modification de celui-ci.

La classe GeoSkillsIndexer utilise ensuite la méthode getWriter de la classe IndexHome pour écrire dans l'index de SearchI2G.

L'analyse, l'extraction et l'indexation des éléments de l'ontologie sont réalisées automatiquement au moment du démarrage de l'application. L'index des capacités et notions est créé.

2.5.2.3 Analyse de l'index de SearchI2G

L'index Lucene, suite au processus d'indexation de l'ontologie GeoSkills.owl, contient 83 champs pour 1 753 documents. Au total, il y a 10 853 termes dans l'index.

Un document correspondant à un élément de l'ontologie se décompose en plusieurs champs contenant les valeurs listées dans le chapitre précédent. Le tableau 2 est une synthèse générique modélisant les champs d'un document de l'index. Il référence aussi les différents problèmes constatés.

Champs	Valeurs	Paramètres	Observations
ancestorTopic	Les URI des ancêtres.	Store : YES Index : UN_TOKENIZED	La même valeur peut être présente plusieurs fois dans le document.
name-<langue>	Les différents noms en fonction de la langue.	Store : YES Index : TOKENIZED	La même valeur peut être présente plusieurs fois dans le document. Le nom du champ peut prendre la valeur qui doit lui être assignée.
name-x-all	Les différents noms.	Store : YES Index : TOKENIZED	La même valeur peut être présente plusieurs fois dans le document.
ontType	Le type de l'élément.	Store ; YES Index : UN_TOKENIZED	La même valeur peut être présente plusieurs fois dans le document.
skillItem-<langue>	Le tableau JSON en fonction de la langue.	Store : YES Index : NO	
title-<langue>	Le nom par défaut en fonction de la langue	Store : YES Index : NO	
uri	L'URI	Store : YES Index : UN_TOKENIZED	
uri-weak	L'URI découpée	Store : YES Index : UN_TOKENIZED	
urlForNav	L'URL	Store : YES Index : UN_TOKENIZED	

Tableau 2 - Index des capacités, notions et niveaux

2.5.2.4 Bilan intermédiaire de l'indexation des notions et capacités

Cette étude a été réalisée suite à la lecture du code du paquetage net.i2geo.index du module SearchI2G et à l'analyse de l'index via l'outil de diagnostic et de développement Luke.

L'indexation ne semble pas optimisée car on retrouve des couples champs/valeurs en double dans les documents, au niveau des champs du type, des ancêtres et des noms référençant les différents noms d'une capacité, notion ou d'un niveau en fonction des langues. Ces couples posent des problèmes au niveau de la pondération des champs de l'index. En effet, si un document contient des doublons, le calcul lengthNorm sera plus bas et la pertinence du document sera donc réduite. Le parcours de l'ontologie doit être corrigé car pour les capacités on remonte une première fois à la racine de l'ontologie pour indexer leurs ancêtres et ensuite plusieurs fois pour indexer les notions qui leurs sont rattachées. Le champ « name-x-all » contenant les différents noms de l'élément dans toutes les langues contient aussi des valeurs en double.

La constitution des documents comporte aussi des problèmes puisque des champs ayant pour nom « name-<valeur> » sont générés en plus des champs « name-<langue> ». La pertinence des champs/valeurs indexées dans les documents devra donc faire l'objet d'une analyse si le processus d'indexation est réutilisé en l'état.

L'application de poids dans le programme lors de l'indexation sur les champs donne une pondération incohérente. Comme le montre la figure 34, les defaultCommonName ont le même poids que les commonName et unCommonName dans les champs name-fr (80) et name-x-all (224).

ancestorTop	Idfp-S---N---	0.25	ConstructionRecipe
ancestorTop	Idfp-S---N---	0.25	ConstructionRecipe
ancestorTop	Idfp-S---N---	0.25	Thing
ancestorTop	Idfp-S---N---	0.25	NamableBit
ancestorTop	Idfp-S---N---	0.25	Topic
ancestorTop	Idfp-S---N---	0.25	Prove_construction_recipe_of_angle_bisector
ancestorTop	Idfp-S---N---	0.25	TransversalCompetency
ancestorTop	Idfp-S---N---	0.25	Justify
ancestorTop	Idfp-S---N---	0.25	NamableBit
ancestorTop	Idfp-S---N---	0.25	Competency
ancestorTop	Idfp-S---N---	0.25	Prove
ancestorTop	Idfp-S---N---	0.25	Justify_or_Prove
ancestorTop	Idfp-S---N---	0.25	Thing
name-	IdfpTS---N---	7.51619	
name-fr	IdfpTS---N---	80.0	justification de la construction de la bissectrice d'un angle
name-fr	IdfpTS---N---	80.0	justification de la construction de la bissectrice d'un angle
name-fr	IdfpTS---N---	80.0	démontrer le procédé de construction de la bissectrice d'un angle
name-fr	IdfpTS---N---	80.0	prouver le procédé de construction de la bissectrice d'un angle
name-x-all	IdfpTS---N---	224.0	justification de la construction de la bissectrice d'un angle
name-x-all	IdfpTS---N---	224.0	justification de la construction de la bissectrice d'un angle
name-x-all	IdfpTS---N---	224.0	
name-x-all	IdfpTS---N---	224.0	démontrer le procédé de construction de la bissectrice d'un angle
name-x-all	IdfpTS---N---	224.0	prouver le procédé de construction de la bissectrice d'un angle
ontType	Idfp-S---N---	1.0	competency
skillItem-fr	-----S-------	---	{"net.i2geo.api.SkillItem":{"readableTitle":"justification de la construction de la bissectrice d'un angle","i
title-fr	-----S-------	---	justification de la construction de la bissectrice d'un angle
uri	Idfp-S---N---	1.0	http://www.inter2geo.eu/2008/ontology/ontology.owl#Prove_construction_recipe_of_angle_bisector
uri-weak	Idfp-S---N---	1.0	prove_construction_recipe_of_angle_bisector
urlForNav	Idfp-S---N---	1.0	http://localhost:8080/comped/show.html?uri=Prove_construction_recipe_of_angle_bisector

Figure 34 - Score erroné dans l'index

Toutes modifications dans l'ontologie en format fichier impliquent obligatoirement une reconstitution de l'index des capacités et notion afin qu'elles soient en compte par le moteur de recherche SkillsTextBox.

L'indexation des ressources par les capacités et notions s'effectue sur un index Lucene différent avec un processus spécifique au composant Curriki. Avant d'aborder cette nouvelle indexation, il convient d'étudier comment sont recherchées les notions et capacité dans l'index de SearchI2G.

2.5.3 Recherche de notions et capacités

Le module SearchI2G amène deux JSP de test permettant d'interroger l'index pour rechercher des notions, capacités et niveau :

- skills-text-box-search.jsp ;
- skills-text-box-editor.jsp.

2.5.3.1 Principes du moteur recherche de notions et capacités

Ces deux IHM présentent une zone de saisie de texte et permettent l'affichage du résultat sous forme d'une liste de suggestions comme le montre la figure 35.

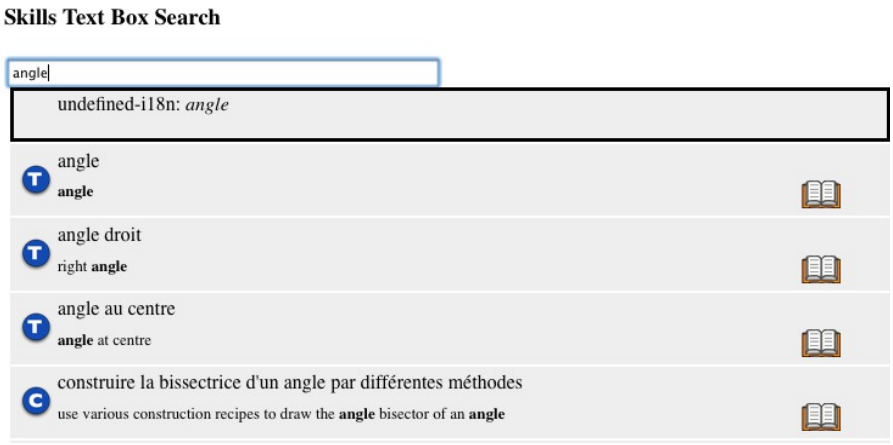


Figure 35 - IHM de test skills-text-box-search.jsp

2.5.3.2 Fonctionnement du moteur recherche de notions et capacités

La recherche de capacités, notions et niveaux est prise en compte par les paquetages net.i2geo.web et skillstextbox-gwt du composant SearchI2G. Le paquetage net.i2geo.web gère la recherche dans l'index Lucene et utilise des classes du paquetage net.i2geo.index. Le paquetage skillstextbox-gwt permet l'affichage en auto-complétion, dans l'IHM de saisie du moteur de recherche, ainsi que la création d'un cache de type cookie dans le navigateur de l'utilisateur. Lors des recherches suivantes sur le même terme, il n'y aura plus d'appel à l'index Lucene, le cache navigateur prend en charge la réponse, si celle-ci n'a pas été supprimée entre temps des cookies. La figure 36 illustre ce processus de recherche.

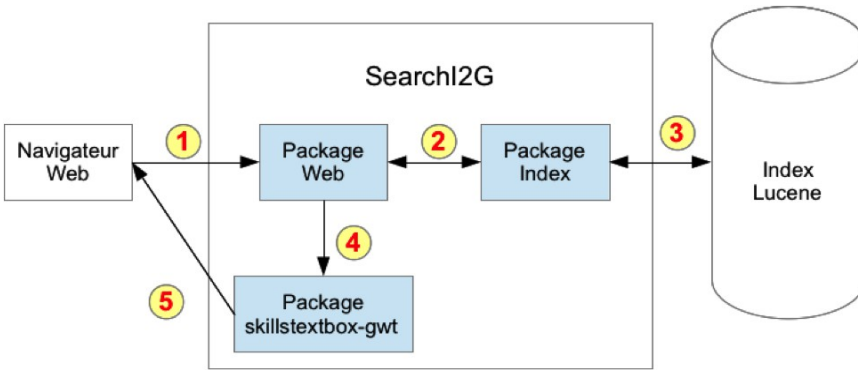


Figure 36 - Processus de recherche de capacités notions

Les principales classes Java utilisée pour la recherche sont « TokenSearchServerImpl.java » et « AutoCompletionServlet.java » du paquetage net.i2geo.web de SearchI2G. Elles utilisent les classes document, search, index et queryParser de l'API Lucene. Elles utilisent aussi la classe « SKBAnalyzer.java » pour l'analyse du texte ainsi que les classes « RSearchQueryExpander.java » et « SKBQueryExpander.java » pour la recherche dans l'index. Ces classes sont implémentées dans le paquetage net.i2geo.index.

La plate-forme étant multilingue, plusieurs analyseurs Lucene sont utilisés :

- analysis.cn.ChineseAnalyzer ;
- ru.RussianAnalyzer ;
- cz.CzechAnalyzer ;
- de.GermanAnalyzer ;
- nl.DutchAnalyzer ;
- fr.FrenchAnalyzer.

2.5.3.3 Exemple d'une recherche d'une notion

L'IHM affiche les 30 premiers documents contenus dans la réponse à la requête. Une réponse est constituée de valeurs classées d'après l'ordre des scores des valeurs dans l'index pondérés et amplifiées par les paramètres de « boost » de la requête recherche. Les résultats sont affichés par ordre décroissant de score.

L'analyse d'une recherche de la notion « angle » par étapes va nous permettre d'analyse plus en détail le déroulement de la recherche.

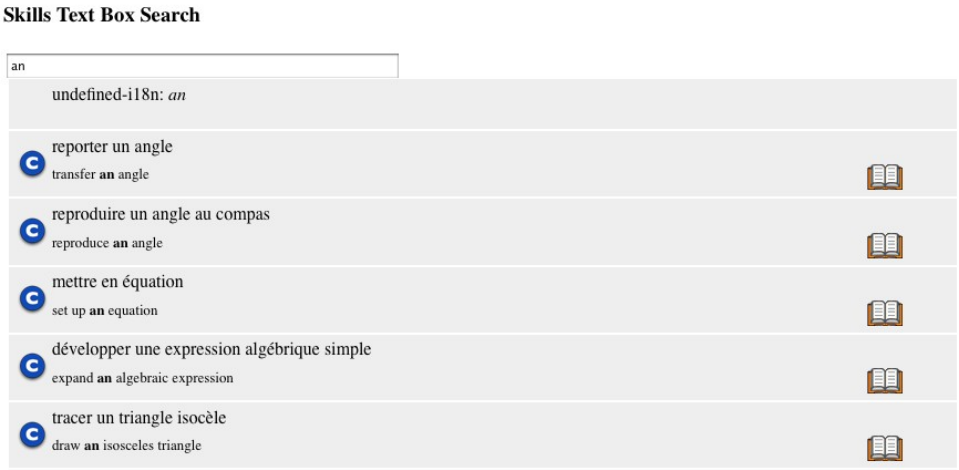


Figure 37 - Recherche sur un fragment de mot

La mise en surbrillance du résultat de la figure 37 démontre que la recherche s'effectue sur le terme anglais « an » et non pas sur le préfixe du mot angle. Il semble donc y avoir un mélange des langues dans le résultat renvoyé.

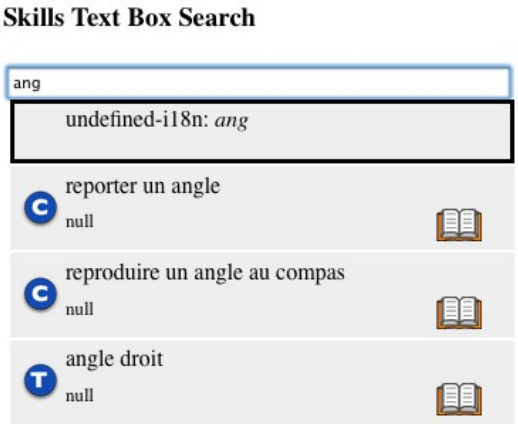


Figure 38 - Recherche sur un fragment de mot suite

La figure 38 montre un dysfonctionnement du retour du résultat avec l'affichage de la valeur « null » en lieu et place du nom par défaut anglais de la notion.

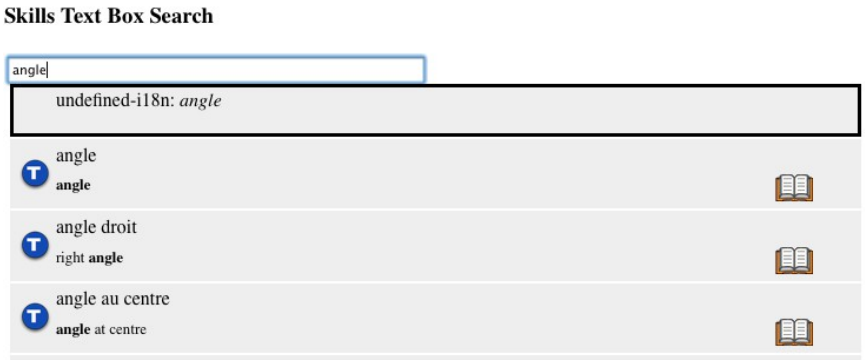


Figure 39 - Résultat de la recherche sur le terme angle

Le résultat final est toutefois cohérent à ce qui est attendu comme le montre la figure 39. La notion angle est bien trouvée et est affichée en premier. Les résultats suivants contenant d'autre terme sont ensuite classés dans l'ordre décroissant lié au score.

2.5.3.4 Pondération lors de la recherche

Une pondération est appliquée sur les champs lors de la recherche au niveau des classes RSearchQueryExpander.java et SKBQueryExpander.java du paquetage net.i2geo.index.

La recherche s'applique donc de manière décroissante sur les champs :

- URI ;
- name-x-all avec les mots complets considérés comme plus importants par rapport aux préfixes de mot ;
- name-<langue> avec les mots complets considérés comme plus important par rapport aux préfixes de mot.

2.5.3.5 Bilan intermédiaire de la recherche de notions et capacités

La recherche permet donc bien de retrouver les notions, capacités ou niveaux extraites de l'ontologie GeoSkills. Cependant, certains problèmes impliquent une analyse plus en profondeur du code source pour correction.

La figure 40 correspondant à une recherche sur deux termes illustre un problème de pertinence et de cohérence dans le classement des réponses. La pondération reste donc à être améliorée.

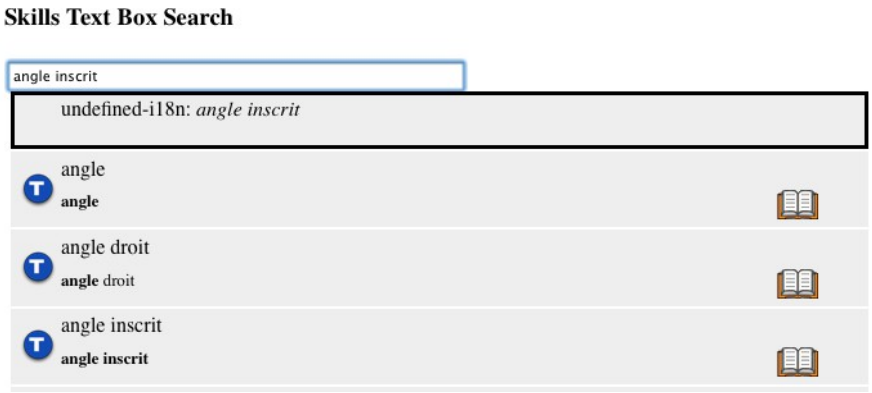


Figure 40 - Recherche de la notion « angle inscrit »

Le mélange des langues ou la prise en compte des termes tels que le, la, les, etc. faussent les premières suggestions retournées par le moteur de recherche puisque le terme exact est préféré au préfixe.

Ce moteur de recherche est intégré au site Curriki par l'intermédiaire du paquetage net.i2geo.xwiki. L'indexation des ressources par les notions et capacités utilise aussi ce moteur de recherche.

2.5.4 Indexation des ressources par les notions et capacités

Dans cette partie, nous allons analyser comment est réalisée l'indexation d'une ressource par les notions et capacités précédemment indexées. L'indexation des ressources est réalisée par l'API Lucene intégrée au module Xwiki du composant Curriki. Ce plugin est au préalable « customisé » par le module i2gCurriki afin de prendre en compte les spécificités d'Intergeo.

2.5.4.1 Processus d'indexation d'une ressource

L'index utilisé pour les ressources est différent de celui généré par SearchI2G. Il n'est pas dédié à l'indexation des ressources et contient donc d'autres champs pour les objets de la plate-forme Curriki tel que la partie blog, wiki, etc.

L'indexation est effectuée au moment de l'installation de la plate-forme. Il n'y a donc pas de ressources à cet instant. L'index est ensuite mis à jour lors de chaque ajout manuel de ressource. Curriki crée le document d'indexation à cet instant. Les classes I2GLuceneProfile et I2GResourceData du composant i2gCurriki font appel au paquetage net.i2geo.xwiki du composant SearchI2G qui interroge l'index ou directement l'ontologie GeoSkills. L'interrogation de l'ontologie rend les mêmes services que l'interrogation de l'index, à l'exception de l'auto-complétion.

2.5.4.2 Exemple d'indexation d'une ressource par les notions et capacités

Lors de l'ajout d'une ressource, le contributeur assigne des notions, des capacités et le ou les niveaux scolaires qui la caractérisent. Le moteur de recherche SearchI2G est donc intégré dans le formulaire d'ajout de la ressource comme l'illustre la figure 41.

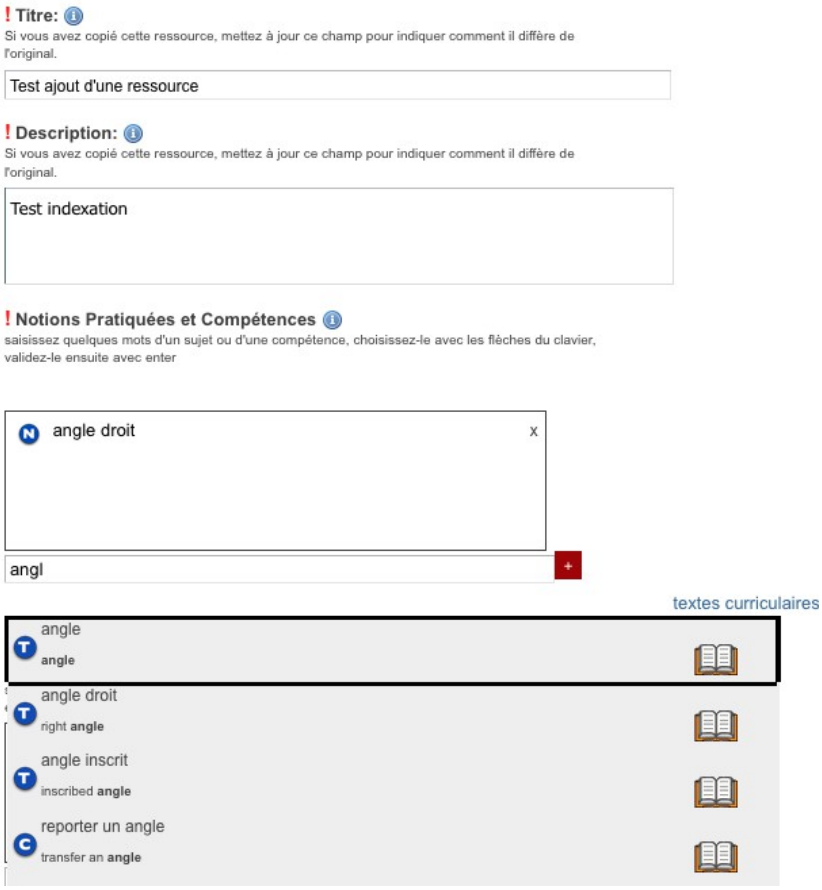


Figure 41 - Indexation d'une ressource par les notions et capacités

L'indexation est ensuite réalisée au moment de l'enregistrement du formulaire.

2.5.4.3 Analyse de l'index des ressources

L'analyse de l'index du Curriki avec l'outil Luke, figure 42, montre qu'il y a 853 champs dans l'index pour 2 227 documents. L'index Lucene compte plus de 94 000 termes.

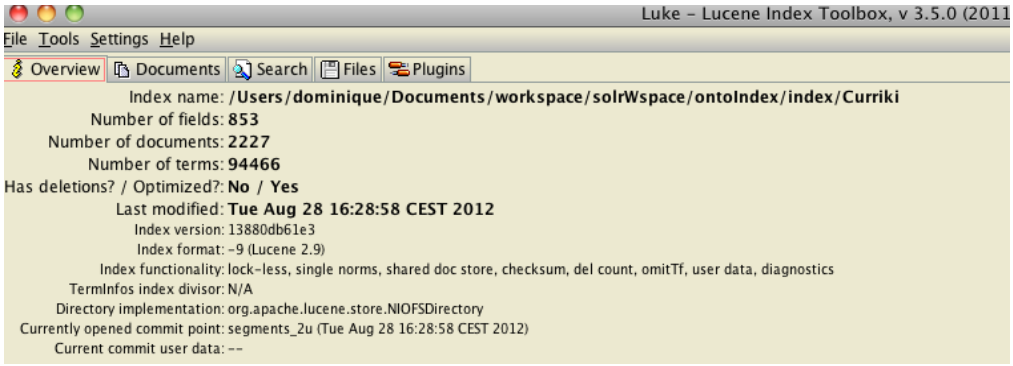


Figure 42 - Index Curriki

L'index Curriki n'est pas réservé à l'indexation des ressources. La majorité des documents de l'index servent pour les différents composants de Curriki. Seulement deux documents caractérisent une ressource. Ces documents s'intitulent « ressource » et resource.objects ».

Le document « ressource » correspond à une brève description de la ressource. Il se compose de 12 champs permettant principalement de faire le lien avec les autres documents de l'index dont « resource.objects ». La structure de ce document est consultable en annexe 2D.

Le document « resource.objects » correspond à la ressource indexée par les notions, capacités et niveau scolaire. Il se compose d'un minimum de 23 champs. La plupart de ses champs sont génériques et définit par Curriki. Le module i2gCurriki introduit 2 champs, spécifiques à Intergeo, intitulés « i2geo.ancestorTopics » et « CurrikiCode.AssetClass.trainedTopicsAndCompetencies ».

Le champ CurrikiCode.AssetClass.trainedTopicsAndCompetencies a pour valeur la liste des uriweak des notions et capacités directement liées à la ressource. Le champ i2geo.ancestorTopics regroupe l'ensemble des uriweak des ancêtres des notions et capacités liées à la ressource. La structure de ce document est consultable en annexe 2E.

2.5.4.4 Bilan intermédiaire de l'indexation d'une ressource par les capacités et notions

L'indexation des ressources par les capacités et notions est spécifique à Curriki et donc dépendante de son code source. Si Sésamath ne souhaite pas réutiliser cette plate-forme parmi ses différents sites, il sera nécessaire de développer un nouveau système d'indexation des ressources qui s'intégrera dans l'infrastructure. Une étude des champs et des valeurs de l'index sera aussi à prendre en compte.

2.5.5 Recherche de ressources par les notions et capacités

Le moteur de recherche implémenté dans le module SearchI2G s'intègre dans le site Curriki grâce au paquetage « net.i2geo.xwiki » et est accessible à partir de la page d'accueil du site pour la recherche de ressources par les capacités, notions et niveaux scolaire. Il est aussi présent dans le formulaire de recherche avancée, illustré par la figure 43, et dans le formulaire d'ajout de ressources étudié précédemment.

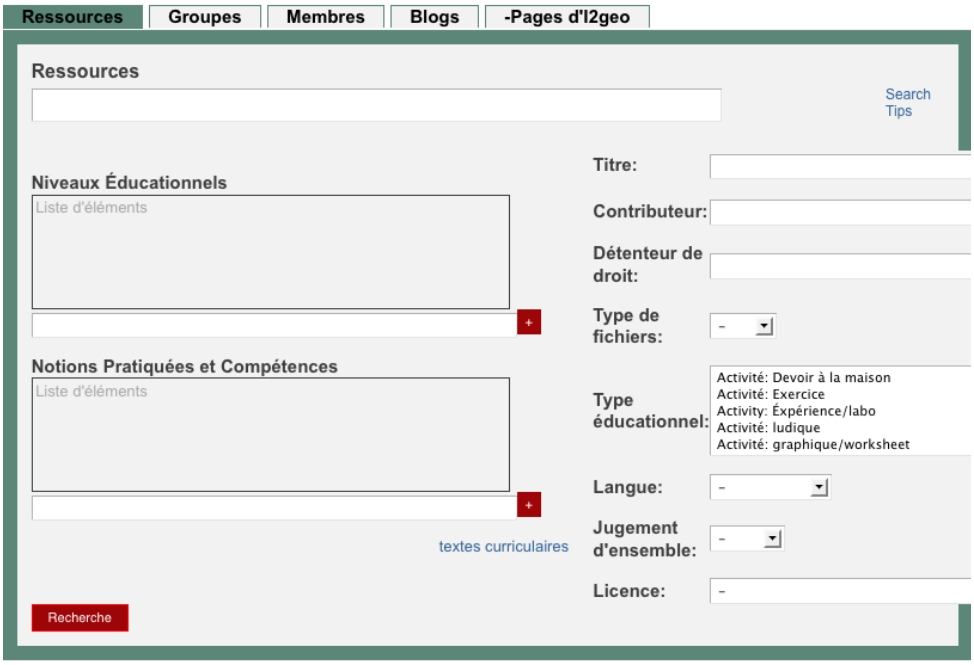


Figure 43 - Formulaire de recherche avancée

Pour ces 2 derniers formulaires, le moteur est découpé en 2 zones de saisie de texte permettant :

- la recherche par notions ou capacités ;
- la recherche par niveau scolaire.

2.5.5.1 Principe de la recherche

La recherche de ressources s'effectue en 2 temps. Dans une première étape, l'index des capacités, notions et niveau est consulté afin de suggérer au chercheur une liste de capacités, notions voire niveaux correspondant aux termes saisies. Lors de la sélection d'une des suggestions, une seconde recherche est lancée sur l'index des ressources. Ce processus est illustré par la figure 44.

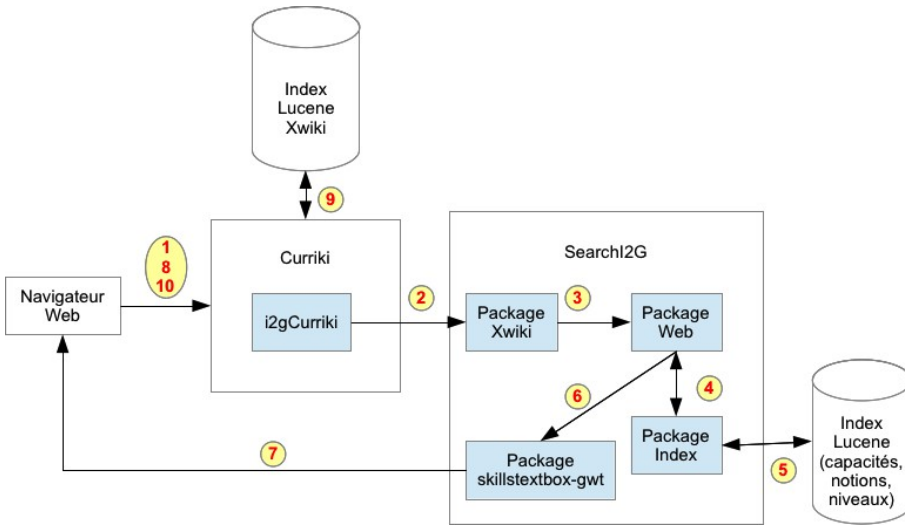


Figure 44 - Processus de recherche d'un ressource par les capacités et notions

La recherche sur l'index des capacités, notions et niveaux renvoie des suggestions indiquant le type, le nom de la capacité, notion ou niveau et le lien vers CompEd modélisé par l'image d'un livre. La liste des fragments d'URI, faisant office d'identifiant des éléments affichés est aussi retournée. Lorsque l'utilisateur sélectionne une capacité, notion ou un niveau, une seconde recherche est réalisée sur l'index Xwiki référençant les ressources. La recherche est réalisée à partir de l'identifiant sur les champs i2geo.ancestorTopics et CurrikiCode.AssetClass.trainedTopicsAndCompetencies.

Une fonctionnalité de recherche plein texte est aussi implémentée. Cette recherche n'accède pas à l'index des capacités, notions et niveaux mais directement sur l'index Xwiki de Curriki.

2.5.5.2 Exemple d'une recherche plein texte

Le principe de recherche par les capacités, notions et niveaux étant similaire à l'étude réalisée dans le chapitre 2.5.3.3, cette partie se focalise sur la recherche plein texte.

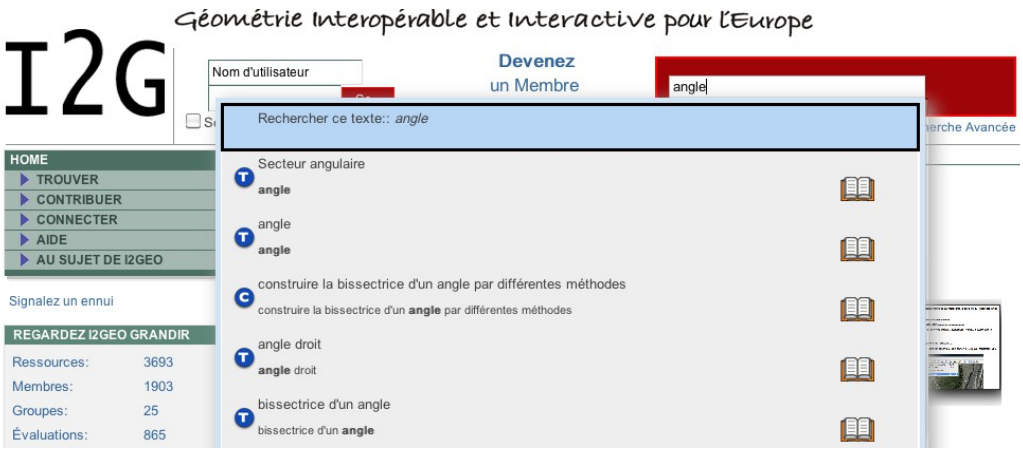


Figure 45 - Recherche plein texte

La figure 45 montre que la première ligne des suggestions affichées correspond à la zone de recherche de type plein texte. L'utilisateur sélectionnant cette zone lance une recherche directement sur l'index des ressources. La recherche est alors réalisée sur les champs :

- title ;
- CurrikiCode.AssetClass.language ;
- ft ;
- title.stemmed ;
- ft.stemmed.

2.5.5.3 Pondération lors de la recherche

Le détail des résultats, figures 46 et 47, montre l'application d'une pondération sur les différents champs consultés.

Figure 46 - Pondération d'une recherche sur la notion « angle »

Lors d'une recherche sur les notions et capacités, la recherche sur le champ i2geo.ancestorTopics est pondérée à 0.8 alors que le champ CurrikiCode.AssetClass.trainedTopicsAndCompetencies garde la valeur de recherche par défaut équivalent à 1. La recherche est ainsi privilégiée sur le champ des notions et capacités directement liées aux ressources puis ensuite sur les ancêtres.

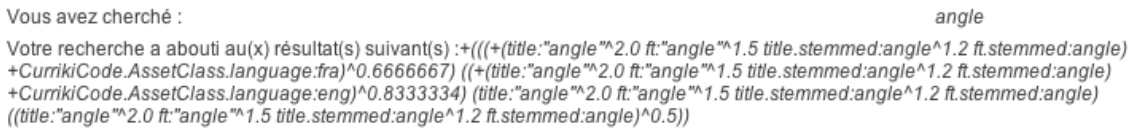


Figure 47 - Pondération d'une recherche plein texte sur le terme « angle »

Lors d'une recherche plein texte, la recherche du terme dans l'index est alors réalisée par priorité décroissante sur les champs :

- title ;
- ft ;
- title.stemmed ;
- ft.stemmed ;
- CurrikiCode.AssetClass.language.

Les ressources renvoyées par le moteur de recherche, figures 48 et 49, sont donc affichées et classées différemment en fonction du mode de recherche.

Figure 48 - Résultat d'une recherche sur la notion « angle »

Figure 49 - Résultat d'une recherche plein texte sur le terme « angle »

2.5.5.4 Bilan intermédiaire de la recherche de ressources par les notions et capacités

La recherche de document par les capacités, notions et niveaux scolaires est fonctionnelle. Elle est réalisée en 2 temps. Une première recherche sur l'index des capacités et notions permet d'afficher une liste de suggestion à l'utilisateur. Lorsque celui-ci sélectionne un élément de la liste une seconde recherche est réalisée sur l'index des ressources afin de retourner l'ensemble des ressources indexées à la notion ou capacité sélectionnée.

La recherche est réalisée en priorité sur le champ contenant les notions et capacités directes. Les ressources trouvées à partir du champ ancêtre seront affichées plus bas dans le classement. La pertinence du classement des ressources est à étudier plus en détail si la solution est retenue. La recherche plein texte apporte un niveau de recherche différent où les capacités, notions et niveaux scolaires ont la même importance que l'ensemble des autres caractéristiques définissant la ressource.

2.6 Bilan et choix de réutilisation de la plate-forme

La plate-forme i2geo.net se décompose en 5 applications Web Java déployées sur un serveur d'applications tel que Tomcat. Le module central, SearchI2G, utilise les API OWLAPI, Pellet pour l'analyse et l'extraction des éléments de l'ontologie, ainsi que l'API Lucene pour la partie indexation et recherche de ces éléments. L'utilisation de l'API Lucene implique du développement pour pouvoir réaliser un système d'indexation et de recherche. L'API Lucene utilisée, en 2009 par i2geo.net, était en version 2.9.3. Les versions stables actuelles sont estampillées 3.6.2 et 4.2. Les changeLog (journaux des modifications) Apache Lucene évoquant les nouvelles fonctionnalités et d'autres rendues obsolètes impliquent une réécriture partielle du code source pour mettre à jour le système.

Malgré quelques problèmes non bloquant, l'installation de l'ensemble des composants a toutefois permis de mettre à disposition une plate-forme de démonstration hébergée et maîtrisée par le LIG afin d'étudier le fonctionnement. L'installation a permis de mettre en évidence la complexité de réutilisation et l'interdépendance des différents modules.

L'indexation des éléments de l'ontologie est à vérifier pour être optimisée et notamment enlever les valeurs doublons sur certains champs ayant un impact sur la pondération lors de la recherche. La pertinence du résultat en début de recherche est très faible car les articles sont pris en compte et considérés comme des mots. La recherche par TermQuery prime sur la recherche PrefixQuery en terme de poids et donc dans le classement des suggestions affichées pour l'utilisateur.

Le système est peu évolutif et manque de souplesse dans l'hypothèse d'ajouter ou retirer un champ dans l'index des capacités et notions. Dans ce cas de figure, il faudrait modifier les différentes classes code source de l'application SearchI2G. Ce qui implique des compétences Java que ne possède pas Sésamath à l'heure actuelle. L'évolution, l'administration et la maintenabilité du système ne sont donc pas pérennes.

Le choix de la plate-forme Curriki et Xwiki ne correspond pas non plus aux besoins de Sésamath. Le module SearchI2G est donc à réécrire en partie pour enlever les liens entre les 2 applications.

Toutefois, une démonstration du fonctionnement, au représentant des utilisateurs de Sésamath, a permis de valider que les principes correspondent aux attentes et besoins de Sésamath mais que les composants i2geo.net ne seraient pas réutilisés suite aux problèmes constatés. De plus, le choix du tout Java ne correspondait pas à l'architecture et aux technologies maîtrisées par Sésamath.

Une étude des moteurs d'indexation Open Source est donc nécessaire afin de déterminer quel système pourra mettre en œuvre, de manière avantageuse et simple, les concepts implémentés dans i2geo.net.

3. Systèmes d'indexation et de recherche – État de l'art

La possibilité de réutiliser des composants de la plate-forme Intergeo ayant été écartée, il convient de recenser et d'étudier les caractéristiques des différents moteurs d'indexation et de recherche libre. Le choix d'une technologie basée sur Lucene semble toutefois incontournable afin de reproduire le plus fidèlement possible les principes mis en œuvre dans le projet Intergeo.

Le chapitre 3.1 de cet état de l'art correspond à un rapport rédigé par le CRIM (Centre de Recherche Informatique de Montréal). Les droits d'auteurs et d'utilisation de ce rapport sont disponibles en annexe 3A. Cette analyse objective permet de découvrir et comparer très clairement les 2 principaux moteurs de recherche basés sur l'API Lucene, que sont Solr et ElasticSearch [CRIM 2012].

3.1 Étude réalisée par le CRIM

3.1.1 Mise en contexte

Ce document présente un aperçu de deux moteurs de recherche basés sur Lucene : Solr et ElasticSearch. L'objectif est de présenter les différences entre les deux projets et d'identifier la technologie la plus appropriée dans un contexte de mise à l'échelle. Il sera question de Lucene, de son intégration dans les moteurs de recherche, des techniques employées pour la mise à l'échelle d'un engin de recherche et des caractéristiques propres à chacune des technologies étudiées.

3.1.2 Lucene

Le moteur d'indexation et de recherche ouvert le plus répandu est Lucene. Il s'agit d'une librairie Java pouvant être intégrée dans d'autres applications pour y ajouter des fonctions de recherche plein-texte personnalisées. On y retrouve des fonctions couvrant le classement des résultats par pertinence, l'analyse de phrase, les caractères joker, la recherche par champ, la lemmatisation, les requêtes complexes, etc.

La librairie est toutefois livrée sans transformateur de contenu, interface utilisateur, gestionnaire de permissions ou mécanisme de distribution des index. Il laisse le soin au programmeur de développer, connecter et configurer tous les composants requis pour offrir une fonctionnalité de recherche à son application.

C'est ce qui a été fait dans une multitude d'applications dans des domaines variés, comme les CMS, la gestion documentaire, les portails Web, etc. La librairie est intégrée dans des produits comme Liferay, Alfresco, Nuxeo, Hadoop, OpenCMS, eXo Platform, xWiki, IBM Enterprise Search, Documentum, JIRA et plusieurs autres. Des sites Web, comme eBay, Amazon, AOL, eHarmony ou Macy's se sont aussi basés sur Lucene pour leurs fonctionnalités de recherche.

3.1.3 Moteurs de recherche autonome

Au lieu d'intégrer directement Lucene à son application ou son site Web, il est possible de passer par une application dédiée à la recherche, intégrant la librairie pré-configurée. Notons, entre autres,

Solr, Nutch ou ElasticSearch qui entrent dans cette catégorie. De telles distributions sont livrées avec leurs propres fonctionnalités, comme la recherche par facettes, la gestion des synonymes, des analyseurs de requêtes complexes, des filtres, des fonctions mathématiques pour le calcul de pertinence, etc. Elles ont aussi leurs propres mécanismes de connecteurs pour y ajouter des fonctionnalités diverses et gérer leur configuration.

Ces « moteurs de recherche » offrent leurs propres API, souvent sous forme de services. Avec une couche REST ou SOAP, on peut alors utiliser d'autres langages que Java pour le développement des interfaces de recherche et l'indexation des documents.

Ce sont ces applications qui ajoutent aussi des outils avancés pour la mise à l'échelle et la distribution des index sur plusieurs serveurs.

3.1.4 Mise à l'échelle d'un moteur de recherche

On peut demander un déploiement distribué (ou mise à l'échelle) d'un moteur de recherche pour diverses raisons :

- Support d'un très grand nombre de requêtes en simultané ;
- Très gros volume de données à conserver (plusieurs dizaines de Go) ;
- Indexation rapide ou instantanée (pour temps réel) ;
- Réplication des données sur plusieurs serveurs ;
- Haute disponibilité (minimiser l'impact de la perte d'un serveur).

Un déploiement distribué implique l'utilisation de plusieurs serveurs qu'on désigne comme « nœuds ». Pour un nombre de requêtes important, une telle architecture permettra de répartir les requêtes sur plusieurs machines qui effectueront l'analyse de la requête, interrogeront l'index et classeront les résultats en parallèle.

Si le volume de données est important, le partitionnement de l'index Lucene en plusieurs morceaux (nommés « shards ») stockés sur plusieurs nœuds permet d'augmenter la performance de manière significative. Cette technique réduit la taille de chaque index pouvant bénéficier d'une mise en cache optimale en mémoire RAM et d'une récupération plus rapide de l'information. Des fonctions présentes dans Lucene permettent aussi de regrouper les résultats provenant de plusieurs sources et d'offrir ainsi un classement normalisé des résultats à l'utilisateur.

Le partitionnement de l'index sur plusieurs nœuds augmente aussi la vitesse d'indexation puisque des serveurs différents peuvent être impliqués pour indexer des documents distincts. Par exemple, si le document X est placé dans l'index A et le document Y dans l'index B, les deux pourront être traités en simultané de manière totalement indépendante. Ce volet dépend des règles de partitionnement mises en place. Il est possible de diviser l'index par type de document, par utilisateur, par champs de recherche ou même en fonction de l'identifiant unique du document (assurant une répartition plus égale entre les nœuds).

Finalement, un déploiement distribué permet une meilleure tolérance aux fautes, évitant les interruptions ou la perte de données causées par le mauvais fonctionnement de l'un des serveurs. Certaines installations distribuées peuvent assurer qu'une réplication sur plusieurs emplacements physiques soit effectuée de manière automatique, offrant ainsi un mécanise de sauvegarde robuste. La reprise en cas de panne sur un serveur peut aussi être automatisée lorsqu'un serveur redevient disponible.

3.1.5 Moteurs de recherche avec fonctions de mise à l'échelle

Solr et ElasticSearch sont deux moteurs de recherche autonomes pouvant être exploités à l'aide de services REST depuis une application externe. Nous comparerons ici les fonctionnalités qu'ils offrent, plus particulièrement dans un contexte distribué pour une mise à l'échelle.

3.1.5.1 ElasticSearch

ElasticSearch a été conçu initialement comme un moteur de recherche distribué. Il gère automatiquement le découpage de l'index en « shards » répliqués sur plusieurs serveurs (ou nœuds). Il est possible de bénéficier de haute disponibilité avec un minimum de configuration. Un serveur responsable d'un « shard » est chargé d'en pousser les nouvelles informations sur les répliques, assurant ainsi une mise à jour pratiquement en temps réel. Lors de la réception d'une requête, un nœud produira des sous-requêtes à exécuter sur chacun des « shards » concernés, pour agréger par la suite les résultats obtenus dans un processus s'apparentant au « map / reduce ». La solution est aussi livrée avec son propre mécanisme de surveillance des serveurs, permettant de détecter qu'un nœud n'est plus disponible. On peut aussi répliquer l'index vers un emplacement externe « gateway » qui ne sera pas impliqué dans l'exécution des requêtes, mais qui pourra être utilisé pour restaurer un environnement en cas de panne.

Parmi les caractéristiques du projet, notons la possibilité d'exécuter des scripts (MVEL, JavaScript, Python et autres) sur les serveurs de recherche afin de calculer le contenu d'un champ ou la pertinence d'un document. Tout le contenu indexé est lu et écrit en format JSON via les API REST ou Java. Il n'impose aucun schéma ou structure particulière sur le contenu et effectue une détection des types de données. ElasticSearch implémente aussi la recherche par Facette, le surlignage des résultats et la recherche géo spatiale. Une fonctionnalité qui lui est propre est le « Percolator » qui permet d'enregistrer des requêtes pour être ensuite notifié d'un changement aux résultats. Cela permet, par exemple, de reproduire le comportement des « Twitter Live Stream » (Widget de recherche Twitter) où les résultats sont actualisés en temps réel lorsque de nouveaux commentaires s'ajoutent.

3.1.5.2 Solr

L'objectif premier de Solr est d'offrir un moteur de recherche d'entreprise, recueillant des données provenant de plusieurs sources et permettant la recherche depuis un site Web ou un intranet. Il ajoute à Lucene la navigation par facettes, le surlignage de liens, l'utilisation des synonymes, la complétion semi-automatique, un correcteur d'orthographe basé sur l'index, des fonctions de calcul de pertinence, les transactions, un traitement des requêtes personnalisé via une configuration XML, des API (REST, Ruby, Java, etc.) pour l'indexation et la recherche, ainsi que des gabarits pour le formatage des résultats. Des connecteurs permettent l'indexation de plusieurs formats (MS Office, OpenOffice, PDF, RTF, etc.) ainsi que le contenu de bases de données. Un schéma est requis mais ce dernier peut comporter des champs dynamiques.

Le projet Solr comporte une communauté très active et du support commercial de la part d'entreprises telles que Lucid Imagination et Constellio). Il est un projet Apache depuis 2006 et son équipe de développement a été fusionnée à celle de Lucene en 2010.

Solr est paramétré pour offrir des performances très élevées grâce à une mise en cache intelligente (auto-warming) et à la distribution des recherches sur plusieurs serveurs. Par contre, le déploiement en mode distribué n'est pas automatique et n'incorpore pas toutes les fonctionnalités. L'administrateur doit lui-même diviser l'index en « shards » et décider du rôle de chacun des serveurs. Les requêtes dans un mode distribué doivent spécifier dans quels « shards » rechercher et l'agrégation des résultats doit passer par un algorithme d'approximation (la comparaison n'étant pas effectuée sur une base uniforme).

3.1.6 Comparaison et conclusion

En termes de maturité du projet, de la quantité d'utilisateurs, de la taille de la communauté de développement, du support commercial et de la quantité de fonctionnalités, Solr se démarque d'ElasticSearch. Les équipes de développement de Lucene et de Solr étant maintenant intégrées, les besoins de ce dernier influencent le développement du cœur de Lucene qui est modifié pour répondre à des exigences de performances et d'analyse de requêtes avancées. Le support des transactions, présent dans Solr, constitue un atout lorsque des données affectant plusieurs documents doivent être enregistrées ou pour s'assurer de l'intégrité d'un processus de mise à jour. Le tout répond bien aux besoins des applications de recherche d'entreprise.

Du côté d'ElasticSearch, on retrouve une communauté réduite mais très dynamique, qui se compose d'un développeur principal et de quelques contributeurs. À l'heure actuelle, aucune entreprise n'offre de distribution commerciale. Son point fort est la facilité avec laquelle il est possible de déployer une grappe de serveurs pour gérer un index distribué de très grande taille. C'est à ce niveau qu'il a fait sa marque et plusieurs développeurs ayant essayé de déployer une telle infrastructure sous Solr ont finalement adopté ElasticSearch, créant un véritable engouement autour du projet.

Lorsqu'il est exécuté sur un index de taille raisonnable (< 1M de documents), Solr offre des performances supérieures à ElasticSearch. C'est toutefois le contraire qui se produit avec une taille d'index plus importante ou lorsqu'un très grand volume de données est indexé en temps réel. Le tout est dû au fait que le traitement est distribué sur des index plus petits, hébergés sur plusieurs serveurs. Si 100 nouveaux documents sont ajoutés en même temps, ceux-ci peuvent être répartis sur 10 « shards » différents effectuant le traitement en parallèle. L'absence de transactions réduit aussi le temps de traitement pour l'ajout de documents.

Une répartition de la charge et une distribution de l'index peuvent aussi être réalisées avec Solr, mais le tout exige beaucoup de configuration et la mise en place de règles définies manuellement. Le projet SolrCloud 4.0 vise à répondre à ce problème en offrant des fonctions pour le découpage automatique de l'index en plusieurs « shards ». La gestion de la grappe de serveurs et de sa configuration ne sont toutefois pas incorporées dans le produit, mais reposent plutôt sur le produit Apache ZooKeeper. À l'heure actuelle, cette solution n'est pas encore comparable à ElasticSearch en termes de convivialité et de facilité de déploiement.

En somme, ElasticSearch représente, à l'heure actuelle, la meilleure solution pour l'indexation temps réel, les notifications en temps réel (fonction « Percolator ») et pour la simplicité de déploiement lorsqu'on est en présence d'un index de très grande taille en mode distribué. Du côté de Solr, il représente un choix sûr pour l'ajout de fonctions de recherche avancées à tout genre d'application. Il peut être répliqué pour répartir la charge sur plusieurs serveurs et offre des performances supérieures dans le contexte d'une utilisation conventionnelle (beaucoup plus de lecture que d'écriture et une taille d'index de moins de 1M de documents).

Étant donné l'intégration des projets Solr et Lucene, il est à prévoir que les fonctionnalités avancées de Solr soient de plus en plus intégrées au noyau de Lucene, ce qui permettra à d'autres solutions, comme ElasticSearch, d'en profiter. Solr continu aussi son évolution et des composants comme SolrCloud, pourraient éventuellement leur permettre de rattraper ElasticSearch pour une utilisation dans un contexte distribué.

3.2 Quelques moteurs Open Source complémentaires

3.2.1 Sphinx

Sphinx est un autre moteur de recherche plein texte, écrit en C++. Ce moteur est capable d'indexer une énorme quantité de documents, plusieurs Gigas, avec une grande rapidité, et d'effectuer des recherches en un temps record. Il est capable d'indexer des données provenant de plusieurs sources, et fournit des API dans la plupart des langages majeurs.

Sphinx est avant tout conçu pour indexer et rechercher rapidement une base de données SQL. Il limite la recherche aux champs définis dans le modèle de données. Sphinx est optimisé principalement pour la lecture ou la plupart du temps la lecture seule. Il ne supporte pas de mises à jour partielles d'un indice. Il est toutefois possible d'utiliser un « index delta », contenant seulement les documents modifiés, et que la recherche s'effectue sur l'index principal. Une tâche planifiée fusionne ensuite les index. Il y a donc un décalage sur le moment où les modifications seront visibles dans l'index.

3.2.2 MNoGoSearch

MNoGoSearch est un moteur de recherche écrit en C, composé d'un indexeur capable de naviguer sur des pages HTML ou en texte pur et d'une interface de requête pour effectuer les recherches. Il peut aussi indexer beaucoup d'autres types de données en utilisant des parsers (analyseurs) externes.

MNoGoSearch fonctionne « à la Google », le robot explore régulièrement le contenu, et construit son index ainsi. Son principal avantage repose sur la gestion de la configuration de l'indexeur. MNoGoSearch est capable d'indexer des sites en plusieurs langues, des groupes de sites. Il supporte plusieurs bases de données, et il est possible d'effectuer des requêtes via des fonctions PHP.

L'inconvénient majeur de MNoGoSearch réside sur le peu d'options disponibles pour les requêtes de recherche. Les résultats peuvent être triés par pertinence, dernière date de modification et par titre. MNoGoSearch reste donc un moteur de recherche basique.

3.2.3 Xapian

Xapian est une API similaire à Lucene diffusée sous licence GPL. Elle est écrite en C++, avec des extensions qui permettent de l'utiliser à partir de langages tels que Perl, Python, PHP, Java, etc. Xapian est un outil très souple qui permet à des développeurs d'ajouter très facilement à leurs applications des fonctions d'indexation et de recherche très sophistiquées.

Xapian supporte des opérations avancées de recherche telles que :
- la combinaison des termes de recherche (et, ou, sauf...) ;
- la stemmatisation (recherche étendue aux pluriels, conjugaison du verbe, etc.) ;
- la recherche avec synonymes ;
- la suggestion orthographique.

Xapian est basé sur un système d'indexation probabiliste tandis que Lucene s'appuie sur un modèle d'espace vectoriel.

Omega est le moteur de recherche, de référence, utilisant sur cette API. Omega peut être comparé à Solr ou ElasticSearch.

3.2.4 Zend Lucene

Zend_Search_Lucene est un moteur de recherche de contenus principalement textuels écrit entièrement en PHP 5. Comme il stocke ses index sur le système de fichiers et qu'il ne requiert pas de base de données, il peut offrir des fonctionnalités de recherche à presque n'importe quel site écrit en PHP.

Zend_Search_Lucene dispose des caractéristiques suivantes :
- « Ranked searching » : les meilleurs résultats sont retournés en premier ;
- Plusieurs puissants types de requêtes : phrase, booléen, joker (wildcard), proximité ;
- intervalle et bien d'autres ;
- Recherche par champ spécifique.

Zend_Search_Lucene est une ré-implémentation du projet Apache Lucene en PHP.

Cette solution est d'une extrême simplicité, il devient possible de mettre en place un moteur de recherche PHP puissant avec une simplicité et une rapidité déconcertante. La documentation de Symfony fournit d'ailleurs un exemple d'implémentation.

En revanche, les performances ne sont pas au rendez-vous, et si le nombre d'objets indexés devient trop important (plusieurs dizaines de milliers), Zend Lucene deviendra vite inutilisable. Zend_Search_Lucene est donc à réserver pour des besoins spécifiques, mais à la volumétrie limitée.

3.3 Bilan et choix du moteur d'indexation et de recherche

Le choix de Solr semble donc être la meilleure solution afin de prendre en compte les contraintes d'architecture matérielle de Sésamath. Le fonctionnement d'ElasticSearch en grappe de serveur ne correspond pas à l'infrastructure de Sésamath qui héberge un maximum de serveurs virtuels sur un seul serveur physique. Le nombre de ressources étant de quelques dizaines de milliers, les avantages apportés par ElasticSearch perdent de l'attrait. De plus Solr apporte toutes les fonctionnalités souhaitées pour reproduire les principes d'indexation et de recherche mis en œuvre dans le projet Intergeo. La forte communauté Apache et la documentation fournie assurent aussi une maintenabilité et une aide précieuse en termes de support pour Sésamath.

Il convient donc d'analyser plus en détail le fonctionnement de Solr et de mettre en œuvre un prototype basé sur cette solution afin d'essayer de reproduire les principes du système Intergeo étudiés dans le chapitre précédent.

4. Indexation des ressources Sésamath avec Solr

La solution Intergeo ayant été écarté, l'objectif de ce chapitre est d'étudier plus en détail le fonctionnement générique du moteur d'indexation Solr sélectionné suite à l'étude réalisée dans le chapitre précédent. La seconde partie de ce chapitre consiste à réaliser des prototypes, illustrés par la figure 50, permettant de reproduire les différentes étapes d'indexation et de recherche implémentées dans le cadre du projet Intergeo.

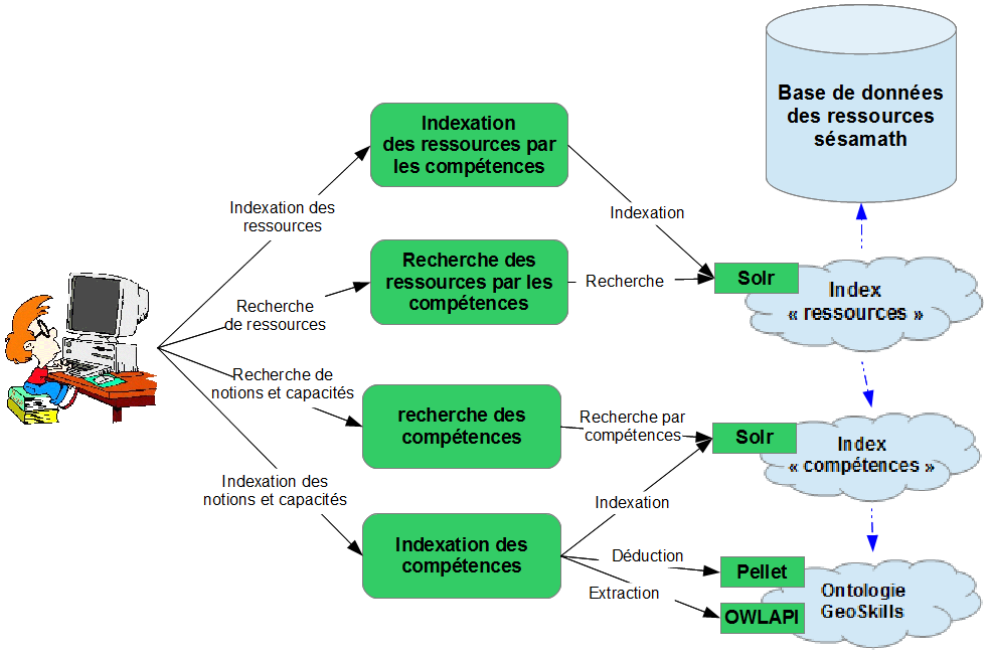


Figure 50 - Mise en œuvre des principes Intergeo

Quatre prototypes sont donc nécessaires. Afin de pouvoir indexer les ressources de Sésamath par les notions et capacités, il est indispensable de mettre en œuvre un système d'indexation effectuant une extraction des notions et capacités de l'ontologie fichier. Une interface de recherche des compétences est aussi pré-requise pour lier les notions et capacités aux ressources. Des bilans intermédiaires des différentes étapes permettent de valider si les principes vus dans Intergeo sont bien respectés. L'objectif final étant de pouvoir retrouver les ressources Sésamath grâce aux notions et capacités.

Au préalable, il convient d'étudier les fonctionnalités Solr et son fonctionnement avant de pouvoir réaliser le premier prototype.

4.1 Présentation de Solr

La couche Solr basée sur l'API Lucene apporte une solution simplifiée de l'utilisation de la librairie Lucene. Depuis mars 2010, le projet apache Solr a fusionné avec le projet Lucene. Toute évolution Lucene est donc prise en compte dans Solr. Solr est un moteur de recherche libre écrit en

Java et basé sur la bibliothèque Lucene, proposant des API XML et JSON par HTTP [SOLR 2012]. Cette application Web se déploie dans un serveur d'applications de type Tomcat, Jetty, etc.

4.1.1 Configuration de Solr

La configuration du service Web Solr se limite à 2 fichiers XML : solrconfig.xml et schema.xml. Des fichiers textes permettent aussi de personnaliser et d'optimiser l'indexation.

4.1.1.1 Fichier solrconfig.xml

Le fichier solrconfig.xml contient la plupart des paramètres de configuration spécifiques à l'application Solr. Ce fichier XML se décompose en sections permettant de configurer le service Web à différents niveaux :

- lib : permet de définir quels plugins Solr peut/doit utiliser ;
- dataDir : précise le répertoire où sont stockés le ou les index ;
- indexConfig : permet de contrôler le comportement de bas niveau de l'indexation (en relation avec l'API Lucene) ;
- Update Handler : gestion du comportement de mise à jour des index ;
- Query : Contrôle tout ce qui est lié à une requête dont les paramètres de mise en cache ;
- Request Dispatcher : indique comment Solr gère les requêtes HTTP ;
- The Highlighter plugin configuration : permet de configurer et personnaliser la mise en évidence des termes lors de la recherche ;
- Admin/GUI : permet la personnalisation de la page d'administration Web ;
- etc.

4.1.1.2 Fichier schema.xml

Il s'agit du fichier qui décrit comment seront indexées les données dans Solr. Il définit les types de champs, quel champ doit être utilisé comme clé unique, quels champs sont obligatoires et comment indexer et rechercher chaque champ.

La section <types> permet de définir une liste de déclarations <fieldType> utilisée dans le schéma, avec la classe Solr sous-jacente qui doit être utilisée, ainsi que les options par défaut pour les champs. Un fieldType est défini par un nom et des paramètres permettant d'indiquer la classe (String, Text, Integer, etc.) et la méthode de tri qui lui est appliquée (ascendant, descendant, etc.). La figure 51 présente la configuration d'un fieldType.

```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">
      <analyzer type="index">
     <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"
enablePositionIncrements="true" />
      <!-- in this example, we will only use synonyms at query time
       <filter class="solr.SynonymFilterFactory" synonyms="index_synonyms.txt"
ignoreCase="true" expand="false"/>
      -->
      <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
     <analyzer type="query">
     <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"
enablePositionIncrements="true" />
      <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt"
ignoreCase="true" expand="true"/>
      <filter class="solr.LowerCaseFilterFactory"/>
```

```
      </analyzer>
</fieldType>
```

Figure 51 - Exemple de déclaration d'un type de champs

D'autres paramètres importants peuvent également être définis lors de la déclaration d'un champ. Il s'agit de la méthode d'analyse lors de l'indexation ou de la recherche. La méthode d'analyse permet de définir comment seront traitées les données du champ, à savoir son mode de découpage en mots (tokenizer) et les filtres qui lui seront appliqués (filter). Les filtres permettent par exemple de supprimer les mots vides, de mettre tous les mots en minuscule, de convertir les mots accentués en leurs équivalents sans accent, etc.

La section <fields> est l'endroit où sont énumérées les déclarations de chaque <field> utilisé dans un document. Chaque <field> a un nom devant être utilisé lors de l'ajout des documents ou d'une recherche et est associé à un <type> qui identifie le nom du fieldType. Il y a des différentes options permettant de compléter la définition du champ. La figure 52 décrit la déclaration de différents champs d'un index générique.

```
<field name="id" type="string" indexed="true" stored="true" required="true" />
<field name="name" type="text_general" indexed="true" stored="true"/>
<field name="manu" type="text_general" indexed="true" stored="true"
omitNorms="true"/>
<field name="cat" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="features" type="text_general" indexed="true" stored="true"
multiValued="true"/>
<field name="includes" type="text_general" indexed="true" stored="true"
termVectors="true" termPositions="true" termOffsets="true" />
<field name="weight" type="float" indexed="true" stored="true"/>
<field name="price" type="float" indexed="true" stored="true"/>
<field name="popularity" type="int" indexed="true" stored="true" />
<field name="inStock" type="boolean" indexed="true" stored="true" />
<field name="store" type="location" indexed="true" stored="true"/>
```

Figure 52 - Exemple de déclaration des champs

Les données indexées dans Solr par le service d'indexation sont contenues dans un datagramme de type XML. Lorsque Solr traite ce datagramme, il rapproche le nom de chaque élément XML avec le nom d'un champ. Il manipule alors la donnée associée à cet élément comme cela est défini par le type du champ.

4.1.1.3 Fichiers TXT

L'analyse textuelle de Solr repose en partie sur l'utilisation de lexiques. Généralement, les termes y sont enregistrés sous forme normalisée sans accents ni majuscules. Quatre lexiques au format fichier « .txt » sont utilisés par Solr :

- contractions.txt : prise en compte des élisions. Les élisions sont une particularité du français, qui consistent en une contraction de mots comme « le » ou « de » quand ils sont suivis d'une voyelle. Il est possible de supprimer ces élisions à l'aide ce fichier ;
- protwords.txt : l'indexation utilise généralement la lemmatisation, qui consiste à réduire les mots à leur racine, par exemple « developp » pour retrouver aussi les articles contenant le mot développer quand on cherche avec le mot développement. Cependant, il arrive qu'il y ait des lemmatisations indésirables, indexant sous un même « lemme » deux mots qui n'ont aucun rapport. Il est possible d'empêcher la lemmatisation de certains mots en les listant dans ce fichier ;
- stopwords.txt : les stopwords sont les mots insignifiants. Un mot considéré comme insignifiant sera ignoré lors de l'indexation si il est listé dans ce fichier ;
- synonyms.txt : Il est possible d'étendre la recherche aux synonymes s'ils sont répertoriés dans ce fichier.

4.1.2 Processus d'indexation

Solr maintient l'index d'une collection d'objets appelés « documents ». Un index est un ensemble de documents analysés et traités suivant un schéma défini. Un document est un ensemble de champs (fields) auxquels sont associées des valeurs.

Solr permet d'indexer par défaut les fichiers XML et CSV, mais il y a possibilité d'indexer des données à partir de documents de type PDF, PPT, DOC … (il faut toutefois mettre en place une étape préalable de conversion de ces fichiers vers un format texte). Solr peut aussi indexer directement une base de données par l'intermédiaire d'un connecteur (JDBC par exemple).

L'indexation des documents, peut s'effectuer à partir d'un fichier XML, mis en entrée de Solr, respectant le schéma défini dans le fichier schema.xml de type :

- noms des champs ;
- valeurs correspondantes aux types des champs.

La figure 53 représente un document de ce fichier XML.

```
<add><doc>
      <field name="id">3007WFP</field>
      <field name="name">Dell Widescreen UltraSharp 3007WFP</field>
      <field name="manu">Dell, Inc.</field>
      <field name="cat">electronics</field>
      <field name="cat">monitor</field>
      <field name="features">30" TFT active matrix LCD, 2560 x 1600, .25mm dot
pitch, 700:1 contrast</field>
      <field name="includes">USB cable</field>
      <field name="weight">401.6</field>
      <field name="price">2199</field>
      <field name="popularity">6</field>
       <field name="inStock">true</field>
      <field name="store">43.17614,-90.57341</field>
</doc></add>
```

Figure 53 - Exemple d'un document XML en entrée de Solr

L'indexation des documents est ensuite réalisée par requête HTTP en postant le ou les fichiers XML nécessaires, comme illustré par la figure 54.

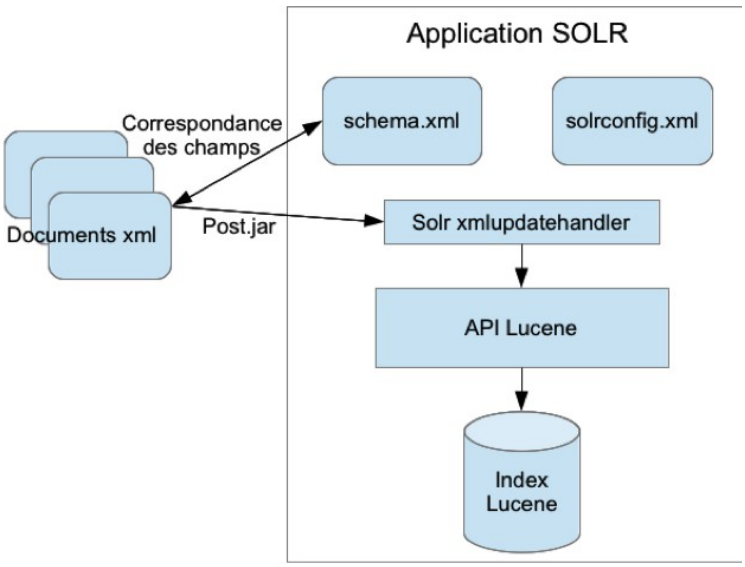


Figure 54 - Processus d'indexation Solr

Solr livre un client Java « post.jar » permettant d'indexer les fichiers XML ou autres en entrée. L'indexation peut donc être lancée par l'intermédiaire de la commande « java -jar post.jar *.xml ». Mais aussi depuis une application cliente écrite dans un autre langage ou encore en automatisant le processus via les tâches planifiées grâce à un script UNIX, comme le montre la figure 55.

```
curl http://localhost:8983/solr/update --data-binary '<commit/>' -H 'Content-type:application/xml'
```

Figure 55 - Commande d'indexation CURL

Solr offre aussi la possibilité de mettre à jour ou supprimer des documents de l'index sans stopper la continuité de service.

4.1.3 Processus de recherche

La réponse est par défaut retournée au format XML, mais de nombreux outils ont été conçus pour traiter la réponse dans différents langages. Ainsi, les langages les plus utilisés sont gérés :

- XML (standard) ;
- JSON (notation JavaScript) ;
- XSLT ;
- etc.

Un index Solr s'interroge par l'URL http://server/solr/select?q=<**champ**>:<**mots recherchés**> appelée avec la méthode HTTP GET. La figure 56 illustre le processus de recherche dans Solr.

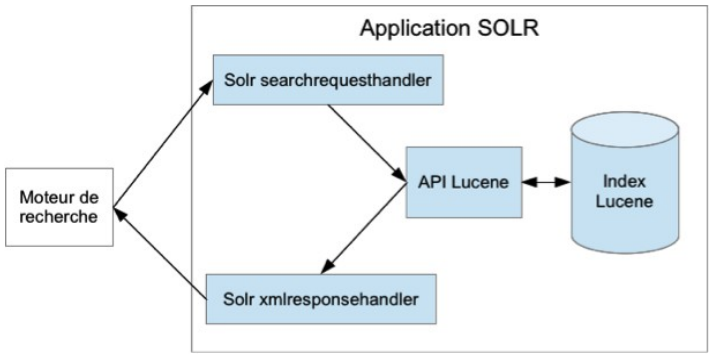


Figure 56 - Processus de recherche Solr

4.1.4 Autres fonctionnalités

En plus des fonctionnalités d'indexation et de recherche Solr apporte différentes fonctionnalités qui lui sont spécifiques et d'autres issues de l'API Lucene. Les principales sont :

- Réplication : un Solr Master (serveur sur lequel est réalisé l'indexation) et plusieurs « Solr Searchers » (serveurs esclaves permettant de réaliser les requêtes) ;
- Highlighting : cette fonctionnalité permet de mettre en valeur les termes recherchés, dans la réponse retournée, afin de mettre en œuvre un affichage en surbrillance. Cette fonctionnalité est amenée par l'API Lucene ;
- SpellCheck : si des termes sont proches de certains présents dans les documents indexés, SpellCheckComponent renvoie des propositions de termes à rechercher. Cette fonctionnalité est amenée par l'API Lucene ;
- Facettes : cette fonctionnalité permet de regrouper les informations par catégories et offre aux utilisateurs les moyens de filtrer une collection de données en choisissant un ou plusieurs critères. Le regroupement dynamique des résultats de recherche dans des catégories permet aux utilisateurs de « forer » dans les résultats par n'importe quelle valeur dans n'importe quel domaine. Cette fonctionnalité est amenée par l'API Lucene ;

- Importations de données : possibilité d'importer et d'indexer des données depuis une base de données ou d'un flux RSS ;
- Scoring : la notion de calcul du score correspond au calcul effectué afin de déterminer l'ordre des résultats suite à une requête. Cette fonctionnalité est amenée par l'API Lucene ;
- Cache warming : pré-remplissage du cache avec des recherches typiques. Le cache Solr est associé à un indexSearcher (« vue » sur l'index). Tant que cette vue est utilisée les objets mis en cache n'expirent pas et sont utilisables. La vue change lorsqu'il y a une nouvelle indexation. L'ancienne vue est fermée uniquement lorsqu'elle aura traité toutes les demandes en cour ;
- Multiindex : Solr offre la possibilité de créer plusieurs index sur une instance de l'application ;
- Gestion dynamique d'index : possibilité d'administrer l'index ou les index sans arrêter Solr ;
- Interface d'administration Web ;
- indexation de document Word, PDF, etc. en couplant Solr avec l'application Apache Tika ;
- etc.

L'ensemble des fonctionnalités Solr sont listées et détaillées sur la page http://lucene.apache.org/solr/features.html.

4.1.5 Calcul des scores

Solr utilise la formule de calcul de Lucene (évoquée dans le chapitre 2.5.1.6 de ce mémoire) pour appliquer un score aux documents lors de l'indexation et classer les documents par pertinence lors des recherches.

Le modèle de notion de base est « TF-IDF ». Les facteurs de notation appliqués de base sont :
- TF représente la fréquence des termes : plus de fois un terme de recherche apparaît dans un document, plus le score du document est élevé ;
- IDF signifie la fréquence inverse dans les documents : les termes rares ont un score plus élevé que les termes communs ;
- coord est le facteur de coordination : s'il y a plusieurs termes dans une requête, plus le document contient ces termes plus son score sera élevé ;
- lengthNorm : moins un champ d'un document contient de termes, plus il est valorisé par rapport au champ d'un autre document contenant plus de terme ;
- boost lors de l'indexation : si un coup de pouce a été spécifié pour un document au moment de l'indexation, les scores pour les recherches qui correspondent à ce document seront renforcées ;
- clause de boost d'une requête : un utilisateur peut explicitement renforcer l'importance d'une partie d'une requête sur une autre.

C'est ce modèle de base qui est utilisé dans le système d'indexation des ressources Sésamath dans les chapitres suivants.

4.1.6 Clients Solr

Il existe plusieurs clients, dans différents langages, permettant d'interagir avec le service Web Solr et implémenter une IHM.

Les technologies utilisées par Sésamath s'apparentant aux langages Web PHP et JavaScript, on peut lister les clients suivants :
- Solr PECL : l'extension Solr PECL est légère et riche en fonctionnalités permettant aux développeurs utilisant Apache Solr via PHP de communiquer facilement et efficacement avec le service Web Solr en utilisant une API orientée objet ;
- Solr-client-PHP : une bibliothèque pour indexer et rechercher des documents ;

- Solarium : une bibliothèque pour les applications PHP qui non seulement facilite la communication Solr, mais aussi tente de modéliser avec précision les concepts Solr ;
- Ajax-Solr : une bibliothèque permettant de créer une interface utilisateur.

La liste complète des clients est disponible sur http://wiki.apache.org/solr/IntegratingSolr.

4.1.7 Outil de test

L'index Solr étant un index Lucene, l'outil Luke présenté dans le chapitre 2.5.1.7 peut être réutilisé.

Il est à noter que Solr amène aussi une interface d'administration Web nommée Solaritas et développée en Java. Cette console d'administration permet, tout comme Luke de consulter la structure l'index, d'effectuer des requêtes complexes sur celui, etc. Solaritas a aussi des fonctionnalités plus spécifiques à Solr telles que la consultation des statistiques ou des traces d'utilisations. La figure 57 présente la page d'accueil de cet outil.

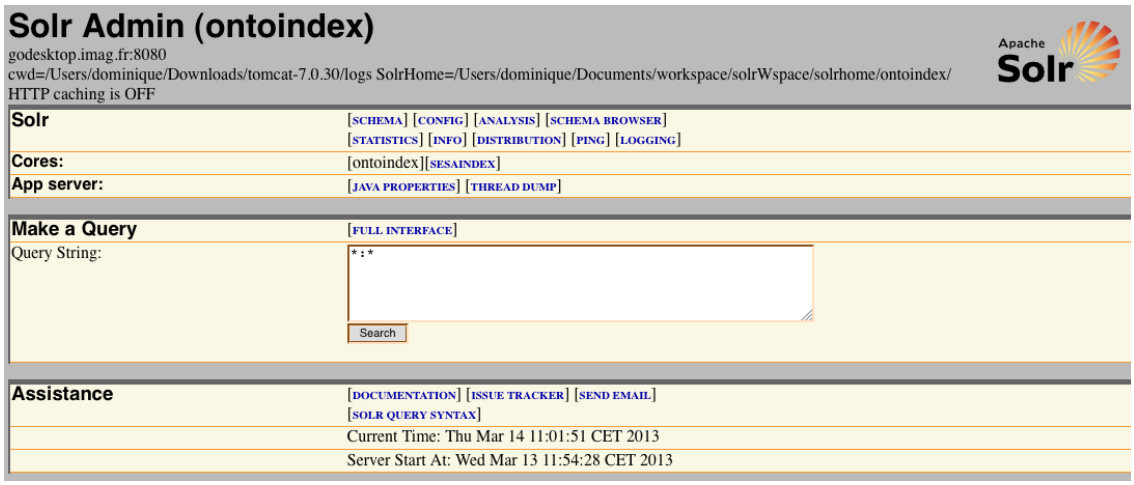


Figure 57 - IHM Solaritas

Cette interface d'administration est accessible par l'URL http://<server>:<port>/solr/admin.

4.2 Développement de l'indexation des notions et capacités

L'objectif premier du prototype est de reproduire l'indexation des notions et capacités réalisée dans le cadre du projet Intergeo. Des itérations successives permettront ensuite d'adapter l'indexation aux besoins du projet Compmep.

4.2.1 Présentation du prototype ontoindexation

L'application « ontoindexation » est un prototype permettant d'analyser l'ontologie GeoSkills et d'extraire les éléments afin de générer des documents XML au format d'entrée de Solr. Ce programme a pour objectif d'effectuer le même service d'indexation que le composant SearchI2G qui utilisait directement l'API Lucene.

Ce programme développé en Java est construit avec Apache ANT pour les tâches de compilation, constitution du JAR et exécution. Il s'appuie sur trois API permettant d'extraire les données de l'ontologie à indexer et de créer le document XML à injecter dans Solr. [ANT 2012]

L'extraction des classes, instances et autres propriétés de l'ontologie GeoSkills.owl a été réalisé à l'aide des API « Pellet » version 2.0.0, pour les raisonnements sur l'ontologie, et « OWLAPI » version 2.2.0-r1317 pour la manipulation de l'ontologie. Cette version d'OWLAPI est livrée par Pellet. L'écriture des données extraites dans un fichier XML s'appuie sur la dernière version (2.0.3) de l'API « JDOM » permettant la manipulation de fichiers XML [JDOM 2012].

Les documentations en ligne, de ces différentes API, sont consultables par les liens suivants :
- PELLET : http://clarkparsia.com/pellet/docs/ ;
- OWLAPI : http://owlapi.sourceforge.net/2.x.x/documentation.html ;
- JDOM : http://www.jdom.org/docs/faq.html.

Les versions utilisées pour les API Pellet et OWLAPI datent de 2009 car les versions plus récentes indiquent que l'ontologie GeoSkills n'est pas cohérente. Mais avant de présenter plus en détail le fonctionnement de ce prototype, il convient de configurer le service Web Solr pour pouvoir réaliser l'indexation.

4.2.2 Configuration de l'index ontoindex

Afin de pouvoir réaliser une indexation reproduisant le fonctionnement d'Intergeo, un index nommé « ontoindex » a été créé dans Solr. Ses différents fichiers de configuration évoqués dans le chapitre 4.1 ont donc été adaptés.

4.2.2.1 Fichier web.xml

A minima, il est nécessaire d'éditer le fichier web.xml présent dans le <répertoire d'installation Tomcat>/webapps/solr/WEB-INF/ afin d'indiquer le chemin vers le répertoire hébergeant les fichiers de configuration indispensables pour le fonctionnement et l'indexation de Solr. La figure 58 donne un exemple de configuration.

```
<env-entry>
      <env-entry-name>solr/home</env-entry-name>
      <env-entry-
value>/Users/dominique/Documents/workspace/solrWspace/solrhome/</env-entry-value>
      <env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

Figure 58 - Paramétrage du fichier web.xml

4.2.2.2 Fichier solrconfig.xml

Pour notre prototype, le fichier utilisé est issu de l'exemple fourni par l'archive de l'application Solr. Il a été épuré de la configuration des cas de test fournis sans autres modifications par ailleurs. Néanmoins, cela reste provisoire en attendant le développement du prototype pour la recherche.

4.2.2.3 Fichier schema.xml

Ce fichier de configuration a été totalement réécrit pour prendre en compte les spécificités des champs et des valeurs qui seront indexées. Un niveau, une capacité ou une notion est défini par les champs listés dans le tableau 3.

Champs	Indexed	Stored	Multivalued	Observations
uri	false	false	false	Identifiant (pour l'ontologie) d'une capacité/notion/niveau avec URL.
uriweak	true	true	false	Identifiant sans l'URL.
ontType	true	true	false	Champ déterminant si le document est pour une capacité, une notion ou un niveau.
ontTypeComp	false	false	true	Type complémentaire d'une notion.
topicRelation	false	false	true	Champ déterminant les notions en relation avec une capacité.
ancestorTopic	false	false	true	Champ déterminant les ancêtres des notions en relation avec une capacité.
ancestor	true	true	true	Ancêtres de la notion, capacité ou niveau.
urlForNav	true	true	false	Champ contenant l'URL vers « CompEd » de la capacité/notion ou vers l'OWLDoc pour un niveau.
pathwaysRelation	false	false	true	Parcours éducatif en relation avec un niveau scolaire.
ancestorPathway	false	false	true	Ancêtres du parcours éducatif en relation avec un niveau scolaire.
regionRelation	false	false	true	Région éducative en rapport avec le niveau.
ancestorRegion	false	false	true	Ancêtres d'une région éducative en relation avec un niveau scolaire.
defaultCommonName	true	true	false	Nom par défaut de la capacité, notion ou du niveau.
commonName	false	false	true	Noms communs de la capacité, notion ou du niveau.
unCommonName	false	false	true	Noms peu communs de la capacité, notion ou du niveau.
rareName	false	false	true	Noms rares de la capacité, notion ou du niveau.
falseFriendName	false	false	false	Noms « faux ami »» de la capacité, notion ou du niveau.
datemodif	false	false	false	Le champ « **datemodif** » est utilisé par 1 seul document spécifique et différent des documents référençant les capacités, notions ou niveaux. Il est créé en prévision d'une éventuelle prise en compte des modifications dans l'index.

Tableau 3 - Liste des champs pour un document représentant une capacité, notion ou niveau scolaire

Nous avons utilisé « StandardTokenizerFactory » pour un découpage « mot par mot ». La documentation en ligne d'Apache Solr référence les différents « Tokenizers » utilisables ainsi que leur fonctionnement (lien : http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters).

4.2.2.4 Fichiers TXT

Pour la configuration de l'index ontoindex, seul le fichier des stopwords est utilisé. La liste des termes qui le composent sont énumérés en annexe 4A.

4.2.2.5 Création de l'index

Un simple redémarrage de l'application Solr prend en compte la configuration et initialise l'index ontoindex vide.

4.2.3 Extraction et indexation des notions et capacités de l'ontologie

Le prototype d'indexation est constitué de 4 classes :

- PropertyLoader.java : classe appelée par la classe principale ontoIndexation.java afin de récupérer les variables d'environnement renseignées dans le fichier de configuration param.properties ;
- ontoIndexation.java : classe principale qui effectue l'analyse, l'extraction et l'indexation des éléments de l'ontologie GeoSkills.owl ;
- doXml.java : classe appelée par la classe principale ontoIndexation.java afin de générer le fichier XML qui sera fourni au service Web Solr lors de l'indexation ;

- updateIndex.java : classe appelée par la classe principale ontoIndexation.java en fin de traitement pour lancer l'indexation du fichier XML ;

4.2.3.1 Fonctionnement du prototype

La classe ontoIndexation.java parcourt, analyse et crée une indexation dans Solr pour l'ontologie GeoSkills. Cette classe s'appuie sur la classe PropertyLoader.java pour lire le fichier « param.properties » afin de récupérer des variables spécifiques telles que le chemin d'accès à l'ontologie, le répertoire de sauvegarde du fichier XML à générer, les URL vers l'éditeur CompEd et les fichiers HTML de documentation de type OWLDoc pour les caractéristiques des niveaux scolaires. Le fichier param.properties regroupe dans un même fichier des URL utilisées lors d'une installation, et évite de les avoir dans le code source du programme, comme c'était le cas dans SearchI2G.

L'ontologie est ensuite parcourue par le programme grâce à des fonctions de l'API OWLAPI. La fonction du raisonneur Pellet « getInconsistentClasses » permet de définir si l'ontologie est cohérente ou non. Si le test de cohérence est réussi, un fichier XML de données d'indexation est initialisé par la classe doXml.java et les éléments de l'ontologie sont chargés en mémoire pour collecter les informations à indexer.

Le fichier XML contient un ensemble de documents où chaque document correspond à un élément de l'ontologie. Les champs et les valeurs de chaque document correspondent aux caractéristiques de l'élément à indexer.

Il a été décidé de ne plus tenir compte des « PURE_ABSTRACT_TOPIC » qui résultaient d'une anomalie de l'ontologie GeoSkills. Les competency processes (classes de compétences) ne sont plus traitée non plus car elles ne sont pas utilisées pour l'indexation des ressources par les capacités et notions.

4.2.3.2 Analyse et extraction de l'ontologie

Le programme est constitué d'une boucle permettant de récupérer les informations des individus de l'ontologie de type Competency, Topic et Level. Dans le traitement des éléments, il a été décidé de ne pas tenir compte des éléments :

- Thing correspondant à la racine de l'ontologie ;
- NamableBit qui avait été mis en place pour l'indexation d'Intergeo ;
- Competency correspondant à la racine des capacités ;
- Topic correspondant à la racine des notions.

En effet, ces éléments n'apportent pas de sens au moment de l'indexation des ressources par les notions et les capacités. L'annexe 4B liste les différents éléments extraits de l'ontologie ainsi que les différentes classes des différentes API utilisées.

4.2.3.3 Génération d'un fichier XML

Les documents du fichier XML sont constitués au fil du programme et stockés en mémoire grâce à l'API JDOM. L'API permet ensuite de créer le fichier grâce à la méthode XMLOutputter qui s'appuie sur la méthode FileOutputStream de la JVM. Le flux des documents est donc enregistré sur un seul fichier en fin de programme ontoIndexation.java.

Des exemples des différents types de document issus du fichier XML, généré suite à l'analyse de l'ontologie GeoSkills, sont consultables en annexes 4C, 4D et 4E.

4.2.4 Indexation des notions et capacités

Une fois le fichier XML généré, la classe ontoIndexation.java fait appel à la classe updateIndex.java pour exécuter la commande d'indexation présentée par la figure 59.

```
proc.exec("java -Durl=" + solrUpdate + " -jar " + pathToPost + " " + pathToXml);
```

Figure 59 - Commande d'indexation de updateIndex.java

Cette commande se compose de différentes variables d'environnement initialement renseignées dans le fichier de propriétés :

- solrUpdate correspond à l'URL de mise à jour de l'index Solr ;
- pathToPost correspond au chemin vers l'application post.jar livrée dans le module ontoIndexation ;
- pathToXml correspond au chemin vers le fichier XML généré suite à l'analyse de l'ontologie.

Les documents du fichier XML, représentant les capacités, notions et niveaux de l'ontologie GeoSkills, sont ainsi injectés pour peupler l'index Solr.

4.2.5 Comparaison index Solr versus index SearchI2G

Le logiciel Luke permet de consulter l'index afin d'analyser comment il s'est constitué. Une comparaison entre l'index Solr et celui généré par SearchI2G montre que le prototype se comporte à l'identique, voire plus finement comme l'illustre le tableau 4.

	Indexation par SearchI2G	Indexation Prototype Solr
Nombre de documents indexés	1701	1377
Nombre de competencies	454	454
Nombre de competency processes	121	-
Nombre de topics	516	416
Nombre de concrete topics	(15)	(15)
Nombre de abstract topics	(601)	(401)
Nombre de abstract topics with representative	(401)	(401)
Nombre de pure abstract topics	(100)	-
Nombre de level	506	506

Tableau 4 - Comparatif des index SearchI2G versus ontoindex Solr

Le tableau 4 permet de constater que le nombre de capacités (competencies) et de niveau (level) sont identiques pour les 2 index.

La différence de 324 documents par rapport à l'index SearchI2G s'explique au niveau de la non prise en compte des classes de compétences (121 documents competency processes) et des « Pure abstract topics » (100 documents). La différence restante (103 documents) après prise en compte de ces 2 catégories provient d'un bogue dans le programme de SearchI2G qui duplique les 100 documents « pure abstract topics » on leur assignant uniquement le type « abstract topics ».

Les 3 derniers documents manquant correspondent à des documents décrivant le site Web des communautés d'enseignants par langue en Belgique ayant pour URI :

- http://www.inter2geo.eu/2008/ontology/ontology.owl#French_Community ;
- http://www.inter2geo.eu/2008/ontology/ontology.owl#Dutch_Community ;
- http://www.inter2geo.eu/2008/ontology/ontology.owl#German_Community.

La non présence de ces 3 documents dans ontoindex n'est pas problématique car ils ne doivent pas être indexés et ne servent pas pour l'indexation et la recherche de ressources par les notions et

capacités. Une comparaison entre les champs des documents des 2 index est consultable en annexes 4F, 4G et 4H.

4.2.6 Bilan intermédiaire

La comparaison entre l'index du prototype et l'index des capacités et notions d'i2geo.net démontre le succès de l'indexation avec Solr. L'index est même plus cohérent que celui généré par SearchI2G. Le nombre de champs est réduit par rapport à Intergeo (17 contre 83 dans l'index SearchI2G) du fait qu'uniquement les éléments français de l'ontologie sont indexés. Les bogues d'indexation de SearchI2G ne sont pas reproduits.

Le programme ontoIndexation s'exécute en moins de 10 secondes pour analyser l'ontologie, constituer le fichier XML regroupant l'ensemble des documents à indexer et réaliser l'indexation.

L'administration simplifiée par les deux fichiers de configuration XML est un autre atout par rapport à la solution proposée par SearchI2G. Il n'est, en effet, pas nécessaire de retoucher le code source pour changer la pondération ou le type d'indexation d'un des champs de l'index. La pondération correspond pour le moment à l'application des scores par défaut de Solr. Toutefois, la gestion des valeurs doublons au niveau des ancêtres lors de la remontée jusqu'à la racine « Thing » de l'ontologie est prise en compte.

Le score appliqué est celui calculé par Solr. Aucune pondération manuelle n'est apportée sur les champs de l'index.

La diversité des clients Solr pour différents autres langages de programmation assure la compatibilité avec l'architecture de Sésamath.

L'unique inconvénient du prototype et de l'indexation Solr est qu'il pourrait impliquer une réécriture partielle des composants ontoUpdate et CompEd. En effet ces composants utilisent SearhI2G notamment lors de la création d'une notion ou capacité dans CompEd. Le moteur de recherche est intégré dans l'IHM CompEd pour lister et sélectionner les notions et capacités en relation avec celle en cours de création.

Un second prototype permettant de tester la recherche dans l'index ontoindex est maintenant nécessaire.

4.3 Recherche des notions et capacités

Une fois l'indexation réalisée, il convient de développer un client de test accédant à l'index afin de reproduire l'interface des suggestions de notions, capacités et niveaux d'i2geo.net étudiée dans le chapitre 2.

4.3.1 Principes de la recherche des notions et capacités

Le prototype de recherche pour Sésamath est écrit en HTML pour le formulaire de saisie du ou des termes à rechercher. La requête est prise en compte dynamiquement par le code JavaScript « solr.js » pour la partie dialogue AJAX avec le service Web Solr et le traitement du résultat retourné. Le code solr.js fait appel à des méthodes fournies par les API Jquery et Jquery UI [JQUERY 2012] [JQUERY UI 2012].

Il existe de nombreux clients, dans différents langages, communiquant avec le service Web Solr. Le choix de ces langages se justifie par l'architecture Sésamath qui est basée sur ces technologies. La partie HTML et JavaScript du prototype est installée sur un serveur Web Apache au niveau de

l'environnement de développement au LIG. Le service Web Solr est déployé sur un serveur d'application Tomcat comme le précise la figure 60 ci-dessous présentant l'architecture générale.

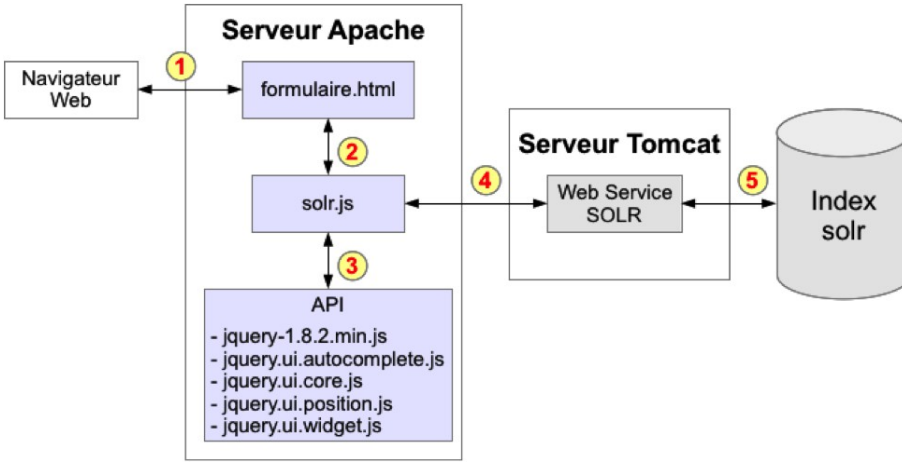


Figure 60 - Architecture pour le processus de recherche

La communication et les requêtes entre le client, le serveur Apache et le serveur Tomcat Solr s'effectuent par l'intermédiaire du protocole HTTP.

4.3.2 Adaptation de l'index ontoindex

La communication entre le client et le serveur Solr étant réalisée par des requêtes HTTP, il est nécessaire d'adapter le serveur Tomcat afin de prendre en compte les caractères spéciaux, tels que les accents, dans les requêtes.

4.3.2.1 Configuration Tomcat

La prise en compte des caractères spéciaux par Tomcat, nécessite la modification du fichier de configuration server.xml en rajoutant une variable URIEnconding="UTF-8" au champ « Connector » du port d'écoute [TOMCAT 2012]. La figure 61 illustre cette configuration.

```
...
<Connector port="8080" protocol="HTTP/1.1"
      connectionTimeout="20000"
      redirectPort="8443"
      URIEncoding="UTF-8"/>
...
```

Figure 61 - Configuration du fichier server.xml sur le port 8080

Un redémarrage du serveur est nécessaire pour la prise en compte de cette modification.

4.3.2.2 Fichier schema.xml

La conception du prototype de recherche des notions et capacités a nécessité quelques ajustements du fichier schema.xml afin de recopier les valeurs des champs defaultCommonName, commonName, unCommonName, rareName et falseFriendName dans un champ appelé name_auto. Il permet de regrouper et rechercher l'ensemble des termes sur un seul champ. L'ajout de ce champ n'a pas d'impact sur le code d'extraction des données de l'ontologie car Solr permet la copie des valeurs d'un

champ vers un autre par l'ajout de la balise « copyField » dans le fichier schema.xml. Cette duplication est illustrée par la figure 62.

```
….
<fields>
….
      <field name="name_auto" type="text_auto" indexed="true" stored="true"
multiValued="true" />
</fields>

      <copyField source="defaultCommonName" dest="name_auto"/>
      <copyField source="commonName" dest="name_auto"/>
      <copyField source="rareName" dest="name_auto"/>
      <copyField source="unCommonName" dest="name_auto"/>
      <copyField source="falseFriendName" dest="name_auto"/>

      <defaultSearchField>name_auto</defaultSearchField>
</schema>
```

Figure 62 - Duplication des champs

Un nouveau fieldType, représenté par la figure 63, a été créé dans schema.xml, nommé text_auto pour configurer l'indexation et la recherche par préfixe sur un champ. Il est utilisé pour typer le champ name_auto et faire dessus des indexations et recherche de type.

```
<schema name="onto" version="1.0">
<types>
      ….
      <fieldType class="solr.TextField" name="text_auto" positionIncrementGap="100">
      <analyzer type="index">
        <tokenizer class="solr.LowerCaseTokenizerFactory"/>
        <filter class="solr.PatternReplaceFilterFactory" pattern="^(\p{Punct}*)(.*?)
(\p{Punct}*)$"
                replacement="$2"/> -->
        <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1"
generateNumberParts="1"
                catenateWords="0" catenateNumbers="0" catenateAll="0"
splitOnCaseChange="1"/>
        <filter class="solr.StopFilterFactory" words="lang/stopwords_fr.txt"
ignoreCase="true"
                enablePositionIncrements="true"/>
        <filter class="solr.LengthFilterFactory" min="3" max="25" />
        <filter class="solr.SnowballPorterFilterFactory" language="French" />
        <filter class="solr.EdgeNGramFilterFactory" minGramSize="2" maxGramSize="25"
side="front"/>
      </analyzer>
      <analyzer type="query">
        <tokenizer class="solr.LowerCaseTokenizerFactory"/>
        <filter class="solr.PatternReplaceFilterFactory" pattern="^(\p{Punct}*)(.*?)
(\p{Punct}*)$"
                replacement="$2"/>
        <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1"
generateNumberParts="1"
                catenateWords="0" catenateNumbers="0" catenateAll="0"
splitOnCaseChange="1"/>
        <filter class="solr.StopFilterFactory" words="lang/stopwords_fr.txt"
ignoreCase="true"/>
        <filter class="solr.LengthFilterFactory" min="1" max="25" />
        <filter class="solr.SnowballPorterFilterFactory" language="French" />
       </analyzer>
```

```
      </fieldType>
</types>
….
```

Figure 63 - Ajout d'un champ dans la partie type

Le mode de découpage en mots choisi pour ce champ est « solr.LowerCaseTokenizerFactory ». Il met tous les termes en minuscule. Plusieurs filtres sont ensuite appliqués afin d'affiner ce découpage :

- solr.PatternReplaceFilterFactory : supprime la ponctuation (entre autres) ;
- solr.WordDelimiterFilterFactory : découpe des mots en sous mot (« Wi-Fi » → « Wi », « Fi ») ;
- solr.StopFilterFactory : ignore les mots communs non significatifs ;
- solr.LengthFilterFactory : élimination des mots dont la longueur ne se situe pas dans une limite minimum et maximum spécifiée ;
- solr.SnowballPorterFilterFactory : génère les formes dérivées de mots à base de modèle (transforme les flexions en leur radical) ;
- solr.EdgeNGramFilterFactory : découpe les mots en différentes chaînes (exemple avec le terme angle : « ang » → « angl » → « angle ») afin de permettre la recherche sur les préfixes.

Toute modification du fichier schema.xml demande une ré-indexation des documents.

4.3.2.3 Fichier solrconfig.xml

Le fichier solrconfig.xml est le fichier permettant de configurer les requêtes HTTP envoyées. Ce paramétrage s'effectue dans la classe « SearchHandler » du champ « requestHandler » comme l'indique la figure 64.

```
...
<requestHandler name="/select" class="solr.SearchHandler">
      <lst name="defaults">
      <str name="echoHandler">false</str>
      <str name="echoParams">explicit</str>
      <int name="rows">10</int>
      <str name="df">name_auto</str>
      </lst>
 </requestHandler>
...
```

Figure 64 - Paramétrage de la recherche

Le tableau « defaults » fournit des valeurs de paramètre par défaut (si le paramètre n'a pas de valeur spécifiée au moment de la demande), comme le champ de recherche par défaut sur l'index (df), le nombre de résultats affichés (rows), etc.

Deux tableaux supplémentaires offre la possibilité d'affiner la recherche :

- « appends » : fournit des valeurs de paramètres utilisées en plus des valeurs spécifiées au moment de la demande ;
- « invariants » : fournit des valeurs de paramètres en dépit de toutes les valeurs fournies au moment de la demande. C'est une façon de laisser l'administrateur Solr verrouiller les options disponibles pour les clients Solr.

L'administrateur Solr peut aussi rajouter dans ces tableaux la pondération des champs pour la recherche, les conditions de « Highlighting », la recherche par facettes, etc. Dans cette version, aucune pondération manuelle des scores calculés par Lucene dans Solr lors de l'indexation et de la recherche n'est effectuée. Les scores des documents correspondent donc à la formule de calcul explicité dans le chapitre 2.

4.3.3 Client de recherche des notions et capacités

Le client de recherche se compose d'un fichier HTML pour l'IHM du formulaire de saisie des termes à rechercher, d'une feuille de style CSS pour sa mise en forme et d'un fichier JavaScript pour la partie traitement des actions « recherche », « traitement » et « affichage » de la réponse renvoyée par le service Web Solr. Le script solr.js, présenté par la figure 65, illustre le fonctionnement d'un client Solr. Ces principes sont réutilisés par la suite au niveau du prototype de recherche des ressources par les notions et capacités.

```
$(function() {
      function log(message) {
       $("<div/>").text(message).prependTo("#log");
       $( "#log" ).scrollTop( 0 );
      }

      $("#autosearch").autocomplete({
       minLength: 1,
       source: function(request, response) {
           $.ajax({
              url: "http://localhost:8080/solr/select",
              //Query
               data: {
                  q: 'name_auto:'+request.term+' OR name_auto:'+request.term+'*',
                  fl: 'defaultCommonName, ontType, urlForNav,score',
                  wt: 'json',
                  omitHeader: 'true',
                  rows: 100,
              },
              dataType: 'jsonp',
              jsonp: 'json.wrf',

               //Parse result
              success: function(data) {
                  response($.map(data.response.docs, function(item) {
                     return {
                         value: item.defaultCommonName,
                         img: item.ontType,
                    url: item.urlForNav,
                           score: item.score
                     }
                  }))
              },
           });
       },
       //Print selected item
       select: function(event, ui) {
           log(ui.item ?
              "Selected: " + ui.item.value :
                  "Nothing selected, input was " + this.value);
           //Clear formula field
           event.preventDefault();
           $(this).val('');
       }
       //Display results
      }).data("autocomplete")._renderItem = function (ul, item) {
       var img = '<img src="img/' + item.img + '.png" width="25" height="25" />';
       var link = 'onclick="window.open(this.href);';
       var urlfornav = '<a href="'+ item.url + '" '+ link + '">
```

```
                    <img src="img/book.gif" width="25" height="25"></a>'
        var score = item.score
        return $("<li></li>")
            .data("item.autocomplete", item)
            .append('<a>' + img + ' ' + item.value + ' ' + urlfornav + ' - score: ' +
score)
            .appendTo(ul);
        };
});
```

Figure 65 - Code source du script solr.js

Le script solr.js se décompose en différentes fonctions. La fonction log gère l'affichage des résultats après sélection d'une des réponses. La communication avec Solr et le traitement des réponses s'effectue au niveau de la partie « autosearch ». La fonction « source » gère les champs nécessaires à la requête HTTP qui est envoyée à Solr :

- q : précise le ou les champs de l'index sur lesquels Solr doit chercher le terme saisit par l'utilisateur ;
- fl : précise les différents champs de l'index à renvoyer ;
- wt : précise le format de la réponse ;
- omitHeader : précise si la réponse doit contenir les entêtes ;
- rows : précise le nombre de résultats à renvoyer.

Les informations nécessaires à l'affichage des résultats sont extraites du tableau JSON retourné par la réponse du service Web Solr, dans la fonction « success ». La fonction « select » gère l'affichage au niveau de la zone de saisie du moteur de recherche. La fonction « renderItem » prend en charge l'organisation des champs pour l'affichage.

4.3.4 Tests de recherche dans ontoindex

Le prototype de recherche des notions et capacités, présenté par la figure 66, est une simple page HTML pour l'IHM du formulaire permettant de tester la recherche des documents contenant les notions et capacité.

Figure 66 - IHM du prototype

La saisie de l'utilisateur est prise en compte par le script « solr.js » qui effectue la requête au serveur Solr via les API Jquery et Jquery UI. Ce script prend aussi en charge le traitement de la réponse et l'affichage du résultat.

La réponse est renvoyée sous forme d'un tableau JSON contenant les valeurs des différents champs renseignée dans la partie « fl » de la requête HTTP. Le script solr.js analyse ensuite ce tableau pour générer l'affichage, illustré par la figure 67, au niveau de l'IHM du moteur de recherche.

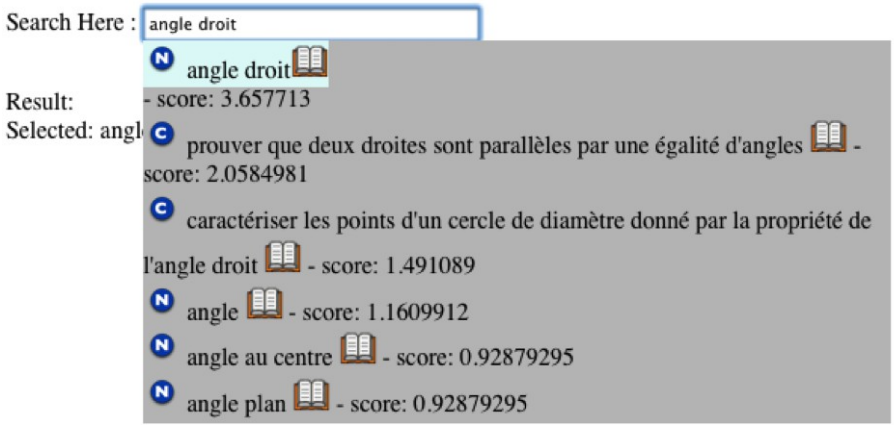


Figure 67 - Affichage du résultat renvoyé par Solr

L'affichage des scores, à l'extrême droite de chaque ligne, servait aux tests et a depuis été supprimé. Ces scores sont présents à titre indicatif afin de visualiser l'évolution du classement ainsi que la pertinence des réponses. La mise en surbrillance de la ligne s'applique lors du parcours de la liste par le passage de la souris sur le champ concerné ou flèches de défilement du clavier. Les liens vers CompEd, représentées par l'icône « livre », sont fonctionnels. Ils renvoient ainsi l'utilisateur vers l'éditeur CompEd.

La pertinence semble correcte car les documents comportant tous les termes recherchés sont bien classés en premier. Le nombre de termes influe ensuite sur ce classement. Plus le champ contient de termes, plus le score décroît. Dans un second temps, on retrouve des suggestions ne contenant qu'un seul des termes recherchés.

4.3.5 Optimisation du prototype de recherche des notions et capacités

La mise en place d'un hôte virtuel Apache faisant office de « proxy inverse » pour accéder au serveur d'application Tomcat permet de configurer un cache serveur contenant les requêtes déjà reçues et les réponses correspondantes.

Ce cache permet donc de limiter les appels à Tomcat et au service Web Solr, comme le montre la figure 68, si le résultat de la requête est déjà présent dans le cache Apache.

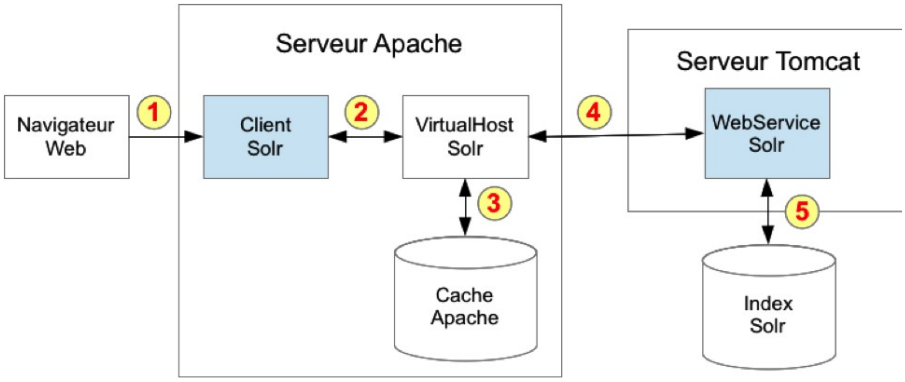


Figure 68 - Architecture avec un cache Apache

La figure 69 présente la configuration de cet hôte virtuel à mettre en œuvre au niveau d'un serveur Apache.

```
<VirtualHost *:8888>
      ServerName localhost
      AllowEncodedSlashes On
      <Proxy *>
    Order deny,allow
    Allow from all
      </Proxy>
      ProxyPass     /solr http://localhost:8080/solr
      ProxyPassReverse /solr http://localhost:8080/solr
  <IfModule mod_cache.c>
    LoadModule disk_cache_module modules/mod_disk_cache.so
    <IfModule mod_disk_cache.c>
       CacheRoot /Applications/MAMP/tmp
       CacheEnable disk /
       CacheDirLevels 5
       CacheDirLength 3
       #Cache size 10 Mo :
       CacheMaxFileSize 10000000
       ExpiresActive On
       ExpiresDefault "access plus 4 weeks"
      </IfModule>
</IfModule>
</VirtualHost>
```

Figure 69 - Configuration du VirtualHost

Il est aussi possible de mettre en place un système de cache sur le navigateur du poste client par l'intermédiaire de cookies stockant les réponses renvoyées par le serveur comme le faisait l'architecture du site i2geo.net. Cette solution n'a pas été retenue car elle implique d'envoyer le cookie intégralement à chaque requête HTTP.

4.3.6 Bilan intermédiaire

Le prototype de recherche des notions et capacités, développé en HTML et JavaScript, démontre qu'il est aisé de communiquer avec le service Web Solr en Java. La mise en place de ce prototype n'est pas complexe en terme de configuration et ne nécessite pas d'importantes ressources machine.

Le service rendu en termes d'affichage des résultats est similaire à ce que propose le moteur de recherche d'Intergeo. Cependant, la mise en cache des résultats au niveau du navigateur client n'est pas implémentée. Il a été validé avec le client Sésamath de privilégier un cache de type serveur au niveau du serveur Web.

La pertinence du classement des résultats est toutefois à étudier plus en détail par des utilisateurs fonctionnels maîtrisant le domaine des mathématiques. Une pondération manuelle pourrait alors être envisagée.

Les principes Intergeo d'indexation et de recherche des notions et capacités sont donc fidèlement reproduits. Un troisième prototype peut maintenant être implémenté pour indexer les ressources Sésamath et les lier aux notions et capacités indexées dans ontoindex.

4.4 Indexation des ressources

Sésamath développe, en parallèle du prototype, une bibliothèque nommée « Bibli » pour référencer l'ensemble des ressources dans une base de données MySQL.

4.4.1 Principe d'indexation des ressources Sésamath

L'objectif de cette étape est de réaliser une indexation des ressources en 2 temps. Le principe est d'indexer les ressources de la base de données de Bibli et ensuite de les lier aux capacités et notions précédemment indexées dans l'index Solr « ontoindex ».

4.4.2 Configuration Solr

Dans le prototype d'indexation des notions et capacités de l'ontologie GeoSkills, le service Web Solr avait été configuré avec un seul index. L'indexation des ressources Sésamath nécessite de faire évoluer la configuration Solr pour prendre en compte les champs des documents correspondant aux ressources.

4.4.2.1 Choix de configuration

Le service Web Solr permet de gérer la double indexation nécessaire à notre application de plusieurs façons :

- aplanissement des données dans un index unique avec des champs communs et d'autres spécifiques aux types de documents ;
- index multiples sur une instance Solr unique (multicore) ;
- multiples ports Solr avec plusieurs serveurs d'applications ;
- multiples instances Solr sur un serveur d'applications.

L'aplanissement des données consiste à indexer différents types de documents dans un même index. Cette méthode s'appuie sur des champs génériques aux documents et d'autres plus spécifiques au type du document. La méthode multicore utilise un seul service Web Solr dans lequel sont configurés plusieurs index en fonction du type de document indexé. Chaque index est isolé et indépendant en terme de configuration. Une configuration multiple port ou instances nécessite le déploiement du service Web Solr sur plusieurs serveurs d'application ou en plusieurs instances.

Le choix d'une configuration multicore a été retenu car elle permet un bon compromis par rapport aux 3 autres possibilités. En effet les options multiple ports et multiple instances sont plus consommatrices en ressources serveurs, alors que l'aplanissement des données sur un index unique est relativement contraignant dans la gestion des documents et leur cycle de vie.

4.4.2.2 Configuration du multicore Solr

La mise en place du mode multicore se configure par l'intermédiaire d'un fichier XML déposé à la racine du répertoire hébergeant les fichiers de configuration Solr. Ce fichier intitulé solr.xml précise le chemin vers les différents index comme le montre la figure 70.

```
<?xml version="1.0" encoding="UTF-8"?>
<solr persistent="false">
      <cores adminPath="/admin/cores">
            <core name="ontoindex" instanceDir="ontoindex">
                  <property name="dataDir" value="/data/ontoindex" />
            </core>
            <core name="sesaindex" instanceDir="sesaindex">
                  <property name="dataDir" value="/data/sesaindex" />
            </core>
      </cores>
</solr>
```

Figure 70 - Configuration Solr en multicore

Le core name « ontoindex » correspond à l'index des notions et capacités de l'ontologie GeoSkills et le core name « sesaindex » définit l'index des ressources Sésamath. Chaque core est représenté par un répertoire et est constitué des fichiers de configuration solrconfig.xml, schema.xml et des fichiers textes (stopwords.txt, synonyms.txt, etc.) nécessaires à Solr pour la gestion des index. Les index sont aussi placés dans chaque core name mais peuvent être déplacés sur un système de fichiers différent.

Le passage en multicore implique le changement des URL au niveau des clients de recherche. Lors du prototype de recherche sur l'index de l'ontologie, le client appelait le service Web Solr par l'adresse « http://localhost:8080/solr/select ». Le core name est maintenant présent dans l'URL afin de différencier l'index des notions et capacités de celui des ressources :

- index des capacités et notions : http://localhost:8080/solr/ontoindex/select ;
- index des ressources : http://localhost:8080/solr/sesaindex/select.

Les fichiers solrconfig.xml et schema.xml de l'index ontoindex restent inchangés. Il faut créer ceux de sesaindex pour effectuer l'indexation de bibliothèque des ressources Sésamath.

4.4.2.3 Configuration de l'index sesaindex

Pour indexer la bibliothèque des ressources Sésamath, la définition du fichier solrconfig.xml est assez proche de celui utilisé pour ontoindex. Une section permettant l'accès à une base de données est rajoutée. La définition de schema.xml est quant à elle plus spécifique à l'index. Un nouveau fichier intitulé « db-data-config.xml » permet la connexion à la base de données afin d'extraire les données à indexer.

4.4.2.3.1 Fichier solrconfig.xml

Une section « dataimport », présentée par la figure 71 a été rajoutée au fichier solrconfig.xml contenant la plupart des paramètres de configuration spécifiques au service Web Solr.

```
….
<requestHandler name="/dataimport"
       class="org.apache.solr.handler.dataimport.DataImportHandler">
      <lst name="defaults">
      <str name="config">db-data-config.xml</str>
      </lst>
</requestHandler>
….
```

Figure 71 - Section dataimport

Cette section permet au service Web Solr de se connecter à une base de données pour en extraire les champs des tables à indexer. Il est nécessaire, pour ce faire, de rajouter la librairie « apache-solr-dataimporthandler-3.6.1.jar » au niveau du répertoire « WEB-INF/lib » de Solr, déployé dans le serveur d'application Tomcat. Cette librairie est fournie par le livrable Solr mais n'est pas inclue par défaut dans l'application.

La balise « config » indique le nom du fichier XML chargé du paramétrage de la connexion à la base et de l'extraction des données. Les autres sections de ce fichier de configuration peuvent être laissées en l'état.

4.4.2.3.2 Fichier db-data-config.xml

Le fichier db-data-config.xml est le fichier permettant l'indexation d'une base de données. Ce fichier se décompose en deux sections, comme le montre la figure 72.

```
<?xml version="1.0" encoding="UTF-8"?>

<dataConfig>
  <dataSource type="JdbcDataSource"
    driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:8889/Compmep"
    user="Compmep"
    password="Compmep"/>

  <document>
  <entity name="id" transformer="RegexTransformer" query="SELECT r.id, r.titre,
r.resume, r.description,
  tt.id AS ttid, GROUP_CONCAT(DISTINCT ts.nom) AS TypeSesa, GROUP_CONCAT(DISTINCT
n.nom) AS niveau
FROM Ressource r
INNER JOIN TypeTech      tt ON r.typeTech_id  = tt.id
 LEFT JOIN ressource_niveau  rn ON r.id      = rn.ressource_id
 LEFT JOIN Niveau       n  ON rn.niveau_id  = n.id
 LEFT JOIN ressource_typesesa rts ON r.id      = rts.ressource_id
 LEFT JOIN TypeSesa      ts ON rts.typesesa_id = ts.id
WHERE r.restriction = 0
GROUP BY r.id
ORDER BY r.id">
      <field column="id" name="id" />
      <field column="titre" name="titre" />
      <field column="resume" name="resume" />
      <field column="description" name="description" />
      <field column="niveau" splitBy="," sourceColName="niveau" />
      <field column="ttid" name="typeTechId" />
      <!-- <field column="ts" name="typeSesa" /> -->
      <field column="ts" splitBy="," sourceColName="typeSesa" />
    </entity>
  </document>
</dataConfig>
```

Figure 72 - Configuration du fichier db-data-config.xml

La section « dataSource » permet de renseigner les informations sur la base de données. Elle précise le SGBD, l'URL d'accès à la base de données ainsi que l'utilisateur et le mot de passe permettant la connexion. Il est nécessaire de copier un driver JDBC du SGBD dans le répertoire « lib » de l'arborescence de la configuration.

La balise « entity » de la section « document » contient la requête SQL permettant l'extraction des données de la base. Les balises « field » permettent au service Web Solr de faire la relation entre le nom des colonnes des tables et le nom des champs de l'index définis dans le fichier de configuration schema.xml.

4.4.2.3.3 Fichier schema.xml

Un type de champ « int » a été rajouté, au fichier schema.xml, pour la prise en compte des identifiants des ressources dans la base de données Sésamath. La section fields a été totalement réécrite pour prendre en compte les champs de la bibliothèque Sésamath, définis dans le fichier db-data-config.xml, et ceux spécifiant les capacités et notions rattachées à la ressource.

Une ressource est définie par les champs listés dans le tableau 5.

Champs	Indexed	Stored	Multivalued	Observations
id	true	true	false	Identifiant de la ressource.
titre	true	true	false	Titre de la ressource.
resume	true	true	false	Résumé de la ressource.
description	true	true	false	Description de la ressource.
typeTechId	false	false	false	ID du type technique de la ressource. Champ non utilisé pour le moment.
typeSesa	true	true	true	Nom du type de la ressource.
niveau	true	true	True	Niveau(x) de la ressource.
capnotion	true	true	true	ID de(s) capacité(s)/notion(s) liée(s) à la ressource.
ancestor	true	true	true	ID des ancêtres des capacités/notions liées à la ressource. Valeurs doublonnées à supprimer.
competency	false	true	true	DefaultCommonName des capacités liées à la ressource.
topic	false	true	true	DefaultCommonName des notions liées à la ressource.

Tableau 5 - Liste des champs définissant une ressource

Les 7 premiers champs concernent la partie spécifique à l'indexation des ressources Sésamath à partir de la base de données. Les quatre derniers sont renseignés lors de l'indexation par les notions et capacités.

4.4.2.3.4 Fichiers TXT

Pour la configuration de l'index sesaindex, seul le fichier des stopwords est utilisé. Il est identique à celui mis en œuvre pour l'index ontoindex (chapitre 4.2.2.4)

4.4.3 Indexation de la base de données

Toutes les interactions avec le service Web Solr s'effectuant par requête HTTP, il est possible de lancer l'indexation de la base de données à partir d'un navigateur Web. La figure 73 montre l'URL d'appel au service Web en précisant qu'il s'agit d'un import de données complet.

```
http://localhost:8080/solr/sesaindex/dataimport?command=full-import
```

Figure 73 - URL d'import des données

L'import peut aussi être lancé en ligne de commande par le biais de la commande UNIX CURL, comme le montre la figure 74.

```
curl http://localhost:8080/solr/sesaindex/dataimport?command=full-import
```

Figure 74 - Import via cURL

L'outil Luke permet ensuite au programmeur d'analyser l'index afin de voir le nombre de documents avec les champs et les valeurs indexés. La figure 75 illustre cette vérification.

Field	IdfpTSVopNLB#	Norm	Value
description	IdfpTS---N---	0.1875	Comment utiliser le crayon de Mathenpoche en 5 étapes.
id	IdfpTS---N--#	1.0	1
nom	IdfpTS---N---	0.25	exercice mep modele 1
titre	IdfpTS---N---	0.4375	Le crayon

Figure 75 - Vérification de l'indexation d'une ressource

Les champs spécifiques aux notions et capacités ne sont pas présents après cette étape. Une indexation manuelle est ensuite nécessaire.

4.4.4 Indexation des ressources par les notions et capacités

En l'absence d'une IHM Sésamath permettant la gestion des ressources (« frontend de gestion »), un prototype d'indexation des ressources par les capacités et notions a été développé afin de tester la mise à jour de l'index sesaindex. Ce prototype « index » est développé avec les langages HTML, JavaScript et PHP sur le même principe que le prototype de recherche des notions et capacités.

Le processus est toutefois plus complexe car les deux index sont interrogés successivement. La figure 76 détaille les différentes interactions.

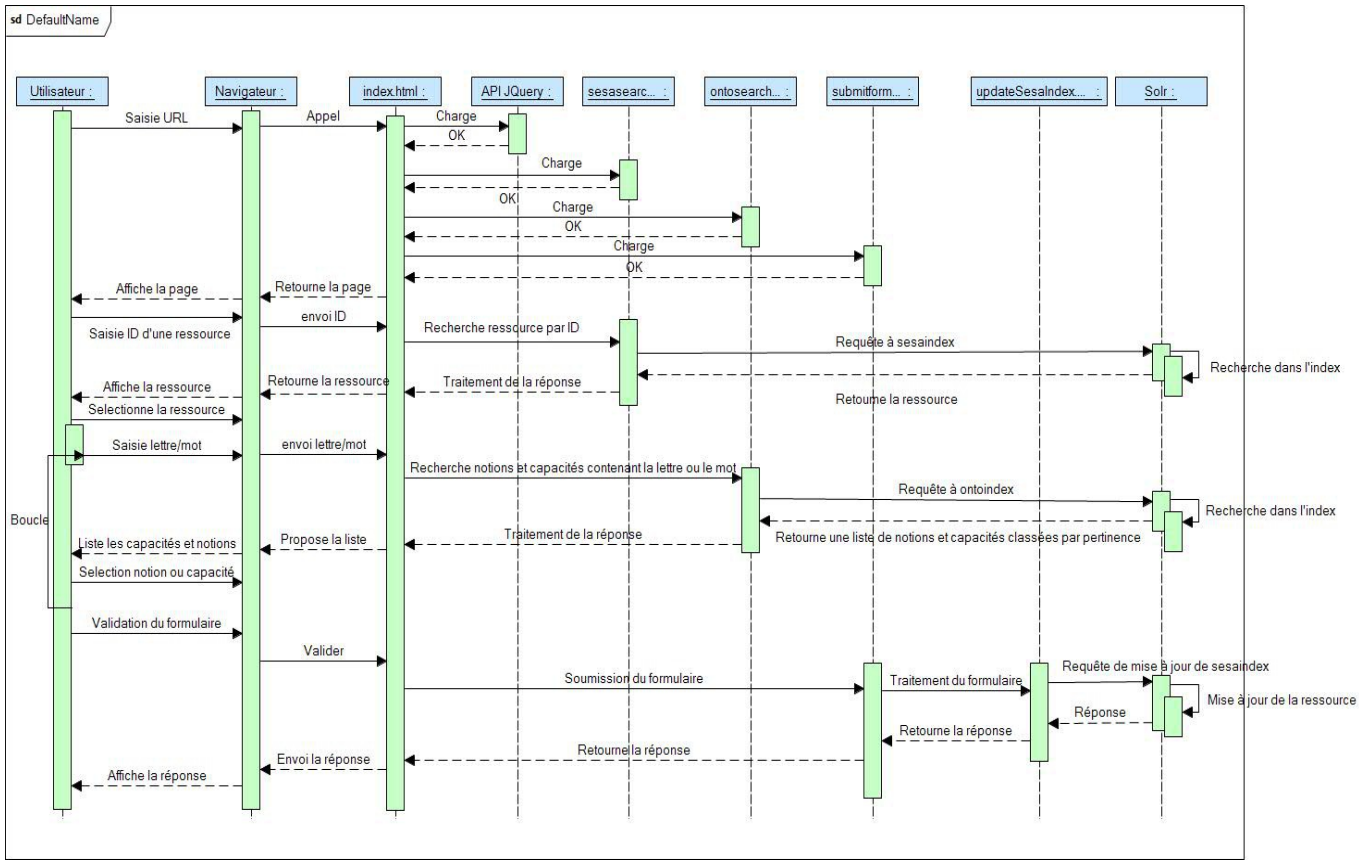


Figure 76 - Processus d'indexation par les notions et capacités

L'indexation d'une ressource par les notions et capacités demande donc au moins trois appels au service Web Solr. Un première requête réalise la recherche de la ressource par son identifiant. Une seconde requête permet de recherche la notion ou capacité à lier. Enfin une dernière requête permet de mettre à jour la ressource pour ajouter la ou les notions ou capacités à lier à la ressource.

4.4.4.1 Recherche d'une ressource

A partir de chaque caractère ou chaîne de caractères saisie par l'utilisateur, une requête est envoyée au serveur Tomcat hébergeant le service Web Solr. Une recherche se compose de l'URL du serveur avec l'accès vers le service Web Solr (/solr/sesaindex/) et de plusieurs paramètres :

- l'indication que la requête est une recherche (select) ;
- la requête sur le champ avec la valeur de la saisie de l'utilisateur (q=id:1) ;
- la liste des valeurs des champs à retourner dans la réponse (fl=id, titre, description, nom) ;
- le format de la réponse (wt=json) ;

- l'indication de ne pas renvoyer les entêtes dans le tableau JSON (omitHeader=true).

La figure 77 présente un exemple de requête pouvant être soumise au service Web Solr sur l'index sesaindex.

http://localhost:8888/solr/sesaindex/select?q=id:1&fl=id,titre,description,nom&wt=json&omitHeader=true

Figure 77 - Exemple d'une requête de recherche d'une ressource

Le service Web Solr effectue une recherche sur le champ ID des documents de l'index sesaindex et retourne les champs composant le document de la ressource indexée. La figure 78 illustre ce processus.

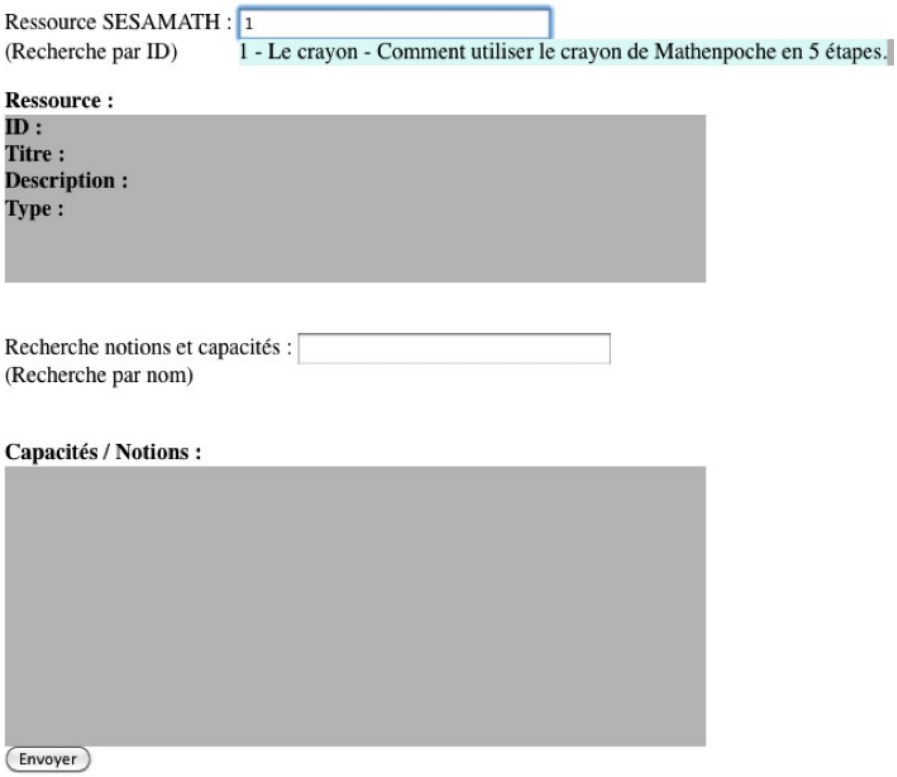


Figure 78 - Recherche de ressources par l'identifiant

La réponse renvoyée, par le service Web Solr, peut être sélectionnée par la souris ou au clavier. Une seconde recherche de ressource effectuée avant soumission de la mise à jour annule et remplace la première recherche.

Le script sesasearch.js, consultable en annexe 4I, gère entièrement l'étape de recherche de ressource. Ce script récupère dynamiquement l'URL du serveur Apache qui transfère la requête au service Web Solr déployé sur le serveur d'applications Tomcat. Le script traite la réponse JSON renvoyée par Solr pour formater la suggestion dans le moteur de recherche ainsi que l'affichage dans la zone « ressource ». Sesasearch.js utilise des méthodes des API JavaScript Jquery et Jquery UI.

Dans cet exemple, il est choisi d'indexer, par les capacités et notions, la ressource ayant pour titre « Le crayon » et descriptif « Comment utiliser le crayon Mathenpoche en 5 étapes ». La seconde partie de l'IHM correspond à la recherche des notions et capacités à rattacher à la ressource avant de mettre à jour l'index.

4.4.4.2 Recherche de notions et capacités

La seconde zone de saisie, illustrée par la figure 79, permet d'effectuer une recherche des notions et capacités dans l'index ontoindex. Cette zone correspond à l'intégration de l'interface de recherche des notions et capacités évoqué dans le chapitre 4.3.

Figure 79 - Recherche de notions et capacités

La réponse renvoyée peut être sélectionnée par la souris ou au clavier. Une seconde recherche de ressource effectuée avant soumission de la mise à jour s'ajoute à la première recherche. Le script ontosearch.js, consultable en annexe 4J, gère entièrement l'étape de recherche des notions et capacités. Ce script récupère dynamiquement l'URL du serveur Apache qui transfert la requête au service Web Solr déployé sur le serveur d'applications Tomcat. Le script traite la réponse JSON renvoyée par Solr pour formater la réponse dans le moteur de recherche ainsi que l'affichage dans la zone « Capacités / Notions ». Ontosearch.js utilise des méthodes des API JavaScript Jquery et Jquery UI.

Dans l'exemple de la figure 80 la ressource est indexée par les notions et capacités suivantes :

- envisager la variation du volume d'un prisme droit en fonction de sa hauteur ou de l'aire de sa base (competency) ;
- angle droit (topic) ;
- angle (topic).

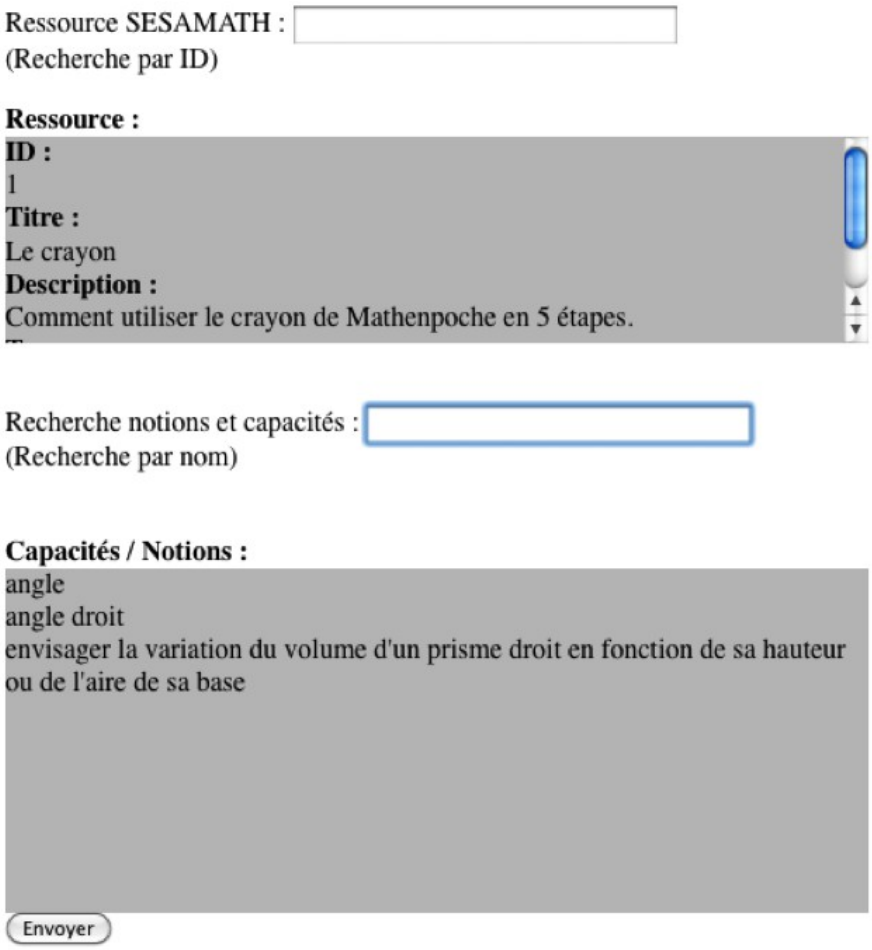


Figure 80 - Ajout des capacités et notions à une ressource

Il est possible de supprimer une notion ou capacité initialement sélectionnée en cliquant sur son nom. Les différents champs et valeurs, du document de la ressource à indexer, sont stockés dans un tableau JSON. La mise à jour de l'index est ensuite exécutée en validant le formulaire par l'intermédiaire du bouton « Envoyer ».

4.4.4.3 Mise à jour de l'index des ressources

La mise à jour d'une ressource dans l'index sesaindex est réalisée par les scripts submitform.js (annexe 4K) et updateSesaIndex.php (annexe 4L). Le script submitform.js récupère dynamiquement l'URL du serveur Apache et ajoute les champs de la ressource au tableau JSON généré par le script ontosearch.js. Ce tableau est ensuite envoyé au script updateSesaIndex.php qui envoie la requête de mise à jour au service Web Solr. La mise à jour de l'index est réalisée par une requête HTTP du type « POST ». Cette opération est réalisée grâce à la bibliothèque CURL fournie avec PHP [PHP 2012].

La mise à jour supprime le document initial pour créer un nouveau document. Cette suppression laisse néanmoins un document vide dans l'index. L'optimisation de l'index permet ensuite de le supprimer totalement.

L'outil Luke permet ensuite au programmeur de vérifier le résultat du programme en visualisant le nombre de documents indexés et les valeurs des champs de chaque document. La figure 81 illustre cette vérification.

Field	IdfpTSVopNLB#	Norm	Value
ancestor	Idfp-S---N---	0.1875	Identify_about_measurement
ancestor	Idfp-S---N---	0.1875	TransversalCompetency
ancestor	Idfp-S---N---	0.1875	NamableBit
ancestor	Idfp-S---N---	0.1875	Recognise_or_Identify
ancestor	Idfp-S---N---	0.1875	Competency
ancestor	Idfp-S---N---	0.1875	Identify
ancestor	Idfp-S---N---	0.1875	Thing
ancestor	Idfp-S---N---	0.1875	Right-Prism
ancestor	Idfp-S---N---	0.1875	Formula_for_volume
ancestor	Idfp-S---N---	0.1875	Variation
ancestor	Idfp-S---N---	0.1875	Angle_fig
ancestor	Idfp-S---N---	0.1875	GeometricObject
ancestor	Idfp-S---N---	0.1875	Topic
ancestor	Idfp-S---N---	0.1875	NamableBit
ancestor	Idfp-S---N---	0.1875	Thing
ancestor	Idfp-S---N---	0.1875	Right_angle
ancestor	Idfp-S---N---	0.1875	GeometricObject
ancestor	Idfp-S---N---	0.1875	Topic
ancestor	Idfp-S---N---	0.1875	NamableBit
ancestor	Idfp-S---N---	0.1875	Angle_fig
ancestor	Idfp-S---N---	0.1875	Thing
capnotion	Idfp-S---N---	0.5	Identify_variation_of_volume_of_right_prism_in_function_of_height_or_base
capnotion	Idfp-S---N---	0.5	Angle_fig
capnotion	Idfp-S---N---	0.5	Right_angle
competency	IdfpTS---N---	0.15625	envisager la variation du volume d'un prisme droit en fonction de sa hauteur ou de l'aire de sa base
description	IdfpTS---N---	0.1875	Comment utiliser le crayon de Mathenpoche en 5 étapes.
id	IdfpTS---N--#	1.0	1
nom	IdfpTS---N---	0.25	exercice mep modele 1
titre	IdfpTS---N---	0.4375	Le crayon
topic	IdfpTS---N---	0.25	angle
topic	IdfpTS---N---	0.25	angle droit

Figure 81 - Vérification de l'indexation d'une ressource par les notions et capacités

Les champs initiaux de la ressource sont toujours présents et les quatre champs spécifiques aux capacités et notions ont bien été ajoutés. Ces champs sont utilisés pour la recherche de ressources par les notions et capacités.

4.4.5 Bilan intermédiaire

L'indexation des ressources est donc fonctionnelle et permet de reproduire les principes d'indexation mis en œuvre dans le système d'Intergeo. L'utilisation de clients légers de type Web correspond aux technologies maîtrisées par Sésamath.

Le score appliqué est celui calculé par Solr. Aucune pondération manuelle n'est apportée sur les champs de l'index.

Une dernière interface Web permettant de rechercher les ressources indexées par les notions et capacités finalise la mise en place des principes implémentés dans Intergeo.

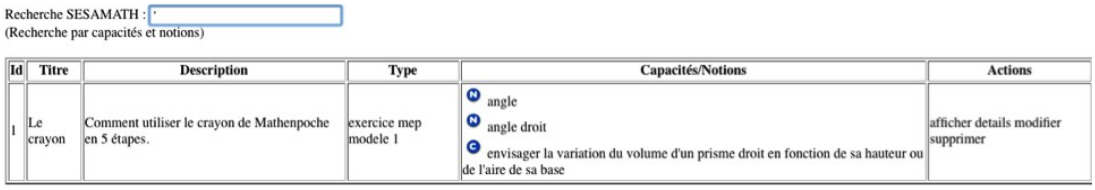
4.5 Recherche de ressources par les notions et capacités

Un prototype « sesaSearch » a été développé pour offrir une recherche de ressources par les notions et capacités pour les éléments de la bibliothèque Sésamath préalablement indexés.

Il a été développé avec les langages HTML et JavaScript. Il utilise aussi les API Jquery et Jquery UI. Le script search.js, consultable en annexe 4M, gère entièrement la recherche et l'affichage des suggestions et résultats. Ce script récupère dynamiquement l'URL du serveur Apache qui transfert les requêtes au service Web Solr déployé sur le serveur d'applications Tomcat. La suggestion renvoyée peut être sélectionnée par la souris ou au clavier.

Le script search.js interroge dans un premier temps l'index ontoindex afin d'afficher des notions et capacités en fonction des caractères et mots saisis dans le moteur de recherche. Après la sélection d'une notion ou capacité par l'utilisateur, un second appel sur l'index sesaindex, permet de collecter l'ensemble des ressources indexées par cette notion. Cette recherche est réalisée sur les champs ancestor et capnotion. La requête de recherche effectue un « OU » entre les 2 champs

Le script search.js analyse la réponse de Solr et formate un tableau HTML des résultats à afficher dans l'IHM. La figure 82 illustre ce tableau des résultats.

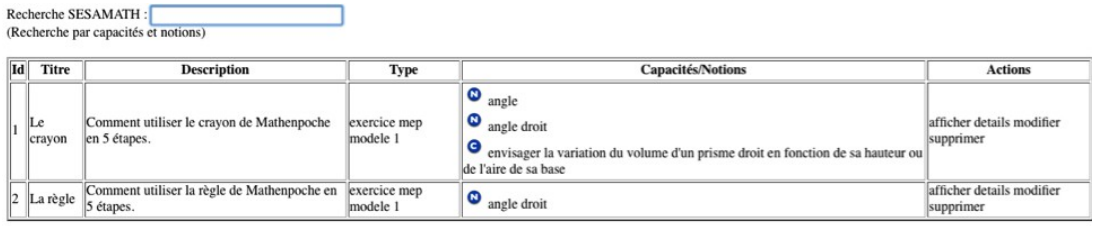

Recherche SESAMATH :
(Recherche par capacités et notions)

Id	Titre	Description	Type	Capacités/Notions	Actions
1	Le crayon	Comment utiliser le crayon de Mathenpoche en 5 étapes.	exercice mep modele 1	N angle N angle droit C envisager la variation du volume d'un prisme droit en fonction de sa hauteur ou de l'aire de sa base	afficher details modifier supprimer

Figure 82 - Résultat de la recherche

Ce tableau reprend l'affichage du prototype actuel de la bibliothèque de ressources de Sésamath. Seule la colonne « Capacités/Notions » a été rajoutée.

L'indexation d'une seconde ressource avec uniquement la notion « angle droit » permet de tester si la recherche est bien faite sur les ancêtres (ici angle) des notions et capacités liées à la ressource.


Recherche SESAMATH :
(Recherche par capacités et notions)

Id	Titre	Description	Type	Capacités/Notions	Actions
1	Le crayon	Comment utiliser le crayon de Mathenpoche en 5 étapes.	exercice mep modele 1	N angle N angle droit C envisager la variation du volume d'un prisme droit en fonction de sa hauteur ou de l'aire de sa base	afficher details modifier supprimer
2	La règle	Comment utiliser la règle de Mathenpoche en 5 étapes.	exercice mep modele 1	N angle droit	afficher details modifier supprimer

Figure 83 - Résultat d'une recherche sur les champs ancestor et capnotion

La figure 83 démontre que la recherche est bien réalisée sur les champs capnotion et ancestor. La ressource intitulée « La règle » est bien retrouvée à partir de la notion angle qui est l'ancêtre de la notion « angle droit ».

4.6 Bilan de la mise en œuvre des principes Intergeo avec Solr

L'objectif de reproduire les principes d'indexation et de recherche par les capacités et notions est totalement atteint. La pertinence de l'indexation a même été améliorée. Le service Web Solr simplifie l'infrastructure faisant abstraction de l'implémentation Lucene. La configuration Solr par fichier XML et la documentation fournie sont aussi des atouts non négligeables pour l'appropriation de la solution.

Les scores par défaut calculés par Solr semblent pertinents. Aucune pondération manuelle n'est apportée sur les champs des index.

Les prototypes prouvent le bon fonctionnement de Solr en mode multi-index ainsi que la recherche et la mise à jour dans les index à partir de requêtes Web. Ils sont portables sur différentes plates-

formes dans une architecture de type « clients serveurs N-tiers » grâce au serveur Apache effectuant l'interface avec les serveurs d'application Tomcat.

Afin de simplifier le déploiement des prototypes, il n'y a aucune URL en entier dans le code source, uniquement des parties génériques qui sont assemblées avec l'URL spécifique du serveur. Une intégration dans la bibliothèque Sésamath, en cours de développement, reste à tester.

Il convient maintenant d'intégrer les différents prototypes dans la bibliothèque des ressources que Sésamath développe en parallèle de nos développements.

5. Intégration au système de recherche de Sésamath

L'objectif de cette partie est de présenter l'intégration du système d'indexation des ressources par les notions et capacités ainsi que les clients de tests développés, au cours du chapitre 4, dans l'interface Web de la bibliothèque de Sésamath. La figure 84 illustre l'architecture finale du projet à implémenter.

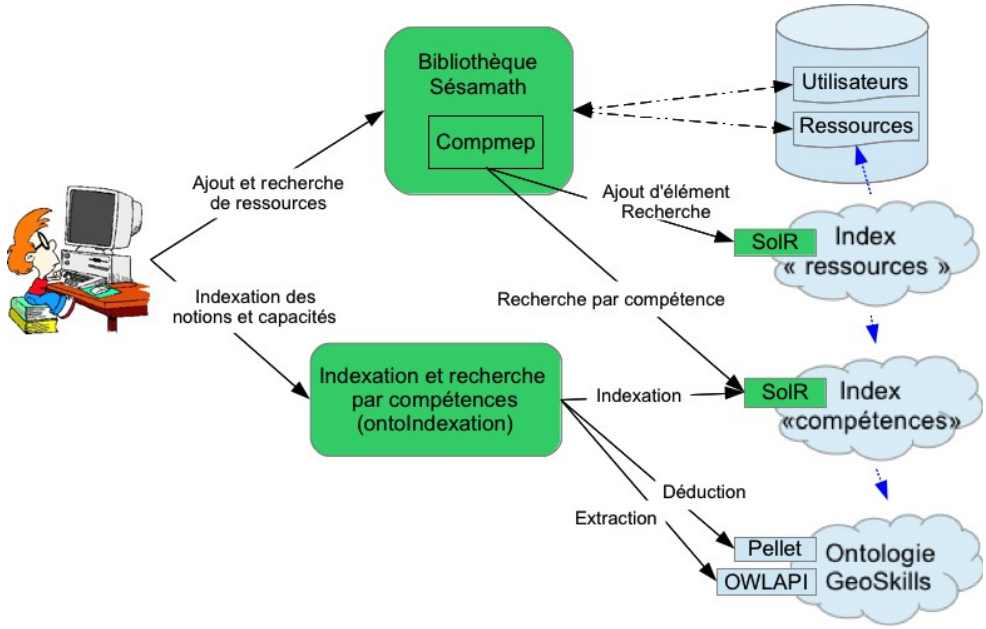


Figure 84 - Intégration de Compmep à la bibliothèque Sésamath

L'intégration et les évolutions du système ont été réalisées par itérations successives faisant suite à des réunions de travail avec la maîtrise d'ouvrage de Sésamath afin de déterminer les nouvelles fonctionnalités à mettre en œuvre et de valider celles précédemment implémentées.

5.1 Présentation de la bibliothèque

Le scénario initialement prévu pour intégrer le moteur de recherche dans l'outil LaboMep a été abandonné car Sésamath a débuté le développement d'un site faisant office de bibliothèque contenant l'ensemble des ressources. Cette bibliothèque a pour objectif de centraliser le référencement de ses ressources éparpillées sur différents sites et dans différents formats. Le moteur d'indexation et de recherche doit donc s'intégrer dans ce nouveau site en cours de développement.

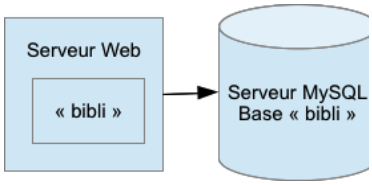


Figure 85 - Architecture Bibli

L'architecture du projet est représentée par la figure 85. La bibliothèque, nommée Bibli, est développée avec les langages Web PHP, JavaScript, HTML et utilise le cadriciel Symfony2 pour le squelette du site. La figure 86 présente cette application.

Figure 86 - Page d'accueil de la bibliothèque Bibli

5.1.1 Cadriciel Symfony2

L'objectif principal d'un framework (cadriciel) est d'améliorer la productivité des développeurs qui l'utilisent. Symfony2 est donc un framework PHP français très populaire. C'est un des cadriciel les plus utilisés dans le monde, notamment dans les entreprises. Il est édité par la société SensioLabs, est Open Source et utilise l'architecture MVC [SYMFONY2 2012].

L'arborescence d'une application Symfony2 est plutôt flexible mais celle de la distribution Standard Edition reflète la structure typique et recommandée d'une application Symfony2 :

- « app » : contient la configuration de l'application. La classe AppKernel est le point d'entrée principal de la configuration de l'application ;
- « src » : contient le code source PHP et autres du projet. Dans ce répertoire, le code est organisé en bundles (briques de l'application) ;
- « vendor » : contient les bibliothèques tierces dont celles de Symfony ;
- « web » : correspond au répertoire Web racine. Tous les fichiers statiques et publics comme les images, les feuilles de styles et les fichiers JavaScript se retrouvent dans ce répertoire après déploiement de l'application.

Bibli utilise par défaut les bibliothèques externes suivantes :

- Assetic : bibliothèque permettant de gérer les « assets » JavaScript et CSS ;
- Composer : bibliothèque de gestion de dépendances pour PHP permettant de faire les mises à jour des composants du framework ;
- Doctrine : ORM permettant de mettre en place une couche d'abstraction à la base de données pour PHP ;
- JMS : bibliothèque permettant notamment la gestion de la sécurité au sein de l'application ;
- Monolog : bibliothèque servant à écrire des logs ;
- Swiftmailer : bibliothèque permettant l'envoi de courriel ;
- Twig : moteur de templates pour la couche de présentation de l'application.

5.1.2 Moteur de templates TWIG

Twig est un moteur de templates PHP permettant de séparer la couche de présentation (Vue du MVC) des applications Web, tout en gardant flexibilité, rapidité et facilités au développement. Ce moteur de templates est directement intégré dans Symfony2 [TWIG 2012].

Twig permet principalement de gérer de l'héritage entre templates et mise en page, de séparer les couches de présentation et couches métiers. Le moteur dispose d'un langage avec 3 types de balises :

- {{ maVar }} : les doubles accolades permettent d'afficher une valeur dans la page HTML ;
- {% for page in arrPages %} : les accolades pourcentage permettent d'exécuter une fonction dans une condition ou une boucle, de définir un bloc comme par exemple pour la prise en compte du code JavaScript, etc. ;
- {# commentaires #} : la dernière syntaxe permet d'insérer des commentaires dans le code source de la page HTML.

5.1.3 Librairies JavaScript

Plusieurs librairies JavaScript sont utilisées dans l'application Bibli pour exécuter des opérations depuis le navigateur Web de l'utilisateur :

- head.js : permet de charger tous les autres fichiers de script d'un site sans bloquer le chargement de la page. De plus, il permet d'optimiser le temps de chargement des pages Web ;
- jquery.js : librairie portant sur l'interaction entre JavaScript (comprenant AJAX) et HTML, et ayant pour but de simplifier des commandes communes de JavaScript ;
- json2.js : permet d'analyser une chaîne en JSON ;
- jstree.js : librairie permettant la création et la manipulation d'arbres ;
- underscore.js : apporte un ensemble d'une soixantaine de fonctions pour JavaScript très utiles et efficaces permettant de simplifier les développements ;
- swfobject.js : librairie permettant d'afficher du Flash dans une page HTML ;

L'application Bibli utilise aussi plusieurs codes JavaScript développés directement par Sésamath.

5.2 Intégration à Bibli

L'intégration des clients de test du chapitre 4 dans le code source de la bibliothèque a été réalisée en parallèle des développements spécifiques à cette dernière. L'infrastructure a tout d'abord été installée sur la machine virtuelle hébergée par le LIG, où les composants d'Intergeo avaient été installés pour réaliser les démonstrations. Cette machine a été utilisée lors des premières démonstrations du système à la maîtrise d'ouvrage Sésamath.

Un serveur d'applications Tomcat a ensuite été rajouté dans l'infrastructure de développement de Sésamath pour déployer le service Web Solr et réaliser l'indexation de l'ontologie GeoSkills et de la base de données Bibli. La figure 87 présente cette nouvelle architecture hébergée par Sésamath.

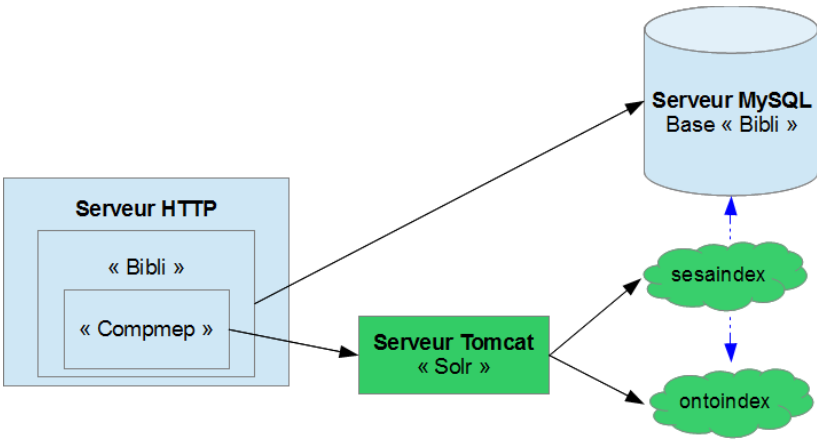


Figure 87 - Nouvelle architecture du projet Bibli

Les JavaScript, images et CSS nécessaire au fonctionnement de Compmep ont été installés dans le répertoire Symfony « src/Sesamath/BibliBundle/Resources/public/Compmep/ ». La figure 88 montre l'arborescence complète du répertoire contenant l'ensemble du code source de Bibli.

```
bibli/src/Sesamath/BibliBundle
        |--- Tests/Controller
        |--- DependencyInjection
        |--- Resources
            |--- translations
            |--- doc
            |--- public
                |--- Compmep
                |--- css
                |--- html
                |--- images
                |--- js
                |--- xml
            |--- phplibs
            |--- views
                |--- Default
                |--- Niveau
                |--- SolrRessource
                |--- TypeTech
                |--- Form
                |--- Ressource
                |--- Tree
            |--- config
        |--- Entity
        |--- Form
        |--- Services
        |--- Controller
        |--- Command
```

Figure 88 - Arborescence du répertoire « src » de Bibli

La partie présentation HTML a été intégrée dans les vues TWIG de la bibliothèque.

5.2.1 Vue layout.html.twig

La vue « layout.html.twig », représentant la page d'accueil du site illustrée par la figure 89, a été modifiée pour prendre en compte la zone de saisie des notions et des capacités. Cette page HTML charge aussi les différents JavaScript développés pour les différents prototypes présentés dans le chapitre 4 de ce mémoire

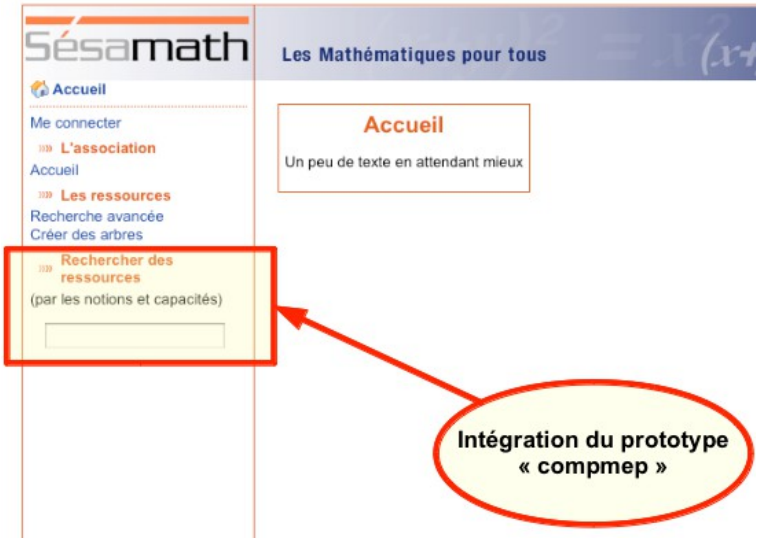


Figure 89 - Intégration du moteur de recherche dans Bibli

Sous la zone de saisie, un formulaire HTML caché, du type <input type="hidden">, permet de collecter l'URI, le nom par défaut et le type de la notion ou capacité sélectionnée suite à la recherche dans l'index ontoindex. Ces valeurs sont écrites dans le formulaire sous la forme d'un tableau JSON transformé en chaîne de caractères. Cette transformation est réalisée grâce à la méthode JSON.stringify de l'API Jquery afin de pouvoir transférer l'objet JSON au serveur.

Le formulaire est ensuite dynamiquement soumis par le JavaScript à un contrôleur pour rechercher les ressources liées à la capacité ou notion choisie dans l'index sesaindex.

5.2.2 Contrôleur

Un contrôleur Symfony prend l'information provenant de la requête HTTP, effectue un traitement et retourne une réponse HTTP.

L'intégration du prototype Compmep a nécessité l'ajout d'une méthode spécifique au contrôleur Symfony pour la recherche de ressources par les notions et capacités sur sesaindex. Initialement, le prototype de recherche, évoqué dans le chapitre 4.5, avait été écrit uniquement en HTML et JavaScript. L'interrogation de l'index sesaindex s'effectuait directement depuis le JavaScript par des requêtes AJAX. L'affichage du tableau des résultats était aussi géré JavaScript. L'intégration dans Symfony nous a obligé à mettre en place un système respectant le modèle MVC. La communication entre le contrôleur Solr et le serveur Solr est assurée par le bundle Symfony « Nelmio » qui utilise le client Solr Solarium évoqué dans le chapitre 4.1.6 de ce mémoire [NELMIO 2012].

La fonction filtreResultsAction, consultable en annexe 5A, du contrôleur solrcontroller.php reçoit le formulaire envoyé, suite à la sélection d'une notion ou d'une capacité par l'utilisateur, et transforme à nouveau la chaîne de caractères JSON en objet JSON grâce à la fonction PHP json_decode.

La fonction utilise la fonction initSolRService pour s'allouer une instance de recherche sur l'index sesaindex. Une requête de recherche Solr est ensuite constituée à partir de l'URI de la capacité ou notion issue du formulaire. La recherche est effectuée sur les champs « capnotion », pour les notions et capacités directement liées à la ressource, et ancestor. Cette requête effectue un « OU » entre ces 2 champs. Le service Web Solr retourne une liste des ressources, classées par pertinence. Les champs capnotion sont privilégiés par Solr car ils sont moins nombreux que les champs ancestor.

Le classement est réalisé selon les règles Solr suivantes :

- une ressource trouvée par le champ capnotion ne contenant que l'URI sera classée en premier ;
- une ressource trouvée par le champ capnotion contenant l'URI et d'autres sera classée en second. Plus il y a de champs capnotion liée à la ressource, plus le score et donc le classement de la ressource décroîtra ;
- une ressource trouvée dans le champ ancestor sera dans la fin du classement. Plus il y a de champs ancestor moins la ressource est pertinente.

Le résultat de la réponse Solr est ensuite formaté dans un tableau pour être envoyé à la vue filtresolr.html.twig qui affichera les ressources avec toute les caractéristiques souhaitées.

5.2.3 Vue filtresolr.html.twig

Les résultats sont affichés dans la vue filtresolr.html.twig spécialement créée à cet effet. La figure 90 illustre l'affichage d'une partie des ressources trouvées à partir de la notion angle.

Id	Titre	Résumé	Description	Type	Niveau	Capacités/notions	Actions
10541	Quadricalc n°4		Ressource du site Calcul@TICE			N angle	aperçu details
1110200	Correction exercice 5 p 37					N angle	aperçu details
1110649	Correction exercice 2 p 145					N angle	aperçu details
1110656	Correction exercice 9 p 146					N angle	aperçu details
1115039	Correction exercice 28 p 19					N angle	aperçu details
1110195	Correction exercice 40 p 35					N mesure d'angle N angle	aperçu details
1115513	Correction exercice 12 p 123					N fonction hyperbolique N angle	aperçu details
1115882	Correction exercice 4 p 207					N angle N diagonale	aperçu details

Figure 90 - Vue filtresolr.html.twig

5.2.4 Autres fonctions

Lors de l'intégration de Compmep, deux fonctions du contrôleur RessourceController.php ont été modifiées pour prendre en compte la problématique de cohérence entre la base de données et l'index sesaindex.

Les fonctions updateAction et deleteAction permettant respectivement de modifier et supprimer une ressource au niveau de la base de données Bibli ont été enrichies afin d'appliquer ses actions en

parallèle dans l'index. Les actions de mise à jour et de suppression d'une ressource dans sesaindex font appel à des fonctions d'un service Symfony développé spécialement pour réaliser ces opérations.

5.2.5 Service Solr

Un service Symfony permet de standardiser et centraliser des fonctionnalités pouvant être appelées par différentes fonctions de l'application.

Les fonctions de mise à jour et de suppression dans solrservice.php utilisent aussi le bundle Nelmio et la librairie Solarium pour réaliser les actions. Le principe est similaire à la fonction filtreResultsAction évoquée dans le chapitre 5.2.3.

5.2.5.1 Mise à jour d'une ressource

La fonction updateAction du contrôleur des ressources envoie l'objet « entity » et un tableau JSON des capacités et notions de la ressource à mettre à jour à la fonction updateSesaIndex du service. Cette fonction utilise plusieurs fonctions de la librairie Solarium pour constituer la requête de mise à jour :

- createUpdate : initialisation de la requête ;
- createDocument : initialisation du document de la ressource à mettre à jour ;
- addField : ajout des capacités et notions au document. La fonction addField permet de prendre en compte les champs à valeur multiple ;
- addCommit : information indiquant à Solr de valider l'opération après traitement ;
- addOptimize : information indiquant à Solr d'optimiser (suppression du document vide) l'index après traitement [SOLARIUM 2012].

L'identifiant, le titre, la description et le type de la ressource sont extraits de l'objet entity et ajoutés au document de la requête de mise à jour. Les capacités et notions sont extraites du tableau JSON et ajoutées au document. L'exécution de la requête s'effectue par l'appel de la fonction « update » fournie par la librairie Solarium. Si la modification échoue, la requête est sauvegardée dans un fichier pour relancer l'opération par un programme journalier qui reste à développer.

5.2.5.2 Suppression d'une ressource

La fonction deleteAction du contrôleur des ressources envoi l'identifiant de la ressource à supprimer à la fonction deleteSesaIndex du service. La fonction deleteSesaIndex utilise plusieurs fonctions de la librairie Solarium pour constituer la requête de suppression :

- createUpdate : initialisation de la requête ;
- addDeleteById : identifiant de la ressource à supprimer ;
- addCommit : information indiquant à Solr de valider l'opération après traitement ;
- addOptimize : information indiquant à Solr d'optimiser (suppression du document vide) l'index après traitement.

L'exécution de la requête s'effectue par l'appel de la fonction update fournie par la librairie Solarium. Si la suppression échoue, la requête est sauvegardée dans un fichier pour relancer l'opération par un programme journalier qui reste à développer.

5.2.5.3 Autres fonctions

Le service solrservice.php est composé de plusieurs autres fonctions utilisées dans le cadre du peuplement de l'index sesaindex :

- initSesaIndex : réalise l'import des ressources depuis la base de données ;
- searchSolR : effectue des recherches dans les index en fonction du nom de l'index passé en paramètre ;

- populateSesaIndex : lie des capacités et notions à une ressource ;
- popUpdate : mise à jour de certaines ressources avec des capacités et notions après l'exécution de populateSesaIndex ;
- commitSesaIndex : valide et optimise l'index sesaindex lors de l'exécution de populateSesaIndex.

Ces fonctions sont appelées par le programme PopulateSolRCommand.php qui réalise le peuplement de l'index sesaindex.

5.3 Peuplement de sesaindex

Le composant Console de Symfony facilite la création scripts PHP utilisables en ligne de commandes pouvant être exécutés pour n'importe quelle tâche récurrente, comme des tâches planifiées, des imports, ou d'autres processus à exécuter par lots.

Le programme PopulateSolRCommand.php a été développé afin d'importer les ressources de la base MySQL dans sesaindex. Ce programme permet aussi l'affectation aléatoire de capacités et de notions sur les ressources afin de pouvoir réaliser des tests fonctionnels sur d'importants volumes de données. PopulateSolRCommand utilise par défaut le bundle Nelmio et la librairie Solarium.

5.3.1 Import de la base de données

L'import des ressources de la base de données est réalisé par la fonction « execute » qui utilise la fonction initSesaIndex de solrservice.php. La fonction initSesaIndex s'appuie sur la bibliothèque PHP « cURL » pour exécuter la requête HTTP indiquant à Solr de lancer l'import configuré dans le db-data-config.xml présenté dans le chapitre 4 de ce document. La ligne de commande permettant de réaliser cette opération est présentée par la figure 91.

```
php app/console Compmep:populateSolR --import --attach
```

Figure 91 - Ligne de commande Symfony pour importer les ressources

La commande « php app/console compmep:populateSolR » est une commande Symfony permettant d'appeler le programme PopulateSolRCommand.php. L'option « --import » indique au programme d'indexer la base de données des ressources dans l'index sesaindex. Le paramètre optionnel « --attach » permet au programme de lier aléatoirement des notions et capacités aux ressources.

5.3.2 Indexation aléatoire

Plusieurs fonctions du programme PopulateSolRCommand.php sont utilisées pour effectuer ce traitement :

- execAttachNelmio : effectue le traitement d'indexation aléatoire ;
- loadOntoNelmio : liste les capacités et notions indexées dans ontoindex ;
- getSomeOntoIds : renvoie entre 1 et 4 capacités et notions tirées au sort à lier à 1 ressource ;
- execUpdate : effectue un traitement post-indexation aléatoire pour affecter des notions et capacités définies à une vingtaine de ressources définies.

L'index ontoindex est interrogé une seule fois pour lister l'ensemble des capacités et notions dans un tableau. L'index sesaindex est interrogé une première fois pour lister l'ensemble des ressources dans un tableau. Le tableau des ressources est ensuite parcouru. Pour chaque ressource, des capacités et notions sont tirées au sort aléatoirement depuis le tableau généré lors de l'interrogation de ontoindex.

L'index sesaindex est ensuite sollicité pour les mises à jour. Une requête HTTP est envoyée pour chacune des 35 838 ressources. Une mise à jour complémentaire (fonction execUpdate) est enfin lancée pour permettre de lier les ressources ayant pour identifiant 27 à 37 et 430 à 440 avec la notion « angle aigu » et les capacités « utiliser la définition du cosinus » et « construire l'axe de symétrie d'une figure ». Le peuplement s'exécute pendant 10 minutes environ.

5.4 Mise en œuvre du besoin fonctionnel

Le peuplement aléatoire de l'index des ressources a permis de valider que le scénario initial de recherche de ressources par une notion ou capacité correspondait aux attentes de Sésamath. Différents scénarios ont ensuite été écrits afin de faire évoluer le moteur de recherche en affinant les résultats par l'intermédiaire de filtres.

Le choix des filtres a été statué sur le niveau scolaire et le type caractérisant la ressource. Les scénarios ont ensuite été réalisés par itérations successives. Quatre scénarios ont été implémentés.

5.4.1 Scénario 1 : Recherche avec filtres d'abord

Dans ce premier scénario, l'utilisateur coche les filtres avant de lancer la recherche d'une capacité ou notion. La figure 92 montre l'ajout des arbres de filtres dans l'interface de Bibli.

Figure 92 - Interface avec les filtres

Le processus de recherche peut être analysé suivant les règles :

- si aucun filtre n'est coché, la recherche s'effectue sur les ressources indexées par la notion ou capacité sélectionnée « ET » ne possédant pas de niveau « ET » ne possédant pas de type ;
- si le filtre 4e, par exemple, est coché, la recherche s'effectue sur les ressources indexées par la notion ou capacité sélectionnée « ET » possédant le niveau 4e « ET » ne possédant pas de type ;

- si les filtres 4e et 5e sont cochés, la recherche s'effectue sur les ressources indexées par la notion ou capacité sélectionnée « ET » possédant le niveau 4e « OU » 5e « ET » ne possédant pas de type.

Le principe des ET et des OU des filtres pour le niveau scolaire s'applique à l'identique pour les filtres par types de ressources. La recherche Solr sur les documents ne possédant pas de champs niveau ou de types de ressources a pour syntaxe : « -<nom du champ>:[* TO *] ».

Ce scénario a ensuite évolué pour permettre une recherche de l'ensemble des ressources sans filtrer par une capacité ou notion. Si l'utilisateur coche un filtre du niveau scolaire ou des types de ressources, la recherche sera dynamiquement lancée avec le caractère « * » sur les champs capnotion et ancestor correspondant aux capacités et notion dans l'index sesaindex.

L'arbre des filtres a été implémente avec l'API JavaScript Jstree [JSTREE 2013]. Tout changement sur l'arbre est analysé par du JavaScript qui met à jour les champs du formulaire caché, de la vue layout.html.twig, et le soumet automatiquement au contrôleur Solr. La fonction filtreResultsAction a dû être adaptée pour gérer ces filtres.

5.4.2 Scénario 2 : Filtre avec critères après une première recherche

Ce scénario découle du précédent. Si après une première recherche l'utilisateur coche ou décoche un filtre, la nouvelle recherche est automatiquement relancée avec le filtre en plus ou en moins. La figure 93 montre que l'utilisateur a initialement recherché des ressources avec la notion angle et ne possédant pas de niveau et de type.

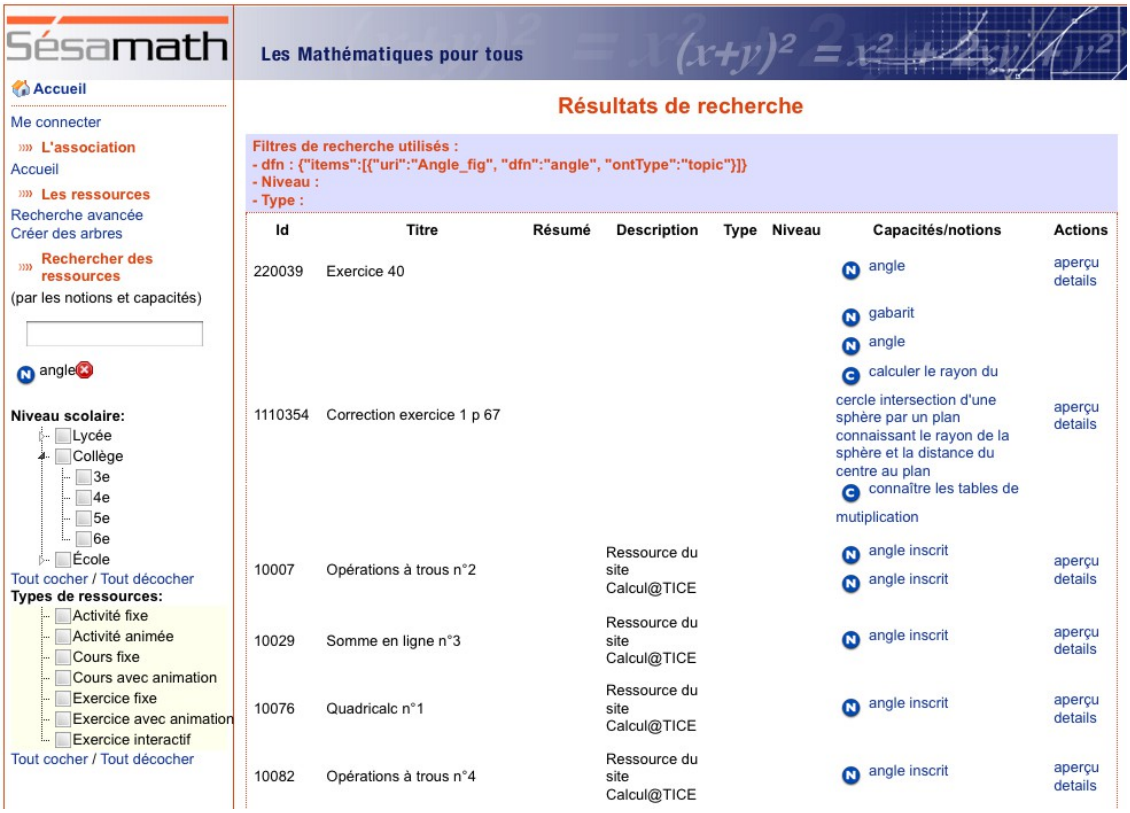


Figure 93 - Recherche de ressources indexées avec la notion angle

Le système affiche une liste de ressource indexée directement avec la notion angle (les 2 premières) ou indirectement par les ancêtres (angle étant un ancêtre de la notion angle inscrit). Si l'utilisateur coche le filtre 6e, le résultat illustré par la figure 94 devient tout autre.

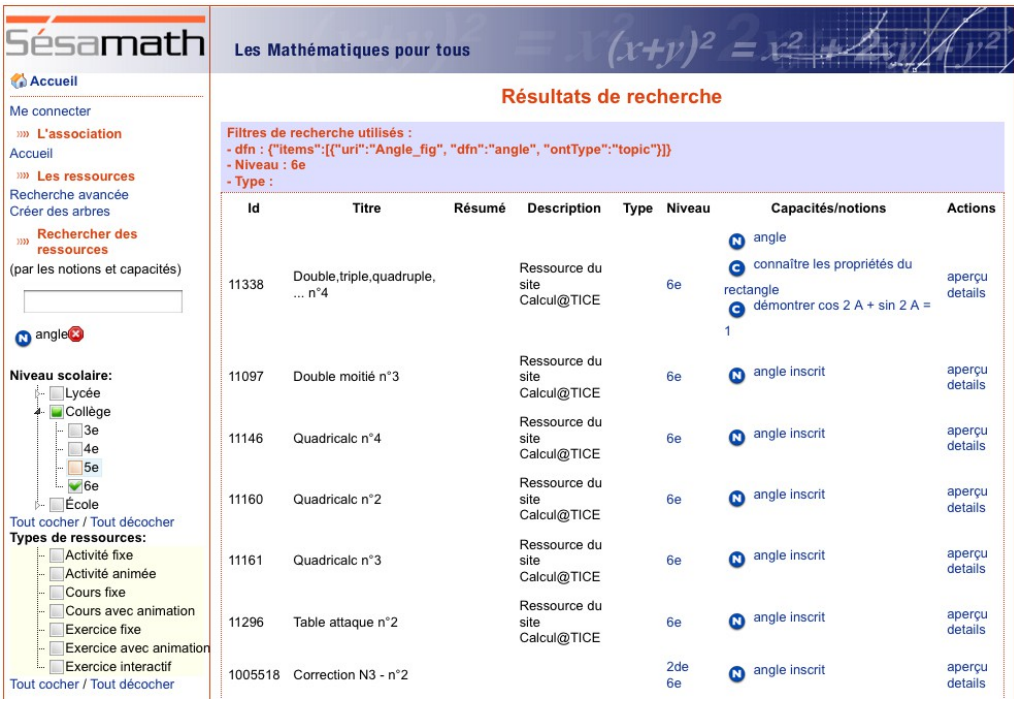


Figure 94 - Recherche de ressources indexées avec la notion angle et le filtre 6e

Les événements sur l'arbre des filtres sont gérées par du JavaScript qui met à jour les champs du formulaire caché, de la vue layout.html.twig, et le soumet automatiquement au contrôleur Solr.

5.4.3 Scénario 3 : Recherche de ressource avec filtres par clic sur une valeur d'un champ d'une des ressources retournées

Ce troisième scénario ne s'applique que si une première recherche par les notions et capacités avec ou sans filtres a déjà été réalisée. L'objectif est de pouvoir appliquer les filtres capacités ou notions, niveau et type de ressources à partir de la vue affichant les résultats. La figure 95 illustre une recherche sur la notion angle avec l'ensemble des niveaux scolaires et avec tous les types de ressources.

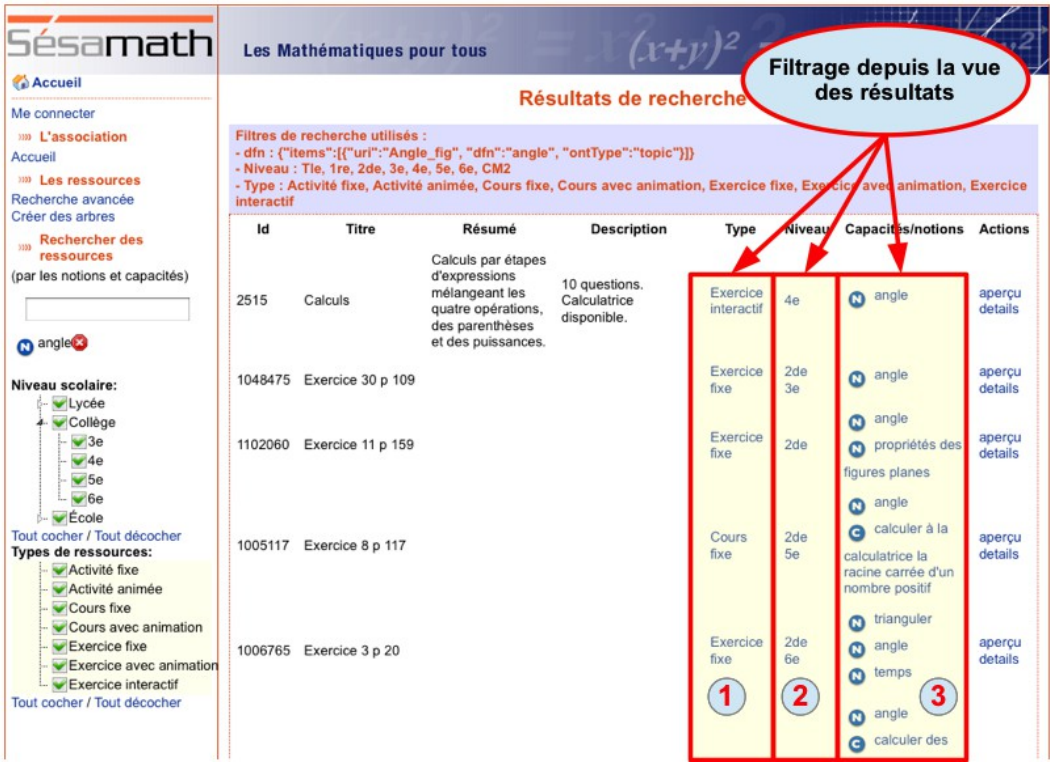


Figure 95 - Recherche tout filtres depuis la vue des résultats

Si l'utilisateur clique sur l'un des liens, dans les colonnes 1, 2 ou 3, une nouvelle recherche est automatiquement lancée en prenant en compte le filtre sélectionné. La figure 96 montre que l'utilisateur a sélectionné le lien de la notion « propriétés des figures planes ».

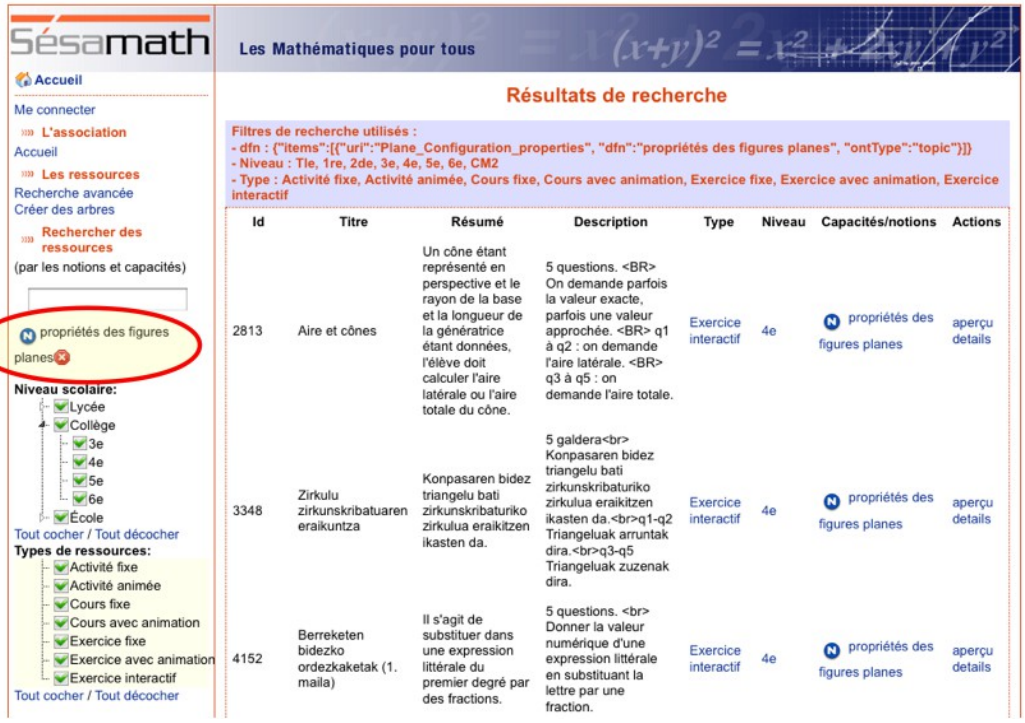


Figure 96 - Filtre depuis la vue des résultats

La notion angle a été remplacée par la notion « propriétés des figures planes », mais les autres filtres restent inchangés. Si l'utilisateur sélectionne ensuite 4e la recherche se réduit aux ressources indexées par la notion « propriétés des figures planes » et étant d'un niveau 4e et possédant un type. Le comportement est le même si l'utilisateur choisit un lien de la colonne des types.

Les événements sur les liens sont gérés par JavaScript qui met à jour les champs du formulaire caché, de la vue layout.html.twig, et le soumet automatiquement au contrôleur Solr.

5.4.4 Scénario 4 : Recherche par combinaison de plusieurs notions et capacités

Jusqu'à présent la recherche n'est réalisée qu'avec une seule capacité ou notion. Ce scénario a pour objectif d'affiner la recherche de ressources en combinant plusieurs notions et capacités. La figure 97 illustre une recherche sur une capacité et une notion à la fois.

Figure 97 - Filtres sur plusieurs capacités et notions

La combinaison est un « ET » sur les capacités et notions. Le système renvoie donc toutes les ressources indexées avec la notion angle aigu et la capacité utiliser la définition du cosinus. Les filtres sur les niveaux et les types restent inchangés. L'utilisateur peut combiner les notions et capacités à partir de la zone de saisie ou depuis les liens de la vue des résultats. Le fonctionnement du scénario 3, qui remplaçait la notion ou capacité, a donc été changé pour ce faire.

Le contrôleur Solr a été modifié pour effectuer une requête HTTP avec la syntaxe illustrée par la figure 98.

```
(capnotion:<capacité/notion 1> OR ancestor:<capacité/notion 1>) AND (capnotion:<capacité/notion 2> OR ancestor:<capacité/notion 2>)
```

Figure 98 - Syntaxe de la nouvelle requête

5.5 Tests de Performances

L'objectif de cette partie est de tester les performances de l'infrastructure et plus particulièrement le serveur Solr afin de valider la stabilité et la pérennité de la solution proposée à Sésamath. Pour ce faire, nous avons utilisé l'application « Tsung », que les équipes de Sésamath avaient déjà utilisée par le passé.

5.5.1 Présentation de l'application Tsung

Tsung est une application Open Source écrite en Erlang pouvant être installée sur un serveur Linux. Tsung permet de réaliser des tests de performance en simulant la connexion de nombreux utilisateurs à une infrastructure Web afin de vérifier quelle charge cette dernière peut supporter. Les résultats sont obtenus sous la forme de graphiques et de données brutes, basés sur différentes métriques, afin de valider les possibilités d'accueil de l'infrastructure [TSUNG 2013].

Tsung a pour principales fonctionnalités :

- la charge peut être distribuée sur un cluster permettant à chaque nœud de générer sa propre charge ;
- multi-protocoles utilisant un système de plugin (HTTP, WEBDAV, SOAP, PostgreSQL, MySQL, LDAP, SSL et XMPP/Jabber) ;
- de nombreuses adresses IP peuvent être utilisée en faisant de l'IP aliasing (eth0, eth0:1, eth0:2…) ;
- support du SNMP pour la supervision ;
- système de configuration via XML. De nombreuses sessions peuvent être utilisées pour simuler différents types d'utilisateur ;
- afin de générer un trafic réaliste, l'activité de l'utilisateur et le taux d'arrivée peut être aléatoire en utilisant une notion de probabilité ;
- des rapports HTML peuvent être générés pour voir les temps de réponse, la charge processeur, mémoire, etc.

Un des avantages de Tsung est qu'il ne se contente pas de se connecter au site, il va naviguer dessus comme un utilisateur lambda. Pour ce faire il faut utiliser « tsung-recorder », qui va faire office de proxy Web pour enregistrer la session de navigation sous forme d'un fichier XML.

Une fois tsung-recorder activé, il faut renseigner, dans un navigateur Web, les paramètres du proxy :

- l'adresse du serveur Tsung ;
- le port utilisé par tsung-recorder, 8090 par défaut.

La figure 99 montre le résultat d'une navigation sur le site Bibli où une recherche de ressources à partir de la notion « angle aigu » a été réalisée.

```
<session name='rec20130306-1609' probability='100' type='ts_http'>

<request><http url='http://ressources.devsesamath.net/' version='1.1'
method='GET'></http></request>
<request><http url='/css/3d5f088.css' version='1.1' method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/js/head/head.min.js' version='1.1'
method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/images/Sesamath_bandeau.png' version='1.1'
method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/images/accueil.gif' version='1.1'
method='GET'></http></request>
<request><http url='/favicon.ico' version='1.1' method='GET'></http></request>
<request><http url='/favicon.png' version='1.1' method='GET'></http></request>
```

```
<request><http url='/bundles/sesamathbibli/js/jquery/jquery-1.8.2.min.js'
version='1.1' method='GET'></http></request>
<request><http url='/js/4ce2d44.js' version='1.1' method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/images/sep.gif' version='1.1'
method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/images/puce/h2.gif' version='1.1'
method='GET'></http></request>
<request><http url='/themes/default/style.css' version='1.1'
method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/Compmep/jstree-themes/default/d.png'
version='1.1' method='GET'></http></request>

<thinktime random='true' value='2'/>

<request><http url='/solr/ontoindex/select?
json.wrf=jQuery18201795463314672575l_1362582779953&amp;q=name_auto%3Aan+OR+name_auto
%3Aan*&amp;fl=defaultCommonName%2C+ontType%2C+urlForNav
%2C+uriweak&amp;wt=json&amp;omitHeader=true&amp;rows=10&amp;_=1362582782201'
version='1.1' method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/Compmep/img/book.gif' version='1.1'
method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/Compmep/img/topic.png' version='1.1'
method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/Compmep/img/competency.png' version='1.1'
method='GET'></http></request>

<thinktime random='true' value='2'/>

<request><http url='/Compmep/results?dfn=%7B%22items%22%3A%5B%7B%22uri%22%3A
%22AcuteAngle%22%2C%22dfn%22%3A%22angle+aigu%22%2C%22ontType%22%3A%22topic%22%7D%5D
%7D&amp;filtreNiveau=&amp;filtreTypeSesa=' version='1.1'
method='GET'></http></request>
<request><http url='/bundles/sesamathbibli/Compmep/img/delete.png' version='1.1'
method='GET'></http></request>
<request><http url='/Compmep/themes/default/style.css' version='1.1'
method='GET'></http></request>
</session>
```

Figure 99 - Session Tsung enregistrée au format XML

Les premières séries de balises <request> correspondent au chargement du statique du site. Les balises <thinktime random='true' value='2'/> représente des délais d'attente entre les actions.

La figure 100 représente la requête envoyée sur l'index ontoindex de Solr pour rechercher les capacités et notions contenant le terme « an ».

```
<request><http url='/solr/ontoindex/select?
json.wrf=jQuery18201795463314672575l_1362582779953&amp;q=name_auto%3Aan+OR+name_auto
%3Aan*&amp;fl=defaultCommonName%2C+ontType%2C+urlForNav
%2C+uriweak&amp;wt=json&amp;omitHeader=true&amp;rows=10&amp;_=1362582782201'
version='1.1' method='GET'></http></request>
```

Figure 100 - Requête sur ontoindex

La dernière partie, illustrée par la figure 101, représente la requête envoyée sur l'index sesaindex de Solr pour rechercher les ressources liées à la notion « angle aigu ».

```
<request><http url='/Compmep/results?dfn=%7B%22items%22%3A%5B%7B%22uri%22%3A
%22AcuteAngle%22%2C%22dfn%22%3A%22angle+aigu%22%2C%22ontType%22%3A%22topic%22%7D%5D
%7D&amp;filtreNiveau=&amp;filtreTypeSesa=' version='1.1'
method='GET'></http></request>
```

Figure 101 - Requête sesaindex

Pour que Tsung puisse répéter le scénario, il faut rajouter différents paramètres à la session de navigation, comme illustré par la figure 102.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tsung SYSTEM "/usr/share/tsung/tsung-1.0.dtd">
<tsung loglevel="info" version="1.0">
  <clients>
    <client host="localhost" use_controller_vm="true"/>
  </clients>
  <servers>
   <server host="localhost" port="80" type="tcp"/>
  </servers>

  <!-- Montée en charge des connexions -->
  <load>
    <arrivalphase phase="1" duration="3" unit="minute">
      <users interarrival="5" unit="second"/>
    </arrivalphase>
    <!-- Durée du test 10 minutes -->
    <arrivalphase phase="2" duration="10" unit="minute">
      <!-- connexion de 2 utilisateurs toutes les secondes -->
      <users maxnumber="3000" interarrival="0.5" unit="second"/>
    </arrivalphase>
  </load>
  <options>
    <option type="ts_http" name="user_agent">
      <user_agent probability="100">Test tsung de sesamath
(tech@sesamath.net)</user_agent>
    </option>
  </options>

  <!-- Le test -->
  <sessions>

    <!-- COPIER ICI L'ENREGISTREMENT ISSU DU PROXY ET L'ADAPTER -->

  </sessions>
</tsung>
```

Figure 102 - Exemple de paramètres complétant le test Tsung

Il est aussi possible d'ajouter des variables à partir de fichiers en entrée du test pour diversifier les requêtes. Une boucle « for » à l'intérieur de la balise <session> permet de lancer plusieurs fois les requêtes pour chaque utilisateur simulé.

Un test complet est consultable en annexe 5B.

5.5.2 Infrastructure de test

Pour réaliser ces tests de charge, quatre machines virtuelles OpenVZ ont été mise à disposition pour simuler l'architecture cible de production. La figure 103 présente la répartition des différents serveurs sur une machine hôte.

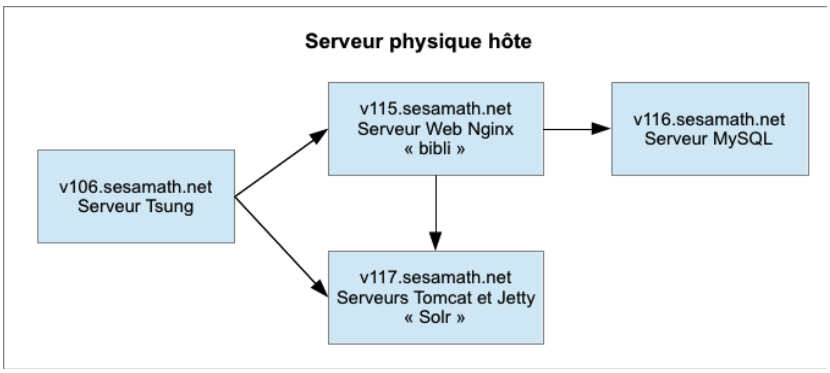


Figure 103 - Architecture de test

Les machines virtuelles utilisent le processeur « Intel(R) Xeon(R) » cadencé à 3.10GHz du serveur hôte. Concernant la mémoire vive, 1 Go est alloué aux machines v115 et v117, 2Go pour la machine v106 et 3Go pour la machine v116. Une rotation des logs a été mise en place sur les machines hébergeant le serveur Web et le serveur d'applications pour palier la saturation de l'espace disque. Pour pouvoir réaliser les tests, la limite maximum de fichiers ouverts a été portée à 65 536 sur l'ensemble des machines.

Cette architecture permet aussi de prendre en compte les contraintes de charge réseau.

5.5.3 Scénarios de tests

Pour réaliser les tests de performances de l'infrastructure, sept scénarios ont implémentés :

- scénario 1 : simulation de requêtes directement sur le serveur d'applications pour interroger ontoindex ;
- scénario 2 : simulation de requêtes directement sur le serveur d'applications pour interroger sesaindex ;
- scénario 3 : simulation de requêtes directement sur le serveur d'applications pour interroger ontoindex puis sesaindex ;
- scénario 4 : reprise du scénario 1 en passant par le serveur Web Nginx pour rediriger les requêtes vers le serveur d'applications ;
- scénario 5 : reprise du scénario 2 en passant par le serveur Web Nginx pour rediriger les requêtes vers le serveur d'applications ;
- scénario 6 : reprise du scénario 3 en passant par le serveur Web Nginx pour rediriger les requêtes vers le serveur d'applications ;
- scénario 7 : reprise du scénario 6 en activant le cache du serveur Web Nginx.

Les scénarios 1, 2, 4 et 5 ont vocation à valider le fonctionnement des tests et lever éventuellement des problèmes spécifiques sur la chaîne de liaison de l'infrastructure. Les scénarios 3, 6 et 7 permettent de définir quelle sera la solution retenue pour l'infrastructure.

Les scénarios ont ensuite été dupliqués en deux séries. La première série consiste à lancer les 7 scénarios en limitant le nombre de connexions simultanées à environ 370. La seconde série correspond à un test de charge de type « production » correspondant à environ 3 000 utilisateurs simultanés.

Enfin, ces tests de performances ont été exécutés sur les serveurs d'applications Tomcat et Jetty afin de comparer le comportement de chacun et choisir quelle solution est la plus performante en vue du déploiement en production.

5.5.4 Comparaison des résultats des tests

Le tableau 6 reprend les résultats des tests pour les scénarios 3, 6 et 7 de la série correspondant à la simulation des 3 000 utilisateurs simultanés.

	Tomcat Scénario 3	Jetty Scénario 3	Tomcat Scénario 6	Jetty Scénario 6	Tomcat Scénario 7	Jetty Scénario 7
Moyenne des pires temps de réponse	41 440 msec	9.95 msec	38.80 msec	76.16 msec	11.88 msec	16.34 msec
Moyenne des meilleurs temps de réponse	0.748 msec	0.833 msec	1.08 msec	1.13 msec	0.267 msec	0.273 msec
Temps de réponse moyen	9 850 msec	5.30 msec	5.52 msec	31.16 msec	4.45 msec	4.18 msec
Maximum de requêtes traitées par seconde	229.9 / sec	1 028.7 / sec	1 018.3 / sec	1 013 / sec	1 014.7 / sec	1 016.8 / sec
Total de requêtes HTTP	920 458	928 284	920 458	923 769	926 177	926 177
Utilisateurs simultanés	3 058	3 084	3 058	3 069	3 077	3 077
Débit réseau en émission	14.00 Mbits/sec	58.17 Mbits/sec	63.40 Mbits/sec	63.05 Mbits/sec	63.04 Mbits/sec	63.08 Mbits/sec
Débit réseau en réception	504.51 Kbits/sec	2.30 Mbits/sec	3.10 Mbits/sec	3.06 Mbits/sec	3.12 Mbits/sec	3.13 Mbits/sec
Code HTTP retourné	200 : 100 %	200 : 100 %	200 : 100 %	200 : 100 %	200 : 100 %	200 : 100 %

Tableau 6 - Bilan des tests de performances

L'analyse du tableau montre que le serveur Tomcat est peu performant dans le cadre du scénario3 avec des temps de réponse pouvant dépasser les 40 secondes. Le serveur d'application Jetty est bien meilleur en accès direct. Toutefois la tendance s'inverse sensiblement lorsque le serveur Nginx effectue la redirection des requêtes HTTP. Le serveur Tomcat donne satisfaction avec un temps de réponse moyen de 5,52 millisecondes contre 31,16 millisecondes pour le serveur Jetty. Avec l'utilisation du cache Nginx, on constate que les serveurs d'applications Tomcat et Jetty ont des temps de réponse quasiment similaires.

Le test de performance montre aussi la fiabilité de Solr puisque les codes HTTP retournés sont tous à 200 : « Requête traitée avec succès ».

5.5.5 Conclusion des tests

Les tests de performance sont donc concluants et permettent de valider la stabilité du système dans un contexte de production correspondant à 3 000 utilisateurs simultanés. L'infrastructure supporte sans problème particulier la charge et les temps de réponses sont satisfaisants. Le temps moyen de réponse inférieur à 5 millisecondes avec le cache Nginx confirme la robustesse du système.

Il n'y a pas de réelle différence entre Jetty (temps de réponse moyen de 4.18 millisecondes) et Tomcat (temps de réponse moyen de 4.45 millisecondes) lorsque la mise en cache Nginx est activée. Le serveur d'applications Tomcat peut tout de même être préféré à Jetty pour la mise en production. La documentation et la forte communauté Apache sont des avantages non négligeables à prendre en compte du fait de la méconnaissance de l'administration d'un serveur d'applications par Sésamath.

Des « stress tests » plus conséquents sont à prévoir pour valider jusqu'à combien d'utilisateurs l'infrastructure peut supporter. Cela permettra ainsi d'affiner les réglages des différents serveurs et notamment au niveau de Tomcat et Solr.

5.6 Bilan de l'intégration Compmep chez Sésamath

L'intégration du prototype Compmep dans la bibliothèque Sésamath est un succès. La solution est satisfaisante et répond aux besoins de Sésamath. Les technologies utilisées sont en grande partie déjà maîtrisées par Sésamath.

Les fonctions de mise à jour et de suppression d'une ressource dans l'index étaient fonctionnelles au moment de l'intégration de Compmep. Toutefois, les développements en cours sur la bibliothèque impliquent une relecture et des modifications potentielles au niveau des variables une fois que la structure de l'objet « ressource » (la principale « entity » Symfony de l'application) sera stabilisée. Ces fonctions doivent aussi être améliorées pour prendre en compte l'échec éventuel de mise à jour ou de suppression dans l'un des deux référentiels. Un programme journalier est à développer pour garantir la cohérence des données entre l'index Solr et la base MySQL.

Les nouvelles fonctionnalités implémentées pour filtrer et affiner la recherche de ressources ont toutes été validées par la maîtrise d'ouvrage de Sésamath. Des tests fonctionnels plus conséquents sont toutefois à planifier avant la mise en production. Nos tests ont été réalisés sur des ressources liées aléatoirement aux notions et capacités.

Les résultats des tests de performance confirment la pérennité du système mis en œuvre. Le choix d'utiliser le serveur d'application Tomcat garantit un support important pour Sésamath dans le cadre d'éventuel problème d'administration ou d'un besoin d'optimisation.

Le système est stable et peut éventuellement être mis en production en l'état. Des évolutions peuvent néanmoins être ajoutées. Les arbres des filtres sont statiques dans le code HTML de la vue de la page d'accueil. Une construction dynamique depuis les objets des niveaux et des types de la base de données apporterait de la souplesse et n'impacterait pas de modifier le code HTML en cas de modification.

L'utilisation de la bibliothèque Solarium a été un choix arbitraire et des tests avec les clients PECL ou Solrbundle permettrait de valider si l'un d'un est le meilleur en terme de performance.

Conclusion

En conclusion, nous revenons sur les différents aspects du stage en établissant une synthèse des objectifs est du travail effectué puis en proposant quelques perspectives d'évolutions. Enfin un bilan personnel termine ce mémoire.

Conclusion

L'analyse de l'infrastructure de Sésamath a montré l'ampleur de l'hétérogénéité des ressources, non centralisées et dans des formats différents. Cette problématique est d'autant plus vraie avec les enjeux d'ouverture vers d'autres pays européens et de ce fait un élargissement de la communauté déjà conséquente au niveau de la France. La multiplication des ressources impose donc un système de recherche rapide et pertinent.

L'éventualité de reprendre la solution i2Geo a été écartée car le code était difficilement maintenable et les technologies totalement en Java ne correspondaient pas vraiment aux contraintes liées à l'infrastructure de Sésamath. De plus l'analyse détaillée du système a permis de mettre en évidence de nombreux problèmes coûteux à corriger. Un mois a été nécessaire pour installer l'ensemble des composants nécessaires au fonctionnement de la plate-forme. La qualité de l'indexation, notamment les valeurs doublonnées et les écarts des scores entre les documents dans l'index des notions et capacités rendent peu fiable la pertinence des réponses. La complexité de réutilisation, d'administration et l'interdépendance des différents modules nous ont confortés dans le choix de mettre en œuvre une solution plus simple. Toutefois, la démonstration réalisée à partir de la plate-forme a permis de valider que les principes correspondaient aux attentes et besoins de Sésamath.

Le choix de Lucene comme moteur d'indexation Open Source a été confirmé par l'étude du CRIM. Le service Web Solr basé sur une configuration à partir de fichiers XML permet une appropriation rapide et assure un transfert de compétences rapide. La communauté Apache est un gage d'évolutions et de support assurant la pérennité du système.

L'objectif de reprendre les principes initiés par le projet Intergeo en utilisant le moteur d'indexation Solr basé sur Lucene est concluant. Le prototype mis en œuvre pour indexer les notions et capacités de l'ontologie GeoSkills a pris en compte les problèmes découvert lors de l'analyse du système Intergeo. La pertinence de l'indexation a même été améliorée. Les variables d'environnement sont déportées dans des fichiers « properties » plutôt que dans le code source pour permettre une éventuelle réutilisabilité. Le service Web Solr simplifie l'infrastructure faisant abstraction de l'implémentation Lucene.

Le choix de clients Web pour les prototypes de recherche a permis de prendre en compte les contraintes matérielles de Sésamath. Ces technologies ont aussi facilité l'intégration dans la bibliothèque en cours de développement chez Sésamath. La solution implémentée offre de la facilité d'administration avec un couplage faible à l'application existante.

Le développement de la bibliothèque, par Sésamath, avait été lancé pour répondre au besoin de normalisation de l'indexation des ressources Sésamath. Le projet Compmep est donc arrivé au bon moment pour s'insérer dans ce chantier. Il y a eu quelques inconvénients à travailler dans un cadre qui manquait encore de stabilité. Nous avons été tributaires des avancements du développement de Bibli et

des changements au niveau des spécifications au cours de la réalisation. Des adaptations de l'index sesaindex et des clients de recherches ont été nécessaires suite aux évolutions du schéma de la base de données.

Au final, l'intégration dans Bibli est un succès et les attentes du client ont été réalisées. Des fonctionnalités non présentes dans le système Intergeo ont pu être ajoutées afin d'affiner la pertinence de la recherche des ressources. Les tests de performance ont démontré la stabilité du système et permis de valider le choix de Tomcat comme serveur hébergeant le service Web Solr.

Chaque composant développé est accompagné d'une documentation et est référencé dans un système de gestion de versions :

- Compmep est référencé avec le projet Bibli dans le SVN Sésamath ;
- Solr et ontoIndexation sont référencés dans le SVN du LIG.

Le système pourra donc être mis en production une fois la bibliothèque finalisée. Malheureusement le temps nous a manqué car le choix des technologies a élargi les perspectives et possibilités du système.

Perspectives et évolutions

Le projet n'est donc pas terminé et peut faire l'objet d'un ou plusieurs stages complémentaires pour implémenter de nouvelles fonctionnalités et développer un système permettant d'éditer collaborativement l'ontologie. CompEd étant dépendant du module SearchI2G, une piste a été explorée avec l'installation du logiciel WebProtégé, développé en Java et s'installant dans un serveur d'application tel que Tomcat. Toutefois, le temps nous a manqué pour développer une interface destinée aux enseignants ou à un groupe de Sésamath.

La solution de mettre l'ontologie dans une base de données, comme le fait CompEd, peut aussi être une piste intéressante. L'interface d'édition pourrait utiliser des technologies Web HTML et PHP pour répondre aux contraintes matérielles de Sésamath si la solution WebProtégé est trop demandeuse de ressources machine. Un programme Java pourrait simplement répliquer les modifications de l'ontologie base de données vers l'ontologie au format « .owl ».

Le système intégré à la bibliothèque ne prend pas en compte les inférences et raisonnement sur l'ontologie. Cette piste pourrait apporter plus de sens au système.

Concernant le système actuel, une interface d'indexation massive pour faciliter le travail des enseignant qui auront la charge de lier les ressources aux notions et capacités lors de la mise en production serait un atout majeur non négligeable. Plusieurs scénarios fonctionnels n'ont pu être ajoutés au système :

- une recherche plein texte sur le champ description et autre de la ressource ;
- étendre la recherche à des notions ou capacités connexes à la notion ou capacité initiale ;
- un « panier » pour que l'enseignant puisse choisir ses ressources et les consulter ultérieurement ;
- les composants d'une ressource composite devront hériter des indexations par capacités/notions de la ressource mère ;
- le niveau ne sera plus codé dans la ressource mais uniquement indirectement dans les capacités et notions ;
- la gestion des statistiques de recherche permettant de mettre en évidence quelles notions ou capacités sont les plus recherchées et inversement. Il faudrait aussi prendre en compte et différencier les notions et capacités directes des indirectes, celles issues de l'ancêtre ;
- la prise en compte des autres pays européens.

Pour améliorer la pertinence du système, on peut envisager que le score prenne en compte la profondeur des ancêtres. Plus l'ancêtre est éloigné de la notion ou capacité moins son score sera important. Actuellement tous les ancêtres ont le même poids dans les index. Des tests utilisateurs avec des données réelles peuvent aussi permettre d'affiner la pondération.

Au niveau technique et pour l'administration de l'infrastructure, un programme journalier pourrait s'assurer la cohérence entre la base de données et l'index des ressources et réaliser les éventuelles synchronisations et reprise sur incident. Des tests de performances avec les clients PECL et Solrbundle permettrait de comparer les résultats par rapport au client Nelmio/Solarium.

Enfin une évolution du service Web Solr peut aussi être envisageable. En effet Solr 4 est passé en version stable vers la fin du projet. Un test a été effectué mais Solr 4 ne prend plus l'option « waitFlush » lors de l'optimisation des index. Ce paramètre, configuré dans Nelmio/Solarium, pose donc problème pour le moment. La commande « optimize » sort en erreur et n'effectue pas l'optimisation de l'index qui devient inexploitable pour cause de verrou non enlevé. Il faut donc surveiller les évolutions du client avant de migrer pour exploiter les nouvelles fonctionnalités apportées dans Solr 4.

Pour conclure, Sésamath est satisfait du déroulement du stage et envisage une collaboration plus accrue avec MeTAH. Sésamath pourra ainsi bénéficier des fruits de la recherche alors que MeTAH profitera de la forte communauté Sésamath pour des mises en pratique.

Bilan personnel

Mon objectif premier était de réaliser un stage orienté programmation afin de mettre en pratique les cours du CNAM et aborder un domaine qui m'était assez méconnu. En effet, mon activité d'intégrateur à la CNAMTS ne m'a jamais permis de me confronter aux problématiques de conception et de mise en œuvre d'une application. J'ai donc profité de l'opportunité du stage de fin d'étude d'ingénieur CNAM pour passer de l'autre côté de la barrière et élargir mon domaine de compétences. Ce stage m'a donc permis de mettre en pratique les langages Java, abordé dans le cadre de mon cursus au CNAM, et PHP que j'utilisais de manière autodidacte. Le langage JavaScript et l'utilisation d'un cadriciel et plus particulièrement Symfony2 furent une totale découverte.

J'ai donc rencontré quelques difficultés liées à ma faible expérience de programmation. L'appropriation du code d'Intergeo fut fastidieuse mais au final une bonne expérience en terme d'expertise. J'ai aussi pu constater qu'il me manquait une certaine connaissance des standards et du respect des bonnes pratiques de programmation. Il y a donc eu beaucoup de choses à découvrir et appréhender en peu de temps au final et qui mériteraient plus d'approfondissement.

La découverte des technologies d'indexation et principalement Apache Solr a été passionnante. Ce service Web offre des perspectives vraiment intéressantes. La collaboration en mode projet à distance lors de l'intégration de Compmep dans la bibliothèque de Sésamath fut aussi une bonne expérience et riche en échanges.

Au final, ce stage fut une expérience très enrichissante et m'a confirmé que j'avais un profil plus orienté « test » et architecture que programmation.

Annexes

Diagramme de Gantt

1A Étude sur la création de ressources dans LaboMep

Message ou question :

Message ou question permet, comme son intitulé l'indique, à l'enseignant de rédiger un message ou de poser une question aux élèves.

Le premier bouton « Titre » permet à l'enseignant de donner un titre et éventuellement un descriptif (facultatif) à la ressource en cours de création. Il indique aussi s'il s'agit d'une question ou d'un message. Le second bouton « Message ou question » permet à l'enseignant de rédiger le message ou la question qu'il souhaite soumettre aux élèves. Le troisième bouton « Enregistrer » permet d'enregistrer le message ou la question pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant son titre et « Tester » pour valider le fonctionnement de l'exercice.

Exercice TracenPoche :

TracenPoche est un logiciel de géométrie dynamique utilisable sur Internet ou en local grâce à la technologie Flash© Adobe. C'est un projet de Sésamath utilisé notamment dans les exercices de Mathenpoche.

Le premier bouton « Titre » permet à l'enseignant de donner un titre et éventuellement un descriptif (facultatif) à la ressource en cours de création. Le second bouton « Consigne » permet à l'enseignant de saisir une consigne adressée à l'élève, de choisir son positionnement dans la fenêtre et d'ajouter éventuellement une zone de texte dans laquelle les élèves peuvent écrire une réponse. L'ajout d'une consigne est facultatif. Le troisième bouton intitulé « Programmation » permet de concevoir la figure initiale de l'exercice, qui sera affichée à l'élève. Cette partie est facultative si l'enseignant souhaite que l'élève effectue une construction à partir d'une page vide. Le bouton intitulé « Boutons » permet de sélectionner les boutons mis à disposition de l'élève pour la construction de figure géométrique. Par défaut tous les boutons sont visibles. Le bouton « Zone » permet de rendre visible à l'élève le script de la figure ainsi qu'éventuellement son analyse. Dans la zone Script, l'élève peut voir la description de la figure (dans le langage propre à TracenPoche). Dans la zone Analyse, il est possible de faire afficher les mesures de segments, d'angles, de faire des calculs, etc. Le sixième bouton « Enregistrer » permet d'enregistrer le message ou la question pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant sont titre et « Tester » pour valider le fonctionnement de l'exercice.

Exercice avec la calculatrice cassée :

Le professeur décide des touches qui sont disponibles ou « cassées » sur une calculatrice virtuelle, pour afficher un résultat donné, ce qui oblige l'élève à développer une stratégie de calcul.

Le premier bouton « Titre » permet à l'enseignant de donner un titre et éventuellement un descriptif (facultatif) à la ressource en cours de création. Le second bouton « Programmation » permet notamment le paramétrage de la calculatrice en rendant certaines touches inutilisables. Le troisième bouton « Enregistrer » permet d'enregistrer l'exercice pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant sont titre et « Tester » pour valider le fonctionnement de l'exercice.

Exercice d'opération posée :

L'exercice d'opération posée permet d'effectuer des opérations d'addition, soustraction, multiplication et division.

Le premier bouton « Titre » permet à l'enseignant de donner un titre et éventuellement un descriptif (facultatif) à la ressource en cours de création. Le second bouton « Programmation » permet la conception de l'exercice. Le troisième bouton intitulé « Paramètre » permet d'apporter quelques paramètres supplémentaires à l'exercice. Le quatrième bouton « Enregistrer » permet d'enregistrer

l'exercice pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant sont titre et « Tester » pour valider le fonctionnement de l'exercice.

Exercice GeoGebra :

GeoGebra est un logiciel libre de géométrie dynamique en 2D, c'est-à-dire qu'il permet de manipuler des objets géométriques du plan (cercle, droite et angle, par exemple) et de voir immédiatement le résultat.

Le premier bouton « Titre » permet à l'enseignant de donner un titre et éventuellement un descriptif (facultatif) à la ressource en cours de création. Le second bouton « Consigne » permet à l'enseignant de saisir une consigne adressée à l'élève, de choisir son positionnement dans la fenêtre et d'ajouter ou non une zone de texte dans laquelle les élèves peuvent écrire une réponse. L'ajout d'une consigne est facultatif. Le troisième bouton intitulé « Programmation » permet de concevoir la figure initiale de l'exercice, qui sera affichée à l'élève. Cette partie est facultative si l'enseignant souhaite que l'élève effectue une figure à partir d'une page vide. Le quatrième bouton « Enregistrer » permet d'enregistrer l'exercice pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant sont titre et « Tester » pour valider le fonctionnement de l'exercice.

Page internet externe à LaboMep :

Ce type d'exercice outil permet d'intégrer facilement dans LaboMep d'autres ressources du Web, de différents types.

Le premier bouton « Adresse du site » permet à l'enseignant de donner un titre et éventuellement un descriptif (facultatif) à la ressource en cours de création. L'enseignant doit aussi indiquer l'adresse du site de la page Internet que l'élève consultera pour réaliser l'exercice. Cette zone permet aussi la pré-visualisation de la page. Le second bouton « Paramètre » offre la possibilité à l'enseignant d'ajouter (ou non) une consigne pour l'élève et/ou une possibilité de réponse de la part de l'élève. Le troisième bouton « Enregistrer » permet d'enregistrer l'exercice pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant sont titre et « Tester » pour valider le fonctionnement de l'exercice.

Exercice de calcul mental :

L'outil « Exercice de calcul mental » permet de proposer aux élèves des exercices de calcul mental en paramétrant très finement un certain nombre de calculs. En particulier, il est possible de fixer certains nombres, de les prendre dans des intervalles, de fixer le nombre de chiffres après la virgule, etc. et régler le temps d'affichage de chaque partie. Le temps de réponse pour l'élève est aussi paramétrable.

Le premier bouton « Titre » permet à l'enseignant de donner un titre et éventuellement un descriptif (facultatif) à la ressource en cours de création. Le second bouton « Programmation » permet le paramétrage de l'exercice en définissant différents tels que :

- nombre de calculs de cette série dans l'exercice ;
- type de calculs de cette série (somme de 2 termes, différence, etc.) ;
- paramétrisation du temps de réponse de l'élève ;
- le nombre de série d'opérations à réaliser ;
- etc.

Le troisième bouton « Enregistrer » permet d'enregistrer l'exercice pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant sont titre et « Tester » pour valider le fonctionnement de l'exercice.

Exercice LaboMep (QCM) :

Le premier objectif de cet outil est de pouvoir créer des exercices type QCM. Pour cela, il est possible de taper facilement des formules mathématiques, d'insérer des images mais aussi des figures dynamiques. Les réponses des élèves sont enregistrées automatiquement dans le bilan des séances contenant ce type d'exercices.

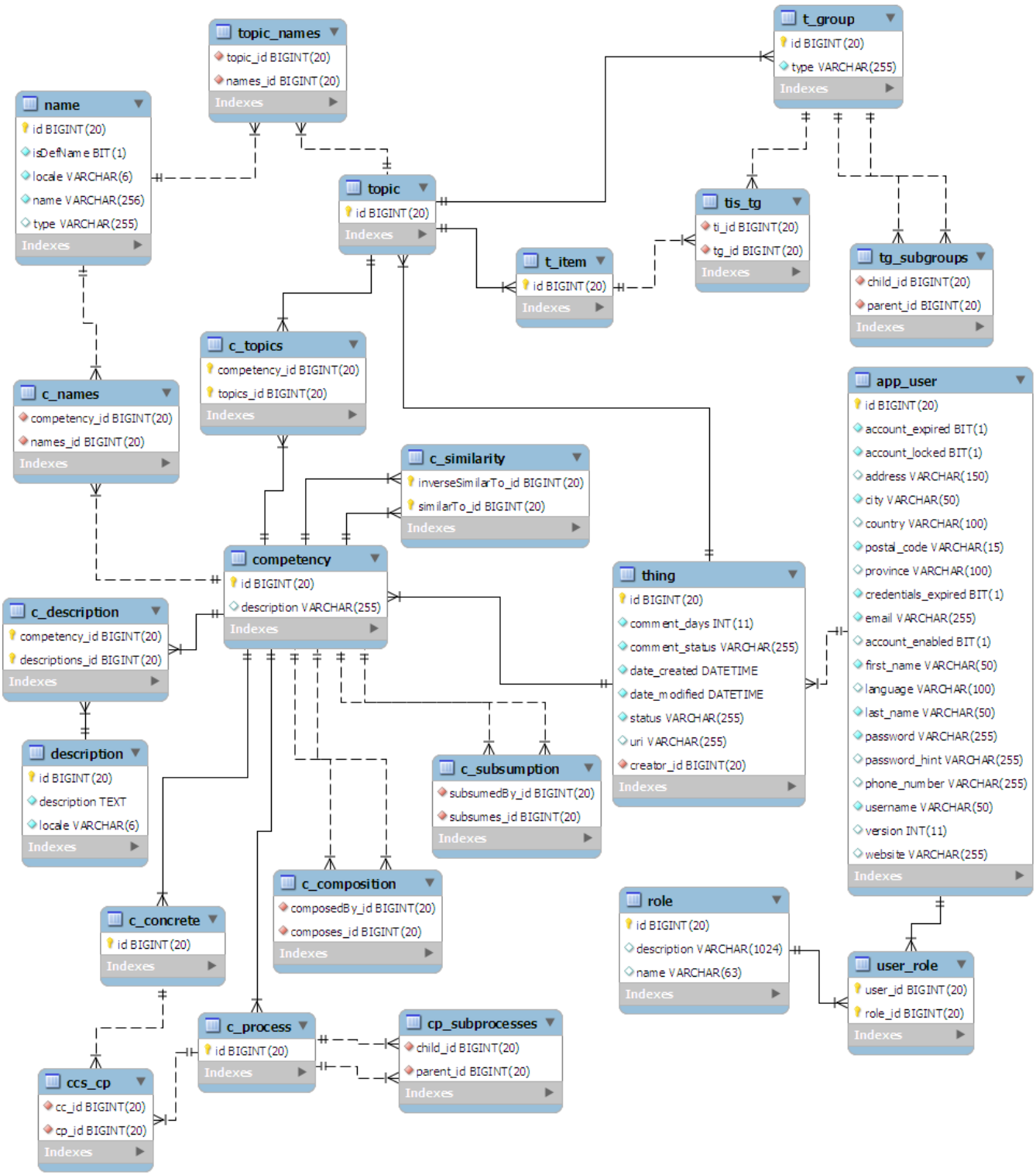
L'interface de création de ce type d'exercices est quelque peu différente des outils exercices vus précédemment. L'enseignant donne un titre à la ressource puis clique sur le bouton « Descriptif » pour donner une description (facultatif) de l'exercice en cours de création.

L'exercice comporte par défaut une partie à paramétrer. Une partie correspond à une question du QCM. Il est possible de rajouter d'autres questions par l'intermédiaire du bouton « Ajouter une partie ».

Le bouton « Programmer cette partie » permet de paramétrer la question, de définir le nombre de réponse possible et notamment celle qui est valide, etc.

Le bouton « Enregistrer » permet d'enregistrer l'exercice pour finaliser la création de la ressource. Deux nouveaux boutons se rajoutent ensuite : « Enregistrer sous » pour dupliquer la ressource en changeant sont titre et « Tester » pour valider le fonctionnement de l'exercice.

2A Modèle Conceptuel de Données de CompEd



2B Déroulement de l'exécution du script « Install-All.sh »

1. Extraction des dépendances Maven dans $HOME/.m2. Cette étape évite le téléchargement des dépendances depuis sur les différents dépôts accessibles Internet.
2. Extraction et nettoyage des sources Curriki.
3. Extraction et nettoyage des sources I2Gcurriki.
4. Synchronisation des sources I2Gcurriki avec Curriki. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
5. Installation du fichier pom.xml pour Xwiki.
6. Installation du fichier pom.xml pour curriki-parent.
7. Installation de la librairie xwiki-xar-handlers-1.9-SNAPSHOT.jar.
8. Installation de la librairie searchi2g-xwiki-component-1.0-SNAPSHOT.jar.
9. Extraction du module WIRIS_HTML_conversion.
10. Compilation du module WIRIS_HTML_conversion. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
11. Extraction des sources SearchI2G. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
12. Compilation de l'API I2GEOAPI.
13. Compilation des plugins Curriki.
14. Compilation de Curriki.
15. Compilation du Wiki de Curriki.
16. Création de la base de données Xwiki et de l'utilisateur d'administration.
17. Configuration du fichier hibernante.cfg.xml.
18. Déploiement de Xwiki Curriki dans les applications de Tomcat.
19. Compilation de SearchI2G.
20. Déploiement de SearchI2G dans les applications de Tomcat.
21. Extraction des sources ServletUtils. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
22. Compilation de ServletUtils.
23. Extraction des sources RootWebapp. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
24. Installation de la librairie i2geo-servletutils-1.0-SNAPSHOT.jar pour RootWebapp.
25. Packaging de RootWebapp.
26. Déploiement de Root dans les applications de Tomcat.
27. Extraction du module Static. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
28. Déploiement du module Static dans les applications Tomcat.
29. Extraction des sources comped-maven-plugin. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
30. Installation du module comped-maven-plugin. Cette étape supprime aussi les dépendances Maven dans $HOME/.m2 et installe celles nécessaires à la compilation de CompEd.
31. Extraction des sources OntoServer. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
32. Installation de OntoServer.
33. Packaging de OntoServer.
34. Déploiement de OntoServer dans les applications de Tomcat.
35. Création du répertoire « ontologies » dans les applications de Tomcat. Cette étape copie aussi les ontologies livrées par Install-i2geo.
36. Extraction des sources CompEd. Cette étape effectue aussi le remplacement des URL « Intergeo » renseignées dans les sources.
37. Configuration automatique du fichier pom.xml.
38. Création de la base de données pour CompEd et de l'utilisateur d'administration.
39. Installation de CompEd. La base de données CompEd est chargée avec les objets de l'ontologie.
40. Déploiement de CompEd dans les applications de Tomcat.
41. Copie des librairies partagées dans le répertoire lib de Tomcat.

2C Liste des problèmes mineurs restant sur Curriki

Liens du site	Problème rencontré
Textes de curriculum	Lien vers le module Static non fonctionnel
Naviguer par Sujet	Page ou ressource non trouvée
Naviguer par Logiciel	Non renommé (affichage : panel.navigation.browse_by_software)
Évaluations récentes	Page ou ressource non trouvée
Système de Revue I2geo	Page ou ressource non trouvée
Voir tous les groupes	Non traduit (affichage : viewAllGroups)
Forum des utilisateurs	Non traduit (affichage : Users Mailing List)
Discussions instantanées	Page ou ressource non trouvée
Mes Évaluations	– Non traduit (affichage : My Reviews) – Page ou ressource non trouvée
Tutoriaux	Page ou ressource non trouvée
Activités d'Intergeo	Page ou ressource non trouvée
Logiciels de géométrie dynamique	Non renommé (affichage : panel.navigation.DGSsoftwares)
Publications	Page ou ressource non trouvée
Groupes de travail	Page ou ressource non trouvée
Conférences	Page ou ressource non trouvée
Connexions	– Non renommé (affichage : panel.navigation.interconnections) – Page ou ressource non trouvée
Participation	Page ou ressource non trouvée
Aide	Page ou ressource non trouvée
Droits d'auteurs	Page ou ressource non trouvée
Signalez un ennui	Lien vers le module Static non fonctionnel
Aide	Page ou ressource non trouvée
Droits d'auteurs	Page ou ressource non trouvée

2D Index Curriki : Document resource

Champs	Valeur	Paramètres
_docid	Id du document	Store : YES Index : UN_TOKENIZEd
author	Dernier contributeur sur la ressource	Store : YES Index : TOKENIZED
creationdate	Date de création de la ressource	Store : YES Index : UN_TOKENIZED
creator	Créateur de la ressource	Store : YES Index : TOKENIZED
date	Date de modification	Store : YES Index : UN_TOKENIZED
fullname	Nom complet de la ressource au format xwiki	Store : YES Index : TOKENIZED
lang	Langue de la ressource	Store : YES Index : TOKENIZED
name	Nom du document	Store : YES Index : TOKENIZED
title	Titre de la ressource	Store : YES Index : TOKENIZED
type	Type du document	Store : YES Index : TOKENIZED
web	Collection	Store : YES Index : TOKENIZED
wiki	xWiki	Store : YES Index : TOKENIZED

2E Index Curriki : Document resource.objects

Champs	Valeur	Paramètres
CurrikiCode.AssetClass.description	Description de la ressource	Store : YES Index : TOKENIZED
CurrikiCode.AssetClass.eduLevelFine	Niveau éducatif de la ressource	Store : YES Index : UN_TOKENIZED
CurrikiCode.AssetClass.instructionnal_component	Type d'instructions liées à la ressource	Store : YES Index : TOKENIZED
CurrikiCode.AssetClass.instructionnal_component.key	Clé du type	Store : YES Index : TOKENIZED
CurrikiCode.AssetClass.instructionnal_component.value	Valeur du type	Store : YES Index : TOKENIZED
CurrikiCode.AssetClass.trainedTopicsAndCompetencies	Fragment de l'URI de la notion ou compétence attachée à la ressource	Store : YES Index : UN_TOKENIZED
CurrikiCode.AssetLicenseClass.rightsHolder	Droits du créateur de la ressource	Store : YES Index : TOKENIZED
CurrikiCode.AttachementAssetClass.file_type	Type de la pièce jointe éventuelle	Store : YES Index : TOKENIZED
_docid	Id du document	Store : YES Index : UN_TOKENIZED
assetType	Type de la ressource	Store : YES Index : UN_TOKENIZED
author	Dernier contributeur sur la ressource	Store : YES Index : TOKENIZED
creationdate	Date et heure de création de la ressource	Store : YES Index : UN_TOKENIZED
creator	Créateur de la ressource	Store : YES Index : TOKENIZED
date	Date de modification	Store : YES Index : UN_TOKENIZED
fullname	Nom complet de la ressource au format xwiki	Store : YES Index : TOKENIZED
i2geo.ancestorTopics	Ancêtres de ou des notion(s) et compétence(s) liée(s) à la ressources	Store : YES Index : UN_TOKENIZED
lang	Langue de la ressource	Store : YES Index : TOKENIZED
name	Nom du document dans l'index	Store : YES Index : TOKENIZED
object	Objets Curriki et Xwiki liés à la ressource	Store : YES Index : TOKENIZED
title	Titre de la ressource	Store : YES Index : TOKENIZED
type	référence au document « ressource »	Store : YES Index : TOKENIZED
web	référence au document « ressource »	Store : YES Index : TOKENIZED
wiki	référence au document « ressource »	Store : YES Index : TOKENIZED

3A Modalités d'utilisation des publications du CRIM

Modalités d'utilisation

Le présent texte établit les termes et conditions d'utilisation de ce site Web. Le CRIM se réserve le droit de faire des mises à jour de cette page et de modifier les modalités et conditions d'utilisation en tout temps et à sa discrétion. IL VOUS APPARTIENT DE LIRE ET DE VÉRIFIER CETTE PAGE RÉGULIÈREMENT, PUISQUE VOTRE UTILISATION DE CE SITE WEB EST ASSUJETTIE AUX MODALITÉS ET CONDITIONS STIPULÉES AUX PRÉSENTES. L'UTILISATION DE CE SITE WEB SIGNIFIE QUE VOUS CONSENTEZ AUX MODALITÉS D'UTILISATION DU PRÉSENT AVIS.

Avis relatif aux droits de propriété intellectuelle

Tous droits réservés © 2008 CRIM.

Le CRIM est propriétaire des droits de propriété intellectuelle du HTML sous-jacent, des textes, photographies, graphiques, extraits audio et vidéo et de tout autre élément contenu et accessible sur ce site Web (« le contenu ») où il détient le droit d'utiliser ces éléments. Le contenu de ce site Web est protégé par les lois sur les droits de propriété intellectuelle et les lois sur les droits d'auteur du Canada, des États-Unis, le Code civil du Québec et/ou d'autres pays. L'utilisation non autorisée du contenu peut constituer une violation des lois sur les droits d'auteur, des lois sur les marques de commerce, des lois sur les droits de propriétés intellectuelles ou d'autres lois.

Droit limité

Le CRIM vous accorde un droit limité d'afficher sur votre ordinateur, d'imprimer, de télécharger et d'utiliser toutes les informations accessibles au public contenues dans ce site, pourvu que ces informations soient utilisées à des fins légales. D'autres éléments du contenu (par exemple les logos, photographies, tableaux et autres éléments de ce site Web) vous sont fournis uniquement pour des fins personnelles et non commerciales, sujet à ce que : ces éléments ne soient pas modifiés, et la notice relative aux droits d'auteur du CRIM et le présent avis sur le droit limité soient incorporés et affichés sur votre copie du contenu.

Aucune disposition de la présente notice sur le droit limité ne doit être interprétée comme conférant un droit en vertu d'un droit d'auteur, d'une marque de commerce ou d'un autre droit de propriété intellectuelle du CRIM ou de toute tierce personne qui possède le droit d'auteur ou autre droit de propriété intellectuelle sur le contenu de ce site.

Exonération de garantie

Le CRIM n'offre aucune garantie quant à la qualité, l'exactitude ou l'exhaustivité de toute allégation, déclaration ou information contenue dans ce site. En outre, il ne fait aucune déclaration quant à l'adaptation de toute information contenue dans ce site, pour quelque fin que ce soit. Les informations sont fournies sur une base «telle quelle», sans aucune garantie ou condition de quelque nature que ce soit. Ce site peut contenir des inexactitudes ou des erreurs typographiques. Le CRIM ne peut en aucune circonstance être tenu responsable des dommages de toute nature, y compris les dommages spéciaux, indirects ou accessoires, résultant directement ou indirectement de l'utilisation du contenu disponible sur ce site. Vous convenez que le CRIM et tout tiers mentionné sur ce site ne seront pas responsables de toute perte ou de tout dommage, y compris des dommages directs, indirects, spéciaux ou accessoires ou des pertes de données ou des pertes de profits, même si le CRIM ou ce tiers a été avisé de la possibilité de tel dommage ou d'une telle perte.

Liens à d'autres sites

Ce site Web peut comporter des liens vers d'autres sites Web. Ces liens ne sont fournis que par souci de commodité et ne signifient pas que le CRIM approuve le contenu de ces sites Web. Le CRIM n'est aucunement responsable du contenu de tout autre site Web et ne fait aucune déclaration ni n'offre aucune garantie quant à ces sites Web, leurs contenus et leurs éléments. Si vous décidez d'accéder à un autre site Web, vous le faites à vos propres risques.

Généralités

Ce site est contrôlé et exploité par le CRIM à partir de ses bureaux situés dans la province de Québec, au Canada. Les présentes termes sont régis par les lois de la province de Québec et doivent être interprétées à la lumière desdites lois.

4A Liste des stopwords

au	pas	étées	soyez	avaient
aux	pour	étés	soient	eut
avec	qu	étant	fusse	eûmes
ce	que	suis	fusses	eûtes
ces	qui	es	fût	eurent
dans	sa	est	fussions	aie
de	se	sommes	fussiez	aies
des	ses	êtes	fussent	ait
du	son	sont	ayant	ayons
elle	sur	serai	eu	ayez
en	ta	seras	eue	aient
et	te	sera	eues	eusse
eux	tes	serons	eus	eusses
il	toi	serez	ai	eût
je	ton	seront	as	eussions
la	tu	serais	avons	eussiez
le	un	serait	avez	eussent
leur	une	serions	ont	ceci
lui	vos	seriez	aurai	cela
ma	votre	seraient	auras	cet
mais	vous	étais	aura	cette
me	c	était	aurons	ici
même	d	étions	aurez	ils
mes	j	étiez	auront	les
moi	l	étaient	aurais	leurs
mon	à	fus	aurait	quel
ne	m	fut	aurions	quels
nos	n	fûmes	auriez	quelle
notre	s	fûtes	auraient	quelles
nous	t	furent	avais	sans
on	y	sois	avait	soi
ou	été	soit	avions	
par	étée	soyons	aviez	

4B Éléments extraits de l'ontologie

URI :

L'URI est un identifiant unique de l'instance ou de la classe dans l'ontologie sous la forme d'une URL (exemple : http://www.inter2geo.eu/2008/ontology/ontology.owl#Represent_data). Cette valeur est collectée par la méthode getURI de l'individu amenée par OWLAPI.

Ce champ est obligatoirement présent une et une seule fois pour chaque document. Il contient une valeur unique.

URIweak :

L'uriweak est une valeur générée suite au découpage de l'URI (exemple : Represent_data) à l'aide de la méthode substring fourni par les librairies de la JVM. Cette valeur correspond à l'URL moins la chaîne http://www.inter2geo.eu/2008/ontology/ontology.owl#.

Ce champ est obligatoirement présent une et une seule fois pour chaque document. Il contient une valeur unique.

Ancestor :

La valeur « ancestor » correspond aux classes supérieures de l'individu. Pour cela, le ou les types de l'individu sont récupérés par la méthode getType d'OWLAPI dans une boucle sous la forme d'une description de classe de l'individu. Un raisonnement Pellet (méthode getAncestorClasses), sur cette description, récupère ensuite la liste des super-classes sous forme d'une collection d'où chacune est extraite pour être ajouté chacun comme un champ de type « ancestor » au document.

ontType :

Ce champ est obligatoirement présent au moins une fois pour chaque document. Le type de l'individu est obtenu à partir de la collection des super-classes. Une vérification est effectuée afin de vérifier si cette collection contient le terme « competency », « topic » ou « level », ce qui indique la super-classe la plus générale et donc le type de l'individu.

Le programme diffère ensuite en fonction du type. Si le type correspond à competency, il cherche les notions (topicRelation) et leurs ancêtres (ancestorTopic) en relation avec la capacité. Si le type correspond à une notion, une vérification sur la chaîne de caractère de l'individu permet d'enrichir le ontTypeComp en « concrete topic » ou « abstract topic » et « abstract topic with representative » si elle se termine par « _r ». Dans le cas d'un type « level », le programme cherche les relations avec les parcours (pathwaysRelation) et les régions (regionRelation) éducatifs.

Si le type ne correspond pas à un de ces 3 cas, l'individu n'est pas traité et le document XML est détaché de l'objet JDOM. Ce champ est obligatoirement présent une et une seule fois pour chaque document. Il contient une valeur unique.

ontTypeComp :

Le champ « ontTypeComp » précise le type de la notion. Une notion peut donc être de type « abstractTopic » et « abstractTopicWithRepresentative » ou « concreteTopic ».

topicRelation :

Ce champ est utilisé pour indexer une capacité. Il permet de lister l'ensemble des notions d'une capacité (relation *hasTopic*). Les valeurs sont obtenues à l'aide de la méthode getRelatedIndividuals fournie par le raisonneur Pellet. Une boucle sur cette méthode permet de lister tous les individus et de les ajouter unitairement au document.

Ce champ est uniquement présent dans un document si le type de l'individu correspond à une capacité et si cette dernière est en relation avec des notions.

ancestorTopic :

Ce champ est utilisé pour indexer une capacité. Il permet de lister l'ensemble des ancêtres des notions d'une capacité. Les valeurs sont obtenues à l'aide de la méthode getAncestorClasses fournie par le raisonneur Pellet.

Ce champ est uniquement présent dans un document si le type de l'individu correspond à une capacité et si cette dernière est en relation avec des notions ayant au moins un ancêtre.

pathwaysRelation :

Ce champ est utilisé pour indexer un niveau. Il permet de lister l'ensemble des parcours éducatifs en relation avec un niveau éducatif. Les valeurs sont obtenues à l'aide de la méthode getRelatedIndividuals fournie par le raisonneur Pellet. Une boucle sur cette méthode permet de lister tous les individus et de les ajouter unitairement au document.

Ce champ est uniquement présent dans un document si le type de l'individu correspond à un niveau et si celui-ci est en relation avec un parcours éducatif.

ancestorPathway :

Ce champ est utilisé pour indexer un niveau. Il permet de lister l'ensemble des ancêtres des parcours éducatifs d'un niveau éducatif. Les valeurs sont obtenues à l'aide de la méthode getAncestorClasses fournie par le raisonneur Pellet.

Ce champ est uniquement présent dans un document si le type de l'individu correspond à un niveau et si celui-ci est en relation avec un parcours éducatif ayant au moins un ancêtre.

regionRelation :

Ce champ est utilisé pour indexer un niveau. Il permet de lister l'ensemble des régions éducatives d'un niveau éducatif. Les valeurs sont obtenues à l'aide de la méthode getRelatedIndividuals fournie par le raisonneur Pellet. Une boucle sur cette méthode permet de lister tous les individus et de les ajouter unitairement au document.

Ce champ est uniquement présent dans un document si le type de l'individu correspond à un niveau et si celui-ci est en relation avec une région éducative.

ancestorRegion :

Ce champ est utilisé pour indexer un niveau. Il permet de lister l'ensemble des ancêtres des régions éducatives en relation avec un niveau éducatif. Les valeurs sont obtenues à l'aide de la méthode getAncestorClasses fournie par le raisonneur Pellet

Ce champ est uniquement présent dans un document si le type de l'individu correspond à un niveau et si celui-ci est en relation avec une région éducative ayant au moins un ancêtre.

urlForComped :

La valeur urlForComped est la concaténation des 2 chaînes de caractères urlComped (http://testbedc0.imag.fr/comped/show.html?uri=) et uriweak. urlComped est pré-renseigné par l'utilisateur dans le fichier param.properties utilisé par le programme.

Ce champ est obligatoirement présent une et une seule fois pour les documents concernant les notions et les capacités.

urlForLevel :

La valeur urlForLevel est la concaténation des 2 chaînes de caractères urlLevel (http://testbedc0.imag.fr:8080/ontologies/dev/individuals-dev/) et uriweak. urlLevel est pré-renseigné par l'utilisateur dans le fichier param.properties utilisé par le programme.

Ce champ est obligatoirement présent une et une seule fois pour les documents concernant les niveaux éducatifs.

defaultCommonName :

Ce champ est obtenu par l'intermédiaire de la méthode getDataPropertyAssertionAxioms fournie par OWLAPI. Le programme boucle sur l'ensemble des propriétés et n'extrait que les « noms communs par défaut » français. Cette boucle permet aussi de collecter les valeurs pour les champs :

- commonName ;
- unCommonName ;
- rareName ;
- falseFriendName.

Ce champ est obligatoirement présent une et une seule fois pour un document.

commonName :

Le programme n'extrait que les « noms communs » français. Si il n'y en a pas, le programme applique le defaultCommonName sur ce champ.

Ce champ est obligatoirement présent une et une seule fois pour un document.

unCommonName :

Le programme n'extrait que les « noms peu courants » français. Ce champ n'est pas obligatoirement présent dans un document. Il peut avoir une ou plusieurs valeurs.

rareName :

Le programme n'extrait que les « noms rares » français. Ce champ n'est pas obligatoirement présent dans un document. Il peut avoir une ou plusieurs valeurs.

falseFriendName :

Le programme n'extrait que les « noms faux-amis » français. Ce champ n'est pas obligatoirement présent dans un document. Il peut avoir une ou plusieurs valeurs.

4C ontoindex : Exemple d'un document competency

```
<doc>
      <field name="ancestor">Use_in_proportionality</field>
      <field name="ancestor">TransversalCompetency</field>
      <field name="ancestor">Use</field>
      <field name="ancestor">Apply_or_Use</field>
      <field name="ancestor">Use_in_calculations</field>
      <field
name="uri">http://www.inter2geo.eu/2008/ontology/ontology.owl#Use_cross_product_to_c
alculate_a_missing_value_in_a_proportion</field>
  <field
name="uriweak">Use_cross_product_to_calculate_a_missing_value_in_a_proportion</field
>
      <field name="topicRelation">Rational_Number</field>
      <field name="ancestorTopic">Topic</field>
      <field name="ancestorTopic">NamableBit</field>
      <field name="ancestorTopic">Number</field>
      <field name="ancestorTopic">Thing</field>
      <field name="topicRelation">Fraction</field>
      <field name="ancestorTopic">Topic</field>
      <field name="ancestorTopic">NamableBit</field>
      <field name="ancestorTopic">Number</field>
      <field name="ancestorTopic">Thing</field>
      <field name="topicRelation">Proportional</field>
      <field name="ancestorTopic">Topic</field>
      <field name="ancestorTopic">NamableBit</field>
      <field name="ancestorTopic">Relationship</field>
      <field name="ancestorTopic">Thing</field>
      <field name="ontType">competency</field>
      <field name="urlForComped">http://testbedc0.imag.fr/comped/show.html?
uri=Use_cross_product_to_calculate_a_missing_value_in_a_proportion</field>
      <field name="defaultCommonName">utiliser le produit en croix pour calculer une
quatrième proportionnelle</field>
      <field name="title">utiliser le produit en croix pour calculer une quatrième
proportionnelle</field>
      <field name="commonName">utiliser le produit en croix pour calculer une
quatrième proportionnelle</field>
</doc>
```

4D ontoindex : Exemple d'un document topic

```
<doc>
      <field name="ancestor">Intersect</field>
      <field name="ancestor">Incidence</field>
      <field name="ancestor">Topic</field>
      <field name="ancestor">NamableBit</field>
      <field name="ancestor">Relationship</field>
      <field name="ancestor">Thing</field>
      <field
name="uri">http://www.inter2geo.eu/2008/ontology/ontology.owl#Intersect_r</field>
      <field name="uriweak">Intersect</field>
      <field name="ontType">topic</field>
      <field name="ontTypeComp">abstractTopic</field>
      <field name="ontTypeComp">abstractTopicWithRepresentative</field>
      <field name="urlForComped">http://testbedc0.imag.fr/comped/show.html?
uri=Intersect</field>
</doc>
```

4E ontoindex : Exemple d'un document level

```
<doc>
      <field name="ancestor">EducationalLevel</field>
      <field name="ancestor">NamableBit</field>
      <field name="ancestor">Thing</field>
      <field
name="uri">http://www.inter2geo.eu/2008/ontology/ontology.owl#College_2</field>
      <field name="uriweak">College_2</field>
      <field name="ontType">level</field>
      <field
name="urlForLevel">http://testbedc0.imag.fr:8080/ontologies/dev/individuals-
dev/college_2.html</field>
      <field name="pathwaysRelation">College</field>
      <field name="ancestorPathway">EducationalPathway</field>
      <field name="ancestorPathway">Thing</field>
      <field name="regionRelation">France</field>
      <field name="ancestorRegion">EducationalRegion</field>
      <field name="ancestorRegion">Thing</field>
      <field name="defaultCommonName">Cinquième</field>
      <field name="title">Cinquième</field>
      <field name="commonName">5eme</field>
</doc>
```

4F SearchI2G vs Solr : Différences sur les champs d'un document de type level

Champs SearchI2G	Valeurs	Champs Solr	Valeurs
uri	http://www.inter2geo.eu/2008/ontology/ontology.owl#College_1	uri	http://www.inter2geo.eu/2008/ontology/ontology.owl#College_1
uriweak	College_1	uriweak	College_1
ontType	level	ontType	level
urlForNav	http://testbedc0.imag.fr:8080/ontologies/dev/individuals-dev/college_1.html	ontType	http://testbedc0.imag.fr:8080/ontologies/dev/individuals-dev/college_1.html
ancestor	EducationalLevel	ancestor	EducationalLevel
ancestor	NamableBit	ancestor	NamableBit
ancestor	Thing	ancestor	Thing
ancestor	College_1	-	-
ancestor	College	pathwaysRelation	College
ancestor	EducationalPathway	ancestorPathway	EducationalPathway
ancestor	Thing	ancestorPathway	Thing
ancestor	France	regionRelation	France
ancestor	EducationalRegion	ancestorRegion	EducationalRegion
ancestor	Thing	ancestorRegion	Thing
name-fr	Sixième	defaultCommonName	Sixième
name-fr	Sixième	-	-
name-fr	Sixième	commonName	6eme
name-x-all	Sixième	-	-
name-x-all	Sixième	-	-
name-x-all	Sixième	-	-
title	Sixième	-	-
skillItem	{""net.i2geo.api.SkillItem"":{""readableTitle"":""Sixième"",""numberOfMatchInStore"":""0"",""shortDescription"":""College_1"",""urlForNavigator"":""http:\/\/localhost:8080\/ontologies\/dev\/individuals\/College_1.html"",""uri"":""College_1"",""type"":""level"",""complete"":""false""}}	-	-

4G SearchI2G vs Solr : Différences sur les champs d'un document de type Topic

Champs SearchI2G	Valeurs	Champs Solr	Valeurs
uri	http://www.inter2geo.eu/2008/ontology/ontology.owl#Logarithmic_function	uri	http://www.inter2geo.eu/2008/ontology/ontology.owl#Logarithmic_function
uriweak	Logarithmic_function	uriweak	Logarithmic_function
ontType	topic	ontType	topic
ontType	abstractTopic	ontTypeComp	abstractTopic
ontType	abstractTopicWithRepresentative	ontTypeComp	abstractTopicWithRepresentative
urlForNav	http://testbedc0.imag.fr/comped/show.html?uri=Logarithmic_function	urlForComped	http://testbedc0.imag.fr/comped/show.html?uri=Logarithmic_function
ancestor	Logarithmic_function	ancestor	Logarithmic_function
ancestor	FunctionTopic	ancestor	FunctionTopic
ancestor	Topic	ancestor	Topic
ancestor	NamableBit	ancestor	NamableBit
ancestor	Function	ancestor	Function
ancestor	Thing	ancestor	Thing
ancestor	Logarithmic_function	-	-
name-en	logarithmic function	-	-
name-en	logarithmic function	-	-
name-es	función logarítmica	-	-
name-es	función logarítmica	-	-
name-fr	fonction logarithmique	defaultCommonName	fonction logarithmique
name-fr	fonction logarithmique	-	-
name-x-all	logarithmic function	-	-
name-x-all	logarithmic function	-	-
name-x-all	función logarítmica	-	-
name-x-all	función logarítmica	-	-
name-x-all	fonction logarithmique	commonName	fonction logarithmique
name-x-all	fonction logarithmique	-	-
title-en	logarithmic function	-	-
title-es	función logarítmica	-	-
title-fr	fonction logarithmique	-	-
skillItem-en	{""net.i2geo.api.SkillItem"":{""readableTitle"":""logarithmic function"",""numberOfMatchInStore"":""0"",""shortDescription"":""Logarithmic_function_r"",""urlForNavigator"":""http:\/\/localhost:8080\/comped\/show.html?uri=Logarithmic_function"",""uri"":""Logarithmic_function_r"",""type"":""topic"",""complete"":""false""}}	-	-
skillItem-es	SkillItem-en en espagnol	-	-
skillItem-fr	SkillItem-en en français	-	-

4H SearchI2G vs Solr : Différences sur les champs d'un document de type competency

Champs SearchI2G	Valeurs	Champs Solr	Valeurs
uri	http://www.inter2geo.eu/2008/ontology/ontology.owl#Manipulate_nets_of_solid_objects	uri	http://www.inter2geo.eu/2008/ontology/ontology.owl#Manipulate_nets_of_solid_objects
uriweak	Manipulate_nets_of_solid_objects	uriweak	Manipulate_nets_of_solid_objects
ontType	competency	ontType	competency
urlForNav	http://testbedc0.imag.fr/comped/show.html?uri=Manipulate_nets_of_solid_objects	urlForLevel	http://testbedc0.imag.fr/comped/show.html?uri=Manipulate_nets_of_solid_objects
ancestor	Manipulate_geometric_objects	ancestor	Manipulate_geometric_objects
ancestor	GeometricCompetency	ancestor	GeometricCompetency
ancestor	NamableBit	ancestor	NamableBit
ancestor	Competency	ancestor	NamableBit
ancestor	Thing	ancestor	Thing
ancestor	Manipulate_nets_of_solid_objects	-	-
ancestor	Net_of_a_polyhedron	topicRelation	Net_of_a_polyhedron
ancestor	Net_of_a_polyhedron	-	-
ancestor	Net_of_a_solid	ancestorTopic	Net_of_a_solid
ancestor	GeometricObject	ancestorTopic	GeometricObject
ancestor	Topic	ancestorTopic	Topic
ancestor	NamableBit	ancestorTopic	NamableBit
ancestor	Thing	ancestorTopic	Thing
ancestor	Right_parallelepiped	topicRelation	Right_parallelepiped
ancestor	Right_parallelepiped	-	-
ancestor	GeometricObject	ancestorTopic	GeometricObject
ancestor	Topic	ancestorTopic	Topic
ancestor	Polyhedron	ancestorTopic	Polyhedron
ancestor	NamableBit	ancestorTopic	NamableBit
ancestor	ThreeDObject	ancestorTopic	ThreeDObject
ancestor	Prism	ancestorTopic	Prism
ancestor	Solidobjects	ancestorTopic	Solidobjects
ancestor	Thing	ancestorTopic	Thing
name-en	manipulate nets of solid objects	-	-
name-en	manipulate nets of solid objects	-	-
name-fr	manipuler un patron d'un cube	defaultCommonName	manipuler un patron d'un cube
name-fr	manipuler un patron d'un cube	-	-
name-x-all	manipulate nets of solid objects	-	-
name-x-all	manipulate nets of solid objects	-	-
name-x-all	manipuler un patron d'un cube	-	-
name-x-all	manipuler un patron d'un cube	-	-
title-en	manipulate nets of solid objects	-	-
title-fr	manipuler un patron d'un cube	-	-
skillItem-en	{""net.i2geo.api.SkillItem"":{""readableTitle"":""manipulate nets of solid objects"",""numberOfMatchInStore"":""0"",""shortDescription"":""Manipulate_nets_of_solid_objects"",""urlForNavigator"":""http:\/\/localhost:8080\/comped\/show.html?uri=Manipulate_nets_of_solid_objects"",""uri"":""Manipulate_nets_of_solid_objects"",""type"":""competency"",""complete"":""false""}}	-	-
skillItem-fr	SkillItem-en en français	-	-

4I Code source sesasearch.js

```
$(function() {
       var url = 'http://' + window.location.host + '/solr/sesaindex/select';
       $("#sesasearch").autocomplete({
      minLength: 1,
      source: function(request, response) {
         $.ajax({
            url: url,
            //Query
            data: {
               q: 'id:'+request.term,
               fl: 'id, titre, description, nom',
               wt: 'json',
               omitHeader: 'true',
            },
            dataType: 'jsonp',
            jsonp: 'json.wrf',
            //Parse result
            success: function(data) {
               response($.map(data.response.docs, function(item) {
                  return {
                     id: item.id,
                     titre: item.titre,
                     desc: item.description,
                     type: item.nom
                  }
               }))
            },
         });
      },
      focus: function() {
         //Prevent value inserted on focus
         return false;
      },
      //Print selected item
      select: function(event, ui) {
         affiche(ui.item.id, ui.item.titre, ui.item.desc, ui.item.type);
         //Clear formula field
         event.preventDefault();
         $(this).val('');
      }
       //Display results
       }).data("autocomplete")._renderItem = function (ul, item) {
      return $("<li></li>")
         .data("item.autocomplete", item)
         .append('<a>' + item.id + ' - ' + item.titre + ' - ' + item.desc)
         .appendTo(ul);
       };
       function affiche(id, titre, description, type) {
      //Remove previous
      $('div').remove('.desc');
      $('div').remove('.id');
      $('div').remove('.titre');
      $('div').remove('.nom');
      //Add Ressource ID, Titre, Description, Type
      $('<div class="id"/>').text( id).appendTo("#id");
      $('<div class="titre"/>').text(titre).appendTo("#titre");
      $('<div class="desc"/>').text(description).appendTo("#desc");
```

```
            $('<div class="type"/>').text(type).appendTo("#type");
        };
});
```

4J Code source ontosearch.js

```
$(function() {
        var jsonArray = {"items": []};
        var capnotion = 'capnotion';
        var ancestor = 'ancestor';
        var url = 'http://' + window.location.host + '/solr/ontoindex/select';
        $("#ontosearch").autocomplete({
       minLength: 1,
       source: function(request, response) {
          $.ajax({
             url: url,
             //Query
             data: {
                q: 'name_auto:'+request.term+' OR name_auto:'+request.term+'*',
                fl: 'defaultCommonName, ontType, urlForNav, ancestor, topicRelation,
uriweak',
                wt: 'json',
                omitHeader: 'true',
                rows: 100,
             },
             dataType: 'jsonp',
             jsonp: 'json.wrf',
               //Parse result
             success: function(data) {
                response($.map(data.response.docs, function(item) {
                   return {
                      value: item.defaultCommonName,
                      img: item.ontType,
                      url: item.urlForNav,
                      topicRelation: item.topicRelation,
                      ancestor: item.ancestor,
                      uriweak: item.uriweak,
                   }
                }))
             },
          });
       },
       focus: function() {
          //Prevent value inserted on focus
          return false;
       },
       //Print selected item
       select: function(event, ui) {
          var dfn = ui.item.value;
          var uri = ui.item.uriweak;
          var anc = ui.item.ancestor;
          var rel = ui.item.topicRelation;
          var type = ui.item.img;
          //Concat relationTopics & ancestors
          if (rel) {
             anc = anc.concat(rel);
          };
          //Display defaultCommonName item
```

```
            $('<li/>').text(dfn).prependTo("#ontoresult");
            //Add item into json array
            $.each(uri, function(i, item) {
               var obj = {};
               obj[type] = dfn;
               obj[capnotion] = item;
               obj[ancestor] = anc;
               jsonArray.items.push(obj);
            });
            //Clear autocomplete form
            event.preventDefault();
            $(this).val('');
            //Remove a display item
            $('#ontoresult').selectable({
               selected: function (event, ui) {
                  $(ui.selected).remove();
                  //Remove item into json array
                  jQuery.each(jsonArray.items, function(i, val) {
                     for (var key in val){
                        if (val[key] == $(ui.selected).text()) {
                           jsonArray.items.splice(i, 1);
                           return false;
                        };
                     };
                  });
               }
            });
         }
         //Display results
          }).data("autocomplete")._renderItem = function (ul, item) {
         var img = '<img src="img/' + item.img + '.png" width="25" height="25" />';
         var link = 'onclick="window.open(this.href);';
         var urlfornav = '<a href="'+ item.url + '" '+ link + '"><img src="img/book.gif"
width="25" height="25"></a>'
         return $("<li></li>")
            .data("item.autocomplete", item)
            .append('<a>' + img + ' ' + item.value + ' ' + urlfornav)
            .appendTo(ul);
          };
          submitupdate(jsonArray);
});
```

4K Code source submitform.js

```
function submitupdate(jsonArray) {
      var url = 'http://' + window.location.host + '/index/';
      $('#form').on('submit', function() {
            //Add resource sesamath
            var $objID = $('div.id').eq(0).text();
            var $objTitre = $('div.titre').eq(0).text();
            var $objDesc = $('div.desc').eq(0).text();
            var $objType = $('div.type').eq(0).text();
            jsonArray.items.push({id: $objID, titre: $objTitre, description:
$objDesc, nom: $objType});
            //Call Ajax
            $.ajax({

                  url: 'updateSesaIndex.php',
                  type: 'POST',
                  datatype: 'json',
                  data: jsonArray,
                  //PHP response and redirect
                  success: function(data) {
                        alert(data);
                        window.location.replace(url);
                  }
            });
            //Avoid to execute the actual submit of the form.
            return false;
      });
}
```

4L Code source updateSesaIndex.php

```
<?php
$jsonIterator = $_POST[items];
$url = "http://" . $_SERVER["HTTP_HOST"] . "/solr/sesaindex/update/json?
commit=true";
$header = array("Content-type:Application/json; charset=utf-8");
$datainit = '{"add":{"doc":{';
$dataend = '}},"optimize": { "waitFlush":false, "waitSearcher":false }}';
$data = "";
      //Init JSON
      foreach ($jsonIterator as $key => $value) {
      foreach ($value as $key => $value) {
         if (!is_array($value)) {
            $data = $data.'"'.$key.'":"'.$value.'",';
         }
         foreach ($value as $key1 => $value) {
            $data = $data.'"'.$key.'":"'.$value.'",';
         }
      }
      }
//Remove last comma
$data = rtrim($data, ',');
//Concat postfield
$data = $datainit.$data.$dataend;
//Update document
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
```

```
curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
curl_setopt($ch, CURLINFO_HEADER_OUT, 1);

$result = curl_exec($ch);
      if (curl_errno($ch)) {
      print "curl_error: " . curl_error($ch);
      } else {
      echo 'Document mis a jour';
      }
curl_close($ch);
?>
```

4M Code source search.js

```
$(function() {
      var url = 'http://' + window.location.host;

      $("#search").autocomplete({
      minLength: 1,
      source: function(request, response) {
         $.ajax({
            url: url + '/solr/ontoindex/select',
            //Query
            data: {
               q: 'name_auto:'+request.term+' OR name_auto:'+request.term+'*',
               fl: 'defaultCommonName, ontType, urlForNav, uriweak',
               wt: 'json',
               omitHeader: 'true',
               rows: 100,
            },
            dataType: 'jsonp',
                jsonp: 'json.wrf',
            //Parse result
            success: function(data) {
               response($.map(data.response.docs, function(item) {
                  return {
                     value: item.defaultCommonName,
                     img: item.ontType,
                     url: item.urlForNav,
                     uriweak: item.uriweak,
                  }
               }))
            },
         });
      },
      focus: function() {
         //Prevent value inserted on focus
         return false;
      },
      //Print selected item
      select: function(event, ui) {
         var uri = ui.item.uriweak;
         var query = "?q=capnotion:"+uri+"%20OR%20ancestor:"+uri+
```

```
"&fl=id,titre,description,topic,competency,competencyProcess,nom&
                        wt=json&json.wrf=?&omitHeader=true";
         var queryresource = url + '/solr/sesaindex/select' + query;
         //Init table
         $("#table").html('<thead><tr><th>Id</th><th>Titre</th>

<th>Description</th><th>Type</th><th>Capacit&eacute;s/Notions</th>
                           <th>Actions</th></tr></thead>');
         //Get Sesamath resources
         $.getJSON(queryresource, function(result){
            $.each(result.response.docs, function(i, obj) {
               var capnot='';
               //Get capacités and notions
               for(var key in obj) {
                  if ((key == 'topic') || (key == 'competency') || (key ==
'competencyProcess')) {
                     $.each(obj[key], function(i, val) {
                        capnot= capnot + '<img src="img/' + key + '.png" width="25"
height="25" /> ' +
                                    val + '<br/>';
                     });
                  }
               }
               //Display resources with their id, titre, description, type, capacités
et notion, actions
               $
("<tbody/>").html("<tr><td>"+obj.id+"</td><td>"+obj.titre+"</td><td>"+
                     obj.description+"</td><td>"+obj.nom+"</td><td>"+capnot+"</td>
                     <td>afficher details modifier supprimer</td></tr>"
               ).prependTo('#table');
            });
         });
         //Clear autocomplete form
         event.preventDefault();
         $(this).val('');
      }
       //Display results
       }).data("autocomplete")._renderItem = function (ul, item) {
      var img = '<img src="img/' + item.img + '.png" width="25" height="25" />';
      var link = 'onclick="window.open(this.href);';
      var urlfornav = '<a href="'+ item.url + '" '+ link + '"><img src="img/book.gif"
width="25" height="25"></a>'
      return $("<li></li>")
         .data("item.autocomplete", item)
         .append('<a>' + img + ' ' + item.value + ' ' + urlfornav)
         .appendTo(ul);
       };
});
```

5A Modèle Conceptuel de Données de Bibli

5B Fonction filtreResultsAction

```
/**
   * Lance une recherche avec des filtre sur SolR (sesaindex) et affiche les
résultats
   *
   * @return array Arguments pour la vue
   * @Route("/results", name="solr_filtreResults")
   * @Template("SesamathBibliBundle:Ressource:filtresolr.html.twig")
   */
  public function filtreResultsAction()
  {
    $this->get('sso_service'); // init de la session SSO
    // en post, c'est
    // $params = $this->getRequest()->request;
    // mais en get faut utiliser
    $params = $this->getRequest()->query;

    // On récupère les données du formulaire de recherche
    $dfn = $params->get('dfn');
    $type = $params->get('filtreTypeSesa');
    $niveau = $params->get('filtreNiveau');

    // le form en a besoin pour mettre éventuellement en cache ses listes
    $em = $this->getDoctrine()->getEntityManager();
    // les variables de la vue
    $viewArgs = array();
    if (!empty($dfn)) {
      // C'est une recherche sur une compétence
      // On récupère les critères de recherche pour les afficher dans la vue des
résultats
      $criteres['dfn'] = $dfn;
      $criteres['Niveau'] = $niveau;
      $criteres['Type'] = $type;
```

```
        // Requète SolR
        $this->initSolRService();
        $select = $this->solr->createSelect();
        // On récupère les ID
        $query = null;
        foreach (json_decode($dfn)->items as $elem) {
          foreach ($elem as $field => $uri) {
            // Si la recherche ce fait sur une capacité/notion
            if (($query == null) && ($field == 'uri')) {
              $query = '(capnotion:'.$uri.' OR ancestor:'.$uri.')';
            } elseif (($query != null) && ($field == 'uri')) {
              $query = $query.' AND (capnotion:'.$uri.' OR ancestor:'.$uri.')';
            }
          }
        }
        //On complète la requête
        $select->setQuery($query)
            ->setStart(0)
            ->setRows(self::MAX_PAGE_RESULTS);

        // Si un filtre "niveau" a été sélectionné
        if ($niveau != '') {
          $niveau = explode(",", $niveau);
          for ($i=0; $i < count($niveau); $i++) {
            if ($i == 0) {
              $requestNiveau = 'niveau:"'.$niveau[$i].'"';
            } else {
              $requestNiveau .= ' OR niveau:"'.$niveau[$i].'"';
            }
          }
          $select->createFilterQuery('level')->setQuery($requestNiveau);
        } else {
          $requestNiveau ="-niveau:[* TO *]";
          $select->createFilterQuery('level')->setQuery($requestNiveau);
        }

        // Si un filtre "typeSesa" a été sélectionné
        if ($type != '') {
          $type = explode(",", $type);
          for ($i=0; $i < sizeof($type); $i++) {
            if ($i == 0) {
              $requestType = 'typeSesa:"'.$type[$i].'"';
            } else {
              $requestType = $requestType.' OR typeSesa:"'.$type[$i].'"';
            }
          }
          $select->createFilterQuery('type')->setQuery($requestType);
        } else {
          $requestType ="-typeSesa:[* TO *]";
          $select->createFilterQuery('type')->setQuery($requestType);
        }

        // Champs retournés
        $select->setFields(
          array('id', 'titre', 'resume', 'description', 'typeSesa', 'niveau', 'topic',
'competency')
        );

        // Résultat
```

```
      $solRresults = $this->solr->select($select);

      // Traitement du résultat
      $results = array();
      $nbTot = 0; // l'index courant dans les résultats
      foreach ($solRresults as $document) {
        // the documents are also iterable, to get all fields
        foreach ($document as $field => $value) {
          // On récupère les champs uniques (ID)
          if (!is_array($value)) {
            $results[$nbTot][$field] = $value;
          } else {
            //On récupère les autres champs
            foreach ($value as $value2) {
              //On traite les notions et capacités
              if (($field == 'topic') || ($field == 'competency')) {
                $results[$nbTot]['capnotion'][] = array ($field => $value2);
              } elseif ($field == 'niveau') {
                $results[$nbTot]['niveau'][] = array ($field => $value2);
              } else {
                $results[$nbTot][$field] = $value; // on laisse le tableau initial ?
              }
            }
          }
        }
        $nbTot++;
      }

      if (isset($results)) {
        // On retourne le résultat de la recherche filtrée
        $viewArgs['filtreresult'] = $results;
        // On retourne les critères de recherche
        $viewArgs['filtrecriteres'] = $criteres;
        //var_dump($viewArgs['filtrecriteres']);
        //exit();
      } else {
        $viewArgs['filtreresult'] = array();
        $viewArgs['filtrecriteres'] = array();
      }
    }

    return $viewArgs;
  }
```

5C Test Tsung

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tsung SYSTEM "/usr/share/tsung/tsung-1.0.dtd">
<!-- Accès par requête HTTP via proxy/cache NGINX sur ontoindex et sesaindex
- utilise les préfixes des capacités/notions renseignés dans wordcut.list
- utilise l'uriweak des capacités/notions renseignés dans CapNot.list -->
<tsung loglevel="info" version="1.0">

    <clients>
        <client host="localhost" use_controller_vm="true" maxusers="10000" />
    </clients>

    <servers>
      <server host="localhost" port="80" type="tcp"/>
    </servers>

    <!-- Montée en charge des connexions -->
    <load>
        <!-- 1 user / 2s sur 3min => 90 users -->
        <arrivalphase phase="1" duration="3" unit="minute">
            <users interarrival="2" unit="second"/>
        </arrivalphase>
        <!-- 20 utilisateurs / 3s (400/min) sur 15 minutes => 6000 -->
        <arrivalphase phase="2" duration="15" unit="minute">
            <users interarrival="0.30" unit="second"/>
        </arrivalphase>
    </load>

    <options>
        <option type="ts_http" name="user_agent">
            <user_agent probability="100">Test tsung de sesamath
(tech@sesamath.net)</user_agent>
        </option>
        <!-- Déclaration du fichier des préfixes de notions/capacités pour la
requête d'ontoindex -->
         <option name="file_server" id='wordcutfile'
value="./randomFiles/wordcut.list"></option>
        <option name="file_server" id='capnotfile'
value="./randomFiles/CapNot.list"></option>
        <option name="file_server" id='niveaufile'
value="./randomFiles/filtreNiveaux.list"></option>
        <option name="file_server" id='typefile'
value="./randomFiles/filtreTypes.list"></option>
    </options>

    <!-- Le test -->
    <sessions>
    <session name='rec20130125-1447' probability='100'  type='ts_http'>

        <!-- Accès machine Tomcat en direct -->
        <request><http url='http://ressources.devsesamath.net/' version='1.1'
method='GET'></http></request>
        <for from="1" to="150" incr="1" var="index">
            <setdynvars sourcetype="file" fileid="wordcutfile" delimiter=";"
order="random">
                <var name="prefix" />
            </setdynvars>
            <setdynvars sourcetype="file" fileid="capnotfile" delimiter=";"
order="random">
                <var name="uriweak" />
            </setdynvars>

            <thinktime random='true' value='3'/>

            <!-- 100 requêtes par utilisateur soit 120.000 requêtes en 10 minutes
```

```
-->
            <request subst="true">
                <!-- <dyn_variable name="uriweak" re='"uriweak":"([^"]*)"' /> -->
                <http url='/solrcache/ontoindex/select?
json.wrf=jQuery18208805036243455864_1359125412718&amp;q=name_auto%3A%_prefix%%
+OR+name_auto%3A%_prefix%%*&amp;fl=defaultCommonName%2C+ontType%2C+urlForNav
%2C+uriweak&amp;wt=json&amp;omitHeader=true&amp;rows=10&amp;_=1359125416288'
version='1.1' method='GET'></http>
            </request>

            <thinktime random='true' value='2'/>

            <!-- 100 requêtes par utilisateur soit 120.000 requêtes en 10 minutes
-->
            <request subst="true">
                <http url='/solrcache/sesaindex/select?q=capnotion%3A%_uriweak%%
+OR+ancestor%3A%_uriweak%%&amp;fl=id%2C+titre%2C+resume%2C+description%2C+typeSesa
%2C+niveau%2C+topic%2C+competency&amp;start=0&amp;rows=100&amp;wt=json'
version='1.1' method='GET'></http>
            </request>
            <!-- 240.000 requêtes passées au proxy/cache NGINX -->
        </for>
    </session>
    </sessions>
</tsung>
```

Glossaire

AJAX : (Asynchronous JavaScript and XML), combinaison des technologies JavaScript, CSS, XML, DOM et XMLHttpRequest permettant de construire des applications Web à l'ergonomie enrichie.

Algorithme de Levenshtein : La distance de Levenshtein mesure le degré de similarité entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre. C'est une distance au sens mathématique du terme, donc en particulier c'est un nombre positif ou nul, et deux chaînes sont identiques si et seulement si leur distance est nulle.

API : (Application Programming Interface), est une interface fournie par un programme informatique. Elle permet l'interaction des programmes les uns avec les autres, de manière analogue à une interface homme-machine.

CSS : (Cascading Style Sheets), langage informatique qui servant à décrire la présentation des documents HTML et XML.

CSV : (Comma-separated values), est un format informatique ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules.

DOM : (Document Object Model), est une recommandation du W3C décrivant une interface standard permettant de manipuler la structure ou le style de documents XML et HTML.

EIAH : (environnements informatiques pour l'apprentissage humain), environnements informatiques ayant pour objectifs de favoriser ou susciter des apprentissages, de les accompagner et de les valider.

Espace vectoriel de recherche d'information : Le modèle de l'espace vectoriel sert de base à la représentation des données textuelles par des vecteurs dans l'espace euclidien. Selon ce modèle, l'élément sémantique de chaque document est le terme. Un terme peut être un mot simple ou un mot composé (un groupe de mots). À partir de cette caractéristique, chaque document est représenté par un vecteur des termes.

HTML : (Hypertext Markup Language), langage dérivé de SGML et conçu pour écrire des page Web.

HTTP: (HyperText Transfer Protocol), protocole de transfert de données hypertextes utilisées dans les sites Web.

IHM : (interactions homme-machine), moyens et outils mis en œuvre afin qu'un humain puisse contrôler et communiquer avec une machine.

JDBC : (Java DataBase Connectivity), est une interface de programmation créée par Sun Microsystems permettant aux applications Java d'accéder à des bases de données.

JSON : (JavaScript Object Notation), est un format de données textuelles, permettant de représenter de l'information structurée.

Lemmantisation : La lemmatisation est le nom du procédé en traitement automatisé de la langue qui consiste à transformer les flexions en leur lemme. Les flexions sont les différentes formes fléchies d'un même mot. Les formes fléchies correspondent aux formes « conjuguées » ou « accordées » d'un mot de base non conjugué et non accordé : le lemme.

Modèle booléen : Un modèle booléen est une méthode ensembliste de représentation du contenu d'un document. C'est l'un des premiers modèles utilisés en recherche d'information, permettant de fouiller automatiquement les grand corpus de bibliothèques.

MVC : (Modèle-vue-contrôleur), est un modèle d'architecture qui cherche à séparer nettement les couches de présentation, métier et d'accès aux données.

Ontologie : Une ontologie est un ensemble structuré des termes et concepts représentant le sens d'un champ d'informations, que ce soit par les métadonnées d'un espace de noms ou les éléments d'un domaine de connaissances. L'ontologie constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Elle est employée pour raisonner à propos des objets du domaine concerné.

ORM : (Object Relational Mapping), technique de programmation faisant le lien entre le monde de la base de données et le monde de la programmation objet. Elle permet de transformer une table en un objet facilement manipulable via ses attributs.

OWL : (Web Ontology Language), est un langage de représentation des connaissances construit sur le modèle de données de RDF. Il fournit les moyens pour définir des ontologies web structurées.

REST : (REpresentational State Transfer), est un patron de conception pour l'implémentation de système connecté permettant l'exposition de ressources sur le web.

Stemmatisation : La stemmatisation ou racinisation est le nom donné au procédé qui vise à transformer les flexions en leur radical ou stemme. La racine d'un mot correspond à la partie du mot restante une fois que l'on a supprimé son préfixe et son suffixe, à savoir son radical. Elle est aussi parfois connu sous le nom de stemme d'un mot. Contrairement au lemme qui correspond à un mot réel de la langue, la racine ou stemme ne correspond généralement pas à un mot réel.

SSO : (Single Sign-On), est une méthode permettant à un utilisateur de ne procéder qu'à une seule authentification pour accéder à plusieurs applications informatiques (ou sites web sécurisés).

SSL : (Secure Sockets Layer), est un Protocole de sécurisation des communications créé par Netscape.

TICE : (Technologies de l'Information et de la Communication pour l'Enseignement), recouvrent les outils et produits numériques pouvant être utilisés dans le cadre de l'éducation et de l'enseignement.

URI : (Uniform Resource Identifier), chaîne de caractères utilisée pour identifier un nom ou une ressource sur Internet.

URL : (Uniform Ressource Locator), identifiant unique d'une ressource sur Internet, en général vers une page web.

XML : (Extensible Markup Language), est un langage informatique de balisage, très utilisé pour stocker ou transférer des données structurées en champs arborescents.

Bibliographie

[ANT 2012] : Apache Ant™ 1.8.2 Manual.

http://ant.apache.org/manual/index.html, visité le 22 août 2012.

[APPFUSE 2012] : AppFuse QuickStart.

http://appfuse.org/display/APF/AppFuse+QuickStart, visité le 11 juillet 2012.

[COMPED 2009] DESMOULINS Cyrille, LIBBRECHT Paul. Comped, a Web - based Competency Ontology Editor for Dynamic Geometry

http://www.hoplahup.net/copy_left/Desmoulins&Libbrecht%20AIEDWS_Vol2_SWEL09Proceedings.pdf, visité le 19 avril 2012.

[CRIM 2012] : Solr et ElasticSearch dans un contexte de mise à l'échelle.

http://www.crim.ca/media/publication/fulltext/rapport_solr_elastic_search.pdf, visité le 22 août 2012.

[GENTAZ 2011] : GENTAZ Dominique. Technologies du Web Sémantique pour la représentation des compétences scolaires. Rapport technique, CNAM Grenoble, Juin 2011.

[GROOVY 2012] : Groovy User Guide.

http://groovy.codehaus.org/User+Guide, visité le 11 juillet 2012.

[GWT 2012] : Google Web Toolkit Overview.

https://developers.google.com/web-toolkit/overview, visité le 11 juillet 2012.

[I2GEO 2009] : EGIDO Santiago, LIBBRECHT Paul, LESOURD Henri. Platform's Administration Manual. 2009.

http://svn.activemath.org/intergeo/Deliverables/WP4/D4.4-AdminManual/D4.4-AdminManual.pdf, visité le 10 avril 2012.

[I2GEO 2012] : Libbrecht Paul. i2geo Platform Overivew. Avril 2012.

http://draft.i2geo.net/static/paultmp/i2geo-platform-ODS.pdf, visité le 19 avril 2012.

[INSTRUMENPOCHE 2012] : Wiki Instrumenpoche.

http://instrumenpoche.sesamath.net/wiki/doku.php, visité le 12 avril 2012.

[JDOM 2012] : Documentation JDOM.

http://www.jdom.org/downloads/docs.html, visité le 29 août 2012.

[JQUERY 2012] : Documentation Jquery.

http://docs.jquery.com/, visité le 8 octobre 2012.

[JQUERY UI 2012] : Documentation Jquery UI.

http://api.jqueryui.com/, visité le 8 octobre 2012.

[JSTREE 2013] : Documentation Jstree.

http://www.jstree.com/documentation, visité le 15 janvier 2013.

[LIG 2012] : Site Internet du LIG.

http://www.liglab.fr/, visité le 02 avril 2012.

[LABOMEP 2012] : LABOMEP : logiciel pour le professeur de mathématiques et ses élèves.

http://www.labomep.net/fiches, visité le 12 avril 2012.

[LUCENE 2012] : Documentation Lucene.

http://lucene.apache.org/core/documentation.html, visité le 16 juillet 2012.

[MAVEN 2009] : John Casey, Brian Fox, Thomas Locher, Tim O'Brien, Jason Van Zyl, Juven Xu. Maven The définitive guide. Edition O'Reilly.

http://www.silverpeas.org/maven-guide.html, visité le 4 mai 2012.

[METAH 2012] : Site Internet de l'équipe METAH.

https://metah.imag.fr/, visité le 02 avril 2012.

[NELMIO 2012] : NelmioSolariumBundle README.

https://github.com/nelmio/NelmioSolariumBundle/blob/master/README.md, visité le 4 novembre 2012.

[OWLAPI 2012] : Documentation d'OWL API.

http://owlapi.sourceforge.net/documentation.html, visité le 20 avril 2012.

[OPENVZ 2009] : DOSTES Thierry, LIBES Maurice. Virtualisation avec openVZ. Journées thématiques SIARS DR12 CNRS. Septembre 2009.

http://cesar.com.univ-mrs.fr/IMG/pdf/jtsiars-openvz_1_.pdf, visité le 23 avril 2012.

[PHP 2012] : Manuel PHP.

ttp://php.net/manual/fr/, visité le 03 octobre 2012.

[SACOCHE 2012] : A propos de SACoche.

https://sacoche.sesamath.net/index.php?dossier=presentation&fichier=accueil__a_propos, visité le 12 avril 2012.

[SESAMATH 2012] : Site Internet de l'association Sésamath.

http://www.sesamath.net/, visité le 02 avril 2012.

[SESAPROF 2012] : SESAPROF : espace de Sésamath pour les professeurs.

http://sesaprof.sesamath.net, visité le 12 avril 2012.

[SOLARIUM 2012] : Solarium 2.x manual.

http://wiki.solarium-project.org/index.php/Solarium_2.x_manual, visité le 4 novembre 2012.

[SOLR 2012] : Apache Solr Wiki.

http://wiki.apache.org/solr/, visité le 20 août 2012.

[SYMFONY2 2012] : The Book for Symfony 2.1.

http://symfony.com/fr/doc/current/book/index.html, visité le 20 novembre 2012.

[TOMCAT 2012] : Apache Tomcat 6.0 version 6.0.35 du 28 novembre 2011.

http://tomcat.apache.org/tomcat-6.0-doc/index.html, visité le 29 mai 2012.

TRACENPOCHE 2012] : Présentation de TracenPoche.

http://tracenpoche.sesamath.net/spip.php?article1, visité le 12 avril 2012.

[TSUNG 2013] : Tsung User's manual.

http://tsung.erlang-projects.org/user_manual.html, visité le 06 février 2013.

[TWIG 2012] : Twig Documentation.

http://twig.sensiolabs.org/documentation, visité le 4 février 2012.

[VELOCITY 2012] : Velocity User Guide 1.5.

http://velocity.apache.org/engine/releases/velocity-1.5/translations/user-guide_fr.html, visité le 11 juillet 2012.

MÉMOIRE D'INGÉNIEUR C.N.A.M. en INFORMATIQUE

Développement d'un système d'indexation par les compétences avec les technologies du Web sémantique pour Sésamath

Dominique Gentaz **Grenoble, le 27 mai 2013**

Résumé

L'association Sésamath a pour vocation essentielle de mettre à disposition de tous, gratuitement, des ressources pédagogiques libres et des outils professionnels libres utilisés pour l'enseignement des mathématiques via Internet. Dans son rôle de diffuseur, Sésamath tente de simplifier l'accès à une multitude de ressources, de promouvoir les échanges autour des pratiques pédagogiques. L'infrastructure Sésamath basée sur une architecture Web 2.0, permet une indexation des données par mots-clés. La pertinence de la recherche et de la réutilisabilité des ressources commencent à poser problème avec l'accroissement des données.

Ce document propose une étude détaillée du système d'indexation des ressources par les compétences réalisé dans le cadre du projet européen « Intergeo ». Les composants de ce système d'indexation basé sur l'API Lucene ne pouvant être réutilisés en l'état, les principes mis en œuvre sont repris et portés sur le système d'indexation Solr. Les clients Web permettant d'effectuer des recherches dans les index sont ensuite intégrés sur le site Bibli. Bibli est la bibliothèque, en cours de développement chez Sésamath, référençant l'ensemble des ressources.Ce mémoire présente ensuite les évolutions fonctionnelles au niveau des principes de recherche et les tests de performance du système implémenté.

Mots-clés : indexation, Intergeo, Apache Lucene, ontologie, pertinence, Apache Solr, ressources, moteur de recherche

---

Abstract

The Sésamath association has primarily intended to make available to all, free of charge, open educational resources and professional free tools used to the teaching of mathematics via the Internet. In his role as a broadcaster, Sésamath attempts to simplify access to a multitude of resources, promote exchanges about teaching practices. Sésamath infrastructure based on a Web 2.0 architecture allows indexing data by keywords. The relevance of the research and the reusability of resources becoming a problem with the growth of data.

This document provides a detailed study of the indexing system resources by skills performed within the European project « Intergeo ». The components of this indexing system, based on Lucene API, can not be reused in the state, the principles implemented are resumed and carried on the Solr indexing system. The Web clients enabling to search in the indexes are then integrated on the site Bibli. Bibli is the library, in development at Sésamath, referencing all ressources. This paper presents then the functional changes in the principles of research and testing performance of the implemented systeml.

Keywords : indexing, Intergeo, Apache Lucene, ontology, scoring, Apache Solr, ressources, Search engine