

Document d'architecture général

Version : 1.0

Date : 29 Décembre 2013

Rédigé par : Jonathan Lemonsu, Yohann Hay,
Clément Jehannet,
Kevin Cauchois,
Hacène Ingrachen,
André Charleroy, Mohammed Amine Laoyene

Relu par : MOE

Approuvé par : Membres de la MOA

MISES À JOUR

Version	Date	Modifications réalisées
0.1	26/11/2013	Création du document
0.2	28/11/2013	Mise en place de l'objet, des documents applicables et de la terminologie
0.3	30/11/2013	Début de la rédaction du contenu
0.4	02/12/2013	Détail de l'architecture matérielle
0.5	03/12/2013	Description des différents constituants
0.6	04/12/2013	Intégration des diagrammes de séquence généraux permettant d'identifier les liens entre les portlets et les services
0.7	05/12/2013	Justification technique et travail sur la chaîne de lancement
0.8	06/12/2013	Relecture et mise en place de la partie Traçabilité
0.9.1	27/12/2013	Intégration du DAL de l'équipe verte
0.9.2	29/12/2013	Intégration des DAL des autres équipes
1.0	29/12/2013	Finalisation, Correction des parties des lots, Traçabilité

Table des matières

I Objet.....	7
A Objet du projet	7
B Objet du document.....	7
II Documents applicables et de référence	8
III Terminologie et sigles utilisés	9
A Terminologie.....	9
B Les signes utilisés	10
IV Configuration requise	12
A Serveur dédié aux services web et à la persistance	12
B Serveur dédié à Petals	12
C Serveur dédié à Liferay	12
D Environnement de test et de production.....	13
E Développement.....	13
F Configuration côté client.....	13
V Architecture générale du démonstrateur.....	14
A Architecture générale	14
B Détails des composants	15
1 Description du constituant Liferay	15
2 Description du constituant Petals ESB	15
3 Description du constituant Collecte.....	16
4 Description du constituant Extraction	16
5 Description du constituant Classification.....	16
6 Description du constituant Taxonomie.....	16
7 Description du constituant Indexation	17
8 Description du constituant Recherche.....	17
9 Description du constituant Alerte.....	17
C Gestion des données de l'application (Clément Jehannet)	17
1 La base de données.....	17
2 Le système de gestion de fichiers	19
3 La gestion des fichiers de log	20
D Diagramme de paquetages.....	21
E Le paquetage Manager : gestion de l'application.....	21
F Fonctionnement dynamique	22
1 Éditer les images	22
2 Gérer les alertes	22

3 Rechercher des images	23
4 Gérer les éditions des utilisateurs.....	23
5 Supprimer une image.....	24
6 Uploader une image.....	24
7 Gérer les paramètres de collecte	25
8 Éditer la taxonomie	26
9 Gérer les moteurs de classification.....	26
10 Évaluer un moteur de classification	27
11 Gérer les utilisateurs	27
12 Accéder aux logs d'un composant	28
13 La chaîne de traitement	28
G Justifications techniques	28
VI Architecture du lot de collecte et d'extraction (Clément Jehannet)	30
A Architecture statique	30
1 Description du constituant Collecte.....	30
2 Description du constituant Extraction	30
3 Description du constituant Alertes	30
4 Description du constituant Log	31
B Fonctionnement dynamique.....	31
1 Effectuer une collecte sur un site web.....	31
2 Extraire les métadonnées d'une image.....	32
3 Traiter les alertes	32
C Justifications techniques.....	33
VII Architecture du lot de classification et de taxonomie (Kevin Cauchois)	34
A Architecture statique	34
1 Description détaillée du composant Taxonomie	34
2 Description détaillée du composant Classification	36
3 Description détaillée du composant Evaluation	39
B Fonctionnement dynamique.....	40
1 Opération de lecture sur la taxonomie	40
2 Opération d'écriture sur la taxonomie	41
3 Ajouter des images à un modèle.....	41
4 Classer une image	43
5 Évaluer un modèle de classification.....	44
C Justifications techniques.....	45
1 Lire.....	45
2 Lucene	45

3 Skos API	45
VIII Architecture du lot d'indexation (Hacène Ingrachen)	46
A Architecture statique	46
1 Indexation des images	46
2 Indexation des métadonnées extraites des images.....	47
3 Composant de recherche	47
B Fonctionnement dynamique.....	48
1 Recherche d'images par similarité des caractéristiques.....	48
2 Recherche d'images par métadonnées.....	49
3 Indexation des images	50
4 Indexation et mise à jour des métadonnées.....	50
5 Ajout de filtre sur les métadonnées dans l'index	51
6 Supprimer une image.....	51
7 Recherche par filtre.....	52
C Justifications techniques.....	52
IX Architecture du lot d'interfaçage (André Charleroy, Mohamed Amine)	53
A Architecture statique	53
1 Détail des pages par type d'utilisateur	53
2 Dossier de maquettage	55
3 Description du composant Interfaçage.....	70
4 Description de la portlet P_LogViewer	71
5 Description de la portlet P_UserManagement	71
6 Description de la portlet P_UserEditions	71
7 Description de la portlet P_CollectConfiguration	71
8 Description de la portlet P_CollectMonitoring	71
9 Description de la portlet P_ModelConfiguration.....	71
10 Description de la portlet P_KitConfiguration	72
11 Description de la portlet P_ModelEvaluation	72
12 Description de la portlet P_TaxonomyConfiguration	72
13 Description de la portlet P_RecentImages.....	72
14 Description de la portlet P_ImageBrowse	72
15 Description de la portlet P_ImageSearch.....	72
16 Description de la portlet P_AlertConfiguration	73
17 Description de la portlet P_ImageDetail	73
18 Description de la portlet P_ImageUpload.....	73
B Fonctionnement dynamique.....	73
1 Consulter les logs : P_LogViewer	73

2 Gérer les utilisateurs : <i>P_UserManagement</i>	74
3 Gérer les éditions des utilisateurs : <i>P_UserEditions</i>	76
4 Gérer les paramètres de collecte : <i>P_CollectConfiguration</i>	78
5 Consulter l'historique des collectes : <i>P_CollectMonitoring</i>	81
6 Gérer les modèles de classification : <i>P_ModelConfiguration</i>	81
7 Gérer les kits d'évaluations : <i>P_KitConfiguration</i>	84
8 Gérer l'évaluation des modèles de classification : <i>P_ModelEvaluation</i>	86
9 Gérer la taxonomie : <i>P_TaxonomyConfiguration</i>	87
10 Gérer les images récentes : <i>P_RecentImages</i>	89
11 Naviguer dans le corpus d'images : <i>P_ImageBrowse</i>	89
12 Rechercher des images : <i>P_ImageSearch</i>	90
13 Gérer les alertes : <i>P_AlertConfiguration</i>	91
14 Avoir les informations d'une image : <i>P_ImageDetail</i>	92
15 Uploader une image : <i>P_ImageUpload</i>	93
C Justifications techniques.....	93
1 Présentation du Framework JSF.....	93
2 Composants du Framework JSF	93
3 Avantages du Framework JSF.....	93
4 Inconvénients du Framework JSF.....	94
5 Présentation de la bibliothèque PrimeFaces de JSF	94
6 Conclusion.....	96
X Vue d'ensemble du démonstrateur SPORTIFS.....	97
XI Traçabilité des exigences	98

I Objet

A Objet du projet

Le projet SPORTIFS (Système Pour l'Organisation, la Recherche et le Traitement d'images de Flux Sportifs) vise à développer un prototype de système dédié à la gestion d'images sportives. L'objectif est de permettre la constitution d'une banque d'images portant sur des sports divers et d'automatiser les différentes tâches d'acquisition, de traitement et d'exploitation de ces documents.

Le projet SPORTIFS vise à développer des fonctions :

- de collecte d'images de sports sur Internet ou dans des bibliothèques d'images personnelles.
- d'organisation et de classement semi-automatique des images au sein d'une taxonomie visuelle des sports qui sera à définir au cours du projet.
- de recherche d'images à la fois sur leurs métadonnées (nom de fichiers, taille du fichier, etc.) mais aussi sur des caractéristiques intrinsèques aux images (recherche par similarité visuelle) et sur des informations de contexte de l'image (légende dans une page web, texte environnant...).
- de veille et d'alerte permettant à un utilisateur d'être notifié de la publication d'une nouvelle image répondant à ses critères.
- d'évaluation des systèmes de classement semi-automatique d'images.

B Objet du document

Ce document fournit une vue globale de l'architecture du logiciel, de haut niveau, grâce à un certain nombre de vues qui décrivent les différents aspects du logiciel. Ces vues montrent les décisions significatives prises concernant l'architecture du logiciel qui influenceront la conception.

L'objectif de ce document est de décrire la macro architecture du projet ARMADA. Cette macro architecture définit les services, les chaînes de traitements ainsi que les portlets nécessaires. Les interfaces offertes par les différents services seront détaillées ici et permettront de vérifier que l'ensemble des exigences exprimées dans la Spécification Technique de Besoin sont bien traitées. L'architecture proposée devra respecter le modèle WebLab. Pour ce faire, nous nous appuierons sur une architecture orientée services et l'utilisation d'un portail web avec plusieurs portlets qui interrogeront ces services.

Ce document s'adresse à la Maîtrise d'Ouvrage qui validera les choix, à la Maîtrise d'œuvre en vue de la rédaction de son Cahier de Recettes ainsi qu'aux architectes des différentes équipes qui auront en charge la rédaction de leurs Documents d'Architecture Logicielle.

II Documents applicables et de référence

Le document de référence de ce document est :

- SPORTIFS_CCTP_Ed01-Rev00.pdf
- Plan de management v 1.0
- Spécification technique des besoins v 1.0
- BAFO de l'équipe rouge
- Notices techniques

III Terminologie et sigles utilisés

A Terminologie

Annotation : Activité de production de la vérité-terrain et résultat de cette activité. L'activité d'annotation consiste à analyser « manuellement » le contenu d'un document et à étiqueter ou à transcrire les éléments qui y figurent. Les objets annotations qui résultent de ce travail contribuent à la définition de la vérité-terrain.

Corpus : Ensemble de documents (image dans le cas du projet SPORTIFS) regroupés dans une optique précise. Il faut distinguer:

- le corpus d'apprentissage, qui sert à constituer le modèle de fonctionnement sur lequel repose un service en s'appuyant sur un nombre suffisant de documents et d'informations a priori connus,
- le corpus de test qui sert à vérifier la qualité du service et du modèle sur lequel il repose.

Descripteur visuel d'une image : Ensemble de caractéristiques techniques, dites de « bas niveau », extraites des images pour en décrire la signature visuelle. Ces caractéristiques peuvent être, par exemple, la texture, la couleur, les formes ou une combinaison de plusieurs de ces caractéristiques.

Indexation : Processus consistant à extraire les caractéristiques de différents documents ou ressources et à les enregistrer dans une base de données afin de pouvoir les retrouver rapidement en fonction de critères significatifs. Dans le cas des moteurs de recherche « plein texte » les caractéristiques extraites correspondent à des mots représentatifs (non vides) qui seront utilisés dans le cadre d'une recherche par mots clefs. Dans le cas des images, les caractéristiques sont extraites des pixels et stockées dans un vecteur numérique appelé descripteur visuel. Dans l'étape de recherche, le système prend en entrée une image qui sert de requête et il retourne comme résultat une liste d'images ordonnée en fonction de la similarité entre leurs descripteurs visuels et celui de l'image en entrée en utilisant une mesure de distance.

Liferay : Portail open source capable de gérer du contenu. Écrit en java, celui-ci s'appuie, au choix, sur un serveur d'application JEE et exploite les EJB, mais il est aussi capable de fonctionner dans un conteneur de servlet comme Tomcat. C'est sur ce portail que nous allons créer toutes nos portlets, et nous nous servirons de ses fonctionnalités de gestion de contenu, de collaboration et de portail afin de réaliser la couche vue de notre démonstrateur.

Matrice de confusion : Matrice permettant de juger d'un système de classification. Très souvent, chaque colonne de la matrice représente le nombre d'occurrences d'une classe prédite par le système de classification, tandis que chaque ligne représente le nombre d'occurrences d'une classe réelle (ou de référence).

Maitrise d'Œuvre (MOE) : Le maître d'œuvre ou l'organisation qui assure la maîtrise d'œuvre est une personne physique ou morale (entreprise, direction, etc.) garante de la bonne réalisation technique des solutions. Il est responsable de la conception du système d'information (SI) et il a un devoir de conseil vis-à-vis du maître d'ouvrage.

Maitrise d'Ouvrage (MOA) : Le maître d'ouvrage ou l'organisation qui assure la maîtrise d'ouvrage est le donneur d'ordre au profit de qui l'ouvrage est réalisé.

Media-Mining : Domaine technologique consacré à la fouille de documents multi-média c'est à dire à l'analyse de contenus numériques pour y découvrir et en extraire des connaissances. Le media-mining peut être considéré comme une généralisation du data-mining, du text-mining ou encore de l'audiomining. Il s'applique aux contenus numériques de toutes natures.

Orchestrator : Processus automatique d'organisation, de coordination, et de gestion de systèmes informatiques complexes et de services.

Portail : Site internet ou intranet qui offre une porte d'entrée unique sur un large panel de ressources et de services (messagerie électronique, forum de discussion, espaces de publication, moteur de recherche) centrés sur un domaine ou une communauté particulière.

Portlet : Module intégré à un portail d'entreprise, qui permet à l'utilisateur de disposer, dans la même fenêtre, d'un accès centralisé et convivial à différentes ressources (données, applications, sites Web, etc.), de modifier l'interface du portail selon ses besoins et de personnaliser ainsi son environnement de travail.

PrimeFaces : Bibliothèque graphique qui fournit une suite de composants JSF.

Rejet en ambiguïté et en distance : En classification, cas pour lesquels un système ne se prononce pas. En ambiguïté, lorsque l'objet à classer est trop proche de la frontière de décision, en distance lorsque cet objet est trop éloigné des données d'apprentissage.

Spring : Framework open source J2EE pour les applications n-tiers, dont il facilite le développement et les tests. Il est considéré comme un conteneur dit « léger », c'est-à-dire une infrastructure similaire à un serveur d'application J2EE. Il prend donc en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets.

Taxonomie : Science des classifications, utilisée par exemple en bactériologie, en botanique et en zoologie, et qui permet de regrouper des objets en fonction de leurs caractéristiques.

Vérité terrain : Ensemble de données pour lequel une référence liée à la tâche de traitement considérée est connue (en général, donnée par un ou plusieurs experts humains). Par exemple en classification automatique d'image, ensemble d'images pour lequel la catégorie de l'image est donnée.

WebLab : Plateforme d'intégration pour la mise en œuvre d'applications de media-mining. Développée à l'initiative de Cassidian, WebLab est fondée sur les technologies des Web Services et du Web Sémantique. Elle permet de construire des applications par assemblage de services et de composants hétérogènes.

B Les signes utilisés

API : *Application Programming Interface* : Interface de programmation permettant d'accéder à des fonctions, des procédures ou des classes d'objets mises à disposition par un composant logiciel.

CBIR : *Content Based Image Retrieval* : Technique permettant de rechercher des images à partir de leurs caractéristiques visuelles déduites de leurs pixels. Celles-ci sont classiquement décrites comme rendant compte de leur texture, couleur, forme. Un cas typique d'utilisation est la recherche par l'exemple (ou par similarité) où l'on souhaite retrouver des images visuellement similaire à un exemple donné en requête. La comparaison consiste à définir diverses distances entre les caractéristiques extraites et à définir une mesure de similarité globale. Ce processus s'oppose à la recherche d'images par mots clés, qui furent historiquement proposée par les moteurs de recherche tels que Google image, où les images sont retrouvées en utilisant le texte qui les accompagne plutôt que le contenu de l'image elle-même.

HTML5 : *HyperText Markup Language 5* : Révision majeure d'HTML actuellement en cours de standardisation par le W3C <http://www.w3.org/TR/html5/>. HTML5 spécifie deux syntaxes d'un même modèle abstrait: HTML5 et XHTML5. Il étend HTML4 en ajoutant de nouvelles balises et de nouveaux attributs.

JSF : Java server Faces et un framework de développement d'applications Web en java. Il est de type MVC, basé sur des composants coté présentation et destiné aux applications web respectant l'architecture J2EE.

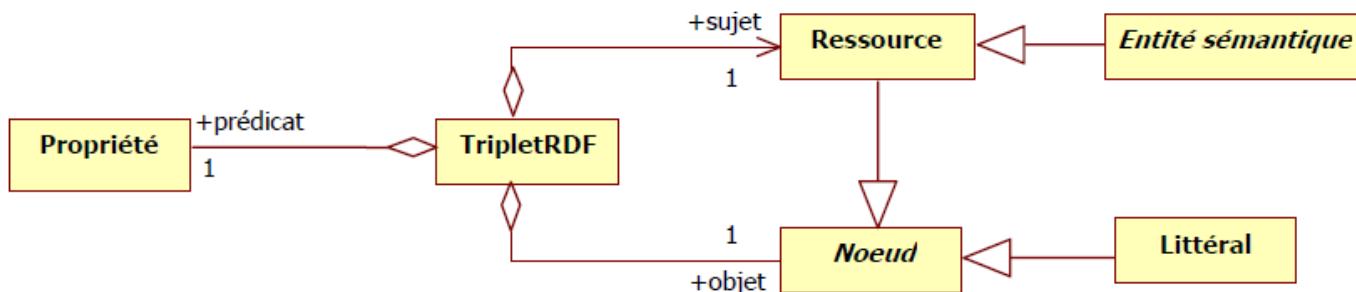
LGPL : *Lesser General Public License* : Licence du GNU utilisée pour certains logiciels libres ou open source. Cette licence n'a pas de caractère contaminant et permet à un programmeur de développer une application propriétaire en utilisant le logiciel mis à disposition sans le contraindre à diffuser son application avec la même licence que le logiciel réutilisé.

OW2 : *Object Web / Orient Ware* : Association internationale dédiée au développement de middleware libres de qualité industrielle. Issue de la fusion en 2006 du consortium français Object Web (fondé par l'INRIA, Bull et France Telecom) et du consortium chinois Orient Ware, elle regroupe aujourd'hui des entreprises et des organismes de

recherche de premier plan. Parmi les projets hébergés dans l'infrastructure OW2, on peut citer, outre WebLab, JOnAS, Joram, XWiki, Bonita, eXo, Platform, etc.

POA - Programmation Orientée Aspect

RDF : Resource Description Framework : Modèle de graphe défini par le W3C pour décrire, de façon formelle, les ressources Web et leurs métadonnées. RDF sert de base à la description des ontologies avec OWL et permet de formaliser des connaissances. Annoter sémantiquement une ressource consiste à lier cette ressource avec une entité sémantique d'une ontologie par l'intermédiaire d'une propriété. Dans le modèle RDF (Resource Description Framework) ceci amène à créer un triplé (ou triple) qui agrège la ressource métier (le sujet), la propriété (le prédicat) et le concept (l'objet). Plus généralement, RDF permet de définir un ensemble de propriétés qui s'appliquent à une ressource et qui permettent de la mettre en relation avec une autre ressource ou un littéral.



RSS - Really Simple Syndication, format de donnée utilisé pour la syndication de contenu Web

SKOS : Simple Knowledge Organization System : Famille de langages formels permettant une représentation standard des thésaurus, classifications ou tout autre type de vocabulaire contrôlé et structuré.

SPORTIFS : Système Pour l'Organisation, la Recherche et le Traitement d'Images de Flux Sportifs

URI : Uniform Resource Identifier -Chaîne de caractères permettant d'identifier de façon unique une ressource sur un réseau en utilisant une syntaxe définie par le standard W3C RFC 3986. Il existe deux types d'URI :

- Les **URL (Uniform Resource Locator)** qui décrivent le mode d'accès primaire et l'emplacement « physique » de la ressource sur le réseau. Par exemple, l'URL <http://www.weblab-project.org/> est un URL qui identifie une ressource qui décrit le projet WebLab et le moyen d'y accéder via le protocole HTTP.
- Les **URN (Uniform Resource Name)** sont des URI qui identifient les ressources par leurs noms dans un espace de nommage. Au contraire de l'URL, une URN peut être employée pour parler d'une ressource sans que cela préjuge de son emplacement ou de la manière de l'accéder.

WSDL : Web Services Description Language : Norme regroupant la description des éléments permettant de mettre en place l'accès à un service web.

IV Configuration requise

Pour pouvoir réaliser l'application dans les meilleures conditions, le projet SPORTIFS possède une architecture matérielle composée de 3 serveurs distincts. Nous avons choisi d'avoir des serveurs différents afin de mettre en évidence le côté distribué de notre solution. Voici le détail de la configuration des serveurs que nous utiliserons :

A Serveur dédié aux services web et à la persistance

Ce serveur requiert une capacité disque conséquente puisque nous allons avoir de nombreuses données à stocker. En effet, si nous imaginons une taxonomie de 150 classes (c'est-à-dire le nombre de sports existants environ) avec une représentation minimum de 100 images par classe), cela représente un total de 15000 images. En évaluant la taille moyenne d'une image à 1Mo (valeur prise arbitrairement), cela signifie un total de 15Go de données. L'installation des logiciels et le stockage des fichiers occupant environ 3-4Go, nous conseillons d'avoir un espace de stockage minimal de 30Go pour une utilisation du démonstrateur consacrée uniquement aux sports. Cependant, comme il est possible de faire évoluer facilement la taxonomie, il est préférable d'opter pour un espace de stockage plus conséquent (minimum 60Go). Ainsi, cette machine aura les caractéristiques suivantes :

Serveur dédié aux services web et à la persistance	
Système	Windows 7 – 64 bits
Processeur	3 GHz
Mémoire	4 Go
Disque Dur	80 Go
Logiciels installés	
Tomcat	7.0.26
Base de données MySQL	5.6.14

B Serveur dédié à Petals

Ce serveur ne nécessite pas forcément d'être très puissant ou d'avoir une capacité de stockage conséquente car il ne servira qu'à réaliser l'orchestration. Ainsi, cette machine aura les caractéristiques suivantes :

Serveur dédié à Petals	
Système	Windows 7 – 64 bits
Processeur	3 GHz
Mémoire	4 Go
Disque Dur	80 Go
Logiciels installés	
Petals ESB	4.0

C Serveur dédié à Liferay

Comme pour le serveur précédent, les capacités requises ne sont pas très conséquentes. En effet, cette machine ne nécessitera simplement la mise en place d'un serveur Tomcat pour déployer Liferay et les portlets. Ainsi, cette machine aura les caractéristiques suivantes :

Serveur dédié à Liferay	
Système	Windows 7 – 64 bits
Processeur	3 GHz
Mémoire	4 Go
Disque Dur	80 Go
Logiciels installés	
Tomcat	7.0.23
Liferay	6.1.0

D Environnement de test et de production

Chaque machine hébergera à la fois le serveur de production et le serveur de test, et ce afin de limiter le nombre de machines utilisés lors de la phase de développement. Les machines qui serviront à développer contiendront un serveur de développement permettant de vérifier le fonctionnement de la réalisation. Il sera ensuite déployé sur le serveur de test. En cas de réussite des tests, le composant sera intégré en production par la MOE lors d'une phase d'intégration.

E Développement

Le projet utilisera l'outil de gestion de dépendance Apache Maven en version 3.1.1. Les réalisations respecteront les standards du W3C avec notamment :

- HTML5
- RDF
- XML
- SKOS

F Configuration côté client

Notre solution utilisera les technologies du web (HTML, CSS, JS) et nécessitera donc d'avoir des navigateurs supportant ces technologies. Nous essaierons d'être compatibles avec les dernières versions des navigateurs majeurs :

- Firefox 25+
- Chrome 31+
- Internet Explorer 11+

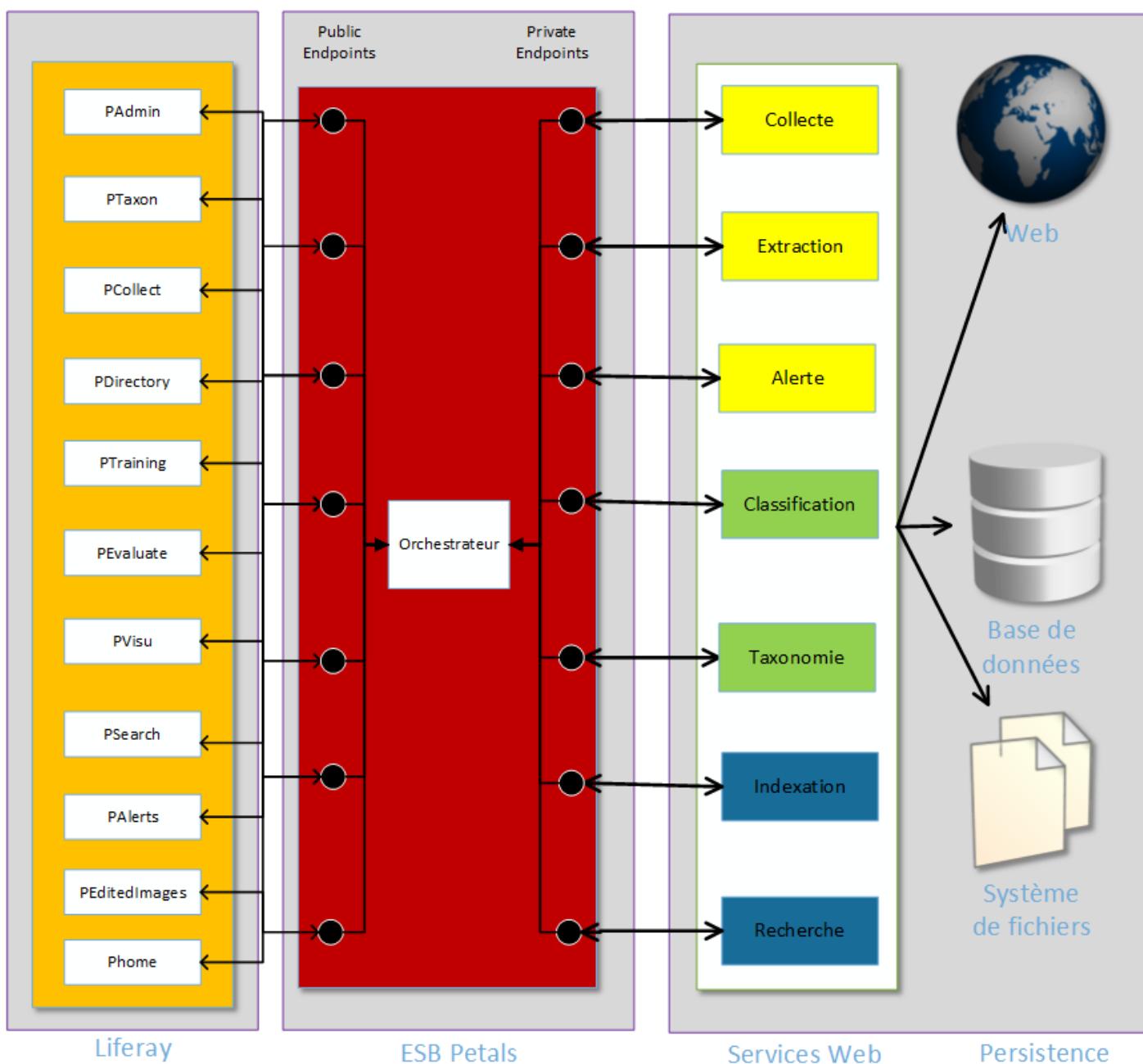
Bien évidemment, le JavaScript devra être actif sur les navigateurs.

Notre application sera normalement multiplateforme (systèmes d'exploitation) dans la mesure où les serveurs Tomcat puissent être installés sur la machine. L'infrastructure importe peu puisqu'il s'agit de mettre en place une architecture orientée services (SOA).

V Architecture générale du démonstrateur

A Architecture générale

Cette section décrit l'architecture générale du démonstrateur en explicitant les différents composants qui seront développés dans le cadre du démonstrateur SPORTIFS, intégré à la plateforme WebLab. La plateforme étant construite selon une architecture orientée services, nous avons choisi de suivre ce modèle d'architecture pour le démonstrateur SPORTIFS afin de rendre notre application aussi modulable que possible. Il s'agira donc d'une application n-tiers. Le diagramme suivant présente les différentes couches du système. Nous expliquerons par la suite le rôle général de chaque composant.



Le premier tiers de l'application sera constitué de portlets, correspondant à la vue de notre application, qui seront mis en place avec Liferay. C'est d'ailleurs la base de données de celui-ci que nous utiliserons afin de gérer les droits des utilisateurs par portlet. Les droits d'accès des utilisateurs seront gérés par la vue. Nous ajouterons les trois types d'utilisateurs à la base de données de Liferay et y affecterons les droits qui conviennent. Nous avons préféré utiliser cette implémentation, disponible par défaut dans Liferay, plutôt que de recréer par nous même un système

de droit contrôlé car elle est plus simple et plus adaptée pour le démonstrateur SPORTIFS et elle permettra de réaliser une certaine économie tout en étant largement suffisante pour notre application.

Le second tiers est l'Enterprise Service Bus (ESB) Petals. Cette couche a pour objectif de permettre la communication entre différentes applications ou services Web qui ne sont pas nécessairement réalisés pour fonctionner ensemble (langages différents, etc.). L'ESB est construit avec le langage XML, la technologie JMS ou encore les services Web. L'ESB permettra la communication entre les différents services Web ainsi qu'avec les portlets. Cette ESB a, en son sein, un orchestrateur BPEL qui se chargera d'effectuer les différentes communications entre les portlets et les web services et entre les web services entre eux, permettant de les découpler au maximum, à l'aide d'une chaîne de processus qui n'est rien d'autre qu'une suite de processus à effectuer entre les différents communicants.

L'élément suivant correspond à l'ensemble des services qui seront rendus par l'application. Il s'agit de services Web qui auront chacun le but de réaliser une tâche précise afin de répondre au mieux aux exigences de la MOA. Les échanges entre les services seront réalisés en SOAP par l'utilisation des services de marshalling fournis par la plateforme WebLab et du composant BC-SOAP de Petals qui exécutera la chaîne de traitements. Ainsi, le contenu des fichiers XML transmis en SOAP sera du RDF pour pouvoir utiliser les interfaces de WebLab.

Le dernier tiers correspond à la couche persistance qui permettra de stocker les données de l'application dans une base de données mais également dans des fichiers. Ce tiers concerne également l'indexation des images et des métadonnées. Plusieurs services web intégré à Liferay communiqueront avec la base de données.

La plateforme WebLab sera déployée sur trois machines distinctes comme présenté précédemment. On peut voir ce découpage sur le diagramme par le biais des rectangles gris qui représente chacun un serveur.

B Détails des composants

Cette partie du document va décrire de manière générale mes différents composants du projet. Ils sont décrits de manière détaillée dans la suite du document par les responsables techniques des lots qui seront amenés à les réaliser.

1 Description du constituant Liferay

Liferay est un portail open source capable de gérer du contenu. Écrit en java, celui-ci s'appuie, au choix, sur un serveur d'application JEE et exploite les EJB, mais il est aussi capable de fonctionner dans un conteneur de servlet comme Tomcat. C'est sur ce portail que nous allons créer toutes nos portlets, et nous nous servirons de ses fonctionnalités de gestion de contenu, de collaboration et de portail afin de réaliser la couche vue de notre démonstrateur.

2 Description du constituant Petals ESB

Petals est un Bus de Services d'Entreprise (ESB) fourni par le Consortium OW2. Concrètement, c'est une plateforme Java qui se base sur les principes SOA pour interconnecter des systèmes, applications et services hétérogènes.

L'ESB Petals permet la mise en place d'une architecture distribuée. La communication entre les systèmes est réalisée à l'aide d'échange de messages entre les fournisseurs et les consommateurs de services, leur relation étant contrôlée par un contrat (WSDL), de cette manière, les applications utilisant Petals se retrouvent avec des services web faiblement couplés. C'est l'orchestre, au cœur de Petals, qui se charge de faire communiquer les services par le biais de différents protocoles de communication : SOAP, HTTP(S), Mail, FTP, SFTP, ...

Petals supporte les standards du web comme WS-Security, XSLT, XML Schema, Il offre également des outils d'aide au développement, à l'administration et à la maintenance des applications. Étant tout d'abord un serveur java standalone, il utilise des composants JBI pour permettre l'intégration du système de communication par message entre les applications. Pour l'administration, Petals CLI est une interface en ligne de commande qui permet de gérer les serveurs Petals. Pour la maintenance, Petals inclut un serveur JMX

Ainsi, pour interconnecter des services web, il nous faut d'abord ajouter les services (WSDL) à Petals. Via le bundle WebLab, il suffit d'utiliser la commande : \$weblab[.bat|.sh] add [URI vers le WSDL]. On préférera utiliser Petals Studio afin de gérer les services plus simplement. Lors de la procédure d'ajout, Petals se charge de créer les endpoints ou points d'entrée publique, sur lesquels se connecteront les portlets, et les endpoints privés, sur lesquels communiqueront les services. (Voir : http://weblab-project.org/index.php?title=WebLab_POJO_Chain pour plus d'informations)

3 Description du constituant Collecte

Le rôle du constituant « Collecte » est de collecter des images et les pages qui les contiennent depuis des sites HTTP(s) ou des flux RSS de référence sélectionnés par le superviseur. Il doit également permettre de gérer les différentes collectes ainsi que leur lancement.

Ce composant à la charge de récupérer des images et les pages associées sur Internet afin de pouvoir extraire des informations par l'utilisation du composant d'extraction. Il permet également d'uploader une image depuis un formulaire par le biais d'une URL ou d'un fichier local. Enfin, l'outil doit permettre de suivre les collectes en sauvegardant notamment le nombre d'images collectés et en erreur pour chaque collecte.

Le développement de ce composant sera basé sur l'utilisation des outils Crawler4J pour les sites et ROME pour les flux RSS.

4 Description du constituant Extraction

Le rôle du constituant « Extraction » est d'extraire des informations depuis les images et pages associées afin de pouvoir la classer dans une catégorie. Pour cela, ce constituant aura la charge d'extraire des métadonnées à la fois physiques (poids, format, etc.), techniques (EXIF, XMP, IPTC) mais également des caractéristiques intrinsèques aux images (locales ou globales).

Il aura la charge d'obtenir les informations depuis les images. Ces informations seront utilisées par le processus de classification.

Ce composant sera basé sur l'utilisation des outils Apache Tika pour l'extraction des informations depuis les pages web ainsi que les métadonnées techniques. LIRE sera utilisé pour les caractéristiques intrinsèques.

5 Description du constituant Classification

Le rôle du constituant « Classification » est de définir à quel sport appartient une image en se basant sur les données visuelles, les données textuelles et les métadonnées de celle-ci. Cet outil se base sur un modèle de classification. Ce modèle peut être évalué par le démonstrateur.

Ce composant à la charge d'attribuer une classe (un sport) à une image. Il doit également permettre d'évaluer l'efficacité d'un modèle de classification sur un jeu d'images de test. On obtient notamment le nombre d'images correctement classées, le nombre d'images en échec (ambiguïté) ainsi que la matrice de confusion qui récapitule le nombre de Vrai-Positif, de Vrai-Négatif, de Faux-Positif et de Faux-Négatif. Cette matrice, appelée matrice de confusion, est exportable au format CSV. Ce composant permettra également de créer un nouveau modèle de classification à partir d'une sélection d'images.

Le développement de ce composant sera basé sur l'utilisation des outils LIRE et Lucene pour classer les images et utilisera l'algorithme KNN (k-plus proches voisins) dont l'utilisation sera détaillée dans la partie dédiée au composant de classification.

6 Description du constituant Taxonomie

Le rôle du constituant « Taxonomie » est de permettre la définition d'une taxonomie pour le démonstrateur SPORTIFS. Pour cela, la taxonomie sera basée sur un fichier SKOS qui sera modifié grâce à l'outil SKOS API. Le service doit être agnostique à la taxonomie gérée. Il doit permettre de modifier la taxonomie simplement.

Ce composant a donc la charge de modifier le fichier SKOS ainsi que la table relative à la taxonomie dans la base de données. Il doit donc permettre d'ajouter un concept, d'en supprimer mais également d'en modifier (ajout d'un sous-concept, d'une alternative, etc.).

7 Description du constituant Indexation

Le rôle du constituant « Indexation » est de mettre en place des index à partir des métadonnées extraites des images et de la classe déterminés par le moteur de classification. Ainsi, celui-ci doit permettre d'indexer les métadonnées ainsi que les caractéristiques intrinsèques associées à une image. L'index doit permettre de détecter les doublons rapidement.

Le développement de ce composant sera basé sur l'utilisation des outils LIRE et Lucene pour indexer les images et les métadonnées textuelles.

8 Description du constituant Recherche

Le rôle du constituant « Recherche » est de mettre de rechercher des images à partir de données textuelles ou par similarités image.

Ainsi, celui-ci doit permettre de réaliser des requêtes sur les métadonnées textuelles par l'utilisation de requête pouvant contenir des jokers, des booléens, etc. mais également approximatives. Il doit également permettre d'effectuer des requêtes sur les caractéristiques des images telles que la couleur dominante ou la similarité d'une image avec une autre. En outre, ce composant doit permettre l'utilisation de filtres prédéfinis telle que le format de l'image, une dimension minimale, etc. Enfin, le constituant doit pouvoir fusionner les résultats des requêtes sur les caractéristiques des images et les métadonnées textuelles en impactant le moins possible le système.

Le développement de ce composant sera basé sur l'utilisation des outils LIRE et Lucene pour recherche les images indexées par le démonstrateur SPORTIFS.

9 Description du constituant Alerte

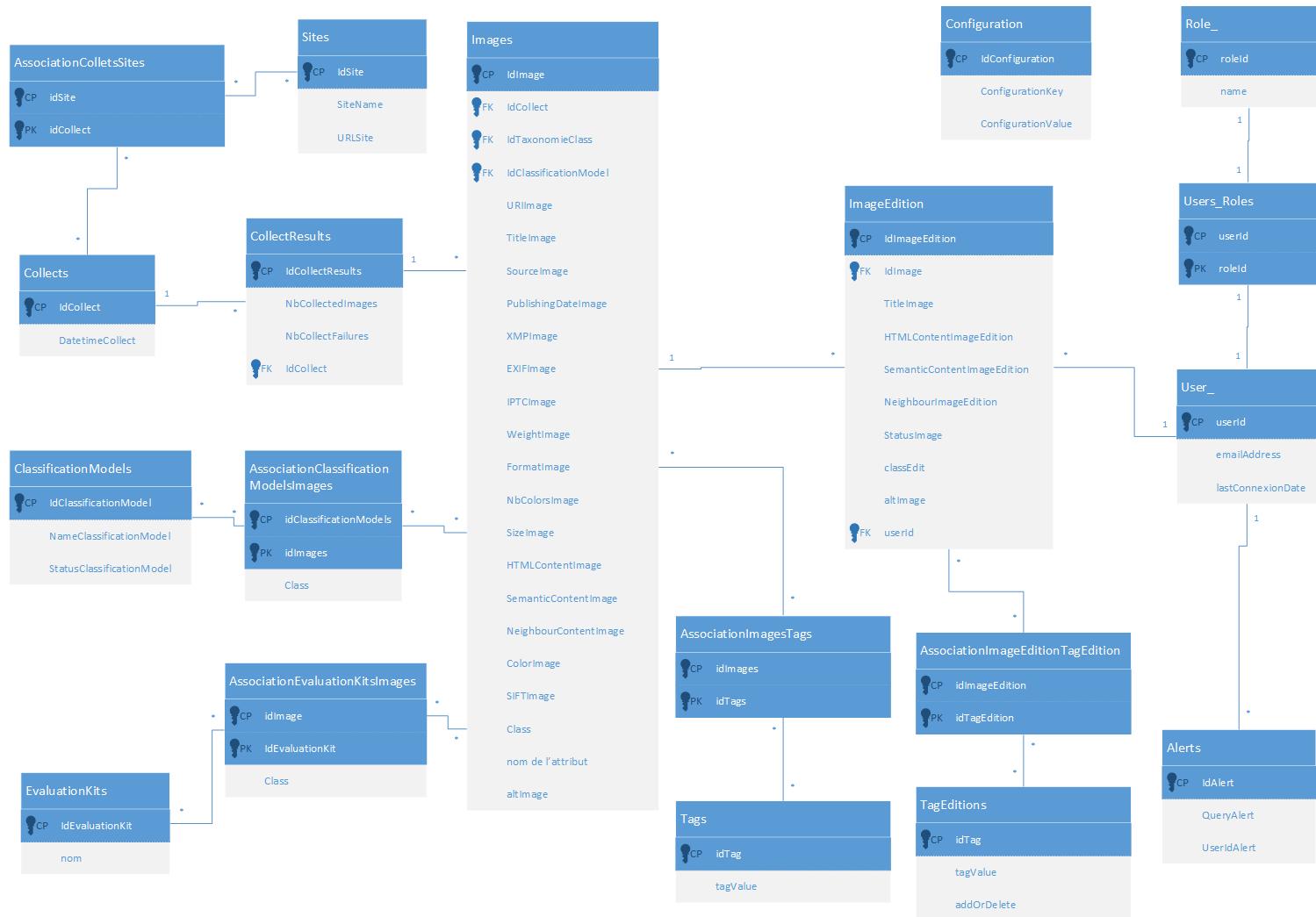
Le rôle du constituant « Alerte » est de gérer les alertes des utilisateurs. Il permet de créer et de supprimer des alertes mais également de prévenir les utilisateurs lorsqu'une image correspondant à une de ses alertes a été collectée. Les messages seront envoyés une fois par jour sous la forme de courriels.

Il sera basé sur l'utilisation de la librairie Quartz afin de permettre la création d'une tâche qui sera lancée de manière quotidienne. Les alertes seront conservées dans la base de données.

C Gestion des données de l'application (Clément Jehannet)

1 La base de données

Le modèle physique de données suivant permet de décrire le modèle de données qui sera utilisé par notre application. Il présente donc les différentes tables qui constitueront notre base de données. Elles sont centrées autour de la table Images qui est la table principale de la base de données.



Voici un récapitulatif du rôle de chaque table qui sera présente dans la base de données :

La table *Images* contient les métadonnées de toutes les images collectées. Elle permet d'accéder rapidement aux images ayant été collectées depuis la dernière collection d'un utilisateur.

La table *Tags* contient les tags des images se trouvant dans la table *Images*. Elle permet d'éviter une trop grande multiplicité de tags pouvant être identique.

La table *AssociationImagesTags* permet de faire l'association entre la table *Images* et la table *Tags*.

La table *ImageEdition* contient les éditions de métadonnées des différentes images par les utilisateurs de l'application.

La table *TagEditions* contient les tags des images se trouvant dans la table *ImageEdition*.

La table *AssociationImageEditionTagEdition* est une table d'association entre *ImageEdition* et *TagEdition*.

La table `User_` contient les caractéristiques d'un utilisateur. Cette table est gérée par Liferay.

La table Role_ : Contient les différents rôles des utilisateurs de l'application. Cette table est gérée par Liferay.

La table `Users_Roles` est une table association entre les tables `User_` et `Role_`. Cette table est gérée par Liferay.

La table *Alerts* contient les différentes alertes créées par les utilisateurs.

La table table *CollectResults* contient les détails des collectes effectuées.

La table *Collects* contient les collectes créées par les administrateurs.

La table *Sites* contient les différents sites à collecter (HTTP et RSS) par le composant de collecte.

La table *AssociationCollectsSites* est une table d'association entre *Collects* et *Sites*. Elle permet de suivre les collectes en conservant le nombre d'images collectées ou en erreur lors d'une collecte.

La table *ClassificationModels* contient les modèles de classification (URI, nom, statut actif ou non).

La table *AssociationClassificationModelsImages* est une table d'association entre *Images* et *ClassificationModels*. C'est dans cette table qu'est enregistrée la classe affectée par un superviseur lors de l'ajout d'une image à un moteur de classification.

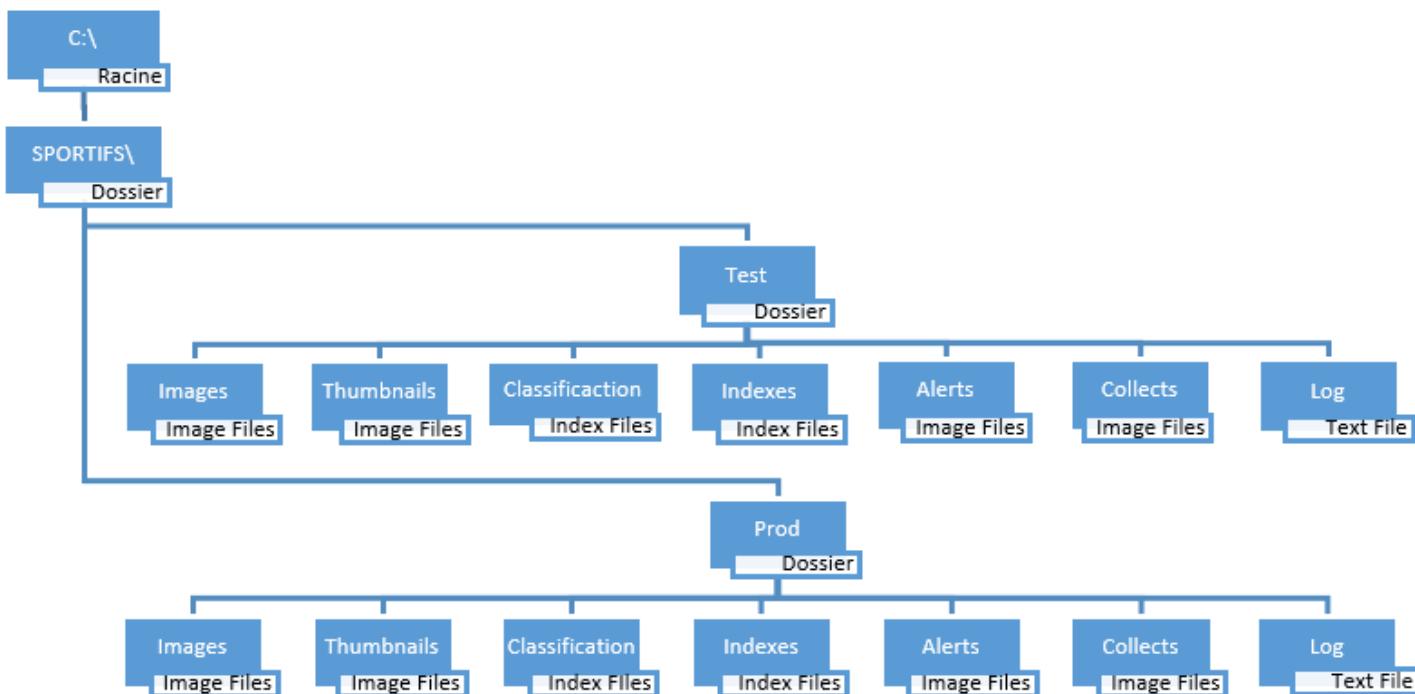
La table *EvaluationKits* contient les différents kits d'évaluation gérés par le démonstrateur (nom). Ils sont utilisés pour évaluer les moteurs de classification.

La table *AssociationEvaluationKitsImages* est une table d'association association entre *Images* et *EvaluationKits*. Comme pour la table *AssociationClassificationModelsImages*, celle-ci enregistre la classe affectée par un superviseur lors de l'ajout d'une image à un kit d'évaluation.

La table *Configuration* est contient les différentes éléments configurables de l'application comme le comportement lors de la rencontre des doublons ou le seuil minimal de classification.

2 Le système de gestion de fichiers

Voici l'organisation du système de fichier qui sera mise en place de le cadre du démonstrateur SPORTIFS :



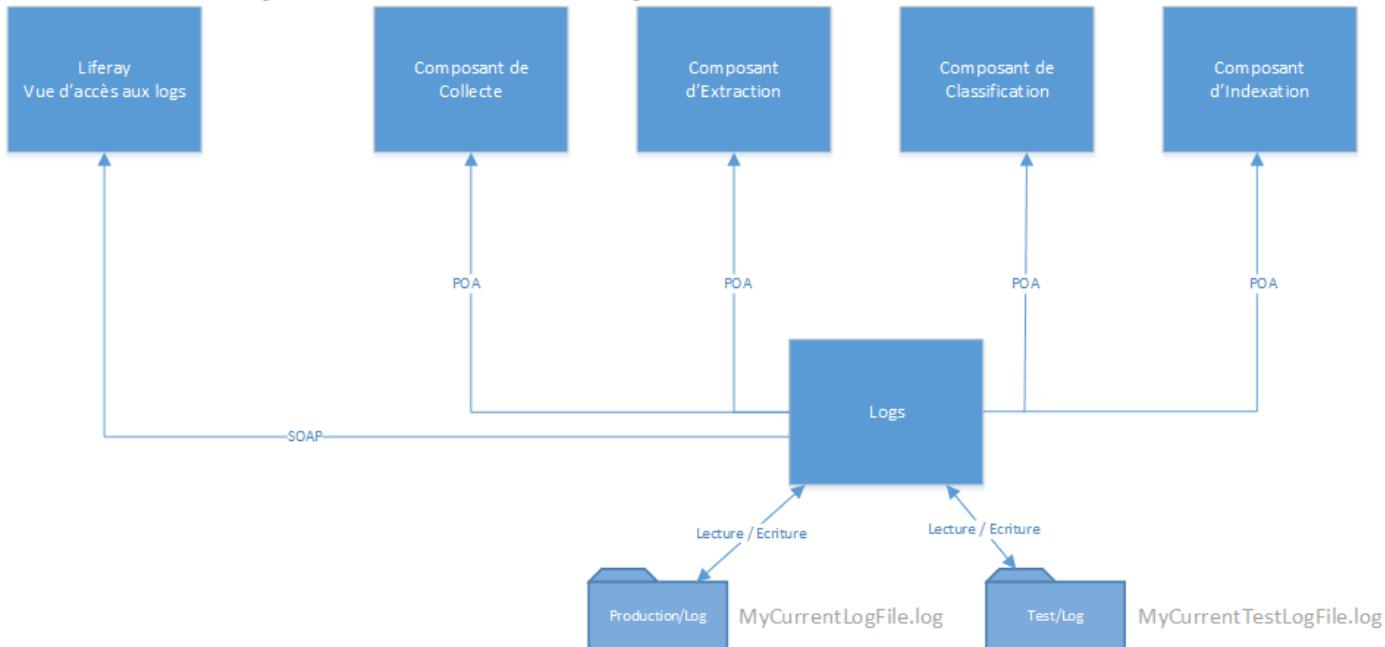
Tous les fichiers sont contenus dans un dossier *SPORTIFS* qui est à la racine de l'ordinateur. Celui-ci est composé des deux sous-dossiers suivants :

- Un dossier *Prod* qui contiendra tous les fichiers de l'application en production.
- Un dossier *Test* qui contiendra les fichiers correspondants à la version de l'application qui est en test.

Dans chacun des deux dossiers, on trouvera les 7 éléments suivants :

- Le dossier *Images* permettant de stocker les images de l'application.
- Le dossier *Thumbnails* servira à stocker les miniatures associées à ces images.
- Le dossier *Classification* contiendra les moteurs de classification.
- Le dossier *Indexes* permettra de stocker les index.
- Le dossier *Alerts* servira à stocker les images correspondant aux alertes créées par les utilisateurs.
- Le dossier *Collects* contiendra les images qui viennent d'être collectées au format RDF et qui sont en attente de traitement par le démonstrateur.
- Un fichier de log qui contiendra les logs de l'application

3 La gestion des fichiers de log

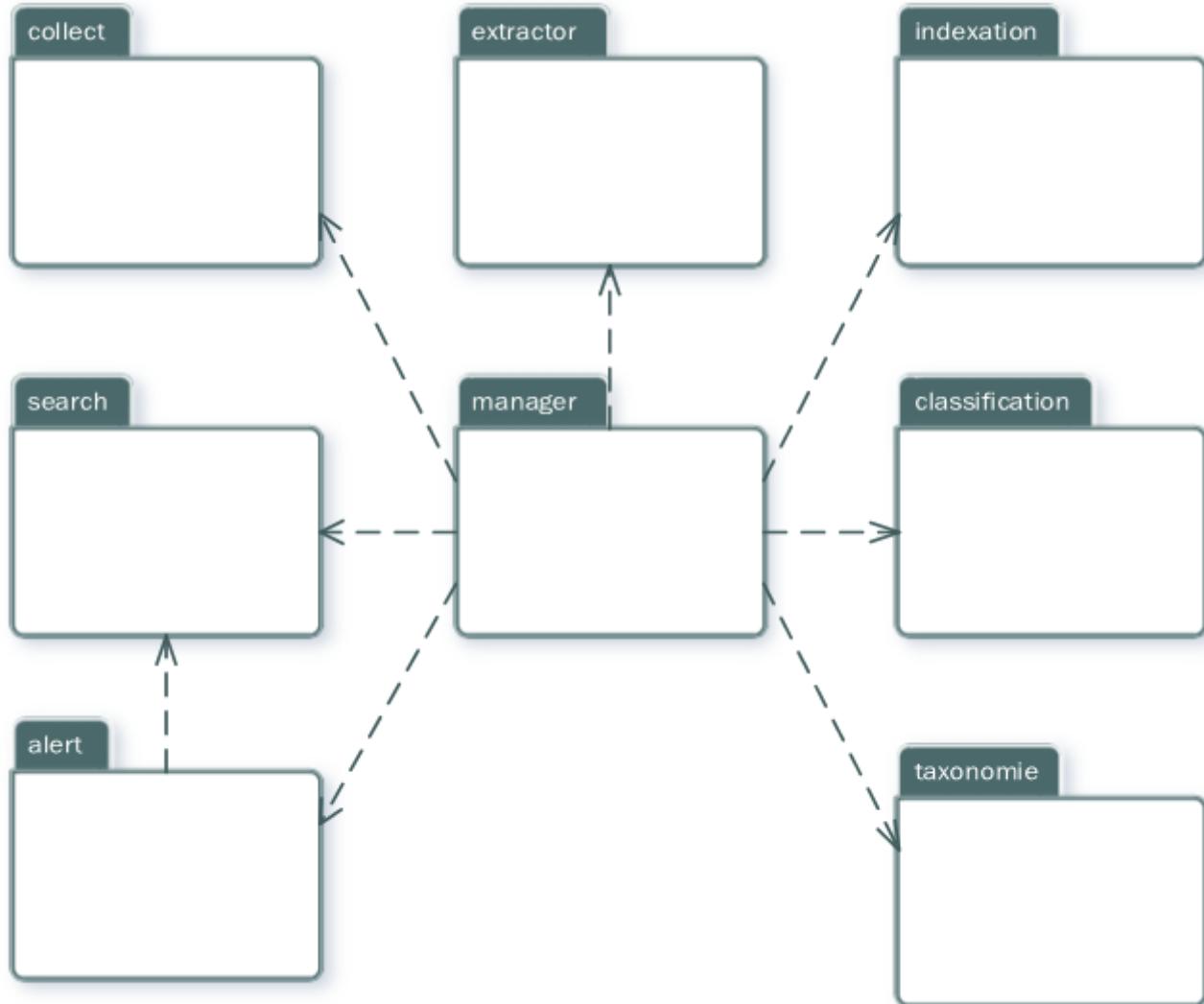


Tout d'abord, la POA (Programmation Orientée Aspect) sera utilisé dans ce projet afin de logger les différents événements. Ceux-ci sont captés à l'aide de points de jonction que nous définirons, suivant les noms des méthodes, pour qu'ils tracent l'exécution de certains composants. Nous avons convenus de tracer ces éléments :

- Les entrées des méthodes publiques des services web ;
- Les sorties de ces méthodes ;
- Les exceptions en détaillant le plus possible l'origine de l'erreur.

Une fois captés, ces événements sont enregistrés dans un fichier de log, situé dans le dossier de log. Lorsque ce fichier atteint sa taille maximale (fixée dans le fichier de configuration de Logback) on reprend l'écriture des logs à son début, par-dessus les plus anciens logs. Ainsi, nous maîtrisons la quantité d'informations enregistrées et la rapidité d'accès aux logs est maintenue.

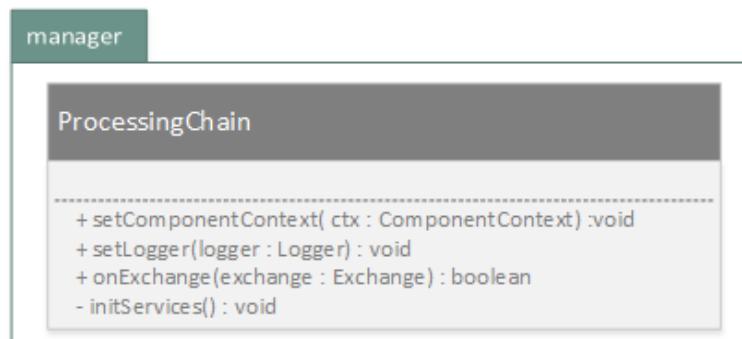
D Diagramme de paquetages



Ce diagramme de package représente les différentes étapes de la vie d'une image dans le démonstrateur SPORTIFS par le biais des différents paquetages qui correspondent chacun à un composant présenté précédemment. Il a donc été découpé en fonction des lots de travaux attribués à chaque équipe, qui seront développés lors du projet. Chaque lot de travaux sera abordé plus en détails dans la suite de ce document. La partie sur l'architecture générale ne présentera que le paquetage **manager**, le cœur de l'application.

E Le paquetage Manager : gestion de l'application

Ce diagramme détaille le paquetage **manager** permettant de gérer l'application, c'est-à-dire mettre en place la chaîne de traitement.



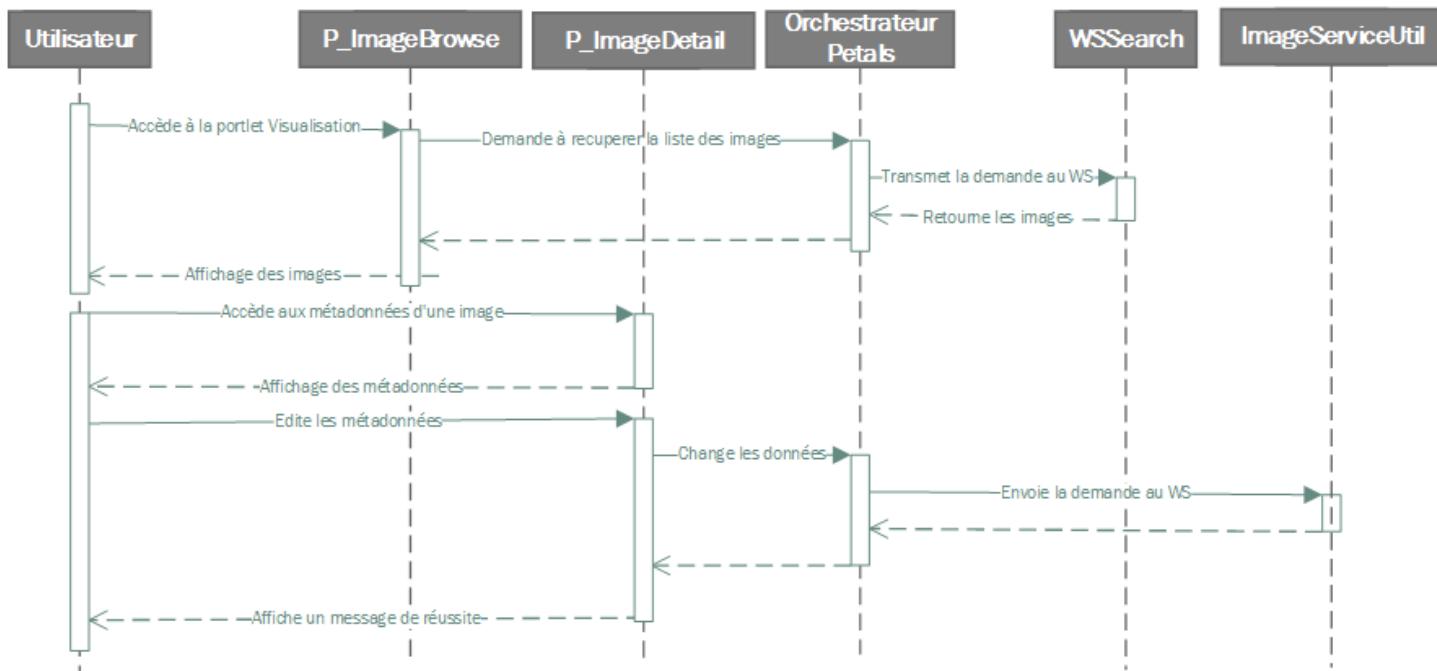
ProcessingChain correspond à la chaîne de traitement. Elle aura la charge de lancer le traitement des images ayant été collectées. Ainsi, elle permettra de lancer les services d'extraction, de classification et d'indexation pour toutes les images collectées ou uploadées.

F Fonctionnement dynamique

Dans la spécification technique de besoin, un certain nombre de cas d'utilisation ont été identifiés et détaillés. Dans ce présent document, nous avons défini précédemment les différents composants rentrant en jeu dans le démonstrateur **SPORTIFS**. Afin de mieux comprendre le fonctionnement de ces composants ainsi que leurs interactions, nous allons détailler ces cas d'utilisation à l'aide de diagrammes de séquence.

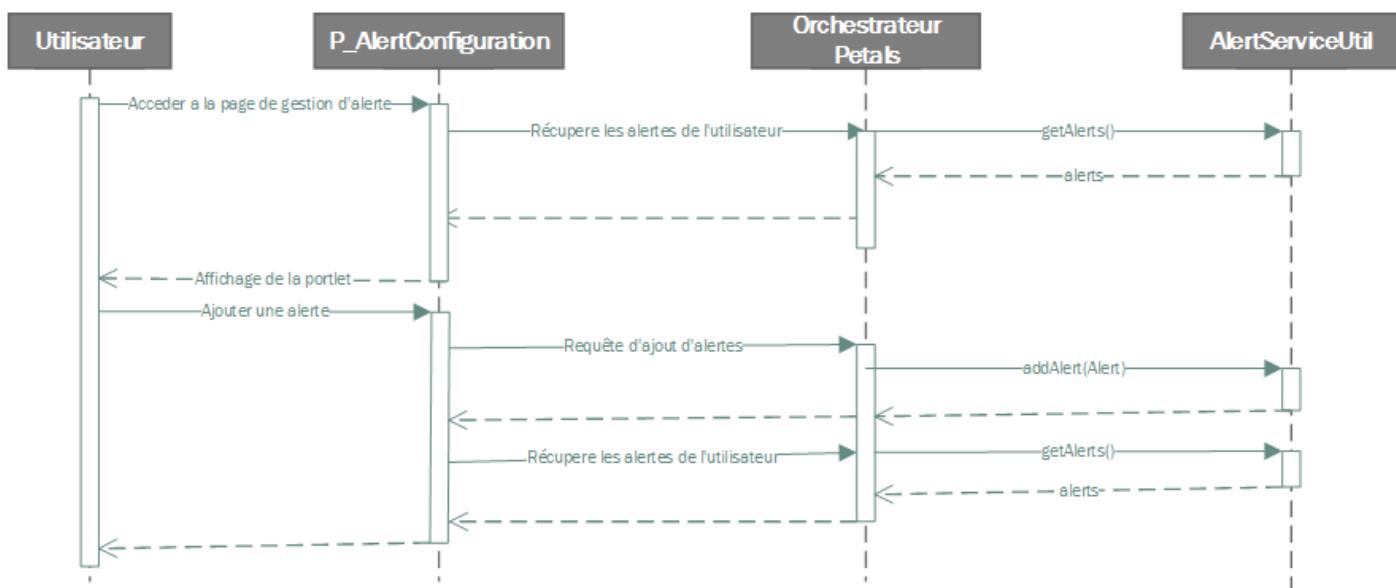
1 Éditer les images

Dans ce cas d'utilisation, nous détaillons les différentes actions entre les composants dans le cadre de l'édition des métadonnées d'une image. L'utilisateur après avoir visualisé les métadonnées détaillées d'une image peut les modifier. Par la suite, lorsqu'il valide son formulaire, le portlet transmet les métadonnées au service permettant d'ajouter les éditions d'image dans la base de données. S'il souhaite que l'image soit supprimée, le mécanisme sera identique.



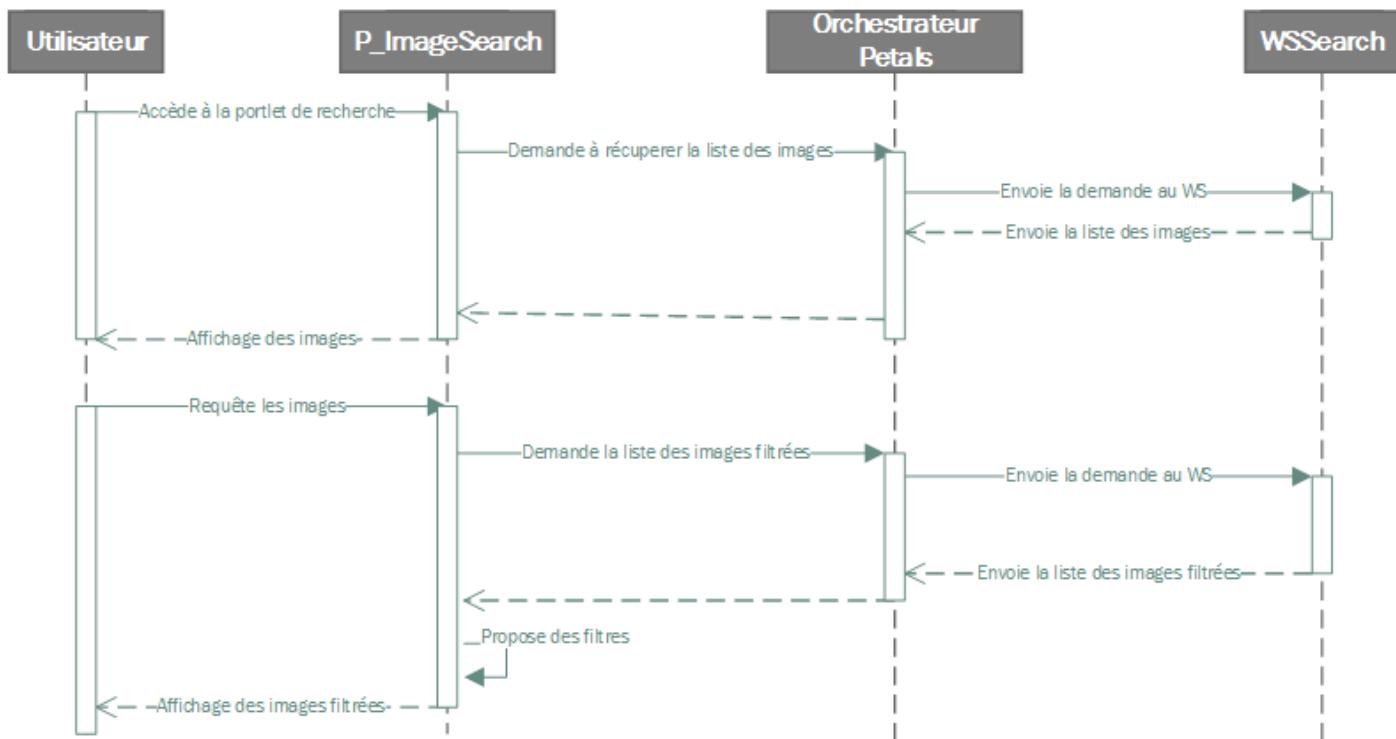
2 Gérer les alertes

Dans ce cas d'utilisation, nous détaillons les différentes actions entre les composants dans le cadre de la gestion des alertes utilisateurs. Celui-ci peut créer, éditer ou supprimer des alertes.



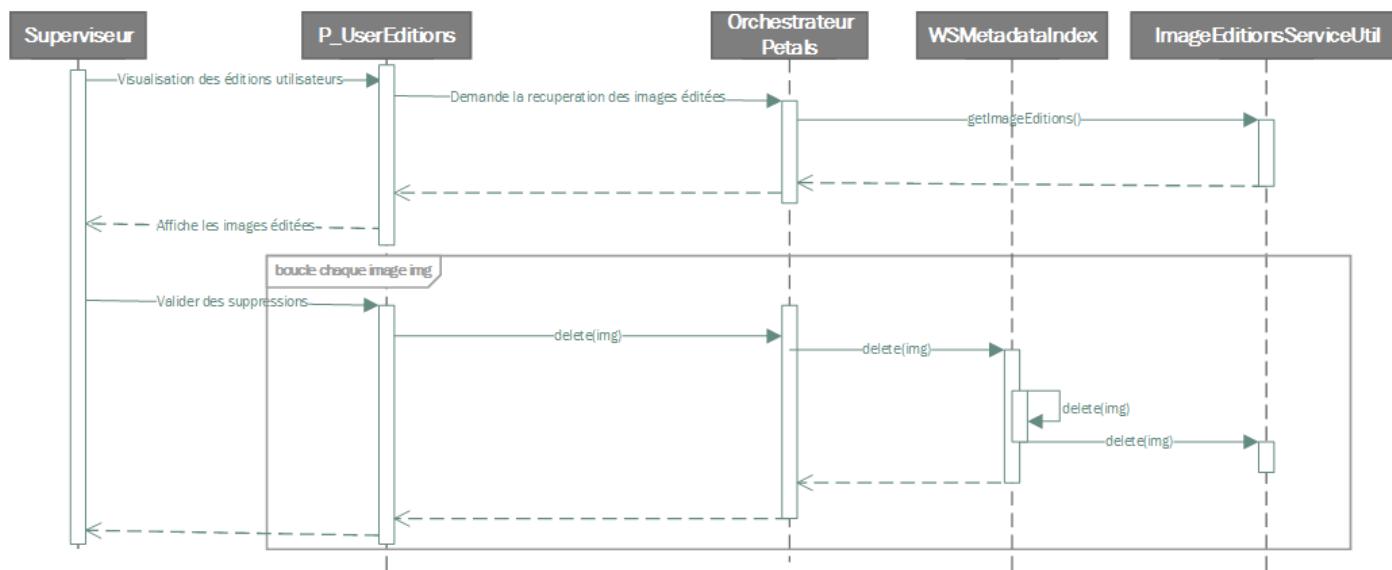
3 Rechercher des images

Ce diagramme de séquence présente le fonctionnement de la recherche d'image en essayant de s'abstraire du type de requête réalisé. En effet, le fonctionnement est quasi identique. Il sera détaillé par l'équipe bleue dans la partie associé au lot d'indexation. Seulement, la portlet de recherche contacte le service de recherche qui effectue la recherche sur l'index.



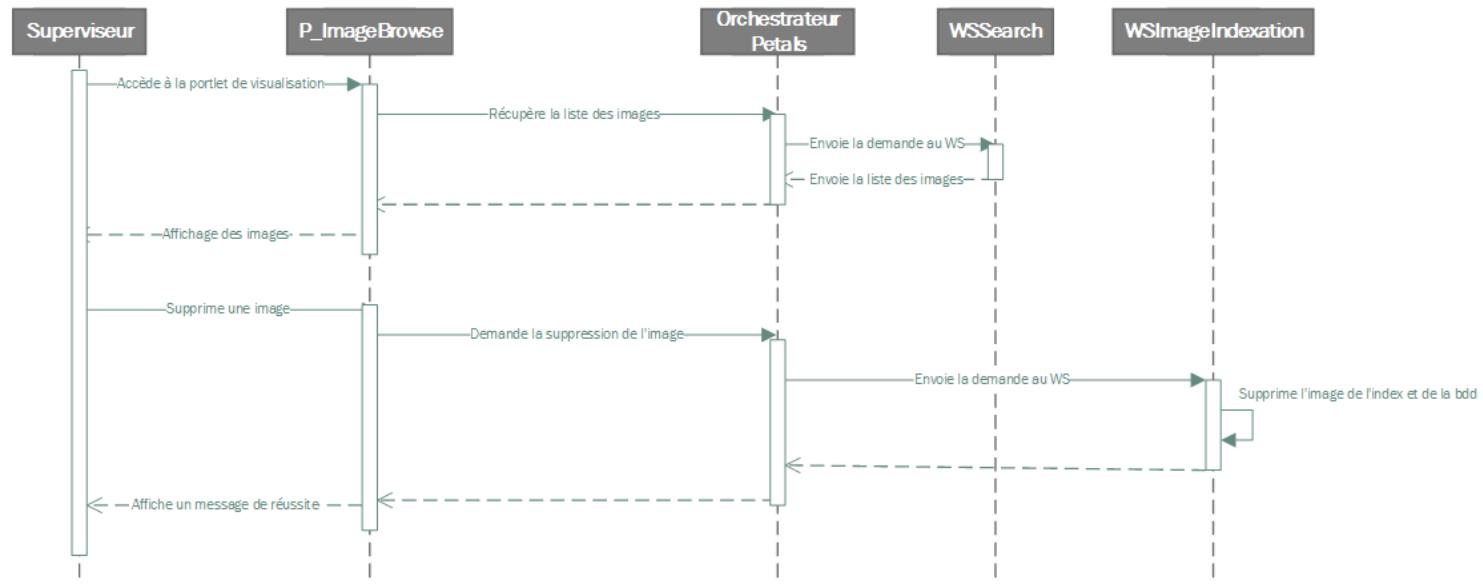
4 Gérer les éditions des utilisateurs

Ce diagramme décrit le fonctionnement du démonstrateur lorsqu'un utilisateur décide de proposer une image à la suppression. Le superviseur a le choix de valider ou non les éditions / suppressions des utilisateurs. Il peut valider les suppressions par lot d'images. La portlet se connectera donc au service d'édition de l'index qui aura également la charge de mettre à jour la base de données.



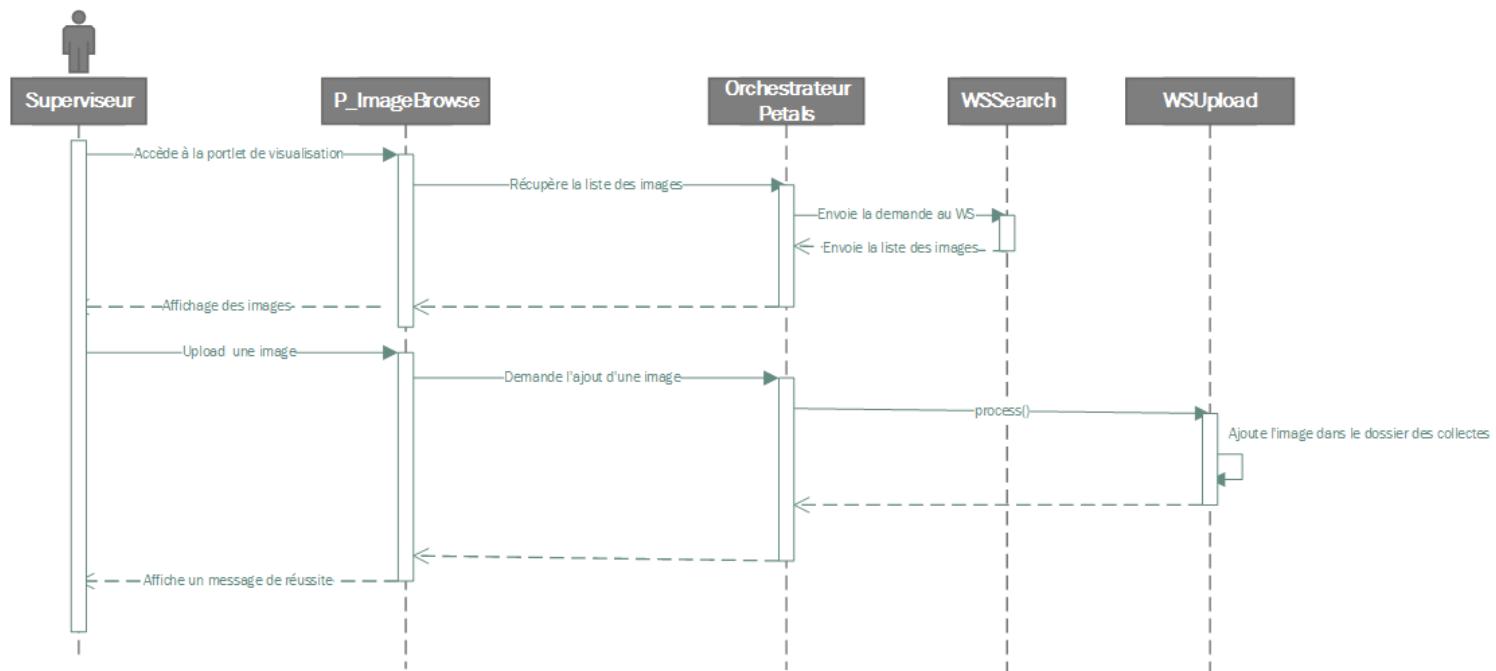
5 Supprimer une image

Ce diagramme de séquence présente le fonctionnement de la suppression d'une image par le superviseur. Il suffit simplement de cliquer sur le bouton de suppression d'une image. Le fonctionnement est similaire au

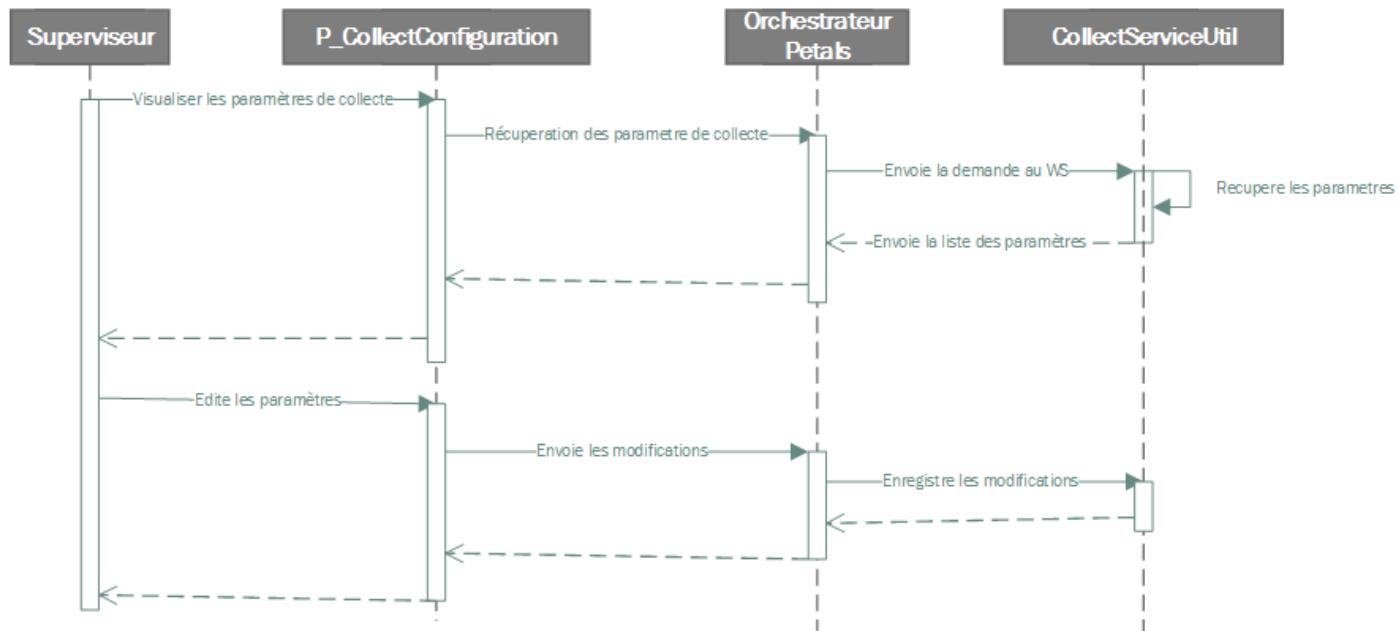


6 Uploader une image

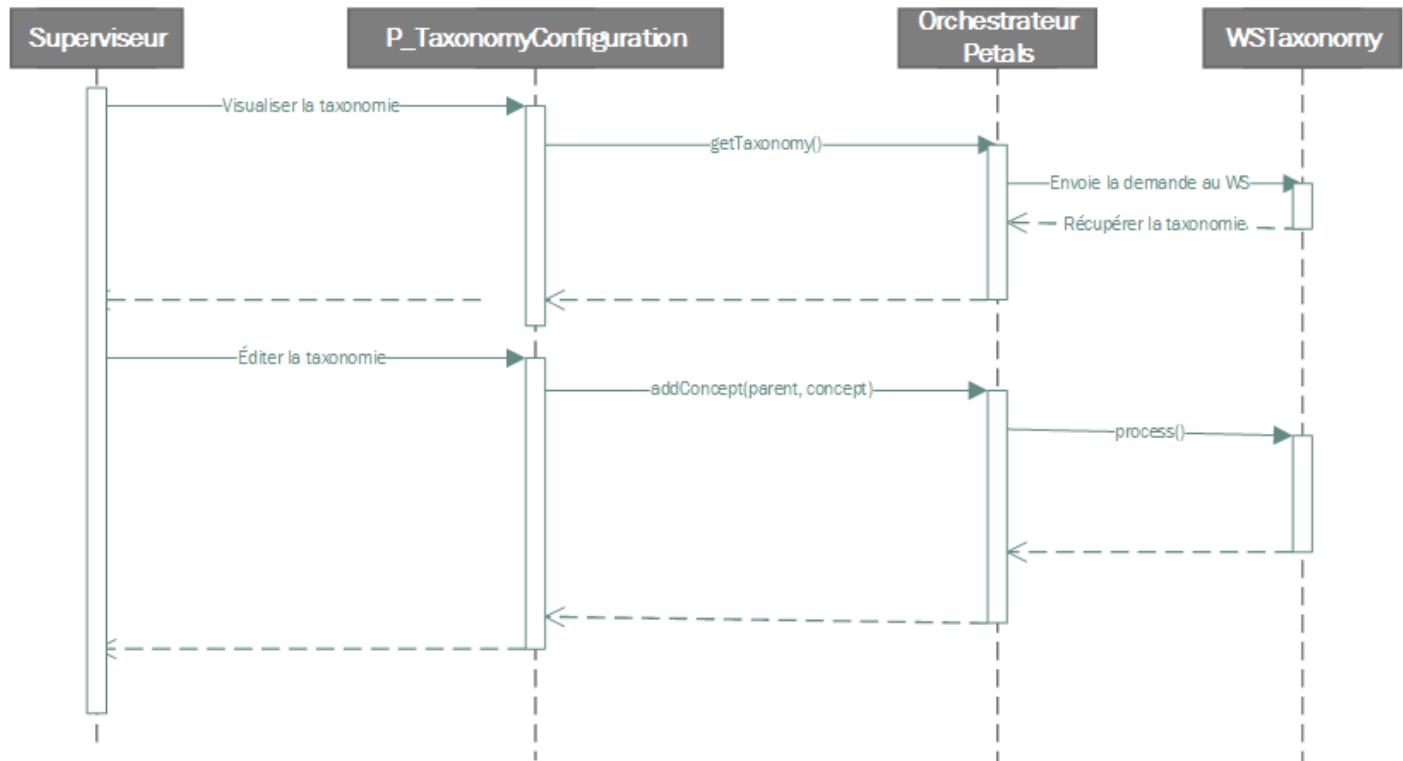
Le démonstrateur permet au superviseur d'uploader une image qui sera traitée par la chaîne de traitement. Pour cela, elle sera ajoutée dans le dossier de collecte par le service dédiée à l'upload d'image.



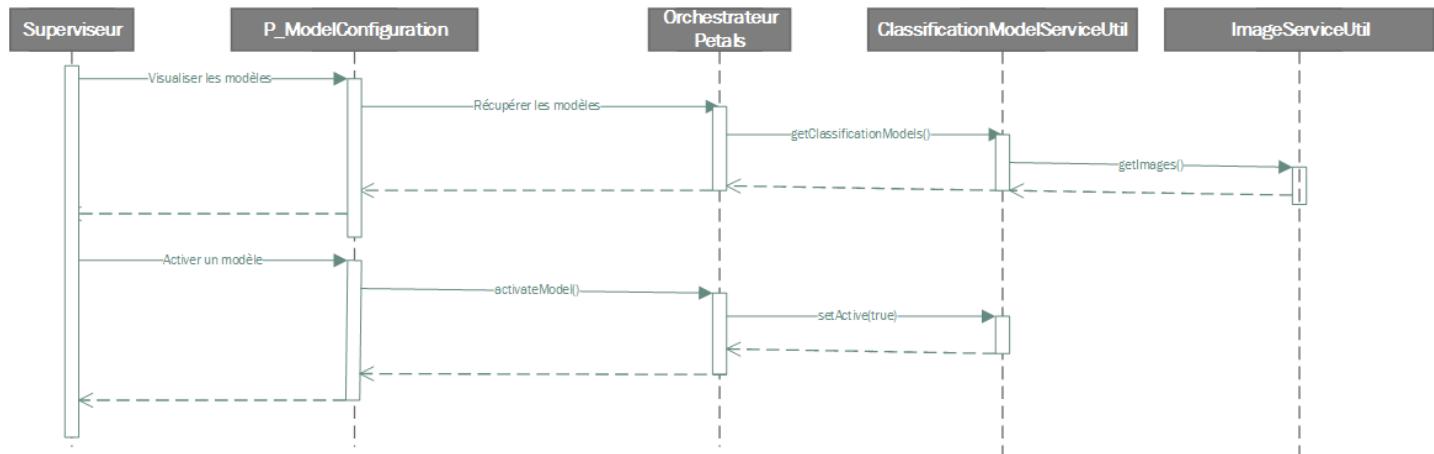
7 Gérer les paramètres de collecte



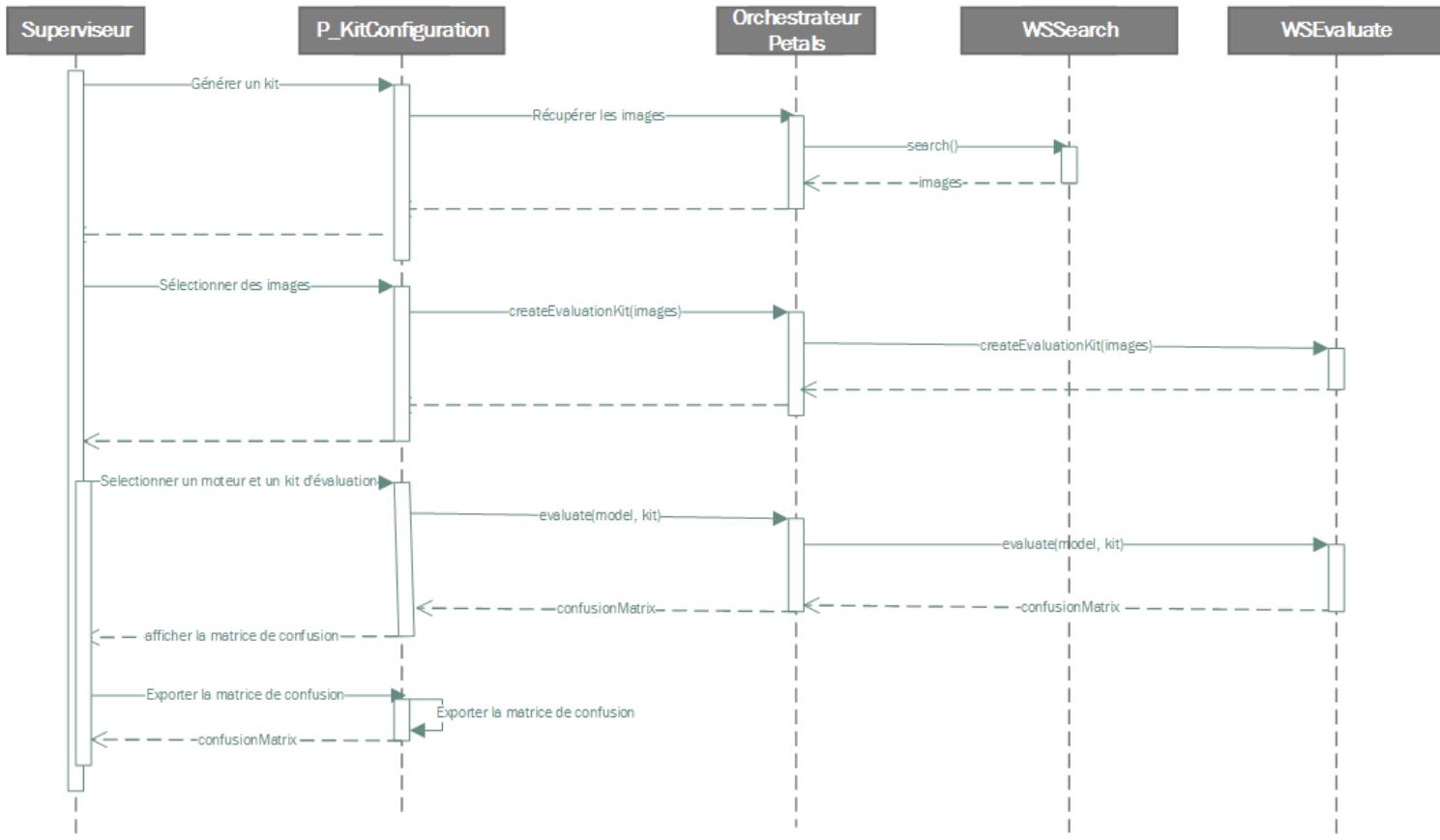
8 Éditer la taxonomie



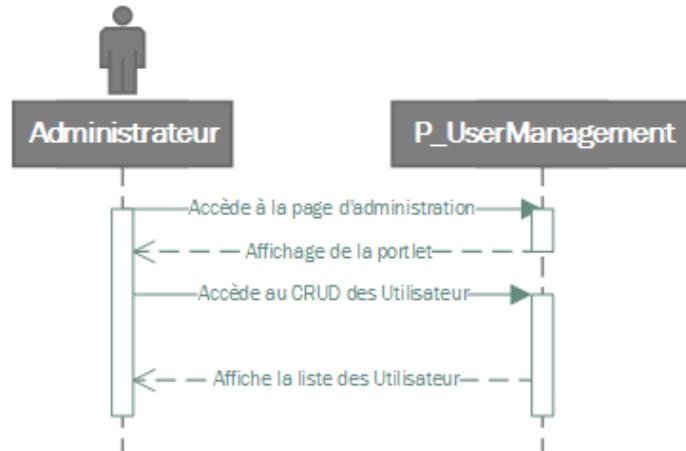
9 Gérer les moteurs de classification



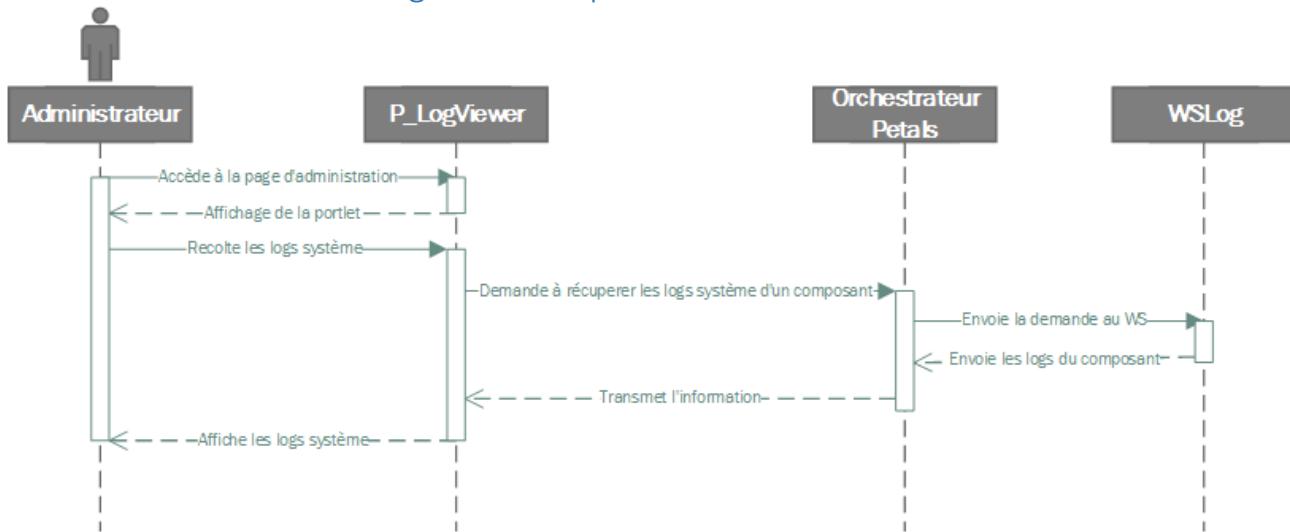
10 Évaluer un moteur de classification



11 Gérer les utilisateurs

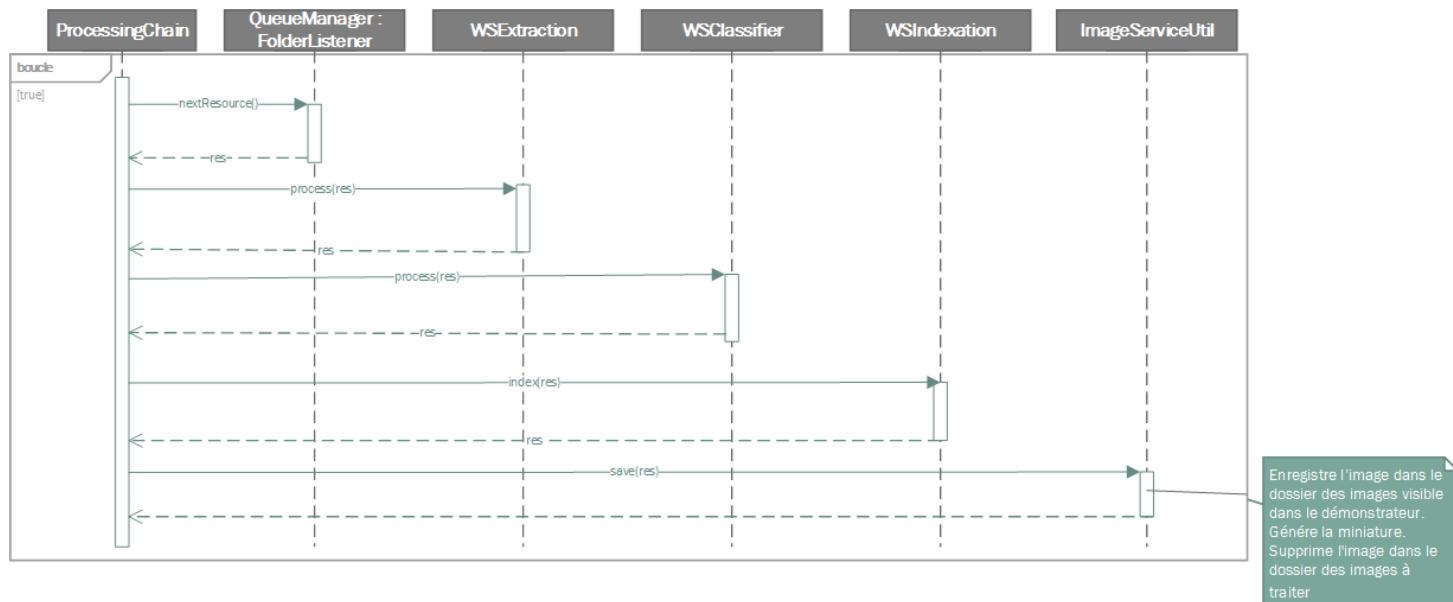


12 Accéder aux logs d'un composant



13 La chaîne de traitement

Le schéma suivant détaille les appels entre les composants dans le cas où une image à traiter est ajoutée dans le dossier écouté par la chaîne de traitement. Le processus traite chaque ressource renvoyée par le service de collecte ou uploadée par un superviseur, et appelle les différents services de traitement.



G Justifications techniques

L'architecture proposée est conforme au format de la plateforme WebLab. Cela signifie que l'application s'appuie sur le socle technique « WebLab Core » et que les services développés sont compatibles avec le modèle d'échange WebLab 1.2.5. Les services implémentent ainsi au moins une interface générique du modèle WebLab. Les données échangées entre les différents composants du démonstrateur sont des Resource WebLab. De plus, le format RDF a été utilisé afin de pouvoir annoter au maximum les documents.

Concernant les vues du démonstrateur, le choix s'est porté sur le portail Liferay qui est un portail open source de gestion de contenu écrit en Java et créé en 2000. Cela permet ainsi une intégration des portlets assez aisée. De plus, Liferay s'intègre à un serveur d'application Tomcat. Enfin Liferay est compatible avec le format HTML5.

Pour la gestion de la chaîne de traitement, le choix a été porté sur Petals ESB qui est un bus d'entreprise orienté services (ESB) hautement distribué. Il permet ainsi d'intégrer les applications de manière interopérable, et flexible.

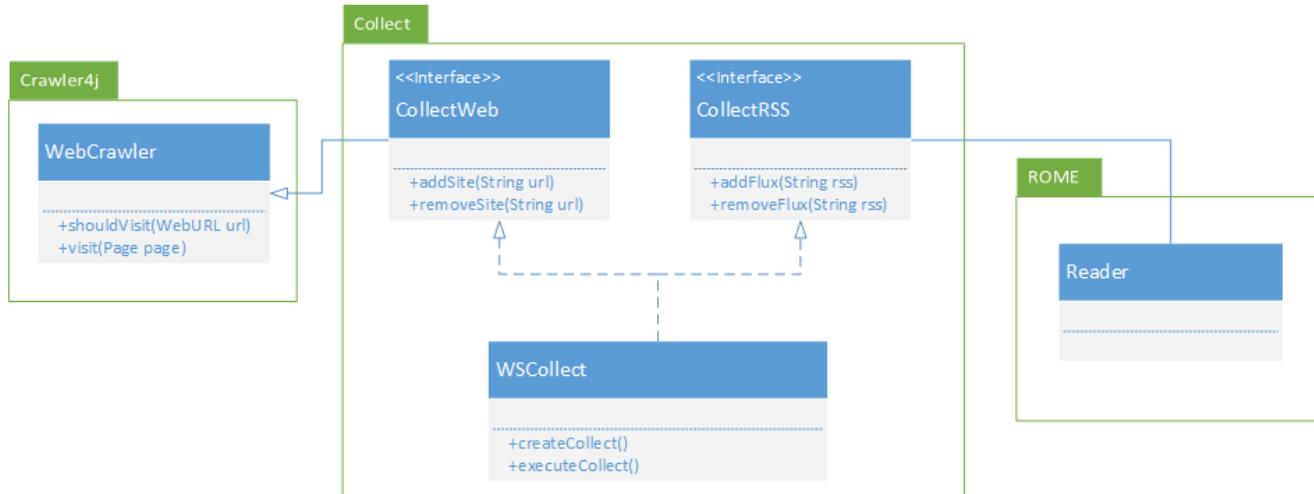
L'outil de log sera LogBack. Il permet de gérer et reporter toutes les activités ainsi que les éventuelles erreurs et crashes de l'application. Il est plus rapide que log4j. C'est une implémentation de slf4j.

VI Architecture du lot de collecte et d'extraction (Clément Jehannet)

A Architecture statique

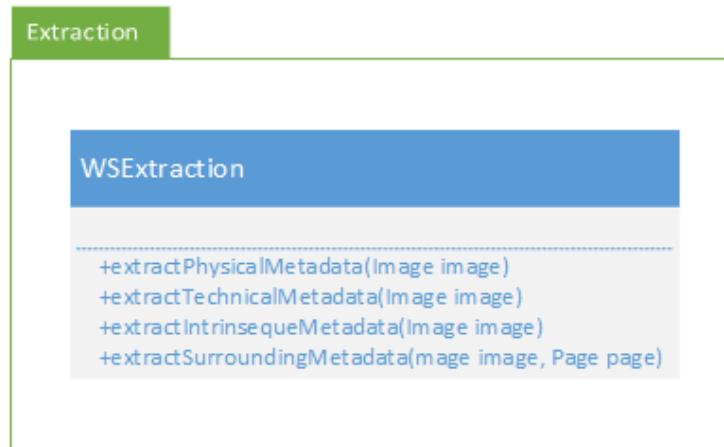
Voici la description de la structure des différents composants constituant ce lot, à savoir les composants de Collecte, d'Extraction et d'Alerte, de Log ainsi que la gestion de la persistance.

1 Description du constituant Collecte

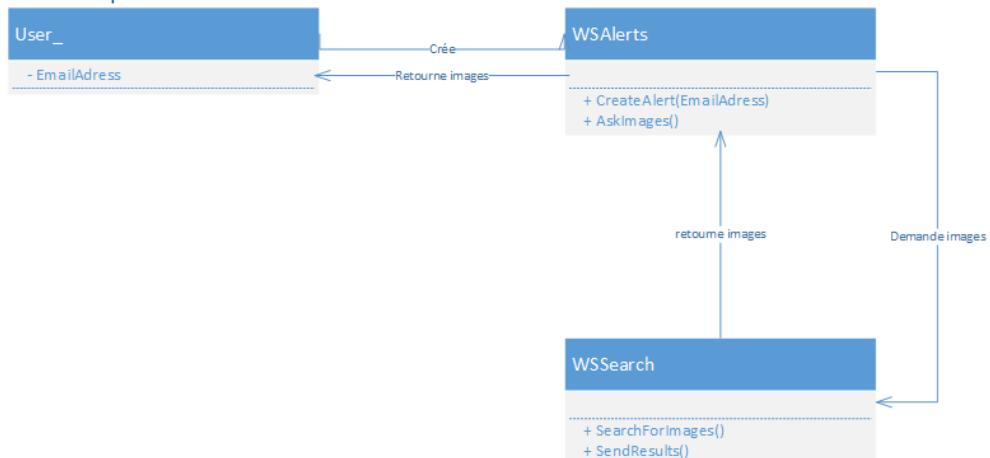


La collecte consiste en l'enregistrement d'images venues de site web ou de flux RSS. Pour ce faire, un web service WSCollect se chargera de l'automatisation de cette tâche et utilisera les outils Crawler4J et ROME.

2 Description du constituant Extraction



3 Description du constituant Alertes



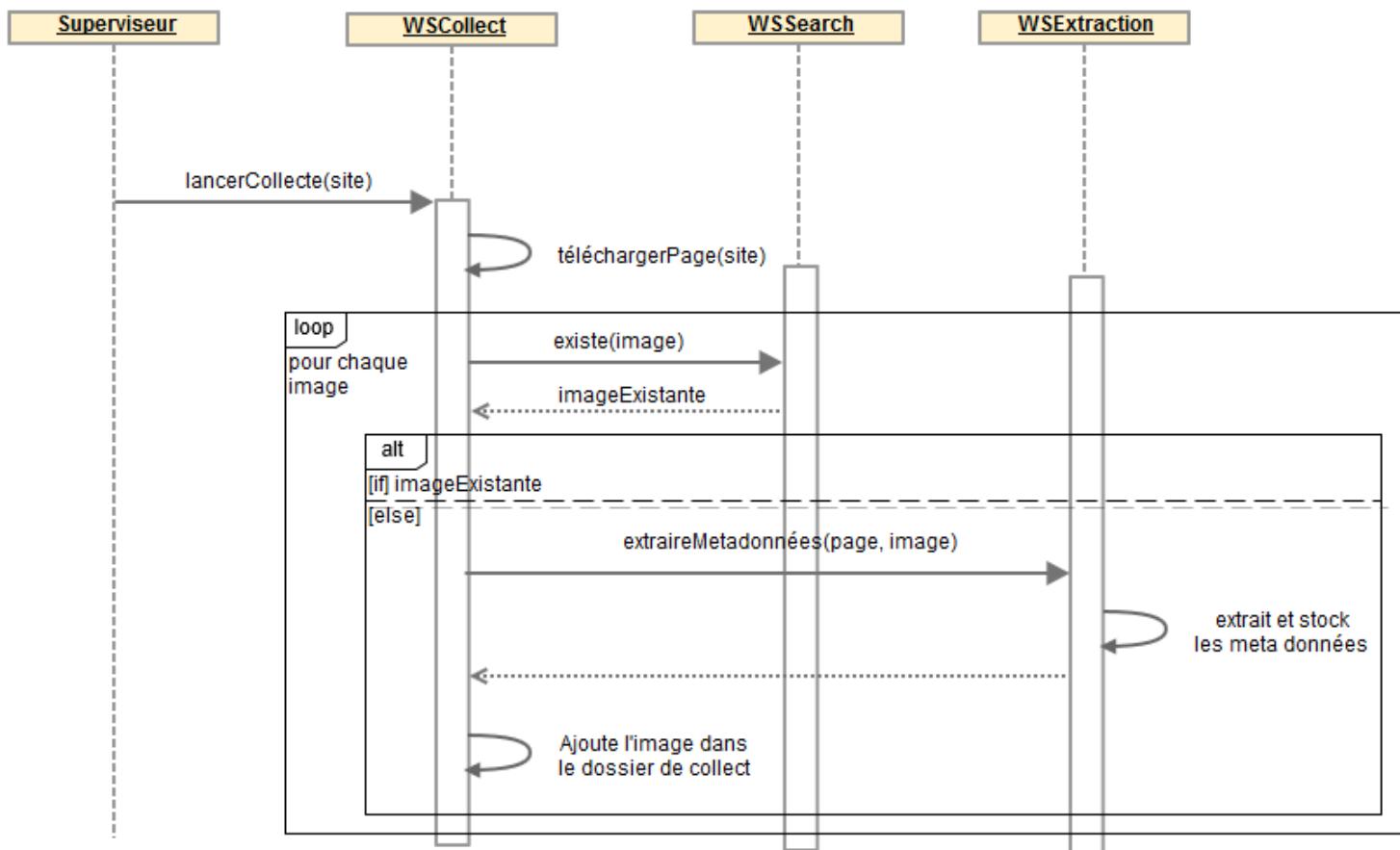
Un utilisateur crée une alerte. L'alerte demande une recherche sur les critères donnés par l'utilisateur. Lorsqu'il y a un résultat à la recherche, le service *WSAlerts* renvoie à l'utilisateur les images correspondantes.

4 Description du constituant Log

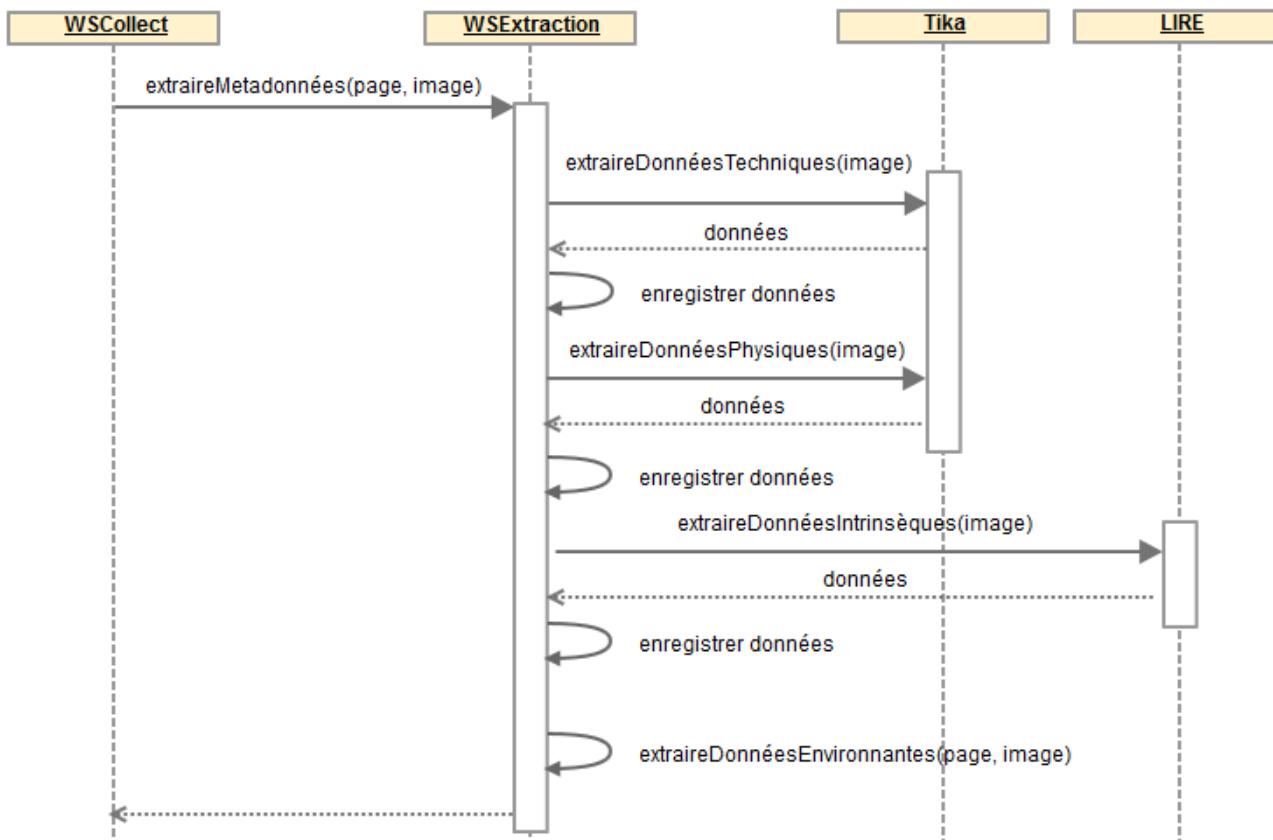
Il est possible d'accéder aux logs depuis Liferay. Quand le service de logs reçoit le message SOAP lui indiquant d'exposer les logs (filtrés ou non), il lit dans le fichier et renvoi ce qui a été demandé.

B Fonctionnement dynamique

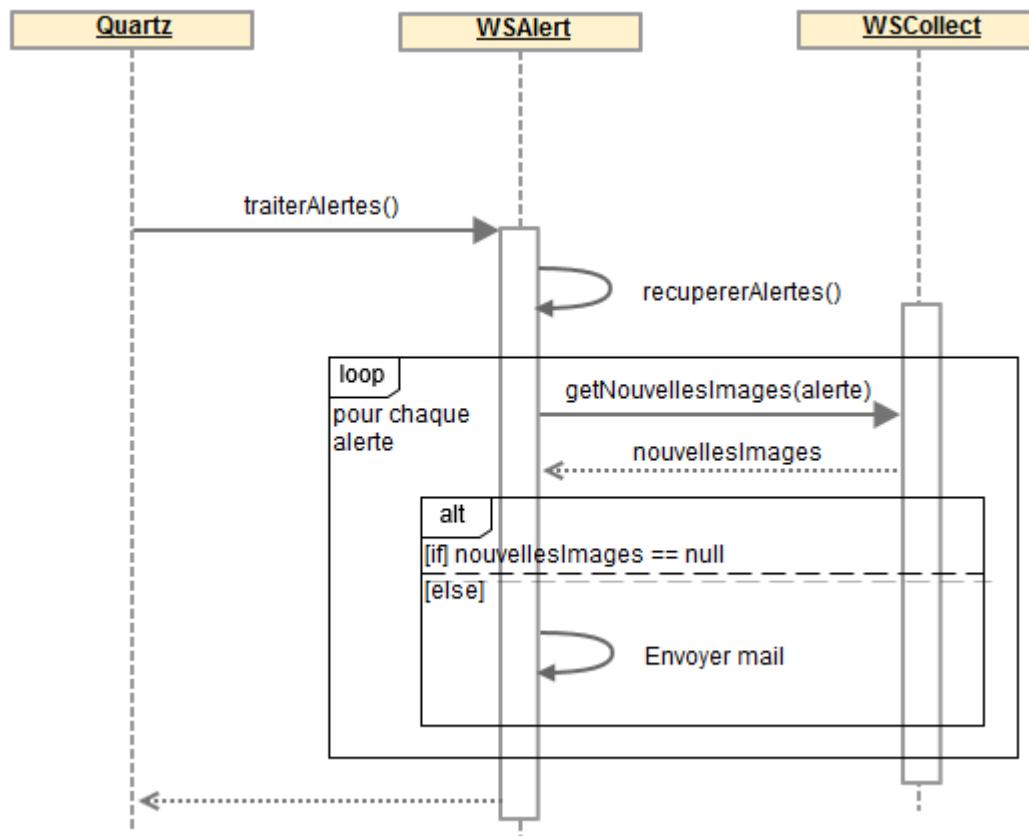
1 Effectuer une collecte sur un site web



2 Extraire les métadonnées d'une image



3 Traiter les alertes



C Justifications techniques

Le composant **Collecte** se sert des outils **Crawler4J** et **ROME** afin de collecter des images respectivement sur une page web et dans un flux RSS.

Le composant **Extraction** utilise l'outil **Apache Tika** afin d'extraire toutes les métadonnées nécessaires à l'application.

Le composant **Alerte** se sert de l'outil **Quartz** afin d'automatiser l'envoie des mails aux utilisateurs ayant créés des alertes.

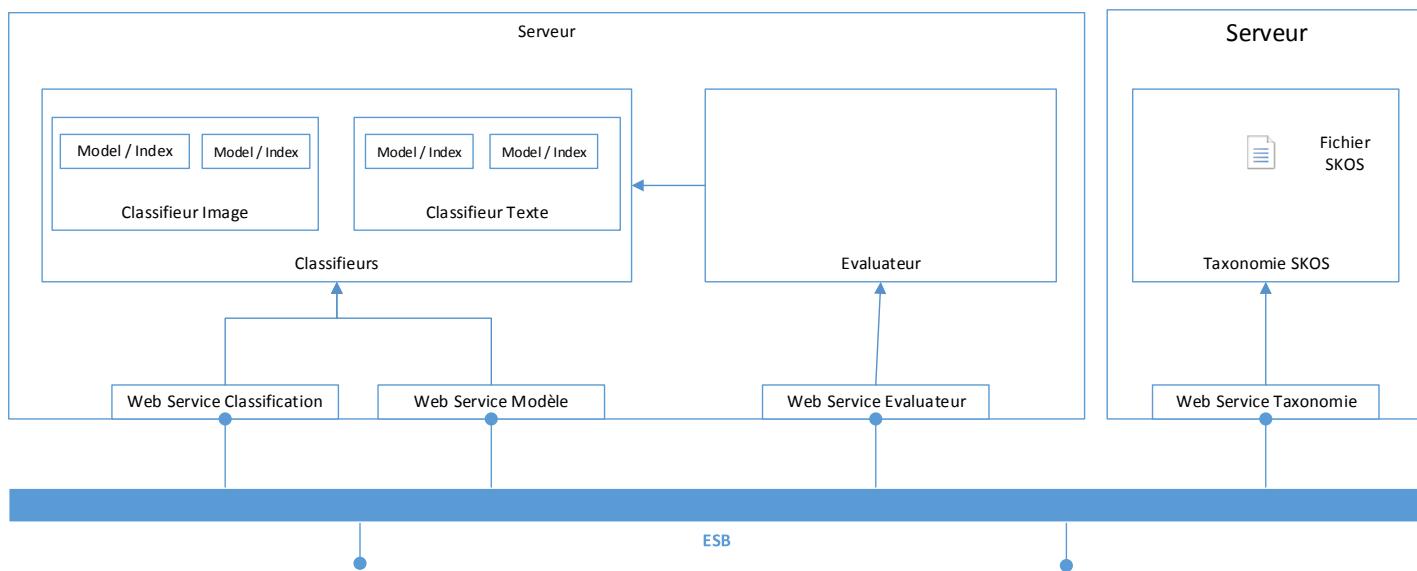
Tous ces outils ont été choisis dans un premier temps pour leur fait d'être écrits en Java, langage utilisé par WebLab et donc facilitant leur intégration. Ce sont tous des outils open-source et leur facilité d'utilisation et de mise en place ont de même contribués à leur sélection.

VII Architecture du lot de classification et de taxonomie (Kevin Cauchois)

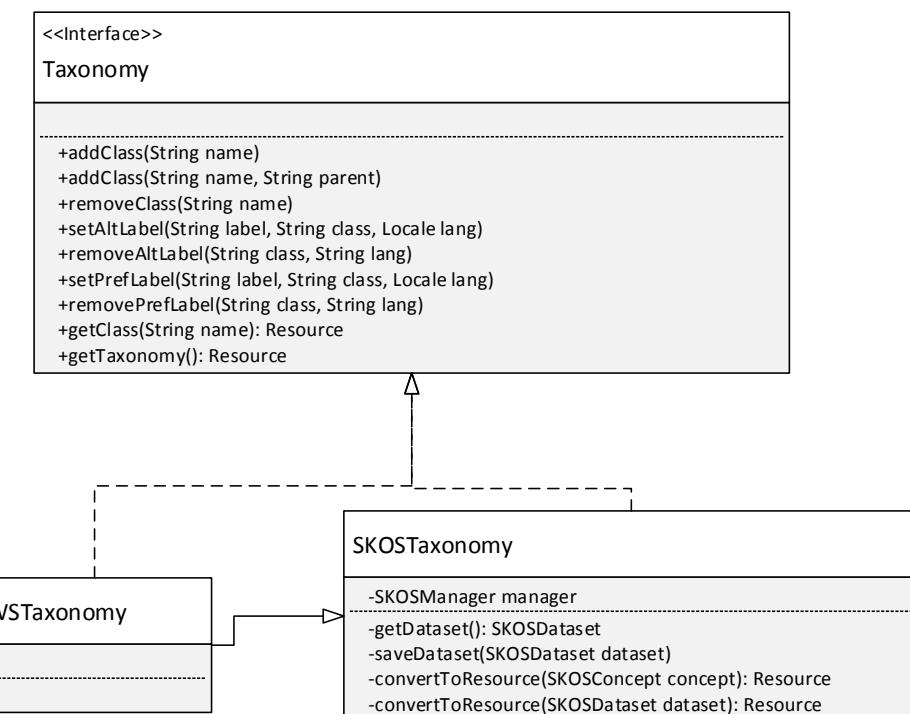
A Architecture statique

En plus de répondre aux exigences du lot de classification et de gestion de la taxonomie et aux services web demandés par l'équipe dirigeante, nous avons pensé notre lot de sorte que celui-ci soit le plus réutilisable possible. Nous souhaitons donc qu'ajouter un nouveau système de classification ou un autre système de gestion de la taxonomie soit le plus simple possible.

C'est pourquoi deux patrons de conception seront utilisés plusieurs fois dans notre architecture. Le premier est le patron proxy qui permettra aux services web de venir s'intégrer au système sans avoir besoin de modifier le code métier de l'application. Le second est le patron composite qui gérera un ensemble de classifieurs. Grâce à ce dernier, ajouter un nouveau système de classification se fera en l'ajoutant au composant composite. Voici le diagramme général du lot de classification et de gestion de la taxonomie :



1 Description détaillée du composant Taxonomie



L'interface *Taxonomy* permet la gestion de la taxonomie, qu'il s'agisse d'une taxonomie des sports ou non. On voit ici l'utilisation du patron proxy car les deux classes implémentent l'interface et la classe *WSTaxonomy* hérite de *SKOSTaxonomy*. La classe *SKOSTaxonomy* s'occupera de manipuler la taxonomie des sports au format SKOS tandis que la classe *WSTaxonomy* contiendra le code nécessaire au web service et déléguera toutes ses méthodes à sa classe héritée. Si, dans une utilisation future, on ne souhaite plus utiliser SKOS pour gérer la taxonomie, il sera alors simple de changer le composant. Elle utilise un *SKOSManager*. C'est un objet de la Skos API permettant de charger et sauvegarder des fichiers SKOS. Voici la description des méthodes de l'interface :

- *addClass(String name)*
 - Ajoute une classe de nom « name » à la taxonomie
- *addClass(String name, String parent)*
 - Ajoute une classe de nom « name » à la taxonomie ayant comme parent le concept « parent »
- *removeClass(String name)*
 - Supprime la classe « name » de la taxonomie
- *setAltLabel(String label, String class, Locale lang)*
 - Met « label » comme label alternatif de la classe « class » dans la langue « lang »
- *removeAltLabel(String class, String lang)*
 - Supprime le label alternatif de la classe « class » dans la langue « lang »
- *setPrefLabel(String label, String class, Locale lang)*
 - Met « label » comme label préféré de la classe « class » dans la langue « lang »
- *removePrefLabel(String class, String lang)*
 - Supprime le label préféré de la classe « class » dans la langue « lang »
- *getClass(String name) : Resource*
 - Retourne une resource correspondant à la classe « name »
- *getTaxonomy() : Resource*
 - Retourne la taxonomie

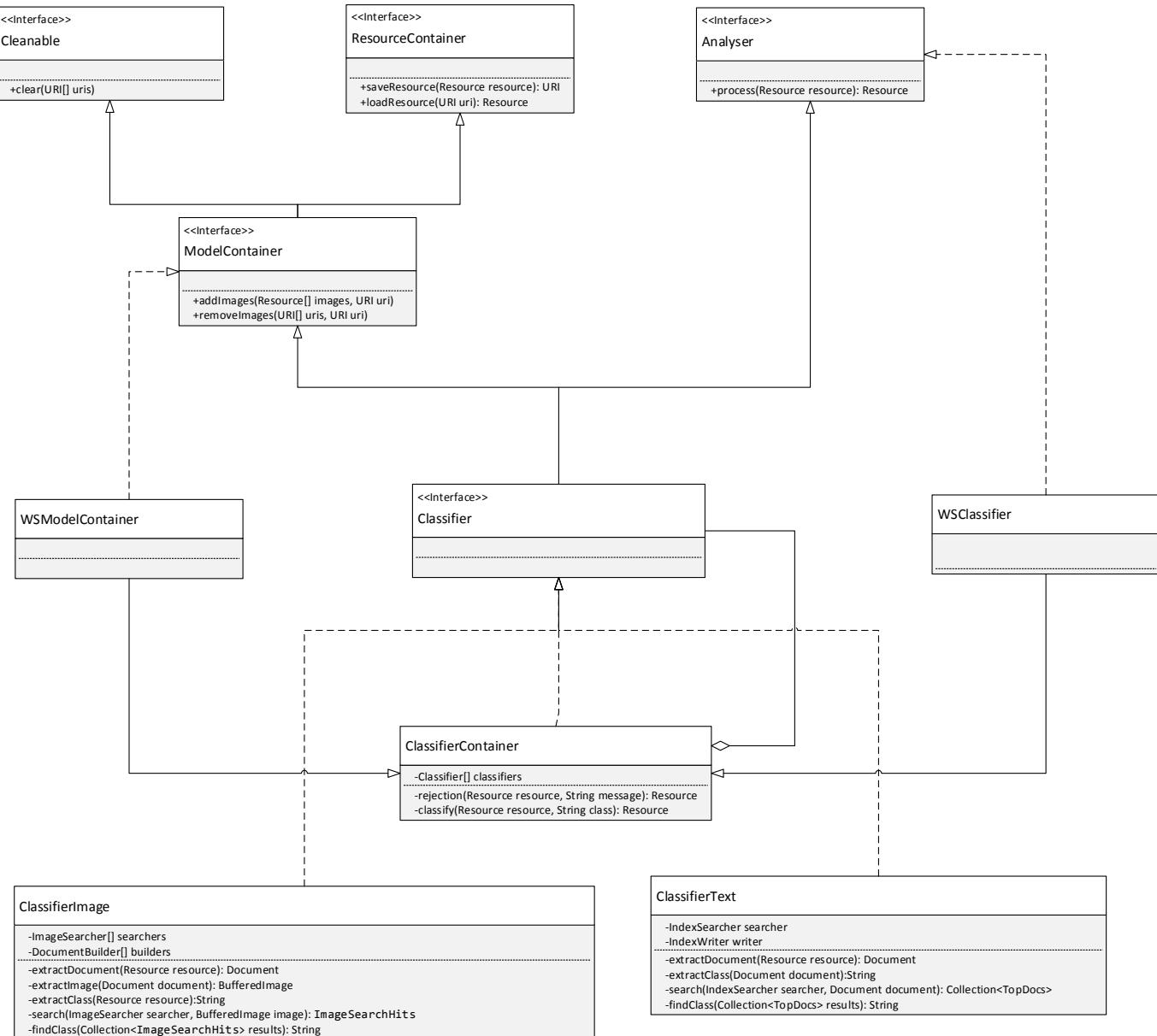
On voit ici l'utilisation du patron proxy car les deux classes implémentent l'interface et la classe *WSTaxonomy* hérite de *SKOSTaxonomy*. La classe *SKOSTaxonomy* s'occupera de manipuler la taxonomie des sports au format SKOS tandis que la classe *WSTaxonomy* contiendra le code nécessaire au service web et déléguera toutes ses méthodes à sa classe héritée. Si, dans une utilisation future, on ne souhaite plus utiliser SKOS pour gérer la taxonomie, il sera alors simple de changer le composant.

La classe *SKOSTaxonomy* possède en plus les méthodes suivantes :

- *getDataset() : SKOSDataset*
 - Retourne le fichier SKOS
- *saveDataset(SKOSDataset dataset)*
 - Sauvegarde le fichier SKOS
- *convertToResource(SKOSConcept concept) : Resource*
 - Converti la classe représentée par le concept en ressource
- *convertToResource(SKOSDataset dataset) : Resource*
 - Converti le fichier SKOS en ressource

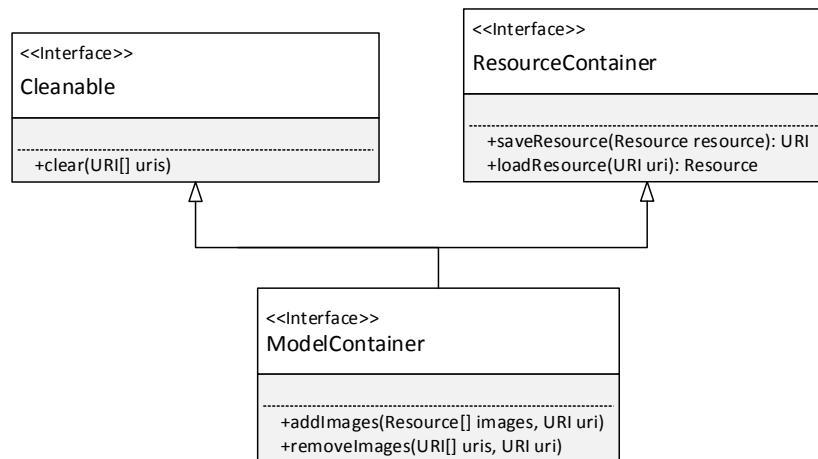
2 Description détaillée du composant Classification

Voici le diagramme répondant aux besoins du lot de classification :



a Description de l'interface ModelContainer

L'interface `ModelContainer` sert à gérer l'ensemble des modèles de classification et des images. Celui-ci étend les interfaces WebLab `Cleanable` et `ResourceContainer`.

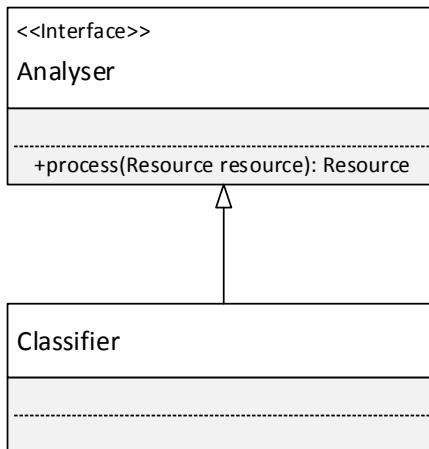


L'interface *ModelContainer* contient l'ensemble des méthodes suivantes :

- *saveResource(Resource resource) : URI*
 - Sauvegarde un nouveau modèle et retourne une URI identifiant celui-ci
- *loadResource(URI uri) : Resource*
 - Retourne le modèle correspondant à l'URI
- *clear(URI[] uris)*
 - Supprime les modèles correspondant aux URI
- *addImage(Resource[] images, URI uri)*
 - Ajoute les images et métadonnées au modèle correspondant à l'URI
 - Les images doivent avoir une URI, une classe, une image, et des métadonnées
- *removeImages(URI[] uris, URI uri)*
 - Supprime les images correspondant aux URI du modèle correspondant à l'URI

b Description de l'interface *Classifier*

L'interface *Classifier* est l'interface que doit étendre tout système de classification. Elle regroupe les méthodes nécessaires à la gestion de modèles, l'ajout de nouvelles images et à la classification. Celle-ci étend l'interface WebLab *Analyser*.



L'interface *Classifier* contient la méthode suivante :

- *process(Resource resource) : Resource*
 - Prend une image en paramètre et retourne la même resource annotée de sa classe et annotée d'une raison de rejet
 - L'image doit avoir une URI, une classe, une image, et des métadonnées

c Description de la classe *ClassifierContainer*

La classe *ClassifierContainer* implémente l'interface *Classifier*, c'est donc un système de classification. Cette classe respecte le patron de conception composite afin de contenir un ensemble d'implémentation de « *Classifier* ». Chacune des méthodes sont transférées à l'ensemble des classifieurs qu'il contient. Par exemple, *addImage(Resource[] images, URI uri)* ne fera qu'appeler la même méthode pour tous les autres classifieurs. La seule méthode qui n'aura pas ce comportement est la méthode *process(Resource resource): Resource*. Celle-ci appellera les méthodes *process* des classifieurs contenu par le composite.

Si tous les résultats sont identiques, il annotera la ressource avec la classe trouvée, si ce n'est pas le cas, il annotera la ressource de la raison du rejet. Pour notre lot, la classe « *ClassifierContainer* » contiendra deux classifieurs : *ClassifierImage* et *ClassifierText*.

d Description de la classe *WSModelContainer*

La classe *WSModelContainer* implémente l'interface *ModelContainer* et étend la classe *ClassifierContainer*. Cette classe est une implémentation du patron proxy et contient le code nécessaire pour exposer le service web. Elle délègue l'ensemble de ses méthodes à la classe *ClassifierContainer*.

e Description de la classe *WSClassifier*

La classe *WSClassifier* implémente l'interface *ModelContainer* et étend la classe *ClassifierContainer*. Cette classe est une implémentation du patron proxy, contient le code nécessaire pour exposer le web service et délègue sa méthode à la classe *ClassifierContainer*.

f Description de la classe *ClassifierImage*

La classe *ClassifierImage* implémente l'interface *Classifier*. C'est un système de classification d'images. Cette classe implémente les fonctions nécessaires à la gestion de son modèle et à la classification par images. Pour cela, elle utilisera la librairie LIRE. Voici la description des attributs de la classe :

- *ImageSearcher[] searchers*
 - Contient la liste des méthodes de recherche de LIRE que nous utiliserons. Elle contiendra à minima :
 - JCD : Recherche par répartition spatiale des couleurs et textures
 - CEDD : Recherche par couleur et la texture
 - OH : Recherche par couleurs et vecteurs
 - CL : Recherche par répartition spatiale des couleurs
 - EH : Recherche par texture, couleur, forme
 - SIFT : Recherche par vecteurs SIFT
 - SURF : Recherche par vecteurs SURF
- *DocumentBuilder[] builders*
 - Contient l'ensemble des builders nécessaire à LIRE pour l'indexation. Ces builders sont ceux correspondants aux méthodes de recherche que nous utiliserons.

Voici le détail des méthodes de la classe *ClassifierImage* :

- *extractDocument(Resource resource) : Document*
 - Extrait le document de la resource
- *extractImage(Document document) : BufferedImage*
 - Extrait l'image de la resource
- *extractClass(Document document) : String*
 - Extrait la classe d'une image
- *search(ImageSearcher searcher, BufferedImage image) : ImageSearchHits*
 - Lance une recherche d'une image sur un searcher et retourne l'ensemble des résultats
- *findClass(Collection<ImageSearchHits> results) : String*
 - Prend un ensemble de résultat pour en extraire la classe sportive de celle-ci. Nous utiliserons l'algorithme K-NN pour cette tâche.

g Description de la classe *ClassifierText*

La classe *ClassifierText* implémente l'interface *Classifier*. C'est un système de classification par le texte. Cette classe implémente les fonctions nécessaires à la gestion de son modèle et à la classification par textes. Pour cela, elle utilisera la librairie LUCENE. Son fonctionnement est très proche de celui de la classe *ClassifierImage*. Voici la description des attributs de la classe :

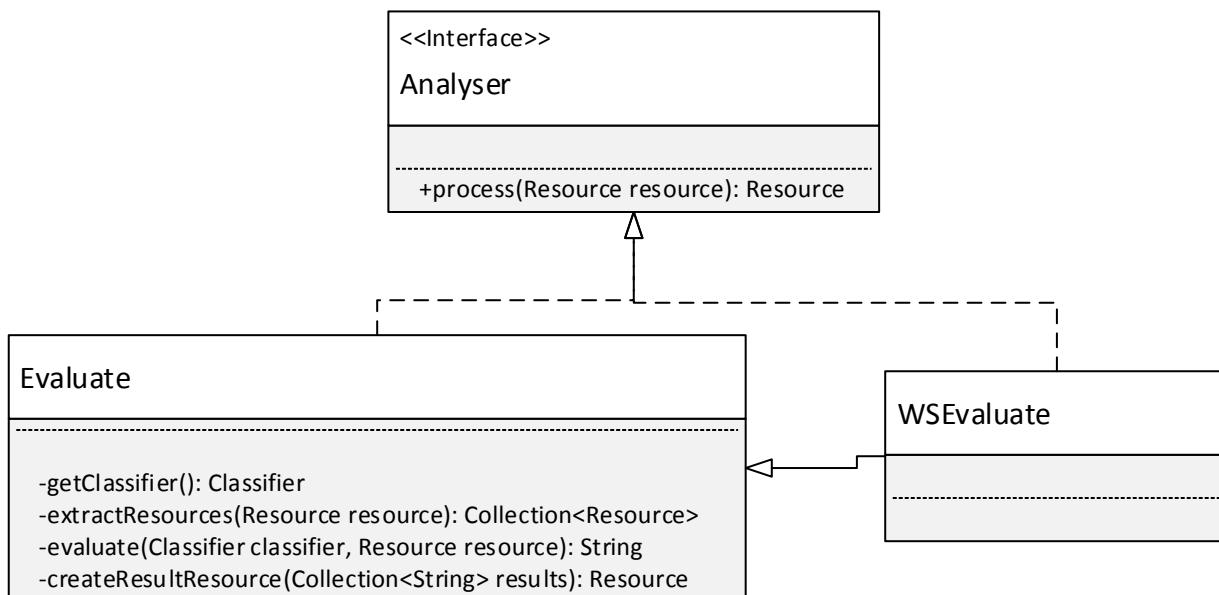
- *IndexSearcher searcher*
 - Permet la recherche dans les documents indexés.
- *IndexWriter writer*
 - Permet l'ajout de document dans l'index.

Voici le détail des méthodes de la classe *ClassifierText* :

- *extractDocument(Resource resource) : Document*
 - Extrait le document de la resource
- *search(IndexSearcher searcher, Document document) : Collection<TopDocs>*
 - Lance une recherche d'un texte sur le searcher et retourne l'ensemble des résultats
- *findClass(Collection<ImageSearchHits> results) : String*
 - Prend un ensemble de résultat pour en extraire la classe sportive de celle-ci. Nous utiliserons l'algorithme K-NN pour cette tâche.

3 Description détaillée du composant Evaluation

Voici le diagramme permettant d'évaluer un moteur de classification :



a Description de la classe *WSEvaluate*

La classe *WSEvaluate* implémente l'interface *Analyser* et étend la classe *Evaluate*. Cette classe est une implémentation du patron proxy et contient le code nécessaire pour exposer le web service. Elle délègue sa méthode à la classe *Evaluate*.

b Description de la classe *Evaluate*

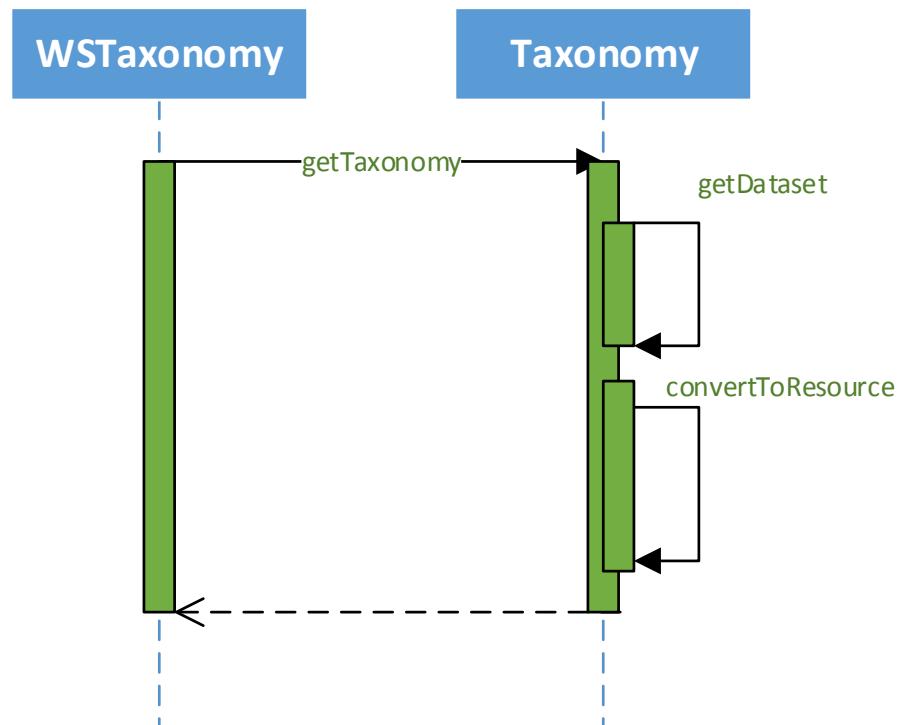
La classe *Evaluate* implémente l'interface *Analyser* et contient le code nécessaire à l'évaluation d'un classifieur. Voici le détail des méthodes de la classe *Evaluate* :

- *process(Resource resource) : Resource*
 - Prend un ensemble d'images annotées de leurs classes en paramètre
 - Retourne une matrice de confusion
- *getClassifier() : Classifier*
 - Retourne le classifieur
- *extractResources(Resource resource) : Collection<Resource>*
 - Extrait les resources de la resource
- *evaluate(Classifier classifier, Resource resource) : String*
 - Evalue le classifieur
- *createResultResource(Collection<String> results) : Resource*
 - Transforme l'évaluation en une ressource qui sera renvoyée

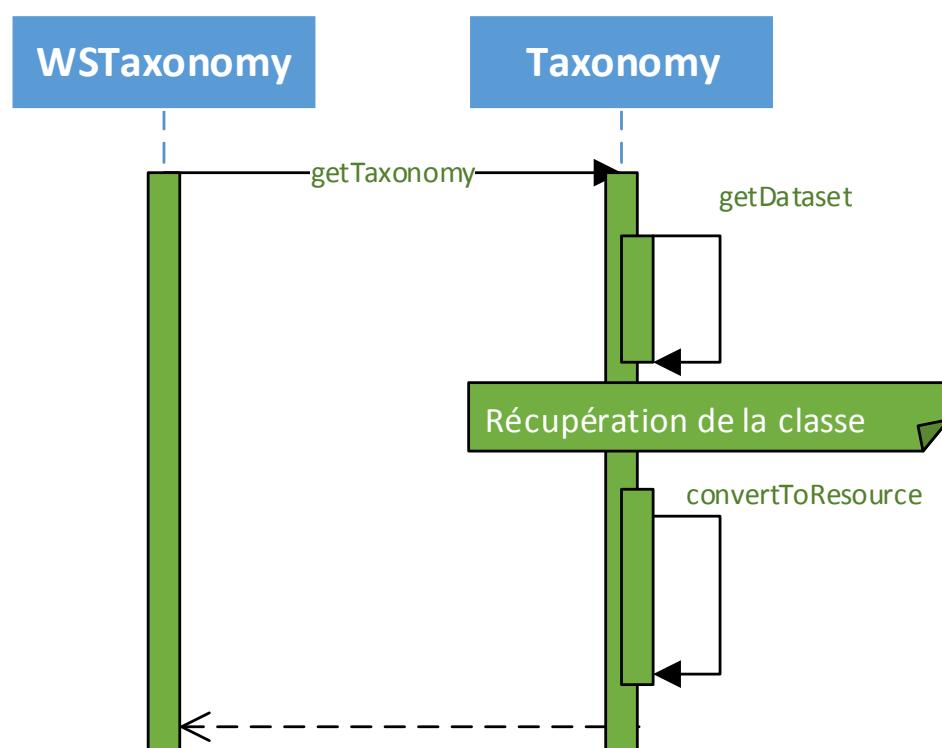
B Fonctionnement dynamique

1 Opération de lecture sur la taxonomie

a Récupérer la taxonomie SKOS

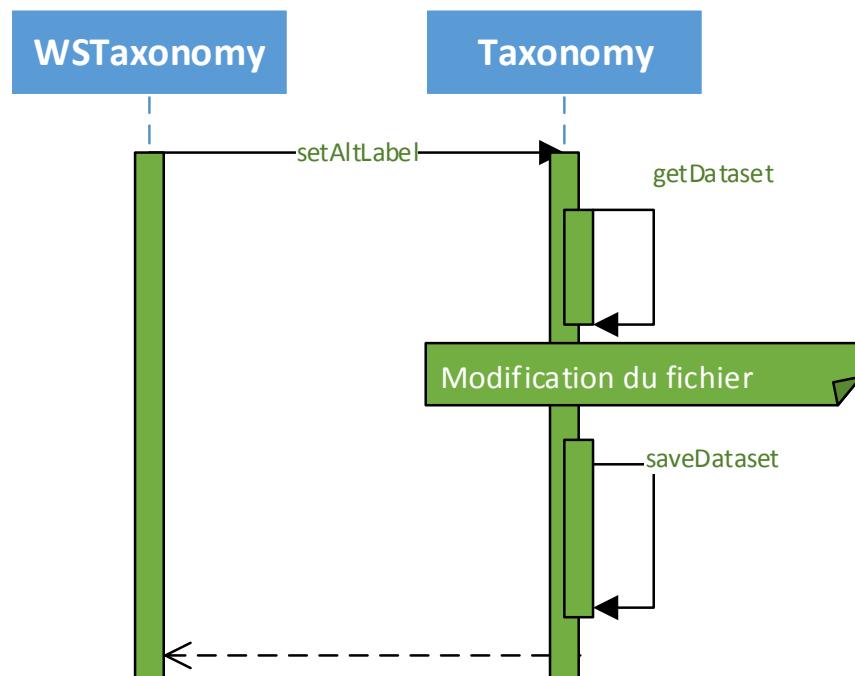


b Récupérer un concept de la taxonomie



2 Opération d'écriture sur la taxonomie

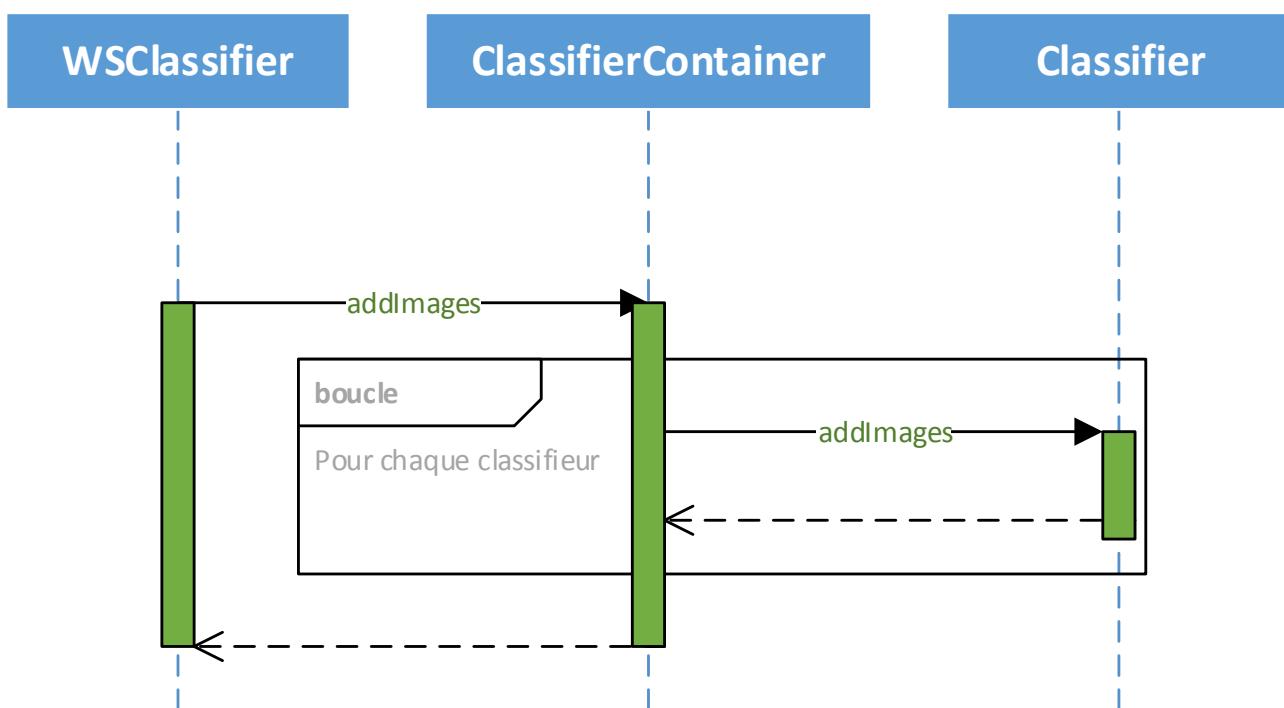
L'ensemble des méthodes de la modification de la taxonomie respectera le même algorithme. Voici un exemple pour l'ajout d'un label alternatif :



3 Ajouter des images à un modèle

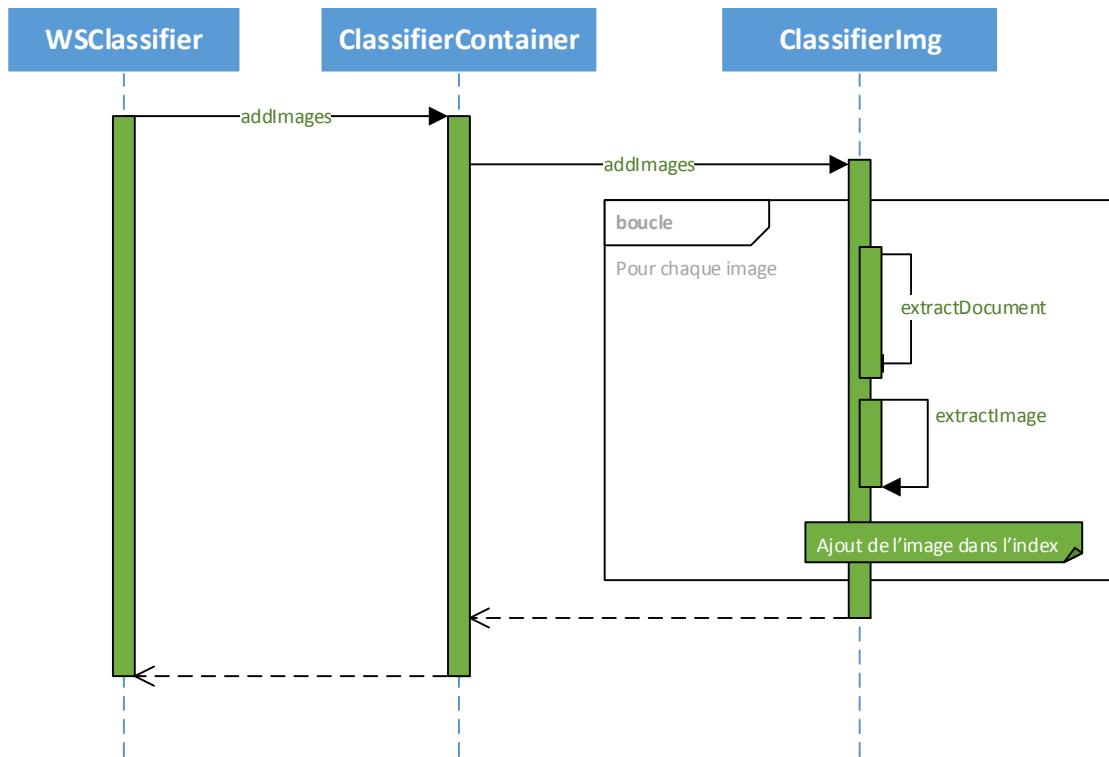
a Ajouter des images au composite

L'ensemble des méthodes de la gestion des modèles pour la classe *ClassifierContainer* respectera le même algorithme. Voici un exemple pour l'ajout d'images à un modèle :



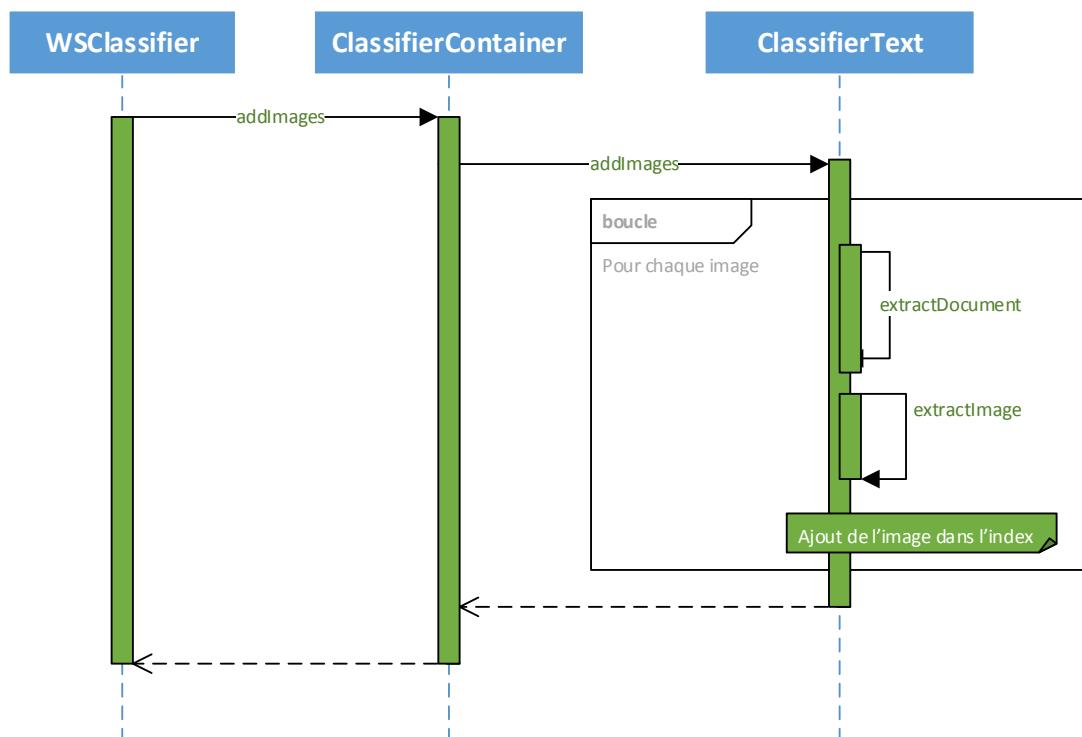
b Ajouter des images au moteur image

L'ensemble des méthodes de la gestion des modèles pour la classe *ClassifierImage* respectera le même algorithme. Voici un exemple pour l'ajout d'images à un modèle :



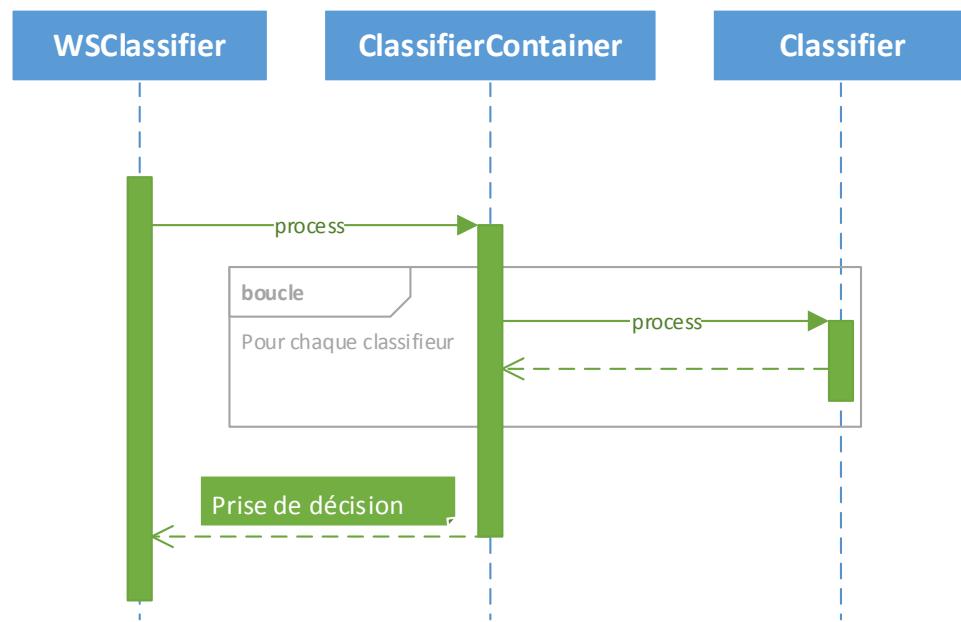
c Ajouter des images au moteur des métadonnées

L'ensemble des méthodes de la gestion des modèles pour la classe *ClassifierText* respectera le même algorithme. Voici un exemple pour l'ajout d'images à un modèle :

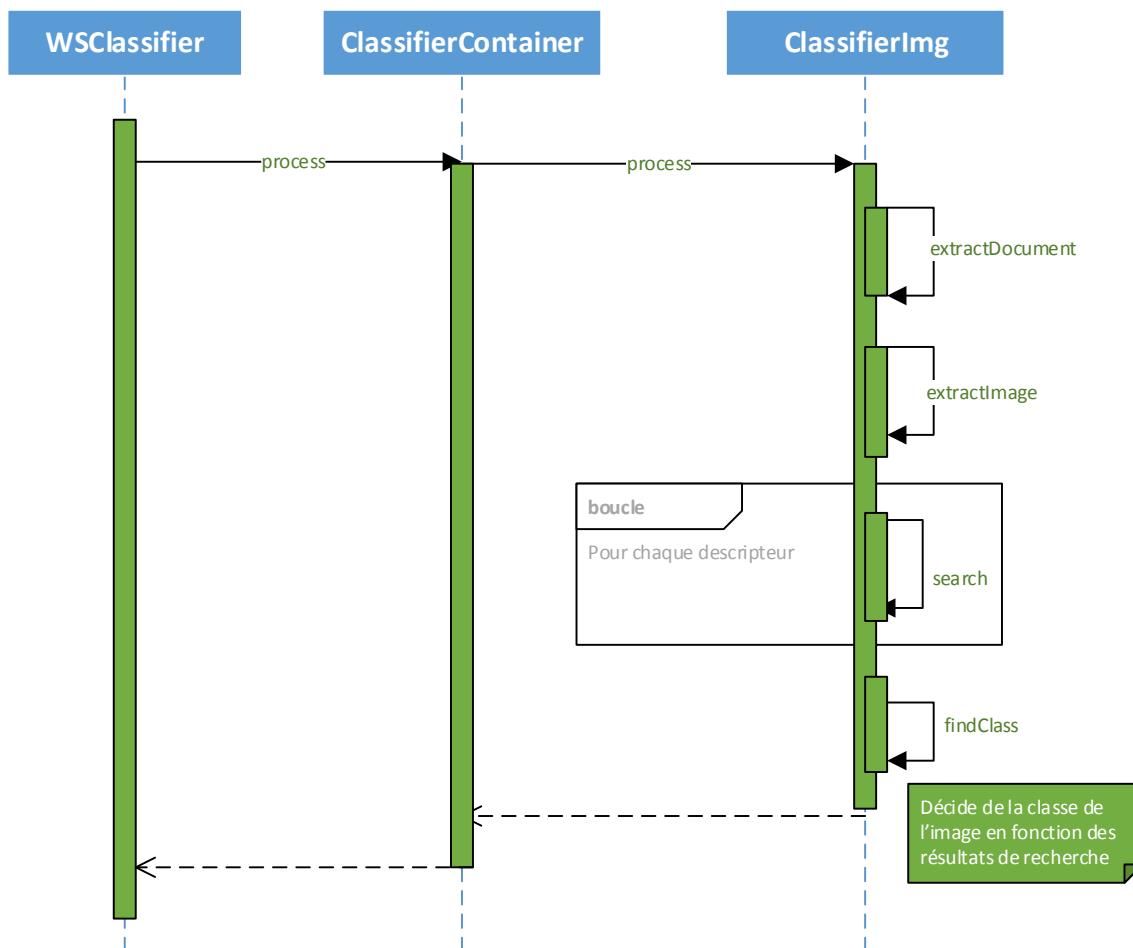


4 Classer une image

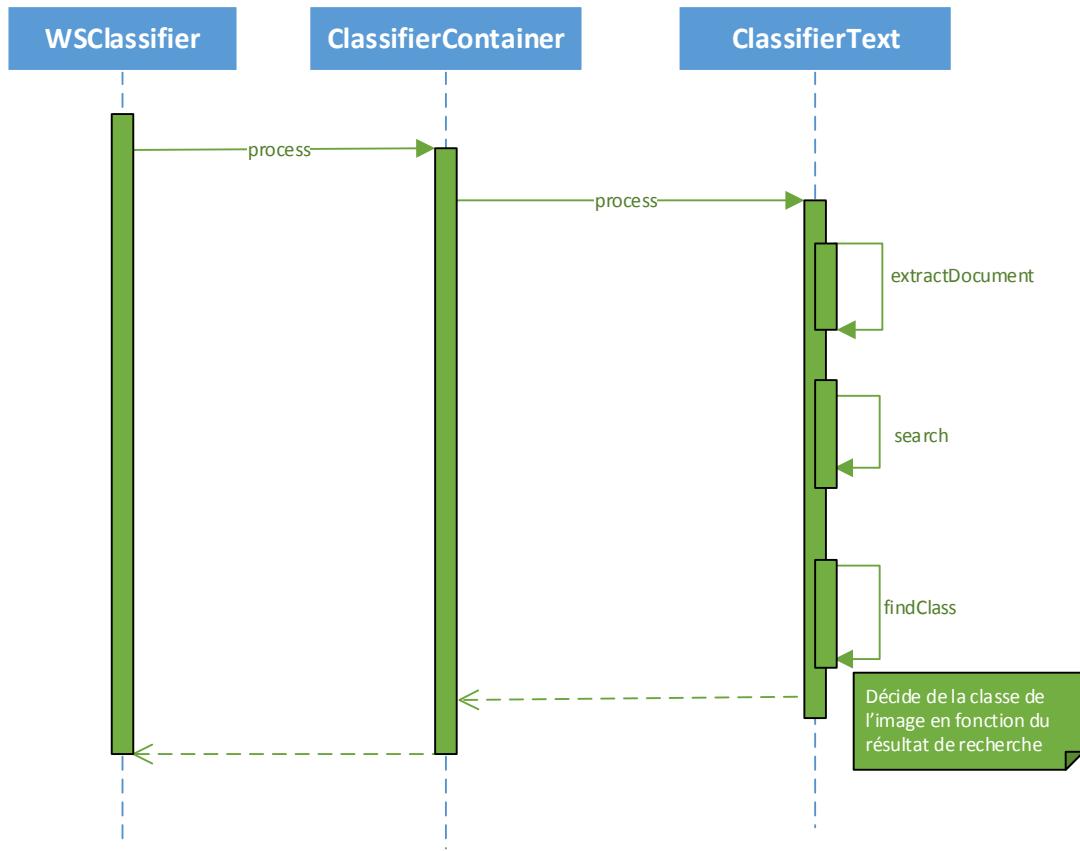
a Classement d'une image par le composite



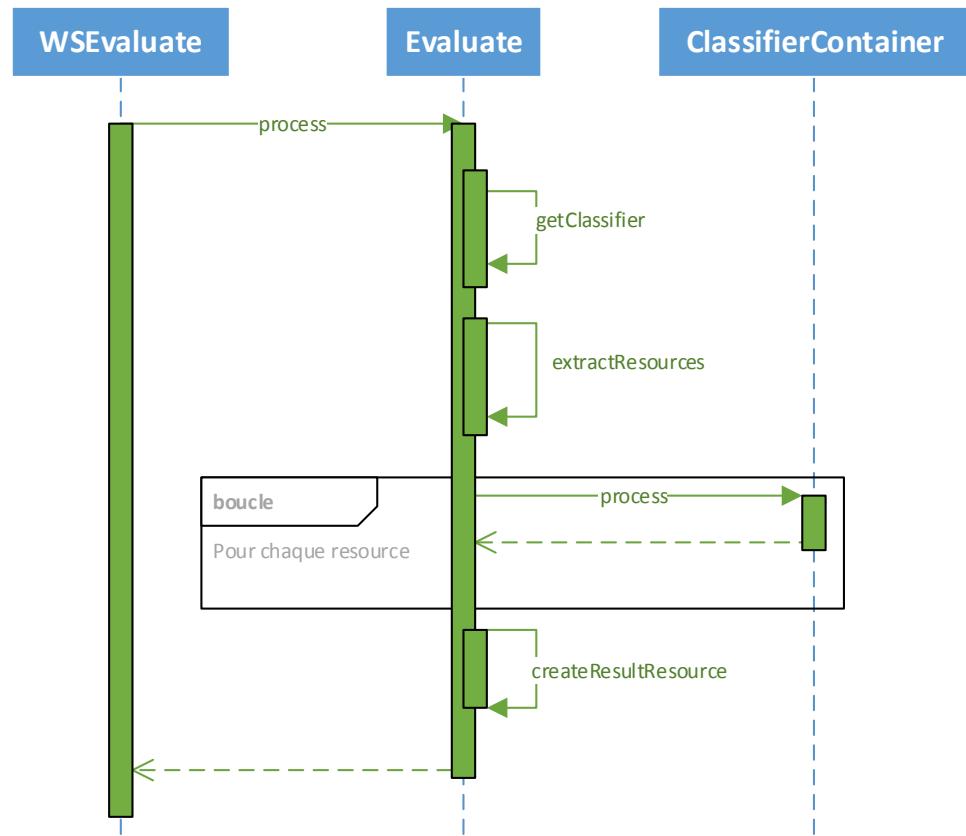
b Classification d'une image par similarités images



c Classification d'une image par comparaison de métadonnées textuelles



5 Évaluer un modèle de classification



C Justifications techniques

Afin de réaliser le lot classification et taxonomie, nous aurons besoin de trois librairies externes.

1 Lire

LIRE est une librairie d'indexation et de recherche d'image. Nous l'utiliserons pour réaliser la classification image. Notre choix s'est porté sur celle-ci car elle permet de répondre à toutes les exigences du lot en plus d'être open-source.

2 Lucene

Lucene est une librairie d'indexation et de recherche de documents textes. Nous l'utiliserons pour réaliser la classification texte. Notre choix s'est porté sur celle-ci car elle permet de répondre à toutes les exigences du lot en plus d'être open-source.

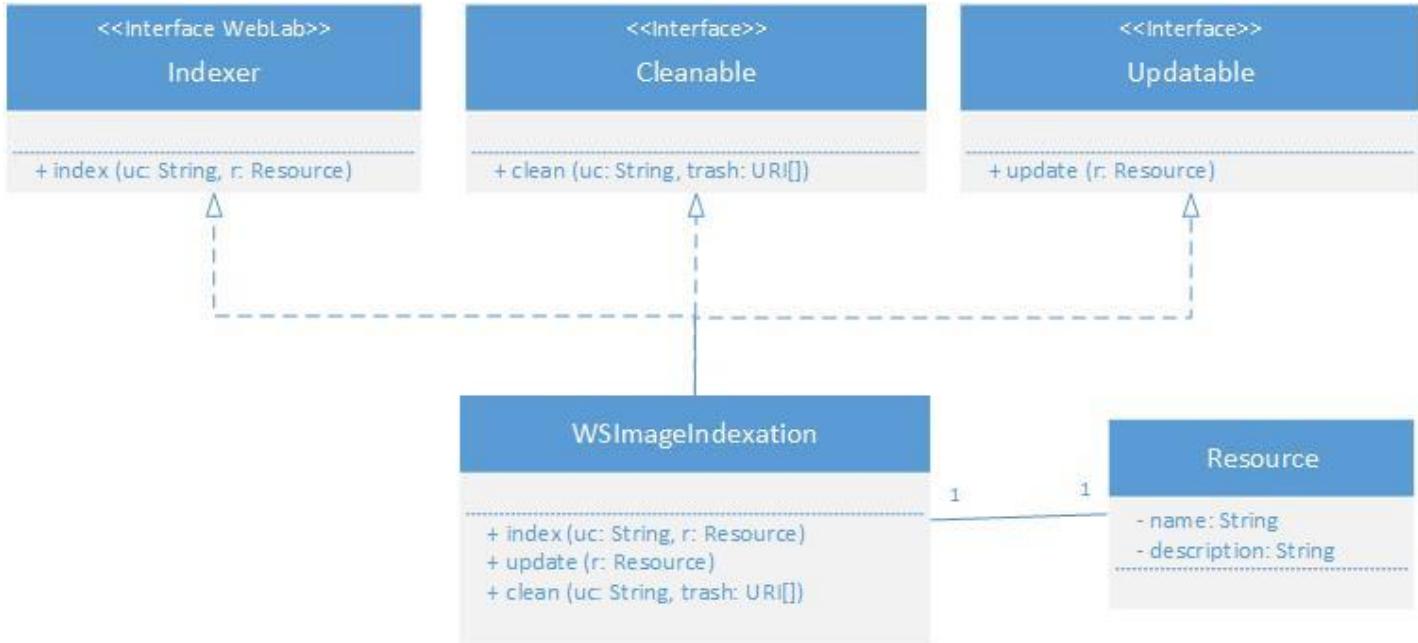
3 Skos API

Skos API est une librairie de manipulation de fichier Skos. Nous l'utiliserons pour manipuler la taxonomie du projet SPORTIFS. Notre choix s'est porté sur celle-ci car elle est très simple d'utilisation et suffit à répondre aux exigences du lot en plus d'être open-source.

VIII Architecture du lot d'indexation (Hacène Ingrachen)

A Architecture statique

1 Indexation des images



Le service web *WSImageIndexation* est composé de l'ensemble des modules de création d'indexes à partir des caractéristiques visuelles des images. Pour cela, le service implémente l'interface *Indexer* fournie par WebLab et utilise les méthodes d'indexation implémentées dans la librairie LIRE. Le service implémente aussi les interfaces *Cleanable* et *Updatable* pour les besoins de la modification ou la suppression d'un document dans l'index.

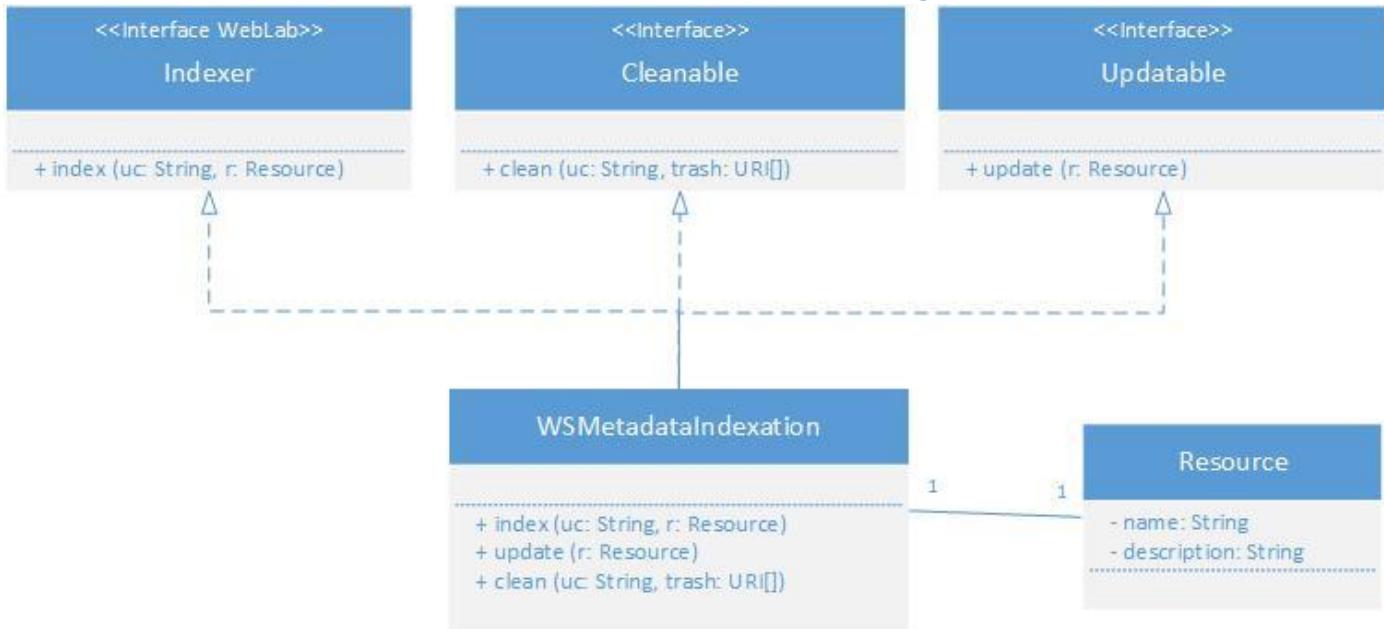
WSImageIndexation reçoit les images à indexer et crée ensuite un document Lucene pour chaque image. L'ensemble des documents sont ajoutés à l'index.

La librairie LIRE nous fournit une classe *DocumentBuilder* qui nous permet de créer un document Lucene. La classe *DocumentBuilderFactory* implémente les méthodes nécessaires pour créer un document spécifique à chaque caractéristique visuelle. Notre service se charge d'instancier les classes et d'utiliser les méthodes pour créer un document qui correspond à la caractéristique visuelle qu'on choisit pour l'image que l'on veut indexer. Le document est ensuite ajouté à notre indexe créé en utilisant la classe *IndexWriter*.

Il est cependant possible d'indexer les images en prenant compte de plusieurs descripteurs visuelles à la fois. Pour cela, LIRE nous fournit la classe *ChainedDocumentBuilder* qu'on peut utiliser pour créer notre builder avec lequel on construit le document. Le document résultant renferme alors plusieurs descripteurs visuels.

Au final, nous avons un index qui est prêt à être utilisé par les fonctions de recherche, d'édition et de suppression des images.

2 Indexation des métadonnées extraites des images

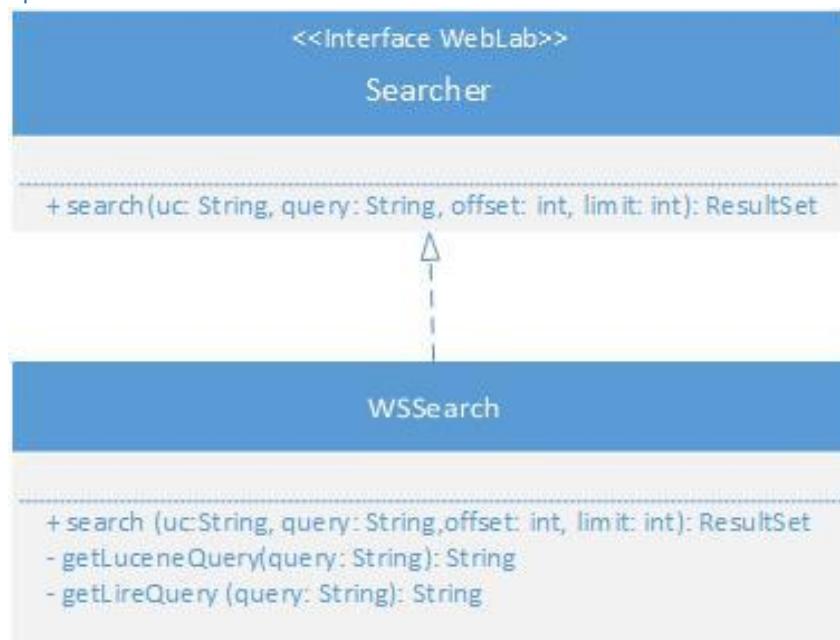


WSMetadataIndexation est le composant responsable de la création, de l'édition et de la suppression de l'index des métadonnées des images. Pareil que *WSImageIndexation*, il implémente les interfaces **Indexer**, **Cleanable** et **Updatable** mais utilise la librairie Lucene puisque il s'agit cette fois du texte.

La librairie Lucene nous fournit notamment:

- *IndexWriter*: nous permet de créer ou modifier un index de métadonnées. Il implémente de méthodes permettant l'ajout, la suppression ou la modification d'un document dans l'index.
- *Analyzer*: Permet un traitement préalable des données avant leur indexation. Les données textes sont converties en termes et sont ensuite stockées dans l'index

3 Composant de recherche

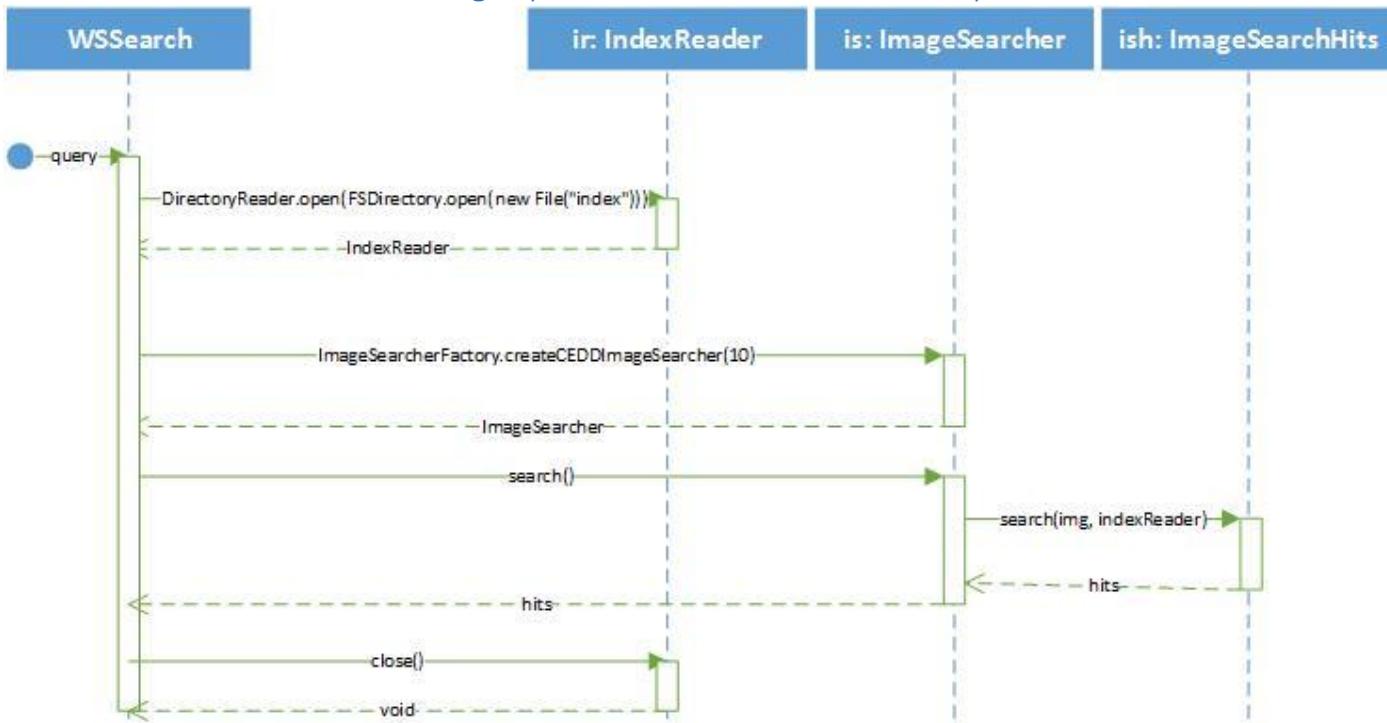


Le service web **WSSearch** englobe tous les modules de recherche des images par similarité en comparant les descripteurs visuels ou par données textuelles. Le service utilise les bibliothèques LIRE et Lucene pour chercher les images avec *IndexSearcher* pour retrouver dans l'index toutes les images correspondant à la requête reçue par celui-ci.

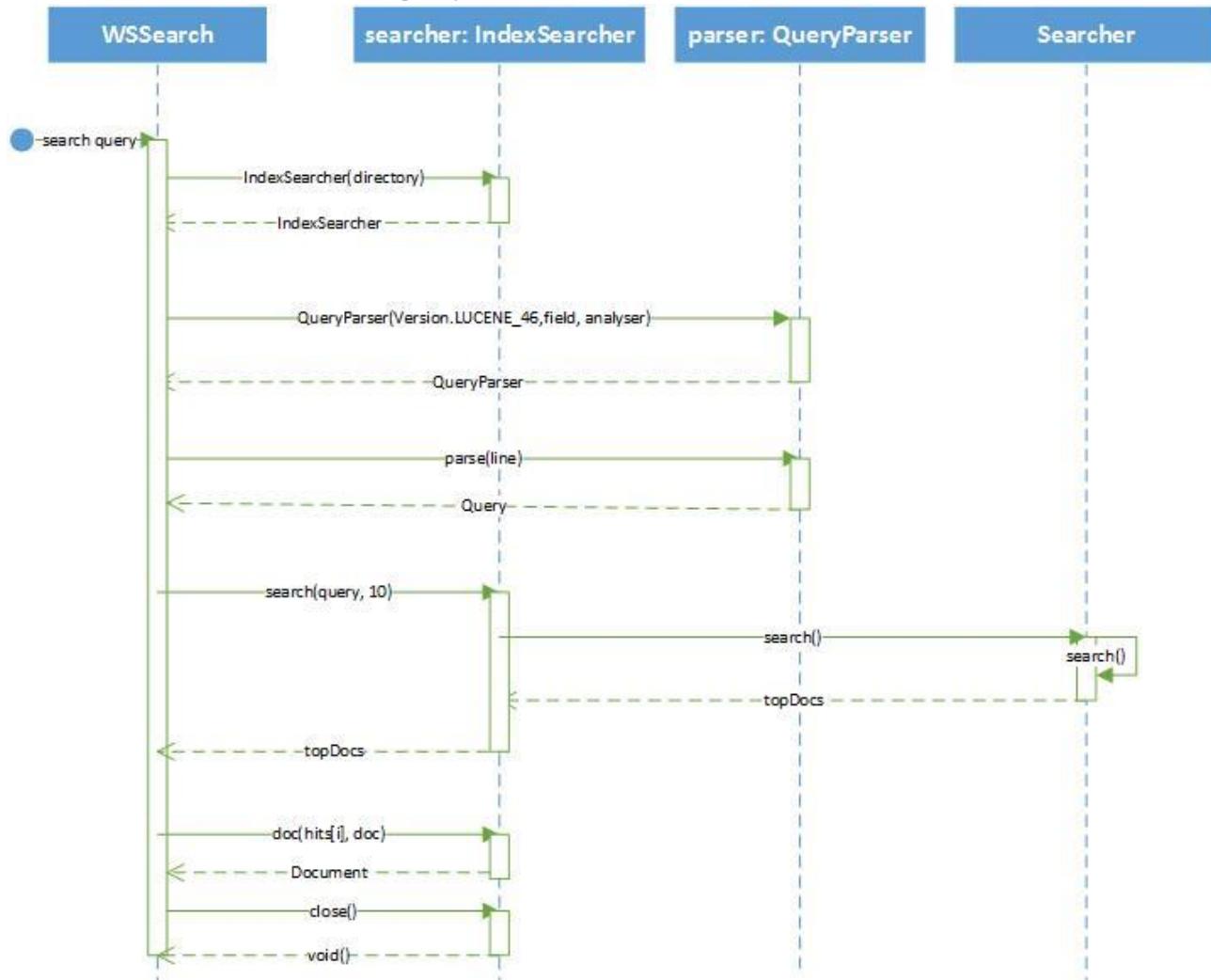
- **Recherche des images par similarité** : Tous les modules nécessaires pour ce type de recherche sont implémentés dans LIRE. *IndexReader* nous permet d'ouvrir l'index en lecture. On crée ensuite un *ImageSearcher* selon un descripteur visuel donnée. *ImageSearcherhits* nous permet retrouver toutes les images correspondantes aux critères de recherche.
- **Recherche des images par texte** : La bibliothèque Lucene nous permet d'utiliser *IndexSearcher* pour chercher l'index des métadonnées stocké dans un répertoire. La classe *QueryParser* nous permet d'encapsuler notre texte de recherche dans un ou plusieurs termes. Ce qui nous donne un objet de type *Query* prêt à être utilisé avec *IndexSearcher* pour effectuer la recherche. *IndexSearcher* utilise ensuite la classe *Searcher* pour retrouver toutes les images.
- **Recherche combinée** : Ce type de recherche nous permet de d'effectuer des requêtes combinant à la fois la recherche sur métadonnée et la recherche sur caractéristiques visuelles. On peut par exemple effectuer deux requêtes séparément, une sur l'index des caractéristiques visuelles en prenant compte uniquement des critères visuels de la requête et une autre requête sur l'index des métadonnées en prenant compte que des critères textuels de la requête. Les résultats des deux requêtes sont ensuite fusionnés pour en prendre que les plus pertinents.

B Fonctionnement dynamique

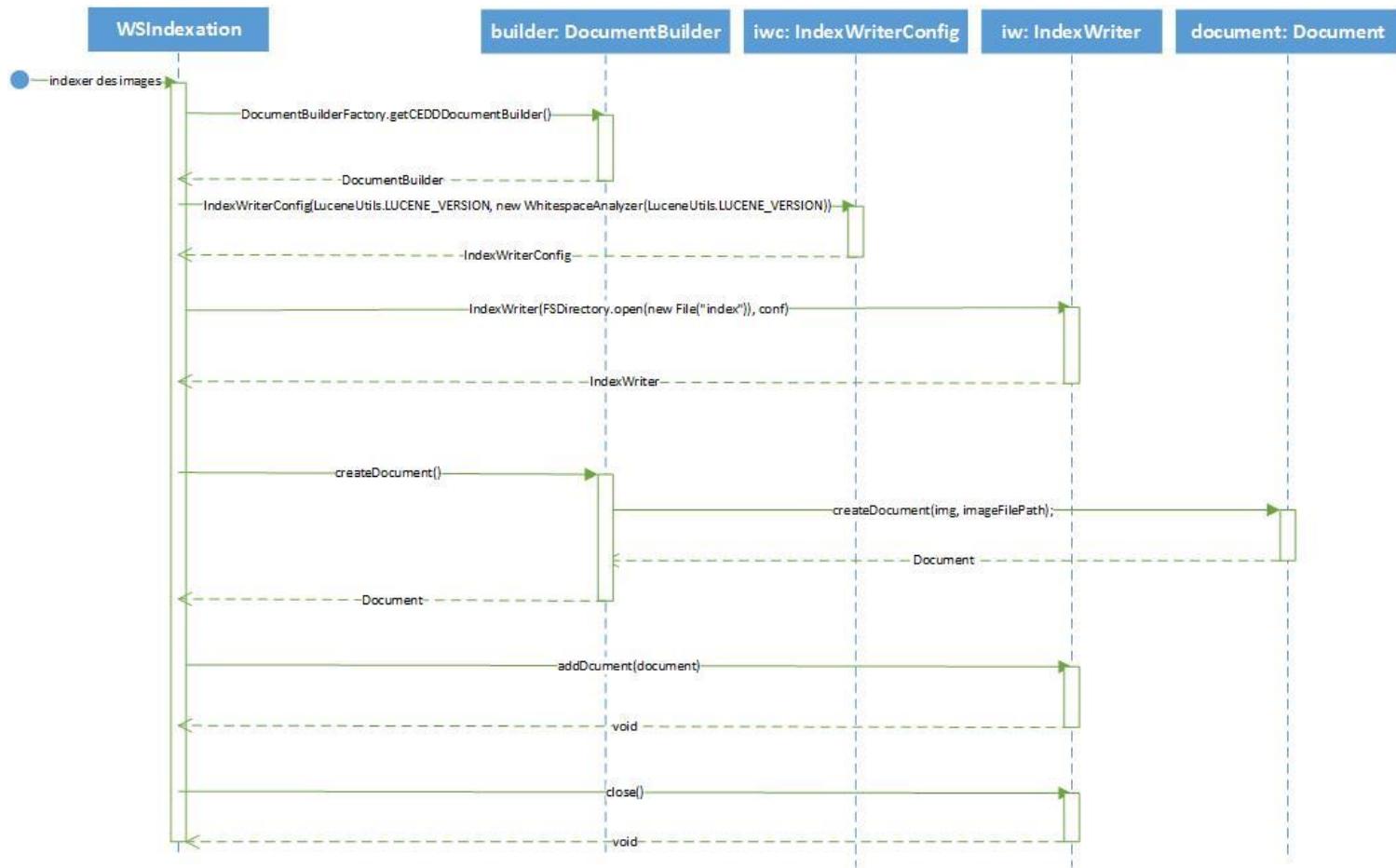
1 Recherche d'images par similarité des caractéristiques



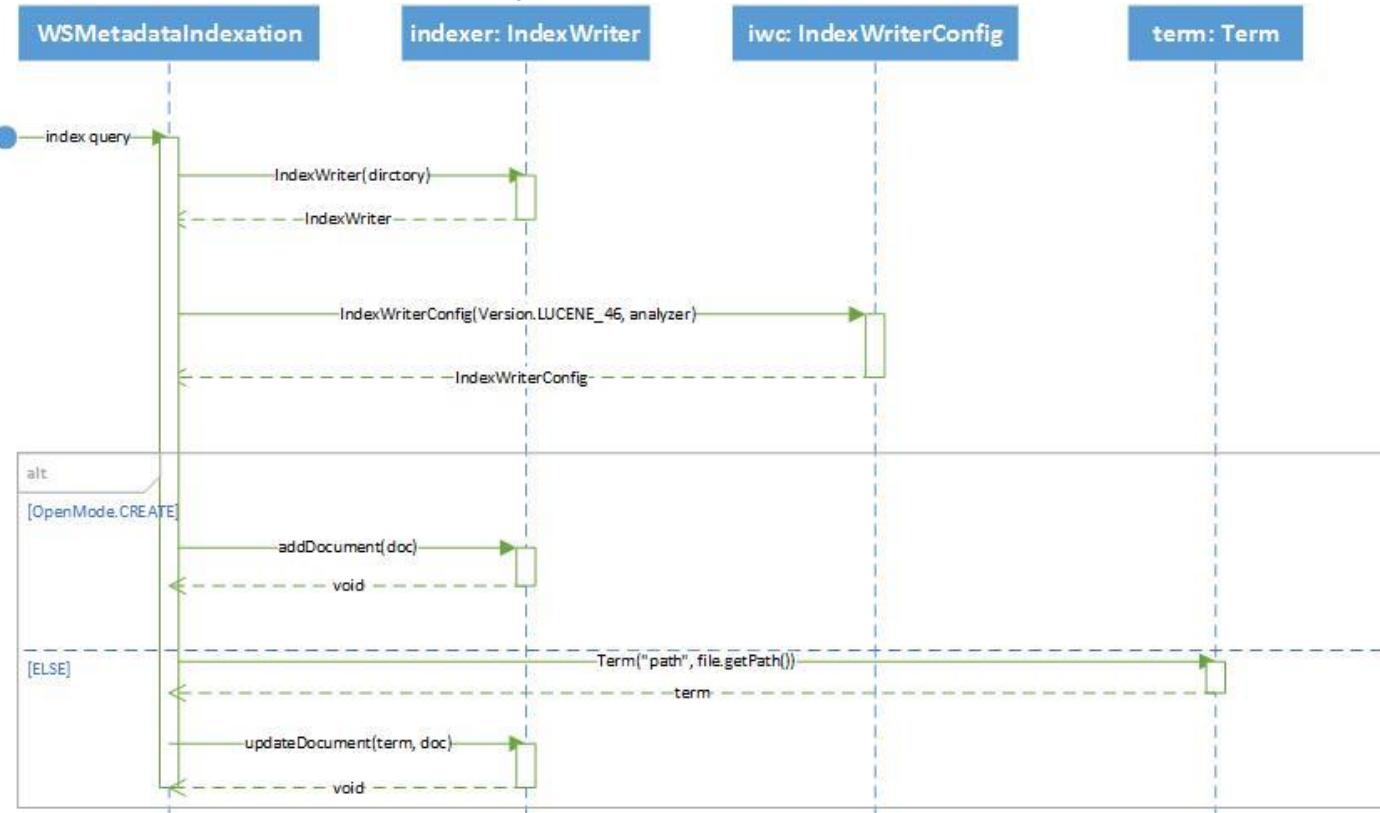
2 Recherche d'images par métadonnées



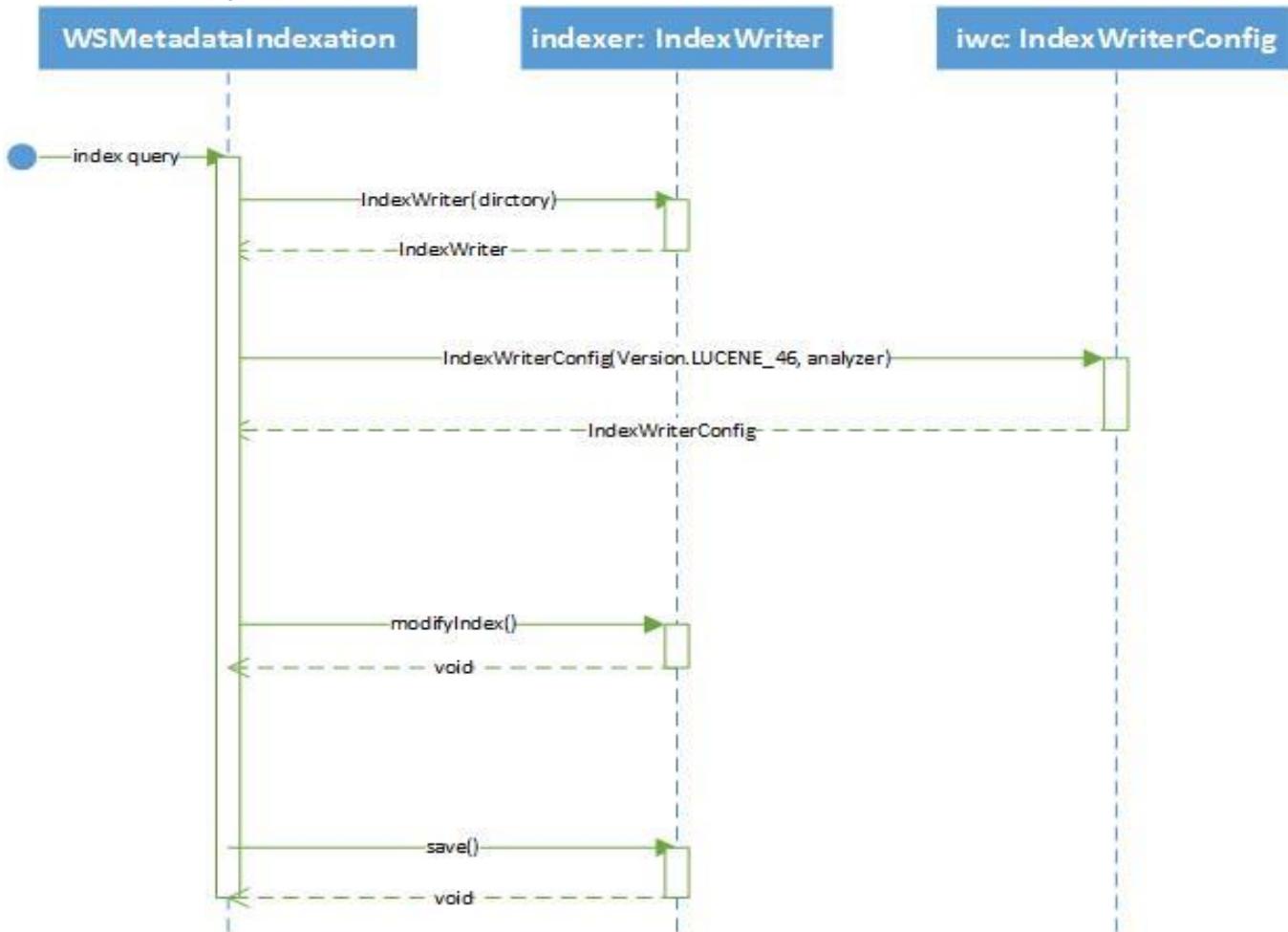
3 Indexation des images



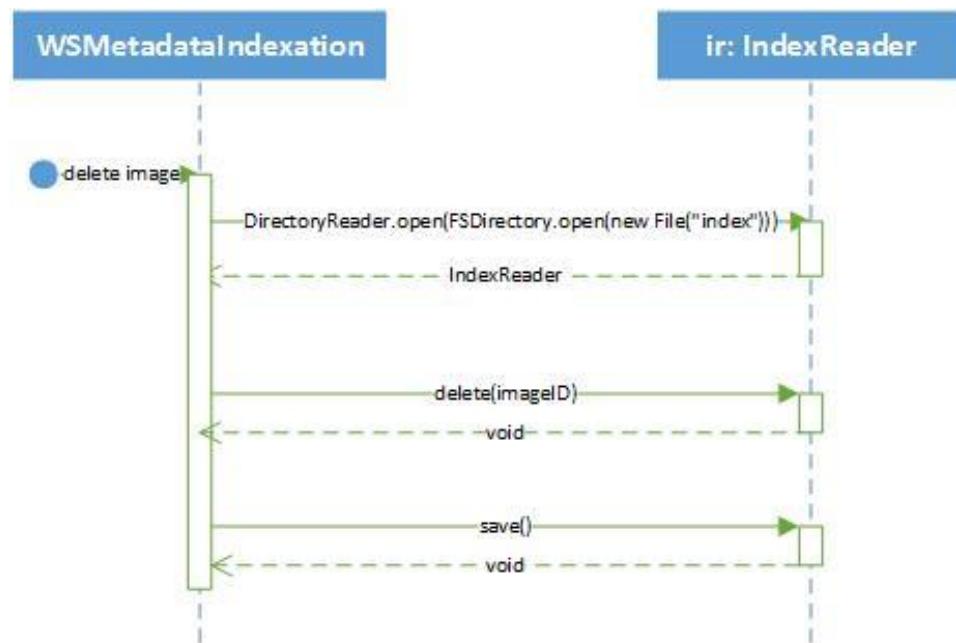
4 Indexation et mise à jour des métadonnées



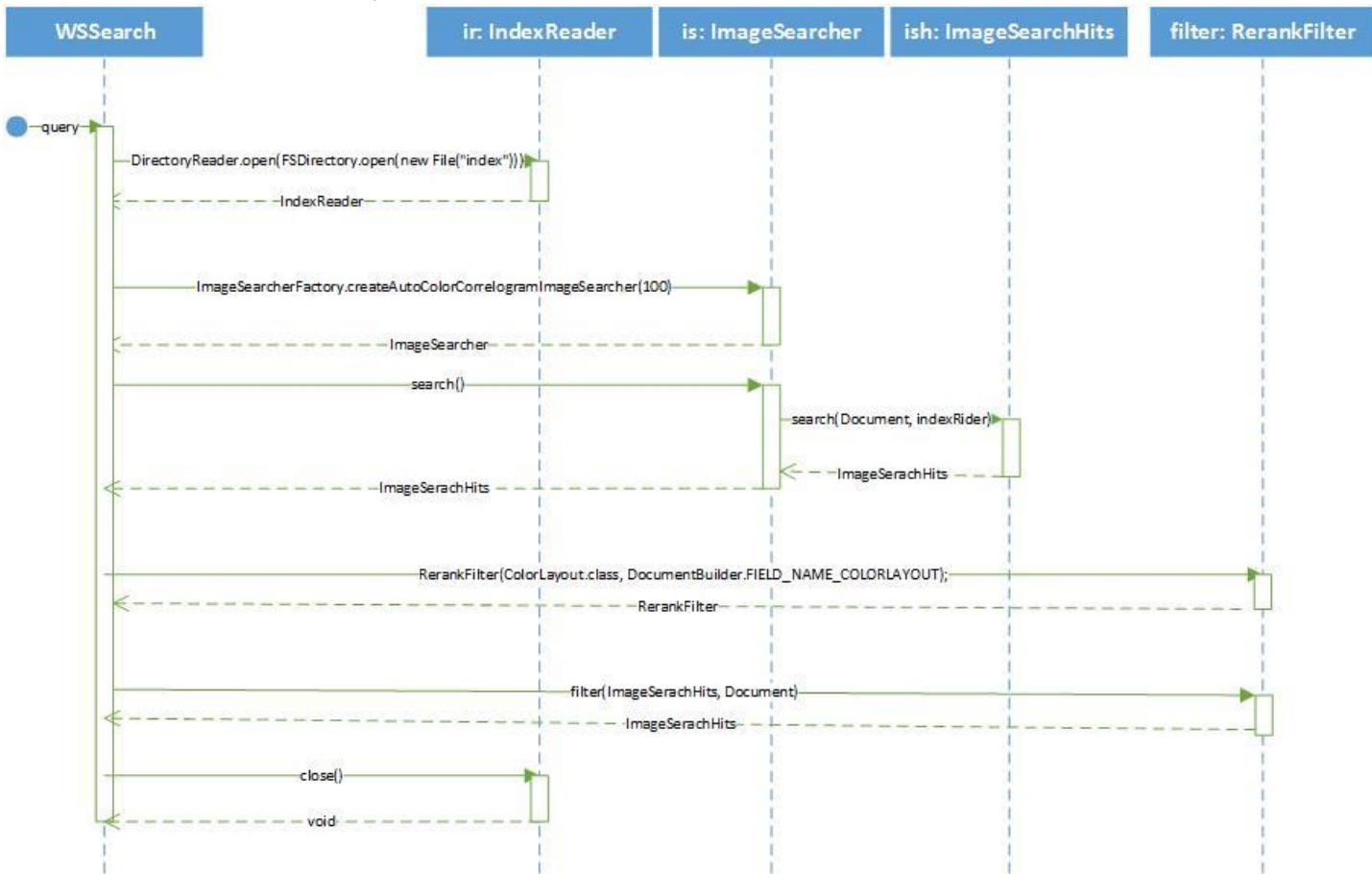
5 Ajout de filtre sur les métadonnées dans l'index



6 Supprimer une image



7 Recherche par filtre



C Justifications techniques

Le choix des technologies s'est basé surtout sur les exigences du client mais aussi sur les compétences des développeurs. Les développeurs ont tous des compétences en Java. Pour cela, nous avons soigneusement choisi d'utiliser des bibliothèques et des outils écrits dans ce langage.

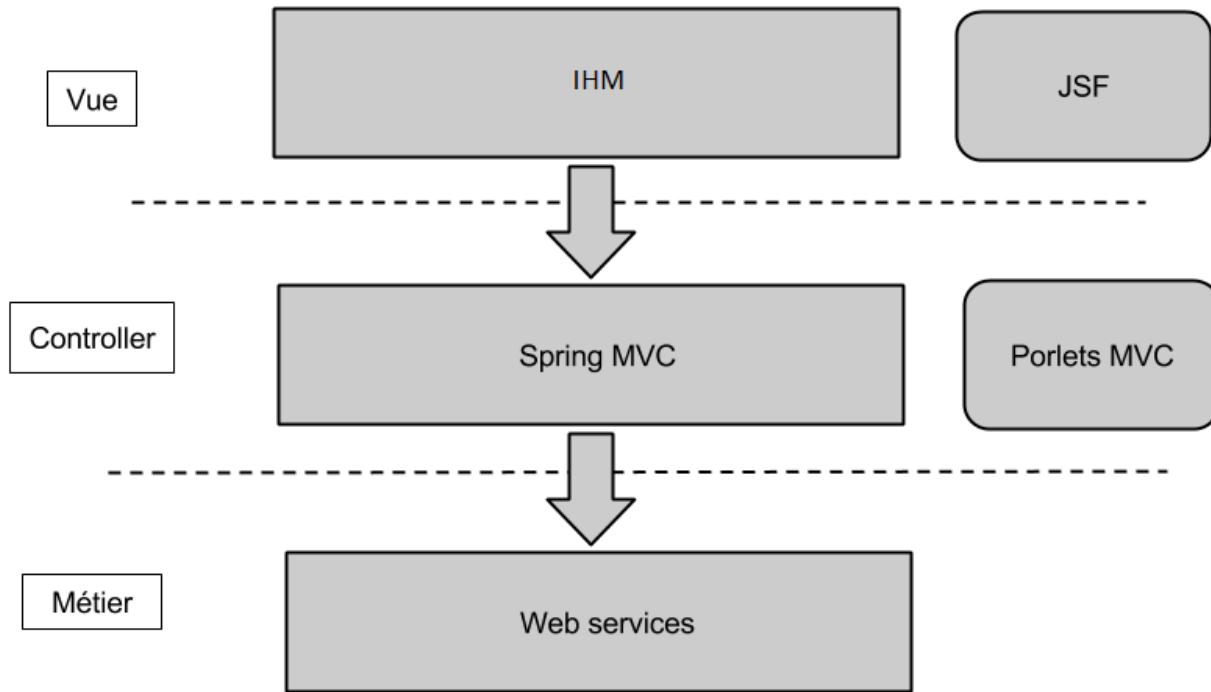
Parmi ces outils :

- LIRE : Lucene Image Retrieval. Basé sur le moteur de recherche Lucene, LIRE nous permet d'extraire les caractéristiques visuelles des images, d'indexer des images, et d'effectuer des recherches par similarité. LIRE implémente un large choix de méthodes d'extraction et d'indexation des caractéristiques visuelles images et des méthodes puissantes permettant la recherche des images par le contenu. LIRE est très simple à utiliser, une documentation riche lui est dédiée.
- WebLab : WebLab est conçu pour être utilisé dans les projets de fouille et de traitement de données audio-visuelles. WebLab nous permettra de faciliter l'intégration des différents composants du système SPORTIFS.

IX Architecture du lot d'interfaçage (André Charleroy, Mohamed Amine)

A Architecture statique

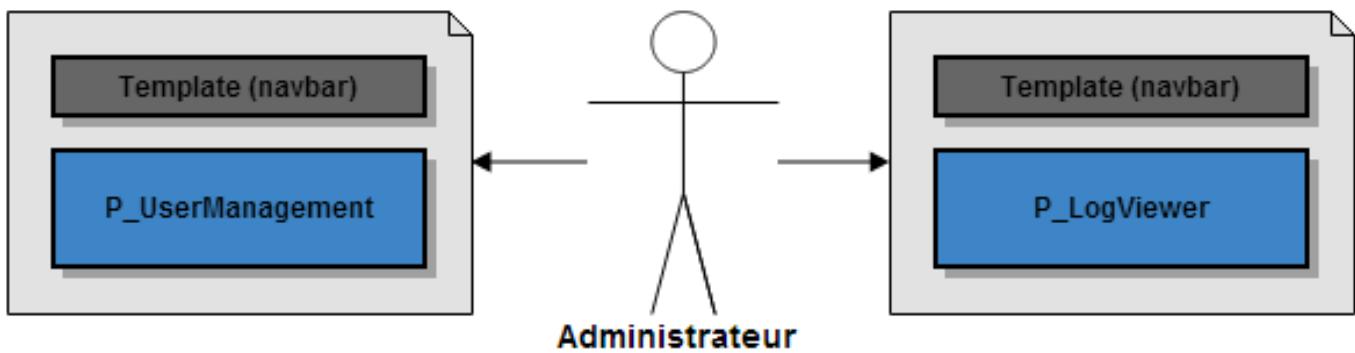
Voici un aperçu général du lot d'interfaçage :



1 Détail des pages par type d'utilisateur

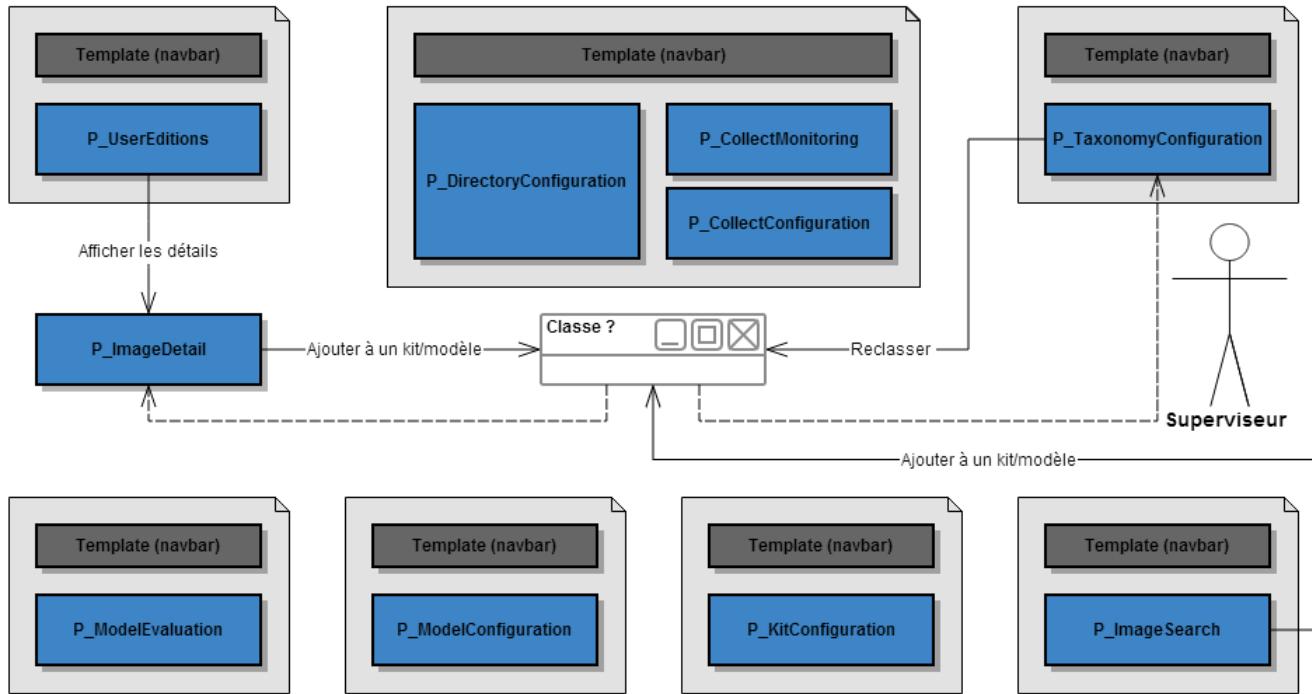
Les différents acteurs de l'application n'ont pas les mêmes besoins, c'est pourquoi différents portlets seront créées afin de répondre à ces attentes.

a L'administrateur



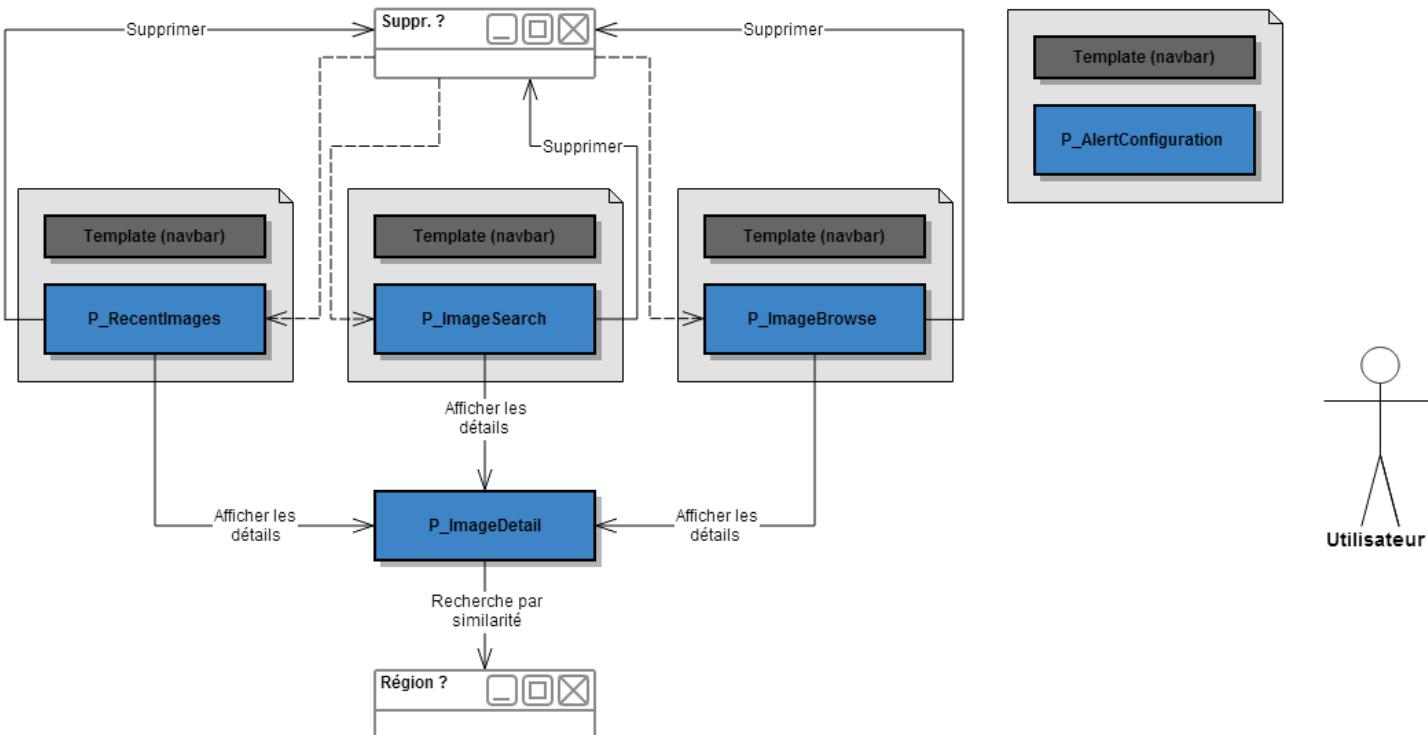
L'administrateur peut gérer les utilisateurs de l'application grâce au portlet *P_UserManagement* et aussi pouvoir accéder aux logs de l'application avec la portlet *P_LogViewer*.

b Le superviseur



Le superviseur peut configurer la taxonomie à travers le portlet *P_TaxonomyConfiguration*, il peut rechercher une image grâce à la portlet *P_ImageSearch*, configurer un kit d'évaluation et un modèle de classification grâce aux portlets *P_KitConfiguration* et *P_ModelConfiguration*, d'évaluer un modèle de classification grâce à la portlet *P_ModelEvaluation*, voir les éditions que les utilisateurs ont effectuées sur les images grâce au portlet *P_UserEditions*, et aussi voir les différentes collectes réalisées, configurer les prochaines collectes grâce aux portlets *P_CollectMonitoring* et *P_CollectConfiguration*.

c L'utilisateur



L'utilisateur peut configurer des alertes qui seront déclenchées lors des récoltes. Il peut visualiser les images récemment collectées avec le portlet *P_RecentImages*, visualiser les images avec le portlet *P_ImageSearch* et les détails des images avec le portlet *P_ImageDetail*.

2 Dossier de maquettage

Un certain nombre de maquettes ont été réalisées afin de pouvoir donner une représentation finale de l'application.

a Maquettes Utilisateur



Sur cette maquette nous avons une représentation du portlet *P_RecentImages* où nous pouvons voir les images récemment collectées concernant les sports contenus dans la taxonomie. Nous pouvons voir aussi la barre de menu qui sera commune à toutes les pages (template navbar). L'utilisateur peut voir une représentation de chaque image collectée quel que soit le sport, puisqu'il se trouve à la racine de l'arbre de taxonomie.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Images récentes

Football (7)



Sports (1)
En équipe (1)
Extérieur (1)
Football (1)
Intérieur (1)
Basket (1)
Handba
En solo (1)
Course (1)
Marathon (1)
Saut de haie (1)

Ici, nous pouvons voir que les images récentes qui sont filtrées selon le sport (Football) sont sélectionnées dans l'arbre de la taxonomie.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Images récentes

Basket (5)



Sports (1)
En équipe (1)
Extérieur (1)
Football (1)
Intérieur (1)
Basket (1)
Handba
En solo (1)
Course (1)
Marathon (1)
Saut de haie (1)

Ici, nous pouvons voir que les images récentes qui sont filtrées selon le sport (Basket-ball) sont sélectionnées dans l'arbre de la taxonomie.

SPORTIFS Accueil Recherche Mes alertes 15 Toto ▼

Détails d'une image

Un superbe pied



Rechercher une image similaire

Métadonnées techniques et de collecte	↑
Forma JPEC Poids: 320,3 K Dimension 1024 * 76	
Date de collect 03/12/2013 09:5 Sourc http://fr.123rf.com/01.i Class: Football	
Informations extraites de la page	↑
Titre Un superbe pied	
Auteur pied de footba	

Ce livre raconte les plus grands moments du Rugby, un sport pas comme les autres.	
C'est avec passion que le Rugby y est raconté, voici la couverture du livre en question comme l'illustre ci-dessous.	
Mots clés	↓
Métadonnées extraits	↓

Ici nous avons une représentation du site au moment où l'utilisateur a cliqué sur une image de son choix. L'utilisateur a accès aux informations concernant l'image.

SPORTIFS Accueil Recherche Mes alertes 15 Toto ▼

Détails d'une image

Un superbe pied



Rechercher une image similaire

Métadonnées techniques et de collecte	↑
Forma JPEC Poids: 320,3 K Dimension 1024 * 76	
Date de collect 03/12/2013 09:5 Sourc http://fr.123rf.com/01.i Class: Football	
Informations extraites de la page	✓ ✗ ↑
Titre	<input type="text"/>
Auteur	<input type="text"/>

Ce livre raconte les plus grands moments du Rugby, un sport pas comme les autres.	
C'est avec passion que le Rugby y est raconté, voici la couverture du livre en question comme l'illustre la photo ci-dessous.	
Mots clés	↓
Métadonnées extraits	↓

La même vue que précédemment mais cette fois-ci dans un mode d'édition afin de pouvoir modifier les informations de l'image.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Détails d'une image

Un superbe pied



Rechercher une image similaire

Métadonnées techniques et de collecte

Forma	JPEG	Poids:	320,3 K	Dimension	1024 * 76
Confirmation suppression					
Etes-vous sûr de vouloir supprimer cette image? La suppression définitive doit être validée par un super					
Confirmer			Annuler		

3rf.com/01.j Class: Football

Ce livre raconte les plus grands moments du Rugby, un sport pas comme les autres.
C'est avec passion que le Rugby y est raconté, voici la couverture du livre en question comme l'illu ci-dessous.

Mots clés

Métadonnées extraites

Ici une pop-up s'affiche lorsque l'on veut supprimer une image, l'utilisateur a alors la possibilité de confirmer ou non la suppression de l'image.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Détails d'une image

L'image a été soumise à suppression. Elle sera supprimée après vérification.

Un superbe pied



Rechercher une image similaire

Métadonnées techniques et de collecte

Forma	JPEG	Poids:	320,3 K	Dimension	1024 * 76
Date de collec	03/12/2013 09:5	Sourc	http://fr.123rf.com/01.j	Class:	Footbal
Informations extraites de la page					
Titre	Un superbe pie				
Auteur	pied de footba				

.....

Ce livre raconte les plus grands moments du Rugby, un sport pas comme les autres.
C'est avec passion que le Rugby y est raconté, voici la couverture du livre en question comme l'illu ci-dessous.

Mots clés

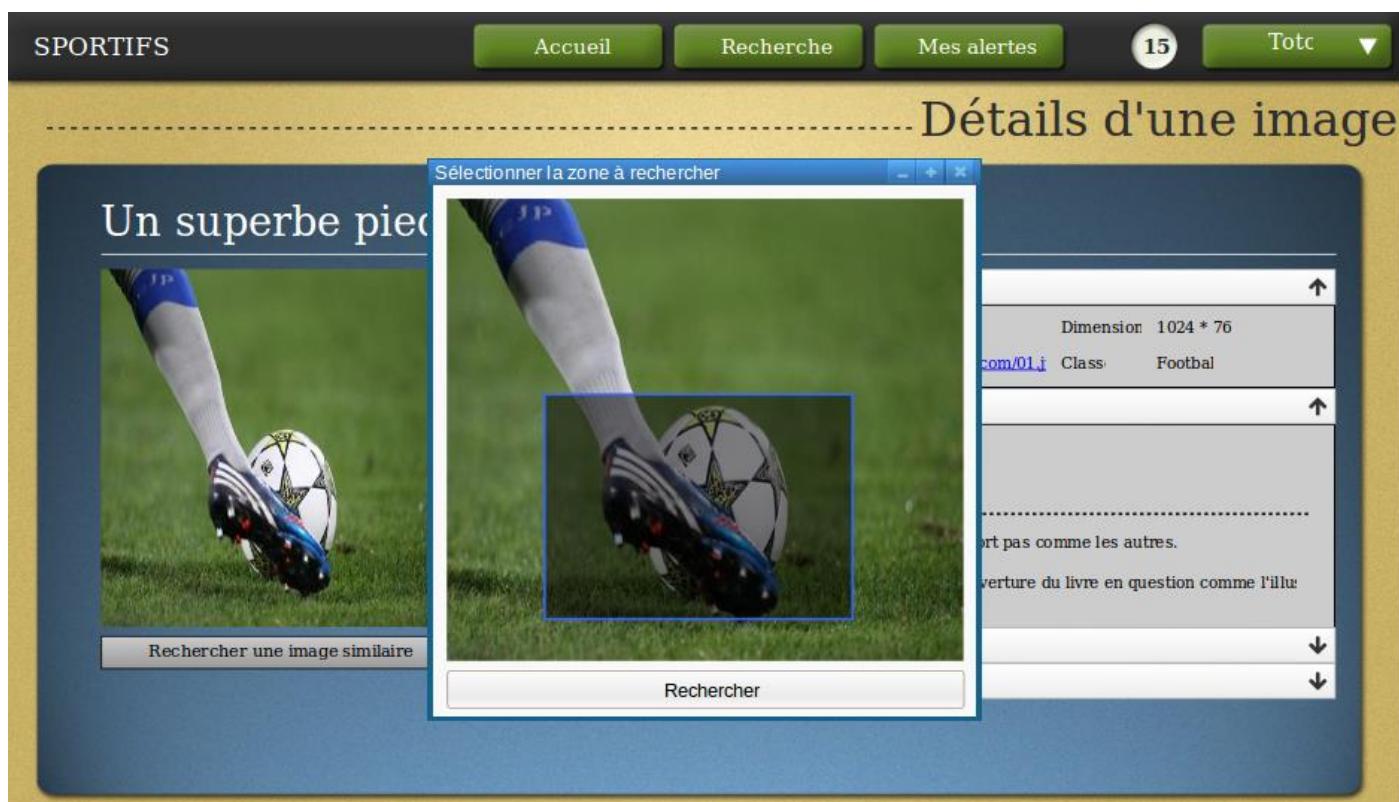
Métadonnées extraites

Une fois que l'utilisateur a confirmé la suppression de l'image, l'image n'est pas supprimée mais marquée comme une image à supprimer par le superviseur qui approuvera ou non le choix de l'utilisateur.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Détails d'une image

Un superbe pied



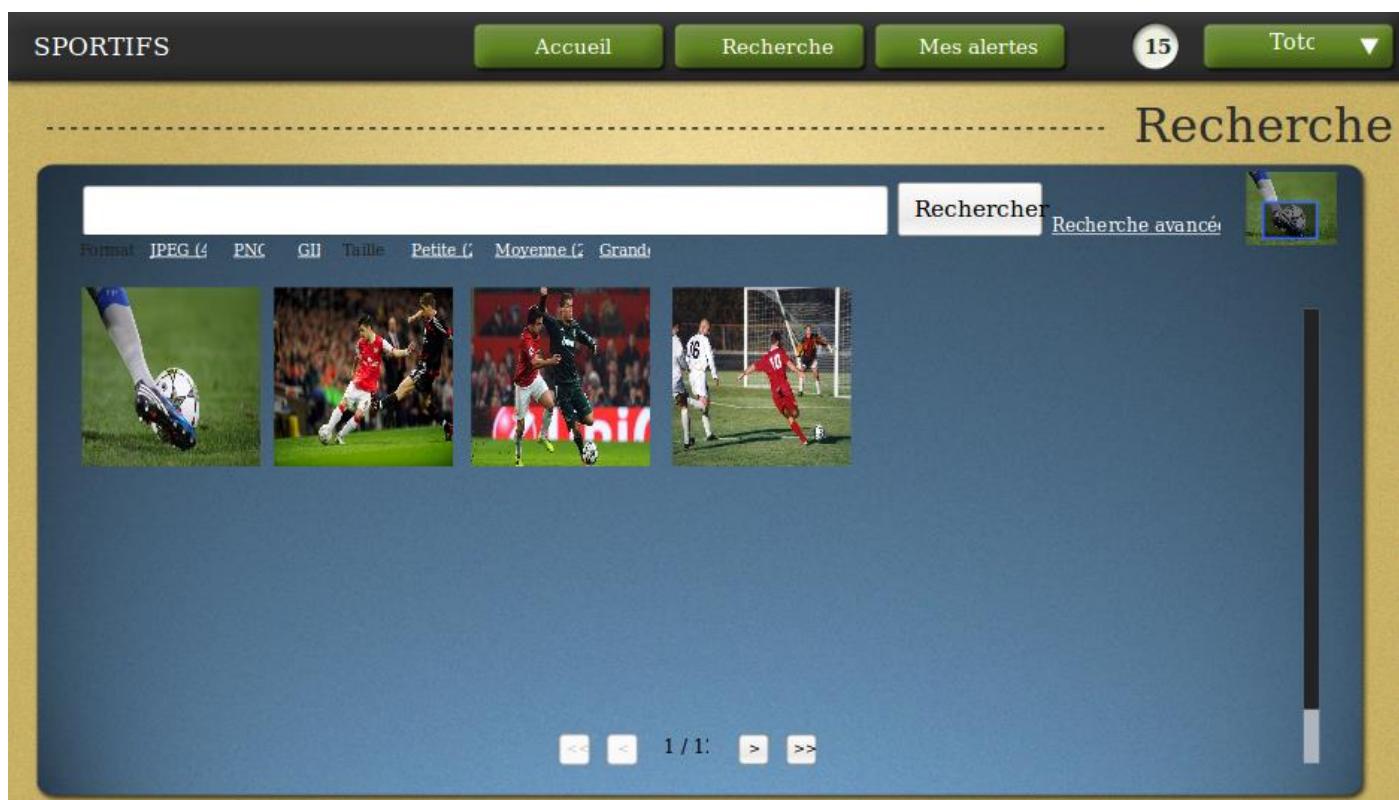
Il est aussi possible d'effectuer une recherche d'images similaire en sélectionnant la zone que l'on souhaite comparer à d'autres images.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Recherche

Format: JPEG (.JFIF) PNG GIF Taille: Petite (C) Moyenne (G) Grande (L)

Rechercher [Recherche avancée](#) 



Une fois la recherche lancée nous retrouvons le portlet *P_ImageSearch* affichant cette fois ci les images similaires à l'image sélectionnée.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Recherche

Rechercher Recherche avancée

Format : JPEG (3') PNG (20 GI) Taille : Petite (0) Moyenne (1) Grande (4)



<< < > >> 1 / 1

Sur cette maquette, nous pouvons voir le portlet *P_ImageSearch* permettant à l'utilisateur de rechercher une image selon des filtres (tailles de l'image, formats, ...). Ici, toutes les images de sports sont affichées (racine de l'arbre de taxonomie).

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Recherche

Rechercher Recherche avancée

Format : JPEG (3') PNG (20 GI) Taille : Petite (0) Moyenne (1) Grande (4)



Voir les détails
250 x 250 - PN
<http://www.lequipe.fr/foot/path/to/image/image>

<< < > >> 1 / 1

Lorsque la souris survole une image une fenêtre agrandissant l'image survolée s'affiche à l'écran permettant à l'utilisateur d'avoir des informations sommaires sur l'image.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Recherche

Format: JPEG (37) PNG (20) GI Taille: Petite (6) Moyenne (10) Grande (4)

Largeur minimale (px):

Rechercher [Recherche avancée](#)



1 / 10 < > >>

Sur cette maquette, nous avons un exemple de recherche avancée sur une image de Football, en définissant par exemple la largeur minimale de l'image.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Recherche

Format: JPEG (6) PNG (5) GI Taille: Petite (6) Moyenne (10) Grande (4)

Largeur minimale (px):

Rechercher [Recherche avancée](#)



1 / 10 < < > >>

Et voici le résultat de la recherche avancée qui a eu pour effet de diminuer le nombre d'images affichées à l'écran (un certain nombre d'images ont été éliminées).

SPORTIFS

Accueil

Recherche

Mes alertes

15

Totc ▼

Mes alertes

Mes alertes

Aucune alerte créée pour le moment

Créer une nouvelle alerte

Type d'images à filter PNG JPEG GIF

Choisir une image similaire ...ou

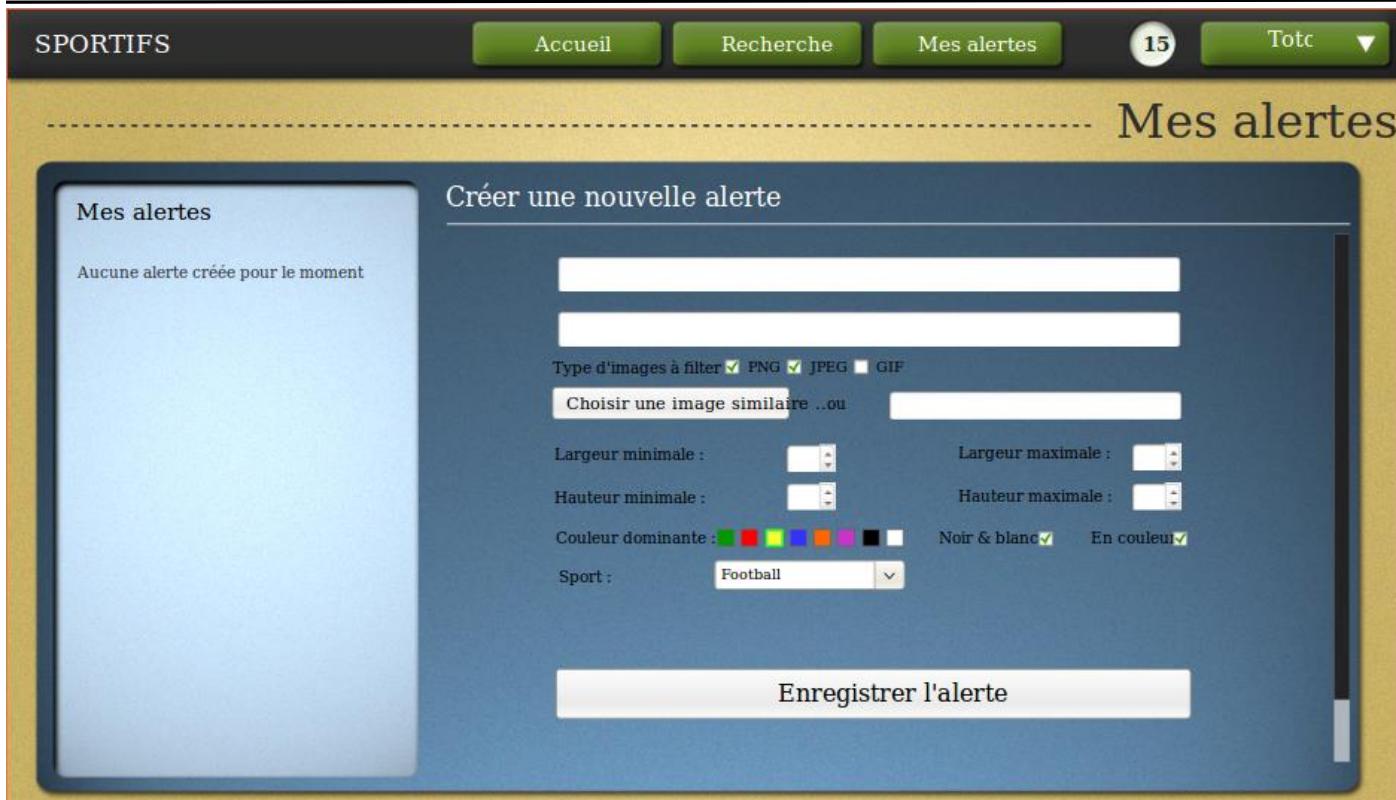
Largeur minimale :

Hauteur minimale :

Couleur dominante : Noir & blanc En couleur

Sport : Football

Enregistrer l'alerte



Sur cette maquette, nous pouvons voir le portlet *P_AlertConfiguration* qui permet la configuration d'une alerte, un certain nombre de paramètres permettent de configurer le plus précisément possible l'alerte selon les besoins de l'utilisateur.

SPORTIFS

Accueil

Recherche

Mes alertes

15

Totc ▼

Mes alertes

L'alerte "Ma première alerte" a été créée

Mes alertes

Ma première alerte  

Créer une nouvelle alerte

Type d'images à filter PNG JPEG GIF

Choisir une image similaire ...ou

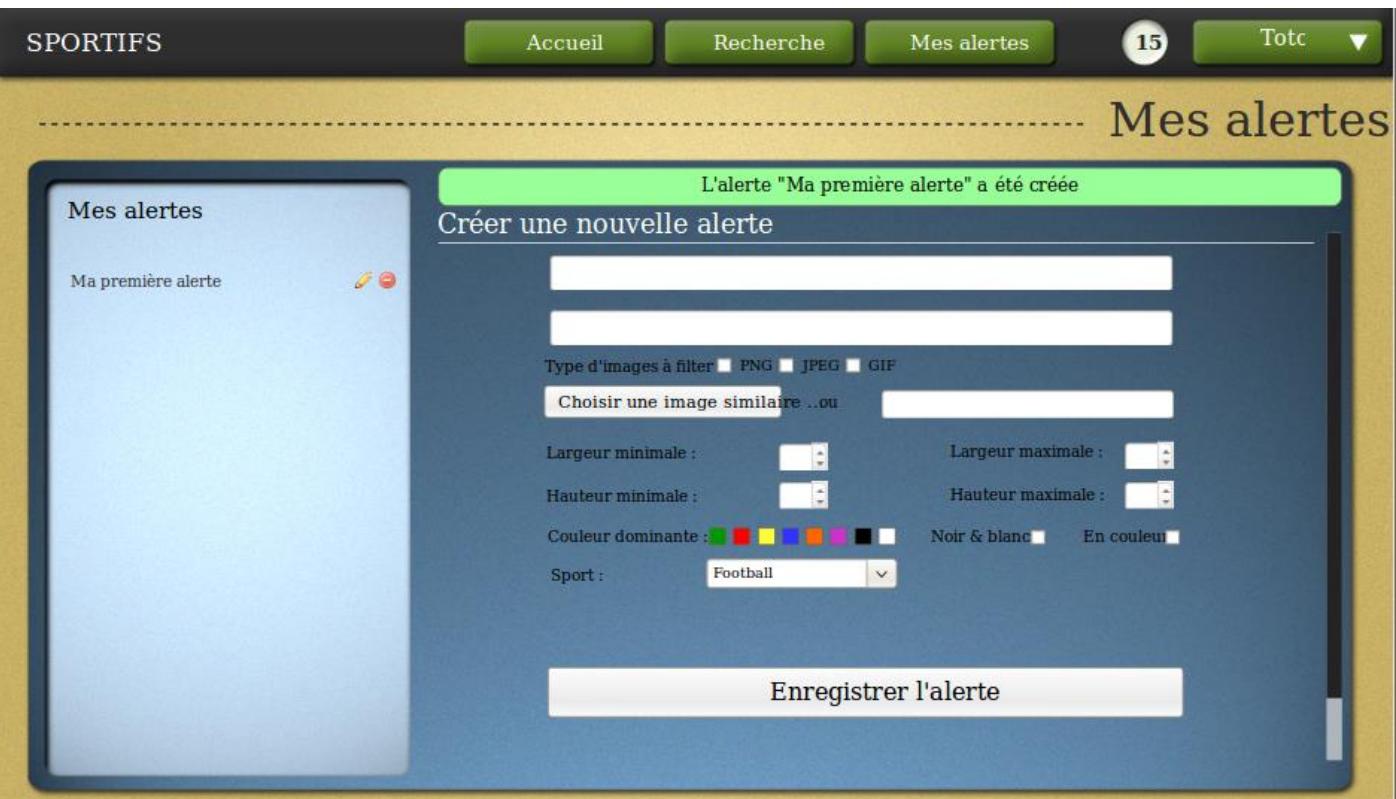
Largeur minimale :

Hauteur minimale :

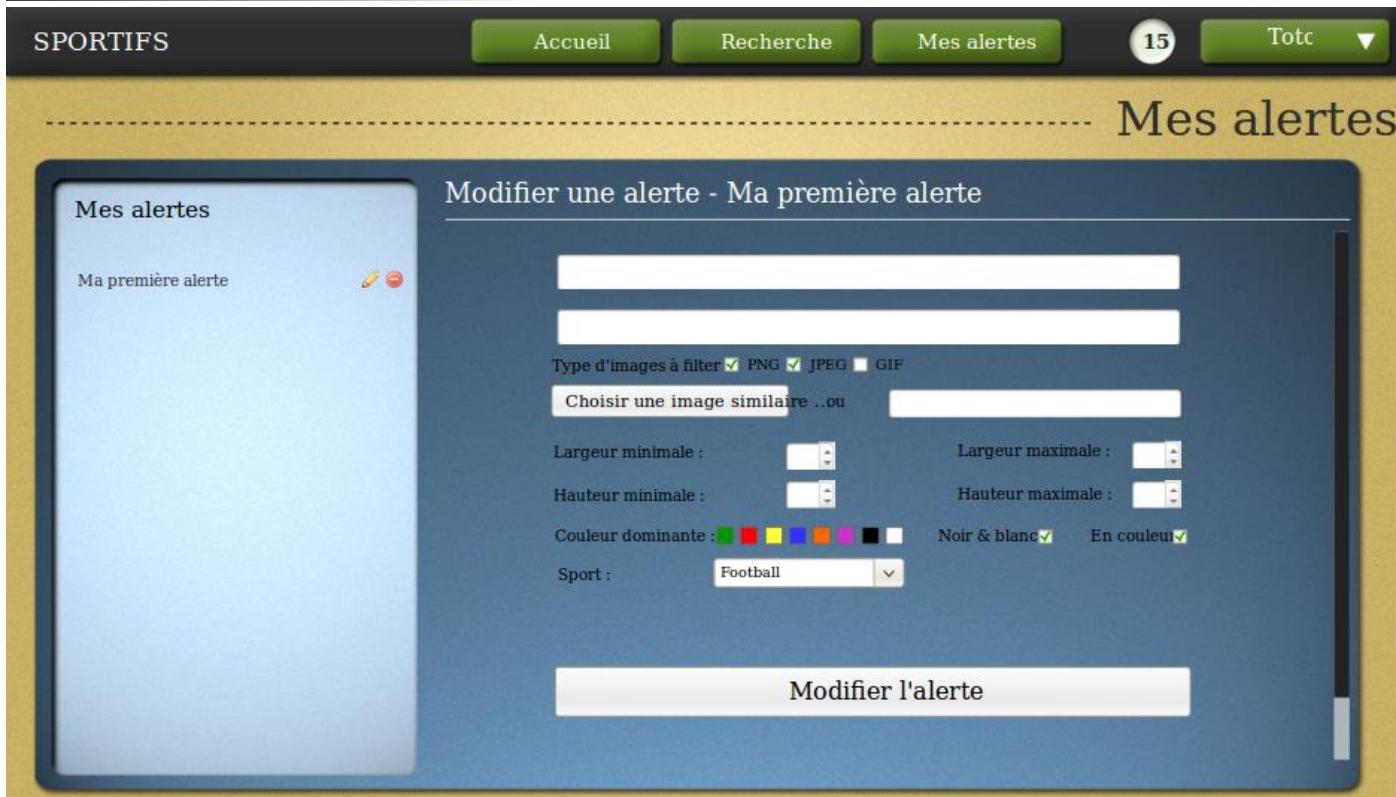
Couleur dominante : Noir & blanc En couleur

Sport : Football

Enregistrer l'alerte



Une fois l'alerte configurée et après que l'utilisateur ai cliqué sur le bouton « Enregistrer l'alerte », le nom de l'alerte s'affiche dans le menu de gauche afin de pouvoir la modifier ou la supprimer par la suite et un message indique à l'utilisateur que l'alerte a bien été créée.



SPORTIFS

Accueil Recherche Mes alertes 15 Totc ▾

Mes alertes

Ma première alerte

Modifier une alerte - Ma première alerte

Type d'images à filter PNG JPEG GIF

Choisir une image similaire ..ou

Largeur minimale : Largeur maximale :

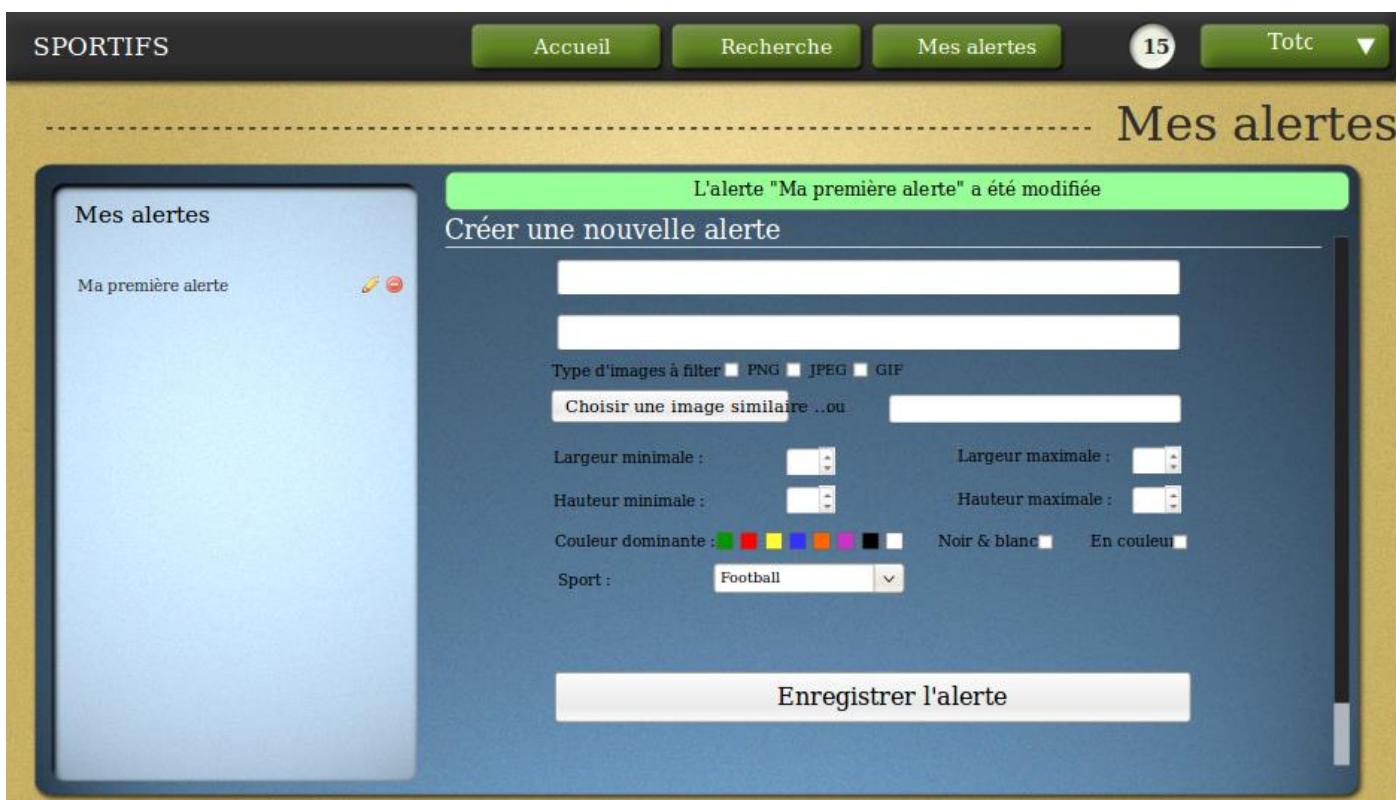
Hauteur minimale : Hauteur maximale :

Couleur dominante : Noir & blanc En couleur

Sport : Football

Modifier l'alerte

Voici une maquette présentant la vue permettant de modifier une alerte qui a été créée. Les paramètres de l'alerte sont alors affichés de nouveau à l'écran.



SPORTIFS

Accueil Recherche Mes alertes 15 Totc ▾

Mes alertes

Ma première alerte

L'alerte "Ma première alerte" a été modifiée

Créer une nouvelle alerte

Type d'images à filter PNG JPEG GIF

Choisir une image similaire ..ou

Largeur minimale : Largeur maximale :

Hauteur minimale : Hauteur maximale :

Couleur dominante : Noir & blanc En couleur

Sport : Football

Enregistrer l'alerte

Une fois l'alerte modifiée et que l'utilisateur a confirmé ses modifications, un message s'affiche à l'écran pour notifier à l'utilisateur que ses modifications ont bien été prises en compte.

SPORTIFS Accueil Recherche Mes alertes 15 Totc ▼

Mes alertes

Mes alertes

Ma première alerte  

Créer une nouvelle alerte

Confirmation suppression

Êtes-vous sûr de vouloir supprimer cette alerte ?

Nombre maximale : Nombre maximale :

Couleur dominante :  Noir & blanc En couleur

Sport :

Il est aussi possible pour l'utilisateur de supprimer une alerte, lorsqu'il décide d'en supprimer une, une pop-up s'affiche pour laisser à l'utilisateur le choix de confirmer ou non la suppression de l'alerte.

b Maquettes Superviseur

SPORTIFS Éditions utilisateurs Collectes Modèles / Kits Supr. ⌂

Editions utilisateurs

Sports (18)

- En équipe (14)
 - Extérieur (8)
 - Football (8)
 - Intérieur (6)
 - Basket (5)
 - Handball (1)
 - En solo (4)
 - Course (4)
 - Marathon (1)
 - Saut de haie (3)



 Supprimer (5)  Annuler la sélection

Sur cette maquette nous pouvons voir le portlet *P_UserEditions* permettant aux superviseurs de visualiser les images qui ont été éditées par l'utilisateur.

SPORTIFS Éditions utilisateurs Collectes Modèles / Kits Supr ⌂

Editions utilisateurs

Cette image a été proposée à la suppression. La conserver ?  

Métadonnées techniques et de collecte

Format	JPEG	Poids	320,3 Ko	Dimensions	1024 * 768
Date de collecte	03/12/2013 09:55	Source	http://fr.123rf.com/01.jpg	Classe	Football [Autre]  

Informations extraites de la page

Titre	Les grands moments du Rugby par Jean-Paul Rey (Collection Sport Passion)
Alt	Grands moments Rugby

Ce livre raconte les plus grands moments du Rugby, un sport pas comme les autres.

C'est avec passion que le Rugby y est raconté, voici la couverture du livre en question comme l'illustre la photo ci-dessous.

Mots clés

Métadonnées extraites

 **Tout valider**  **Tout invalider**

Ajouter à un kit Ajouter à un moteur

Lorsque le superviseur sélectionne une image, le portlet *P_ImageDetail* affiche les détails de l'image. Un message notifie au superviseur l'action réalisée par l'utilisateur (par exemple ici suppression de l'image).

SPORTIFS Éditions utilisateurs Collectes Modèles / Kits Supr ⌂

Editions utilisateurs

Cette image a été proposée à la suppression. La conserver ?  

Métadonnées techniques et de collecte

Informations extraites de la page

Mots clés

- Rugby [Rugby]  
- Collection
- 1996
- [Ballon oval]  
- Livre
- Moments
- Essai
- Sport
- Livre  
- Bleus
- XV de France
- Passion

Métadonnées extraites

 **Tout valider**  **Tout invalider**

Ajouter à un kit Ajouter à un moteur

Il est possible de déployer ou de refermer des volets concernant des types d'informations concernant une image (par exemple les mots clés).

SPORTIFS Éditions utilisateurs Collectes Modèles / Kits Supe ⚡

Collectes

Collecte pendant week-end
 Collecte du mardi
 Collecte quotidienne
 Collecte du lundi
 Collecte pendant semaine
 Collecte du mercredi 12h00
 Collecte du mercredi 00h00

Intitulé	<input type="text" value="Collecte hebdomadaire le samedi soir"/>
Fréquence	Le <input type="button" value="▼"/> Lundi <input type="button" value="▼"/> à 22h00
Dimensions	1024 x 768
Sources	
<input checked="" type="radio"/> http://www.francetvsport.fr <input checked="" type="radio"/> http://www.image-net.org/foot/france <input checked="" type="radio"/> http://www.eurosport.fr/natation <input checked="" type="radio"/> http://www.lequipe.fr/sports/basket/nba <input checked="" type="radio"/> http://www.beinsport.fr/	
<input type="text" value="http://www.sport.fr/championnats/biathlon"/> <input type="button" value="▼"/> <input type="button" value="+"/>	

Date	Intitulé	Nombre d'images collectées	Nombre d'images en doublon
09/02/2013 00:00	Collecte du mercredi 00h00	135	8
08/01/2013 22:00	Collecte du mardi	81	6
02/01/2013 00:00	Collecte du mercredi 00h00	64	4
01/01/2013 22:00	Collecte du mardi	178	13

Cette maquette représente le portlet *P_DirectoryConfiguration* permettant au superviseur de choisir les sites internet où vont être extraits les images. Le portlet *P_CollectMonitoring* permet au superviseur de gérer les différentes collectes créées précédemment. Le portlet *P_CollectConfiguration* permet au superviseur de paramétriser une collecte et de la créer.

SPORTIFS Éditions utilisateurs Collectes Modèles / Kits Supe ⚡

Taxonomie

Aucune image à reclasser pour le moment

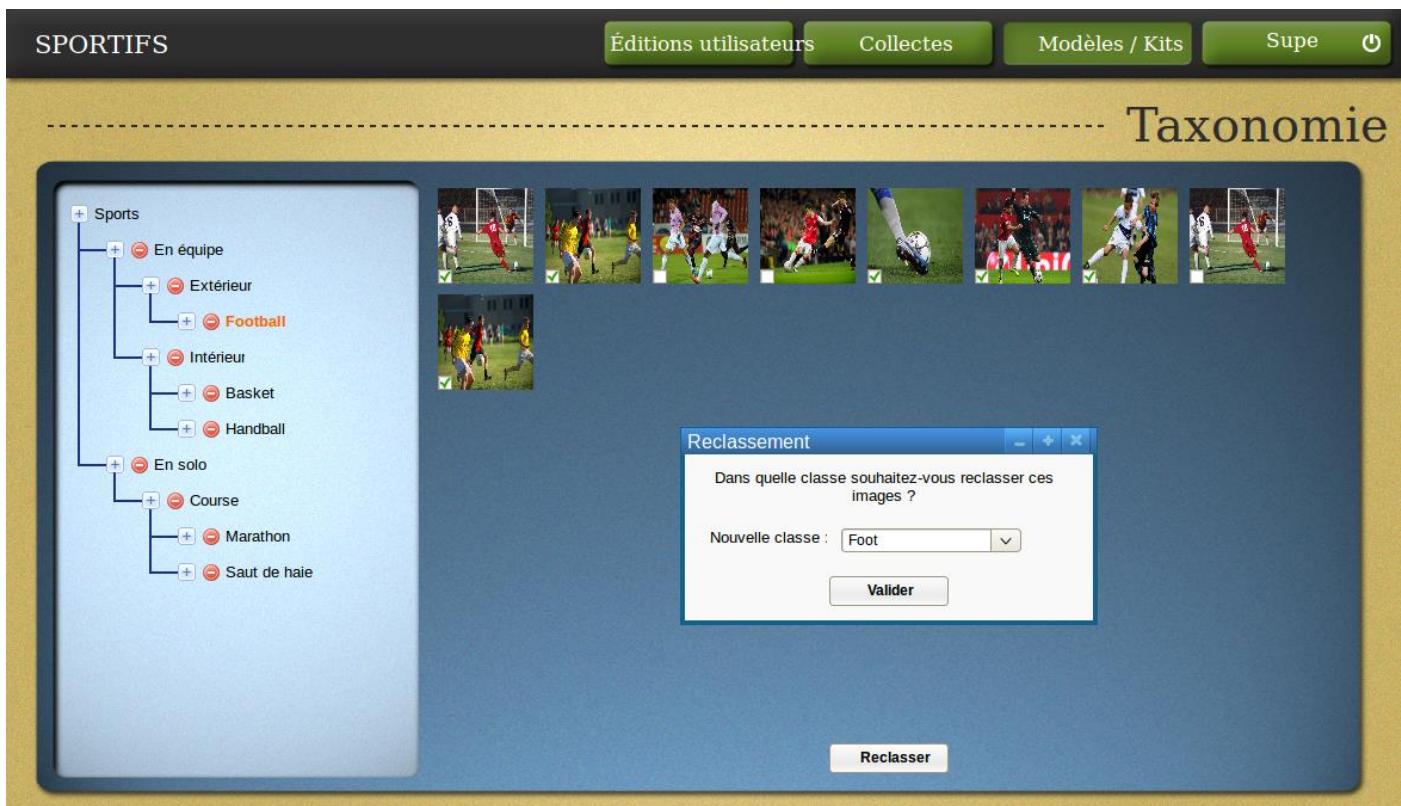
- + Sports
 - + En équipe
 - + Extérieur
 - + Football
 - + Intérieur
 - + Basket
 - + Handball
 - + En solo
 - + Course
 - + Marathon
 - + Saut de haie

Cette maquette illustre la possibilité que le superviseur n'ait aucune image à reclasser.

SPORTIFS

Éditions utilisateurs Collectes Modèles / Kits Supe ⚡

Taxonomie



Reclassement

Dans quelle classe souhaitez-vous reclasser ces images ?

Nouvelle classe : Foot

Valider

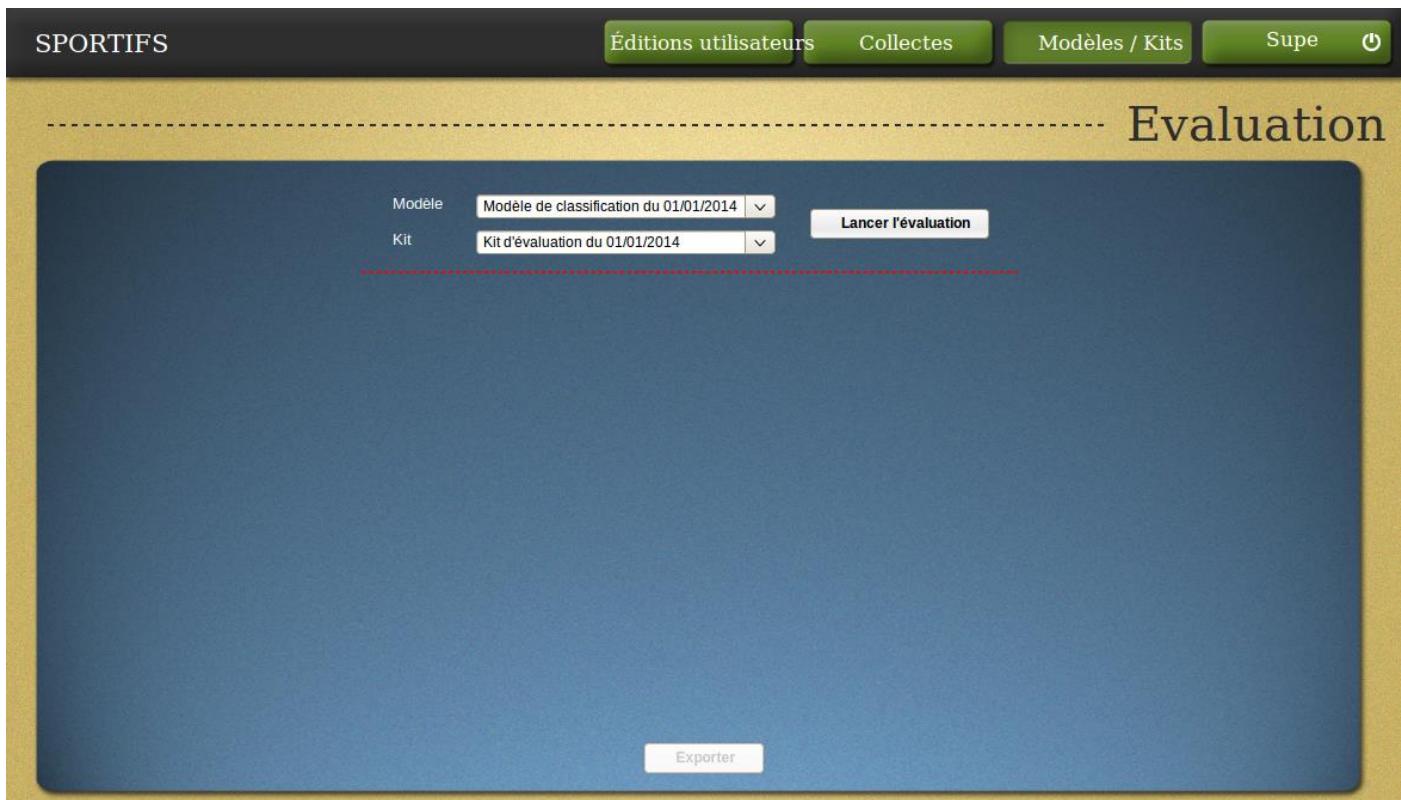
Reclasser

Sur cette maquette est représentée la possibilité au superviseur de pouvoir reclasser les images.

SPORTIFS

Éditions utilisateurs Collectes Modèles / Kits Supe ⚡

Evaluation



Modèle : Modèle de classification du 01/01/2014

Kit : Kit d'évaluation du 01/01/2014

Lancer l'évaluation

Exporter

Cette maquette représente le portlet *P_ModelEvaluation* permettant au superviseur d'évaluer un modèle en fonction d'un kit d'évaluation.

SPORTIFS Éditions utilisateurs Collectes Modèles / Kits Supe ⚡

Evaluation

Modèle Modèle de classification du 01/01/2014 Lancer l'évaluation

Kit Kit d'évaluation du 01/01/2014

Réel\Estimé	Football	Basket	Handball	Marathon	Saut de haie	Total
Football	74 (90,24%)	5 (6,10%)	2 (2,44%)	0 (0,00%)	1 (1,21%)	82
Basket	1 (1,54%)	61 (93,85%)	0 (0,00%)	2 (3,08%)	0 (0,00%)	65
Handball	1 (4,55%)	2 (9,10%)	19 (86,37%)	0 (0,00%)	0 (0,00%)	22
Marathon	0 (0,00%)	0 (0,00%)	1 (7,14%)	12 (85,71%)	1 (7,14%)	14
Saut de haie	0 (0,00%)	0 (0,00%)	0 (0,00%)	1 (5,56%)	17 (94,44%)	18

[Exporter](#)

Ici nous pouvons voir les résultats de l'évaluation du modèle.

SPORTIFS Éditions utilisateurs Collectes Modèles / Kits Supe ⚡

Modèles

Modèle du 01/01/2014 Intitulé : Modèle du 01/01/2014

Classe : Football Afficher



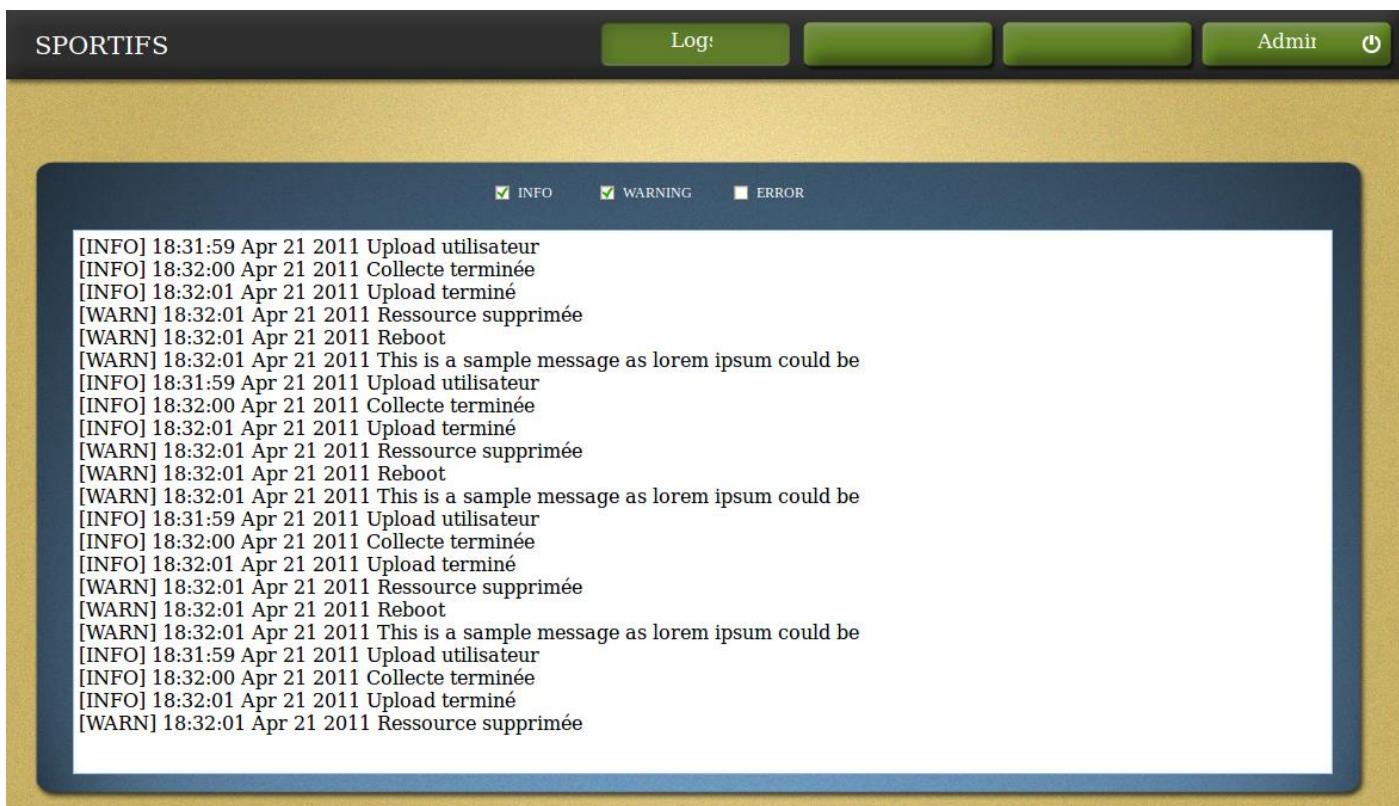
[Supprimer la sélection](#)

Cette maquette représente le portlet *P_ModelConfiguration* permettant au superviseur de paramétriser un modèle qui servira pour la comparaison lors des prochaines collectes.

Sur cette maquette est représenté le portlet *P_KitConfiguration* permettant au superviseur de pouvoir paramétrés un kit d'évaluation qui servira pour les futures collectes.

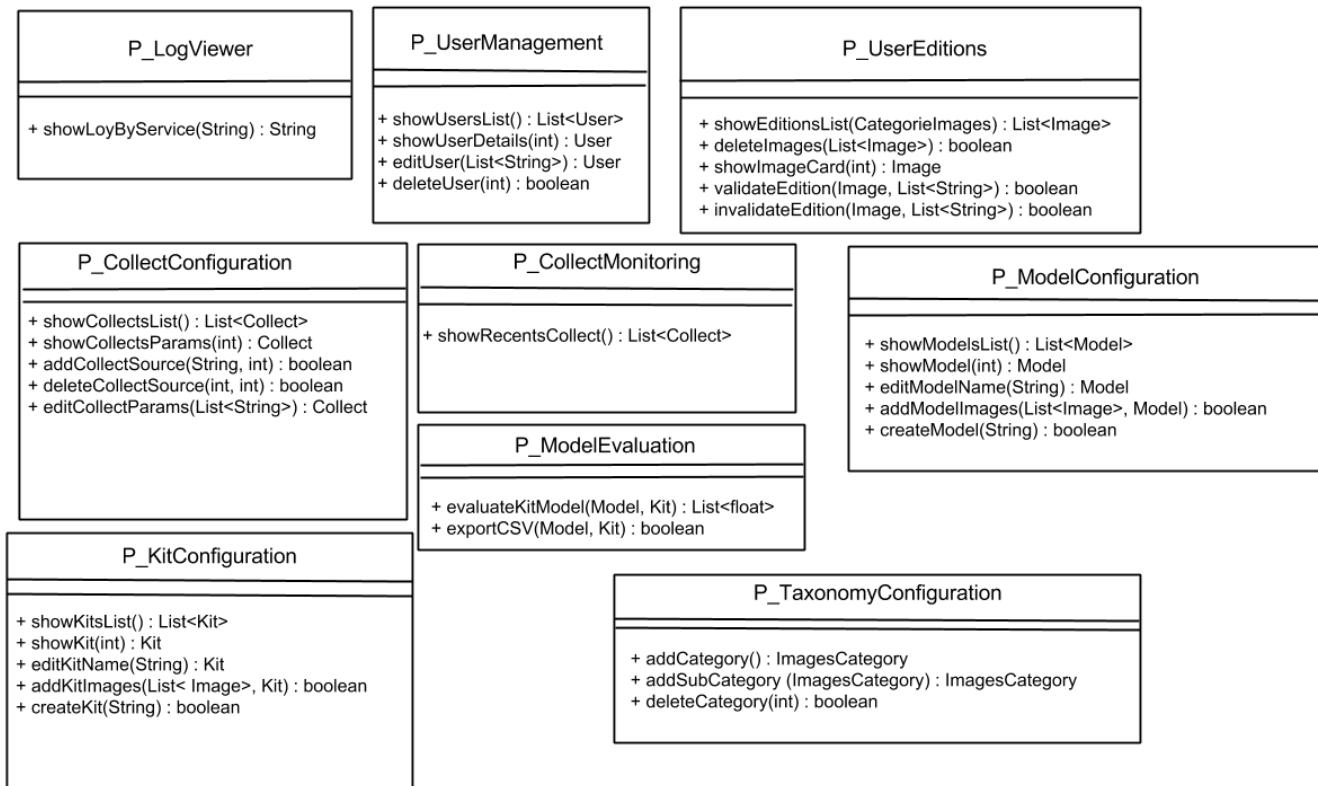
c Maquettes Administrateur

Liferay automatise la gestion des utilisateurs en ajoutant un menu (image 1) à la barre de menus de l'administrateur. Il accède ainsi au panneau de contrôle Liferay, ou Panel Control, et peut créer, modifier et supprimer des utilisateurs depuis cette même interface (image 2).



Cette maquette représente le portlet *P_LogViewer* permettant à l'administrateur de consulter les logs de l'application, que ce soit une information, une erreur ou un avertissement.

3 Description du composant Interfaçage



<p>P_RecentImages</p> <hr/> <p>+ showRecentImages(ImagesCategory) : List<Image></p>	<p>P_ImageBrowse</p> <hr/> <p>+ showTaxonomyTree() : List<ImagesCategory> + editImage(int, List<String>) : Image + deleteImage(int) : boolean</p>
<p>P_ImageSearch</p> <hr/> <p>+ searchFilters(List<Filters>) : List<Image> + searchSimilarImage(Image) : List<Image></p>	<p>P_AlertConfiguration</p> <hr/> <p>+ createAlert(List<String>, Image) : Alert + editAlert(int, List<String>, Image) : Alert + showAlertsList() : List<Alert> + deleteAlert() : boolean</p>
<p>P_ImageDetail</p> <hr/> <p>+ afficherDetailsImage (int) : Image</p>	
<p>P_ImageUpload</p> <hr/> <p>+ UploadImage() : Image</p>	

4 Description de la portlet P_LogViewer

Ce portlet permet à l'administrateur de consulter les logs. Elle affichera une liste des services pour lesquels les logs devront être affichés et un champ de texte qui montrera le contenu des logs du service choisi.

5 Description de la portlet P_UserManagement

Ce portlet permet à l'administrateur de gérer la liste de tous les utilisateurs. Il y aura une liste des utilisateurs contenant seulement les informations essentielles et lorsque l'on clique sur une liste, on affichera la fiche d'un utilisateur sur laquelle les informations pourront être modifiées

6 Description de la portlet P_UserEditions

Ce portlet permet au superviseur de consulter et de valider des éditions utilisateurs. Il pourra naviguer dans la taxonomie via un arbre, faire une distinction visuelle entre éditions et suppressions, il aura la possibilité de sélectionner en masse des suppressions (cases à cocher à côté de la miniature de chaque image), cliquer sur "Valider suppression" où un pop-up apparaîtra pour confirmer la suppression, cliquer sur image affichera la fiche de l'image avec possibilité de la valider (ou l'invalider) entièrement (ou partiellement).

7 Description de la portlet P_CollectConfiguration

Ce portlet permet au superviseur de consulter et de gérer les paramètres de la collecte. Il pourra afficher une liste des collectes, lorsque l'on clique sur une lise il affichera les paramètres de la collecte, modifier les paramètres [nom, fréquence, filtres, doublons, sources], il aura la possibilité d'ajouter ou de supprimer une source via la liste des sites enregistrés dans l'application, lorsque l'on clique sur "Enregistrer" l'enregistrement des paramètres de la collecte s'effectuera.

8 Description de la portlet P_CollectMonitoring

Ce portlet permet au superviseur de consulter l'historique des dernières collectes.

9 Description de la portlet P_ModelConfiguration

Ce portlet permet au superviseur de gérer les modèles de classification. Il pourra afficher une liste des modèles. On pourra ainsi afficher les modèles de classification (des images de référence pour chaque classe). On pourra ainsi modifier le nom. En cliquant sur "Enregistrer", l'enregistrement du modèle de classification s'effectuera. Ce portlet permet également de naviguer dans la taxonomie via un arbre, offrant la possibilité de sélectionner en masse des images (cases à cocher à côté de la miniature de chaque image). Lorsque l'on clique sur "Ajouter à un modèle de classification", une pop-up s'affichera pour choisir le modèle de classification ou le créer le cas échéant et choisir la classe.

10 Description de la portlet P_KitConfiguration

Ce portlet permet au superviseur de gérer les kits d'évaluations. Il pourra afficher la liste des kits d'évaluation. On pourra ainsi afficher les kits (des images pour chaque classe). On pourra modifier le nom. Lorsque l'on clique "Enregistrer", l'enregistrement du kit d'évaluation s'effectuera. Ce portlet permet également de naviguer dans la taxonomie via un arbre, offrant la possibilité de sélectionner en masse des images (cases à cocher à côté de la miniature de chaque image). Lorsque l'on clique sur "Ajouter à un kit", une pop-up s'affichera pour choisir le kit d'évaluation ou le créer le cas échéant (plus choix de la classe).

11 Description de la portlet P_ModelEvaluation

Ce portlet permet au superviseur d'évaluer un modèle de classification. Elle affichera une liste des modèles de classification et une liste des kits d'évaluation. En cliquant sur "Évaluer", elle affichera la matrice avec son de confusion. En cliquant sur "Exporter", on pourra exporter la matrice en CSV.

12 Description de la portlet P_TaxonomyConfiguration

Ce portlet permet au superviseur d'éditer la taxonomie. Elle pourra afficher l'arbre de taxonomie, ajouter ou supprimer un concept ou un sous-concept, lorsque l'on clique sur "Enregistrer" l'enregistrement de la taxonomie s'effectuera.

13 Description de la portlet P_RecentImages

Ce portlet permet à l'utilisateur de consulter et de gérer les images récentes. Elle permettra la navigation dans la taxonomie via un arbre, lorsque l'on survol une image une vue synthétique de l'image (informations principales + "Détails" + "Supprimer") s'affichera, lorsque l'on clique sur "Détails" ou image il affichera une vue détaillée de l'image (informations complètes + "Éditer" + "Supprimer"), lorsque l'on clique sur "Supprimer" une fenêtre pop-up s'affichera pour confirmer la suppression, lorsque l'on cliquer sur "Éditer", on pourra mettre à jour la fiche de l'image puis "Enregistrer" ou "Annuler" les modifications effectués, lorsque l'on clique sur "Enregistrer" l'enregistrement de la fiche s'effectuera, lorsque l'on clique sur "Annuler" les modifications ne seront pas prises en compte.

14 Description de la portlet P_ImageBrowse

Ce portlet permet à l'utilisateur de naviguer virtuellement dans le corpus des images. Il pourra naviguer dans la taxonomie via un explorateur de fichier, lorsque l'on clique sur une catégorie avec sous-catégorie(s) on pourra descendre dans la catégorie, lorsque l'on clique sur une catégorie sans sous-catégorie il affichera les images de la catégorie, lors d'un survol d'une image, il affichera une vue synthétique (informations principales + "Détails" + "Supprimer"), lorsque l'on clique sur "Détails" ou sur une image il affichera une vue détaillée (informations complètes + "Éditer" + "Supprimer"), lorsque l'on clique sur "Supprimer" une fenêtre pop-up s'affichera pour confirmer la suppression, lorsque l'on clique sur "Éditer", on pourra mettre à jour la fiche puis l'enregistrer ou annuler l'action, lorsque l'on clique sur "Enregistrer" l'enregistrement de la fiche s'effectuera, lorsque l'on clique sur "Annuler" l'annulation des modifications.

15 Description de la portlet P_ImageSearch

Ce portlet permet à l'utilisateur de rechercher des images. Lorsque l'on clique sur "Rechercher" il affichera une liste filtrée des résultats et des propositions de filtres, lorsque l'on clique sur "Recherche avancée", on pourra ajouter ou supprimer des filtres et faire des comparaisons avec les images uploader, lorsque l'on clique sur "Comparer avec upload" une fenêtre pop-up de sélection image s'affichera, lorsque l'on survol une image il affichera une vue synthétique de l'image (informations principales + "Détails" + "Supprimer" + "Similarité"), lorsque l'on clique sur "Détails" ou sur image, il affichera une vue détaillée de l'image (informations complètes + "Éditer" + "Supprimer"), lorsque l'on clique sur "Supprimer" une fenêtre pop-up s'affichera pour confirmer la suppression, lorsque l'on clique sur "Similarité" une fenêtre pop-up s'affichera permettant de choisir la région à sélectionner, lorsque l'on clique sur "Éditer" on pourra mettre à jour la fiche puis cliquer sur "Enregistrer" ou sur "Annuler", lorsque l'on clique sur "Enregistrer" l'enregistrement de la fiche s'effectuera, lorsque l'on clique sur "Annuler" aucune modification ne sera pris en compte.

16 Description de la portlet P_AlertConfiguration

Ce portlet permet à l'utilisateur de consulter et de gérer les alertes. Il affichera la liste des alertes, lorsque l'on clique sur une liste d'alertes il affichera l'alerte (et les filtres utilisés), on pourra modifier le nom de l'alerte, lorsque l'on clique sur "Enregistrer" l'enregistrement de l'alerte s'effectuera, il affichera une liste des alertes dans barre de menu, lorsque l'on clique sur une alerte il affichera une page de recherche avec filtres alerte (les images récentes).

17 Description de la portlet P_ImageDetail

Ce portlet permet à l'utilisateur d'avoir toutes les informations concernant une image.

18 Description de la portlet P_ImageUpload

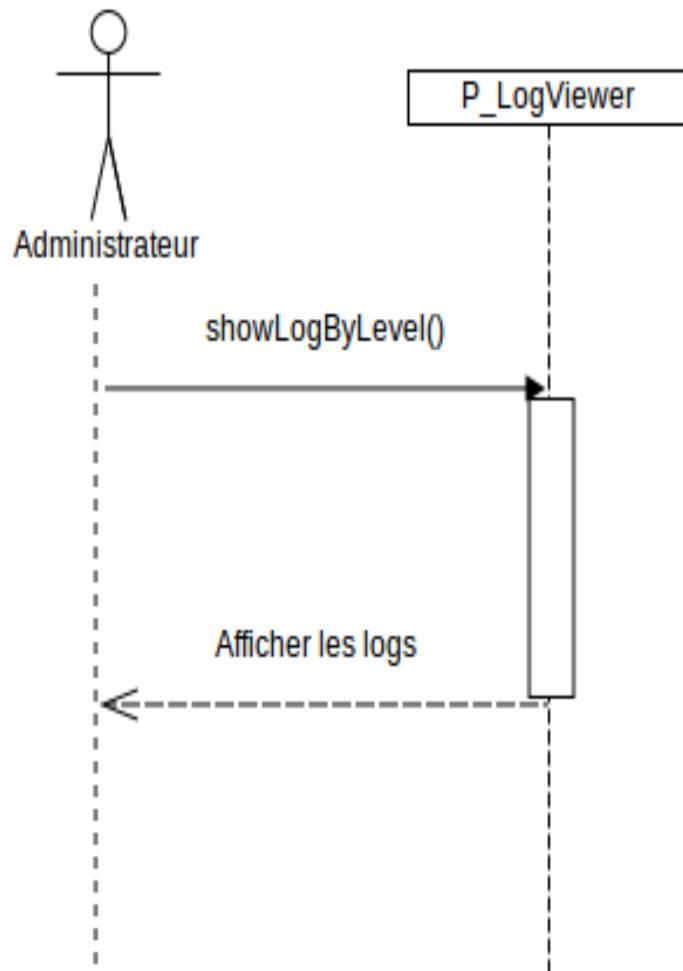
Ce portlet permet à l'utilisateur d'uploader une image pour qu'elle soit collectée.

B Fonctionnement dynamique

Dans la spécification technique de besoin, un certain nombre de cas d'utilisation ont été identifiés et détaillés. Dans ce présent document, nous avons défini précédemment les différents composants rentrant en jeu dans le lot interfaçage du démonstrateur SPORTIFS. Afin de mieux comprendre le fonctionnement de ces composants ainsi que leurs interactions, nous allons détailler ces cas d'utilisation à l'aide de diagrammes de séquence.

1 Consulter les logs : P_LogViewer

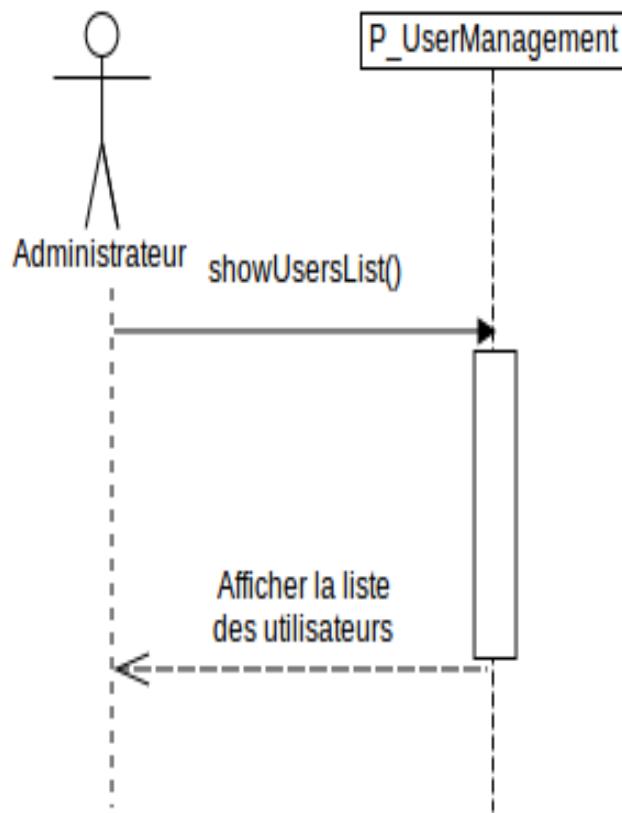
Ce diagramme correspond au fonctionnement du portlet *P_LogViewer* qui permet à l'administrateur de consulter les logs selon un niveau choisi.



2 Gérer les utilisateurs : *P_UserManagement*

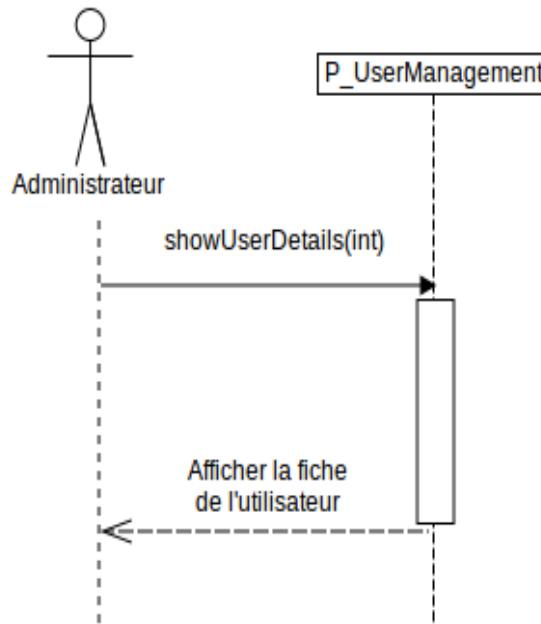
a Afficher la liste des utilisateurs

Ce diagramme correspond au fonctionnement du portlet *P_UserManagement* qui permet à l'administrateur d'afficher la liste des utilisateurs de l'application.



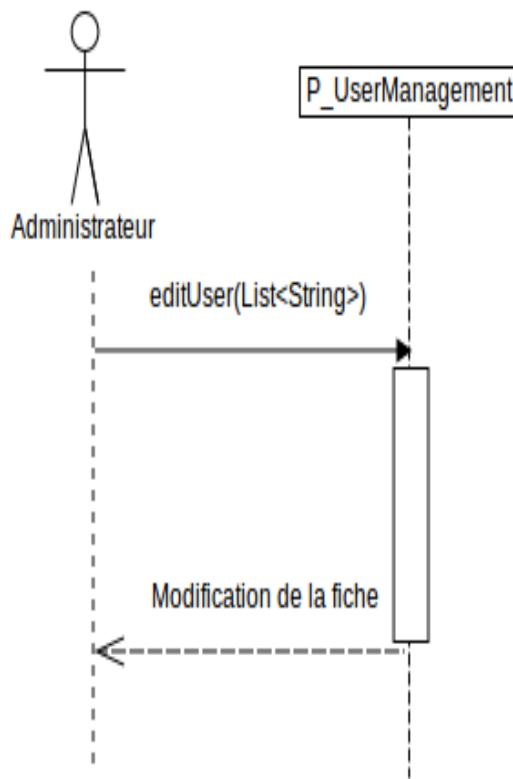
b Afficher les détails d'un utilisateur

Ce diagramme correspond au fonctionnement du portlet *P_UserManagement* qui permet à l'administrateur d'afficher la fiche d'un utilisateur de l'application.



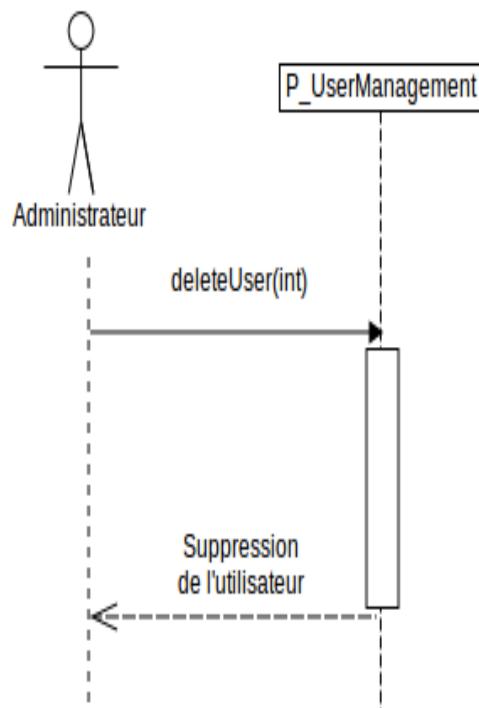
c Modifier les informations de la fiche d'un utilisateur

Ce diagramme correspond au fonctionnement du portlet *P_UserManagement* qui permet à l'administrateur de modifier la fiche d'un utilisateur de l'application.



d Supprimer un utilisateur

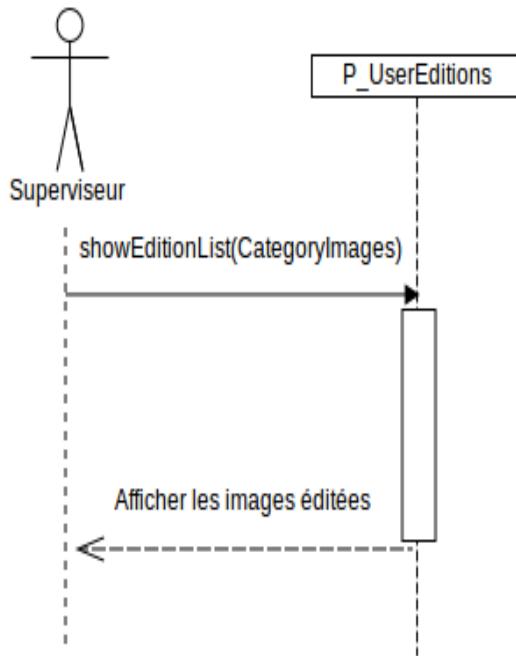
Ce diagramme correspond au fonctionnement du portlet *P_UserManagement* qui permet à l'administrateur de supprimer un utilisateur de l'application.



3 Gérer les éditions des utilisateurs : *P_UserEditions*

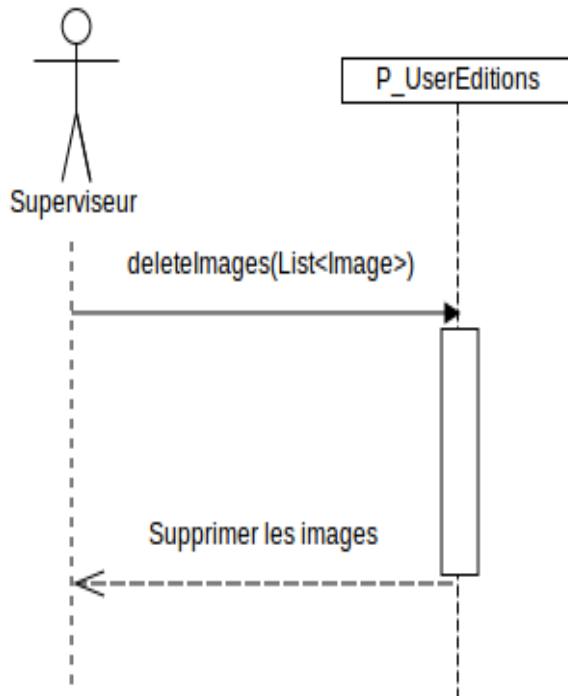
a Afficher les images éditées

Ce diagramme correspond au fonctionnement du portlet *P_UserEditions* qui permet au superviseur de consulter les images éditées par les utilisateurs.



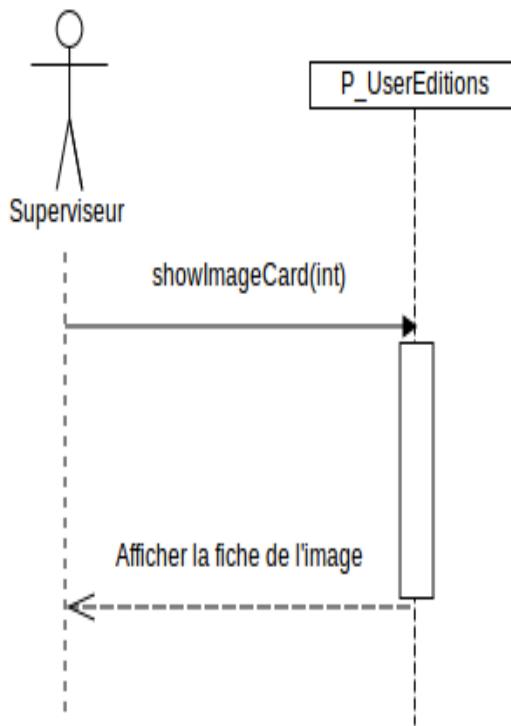
b Supprimer les images éditées

Ce diagramme correspond au fonctionnement du portlet *P_UserEditions* qui permet au superviseur de supprimer les images éditées par les utilisateurs.



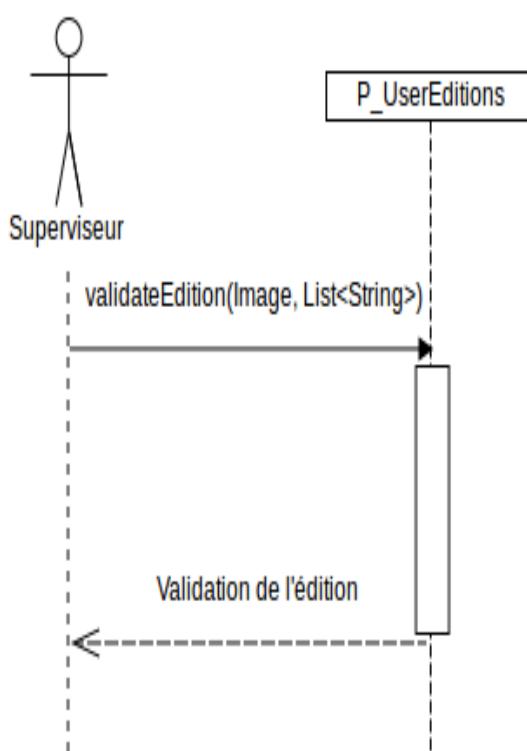
c Consulter la fiche d'une image

Ce diagramme correspond au fonctionnement du portlet *P_UserEditions* qui permet au superviseur de consulter la fiche d'une image.



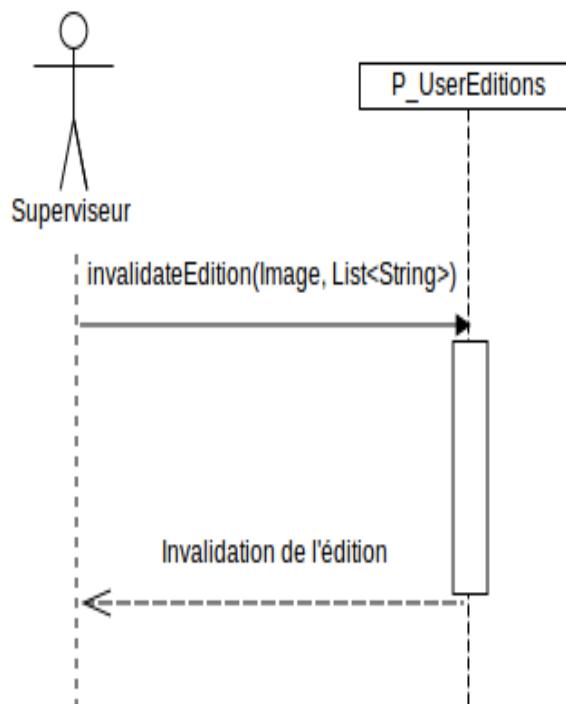
d Validation de l'édition

Ce diagramme correspond au fonctionnement du portlet *P_UserEditions* qui permet au superviseur de valider une édition d'un utilisateur de l'application.



e Invalidation de l'édition

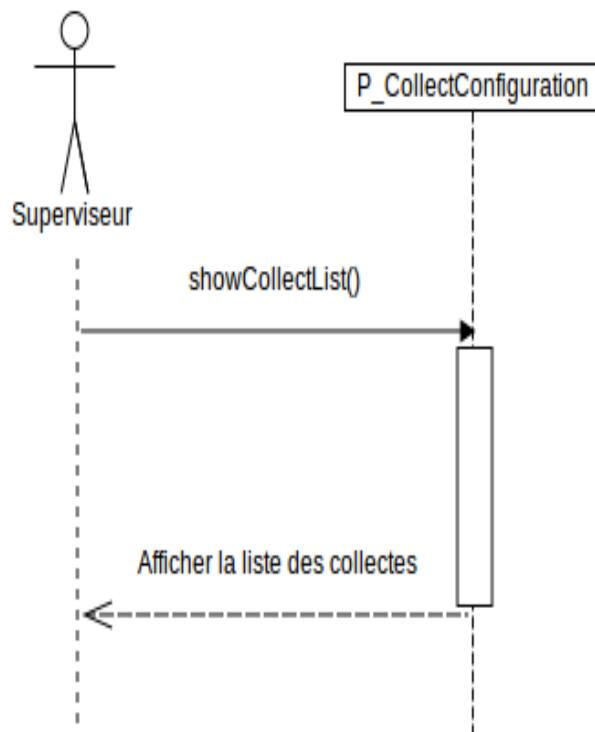
Ce diagramme correspond au fonctionnement du portlet P_UserEditions qui permet au superviseur d'invalider une édition d'un utilisateur de l'application.



4 Gérer les paramètres de collecte : P_CollectConfiguration

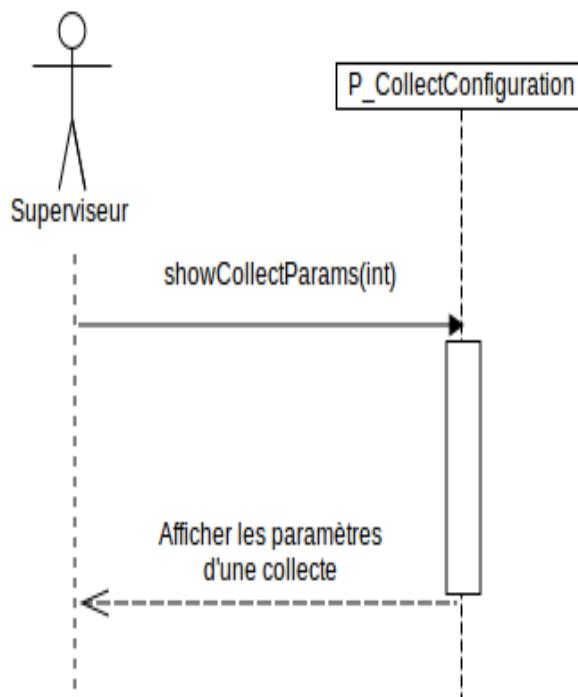
a Afficher la liste des collectes

Ce diagramme correspond au fonctionnement du portlet P_CollectConfiguration qui permet au superviseur d'afficher la liste des collectes précédemment créée.



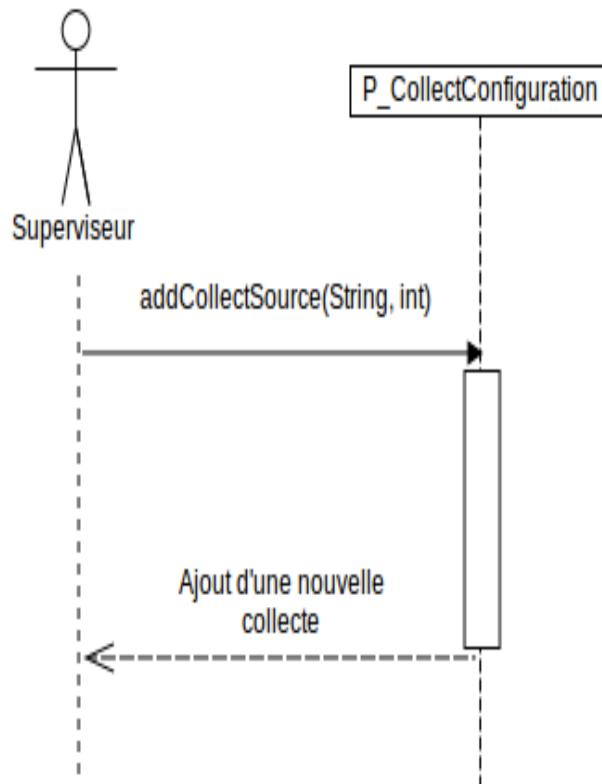
b Afficher les paramètres d'une collecte

Ce diagramme correspond au fonctionnement du portlet *P_CollectConfiguration* qui permet au superviseur d'afficher les paramètres d'une collecte.



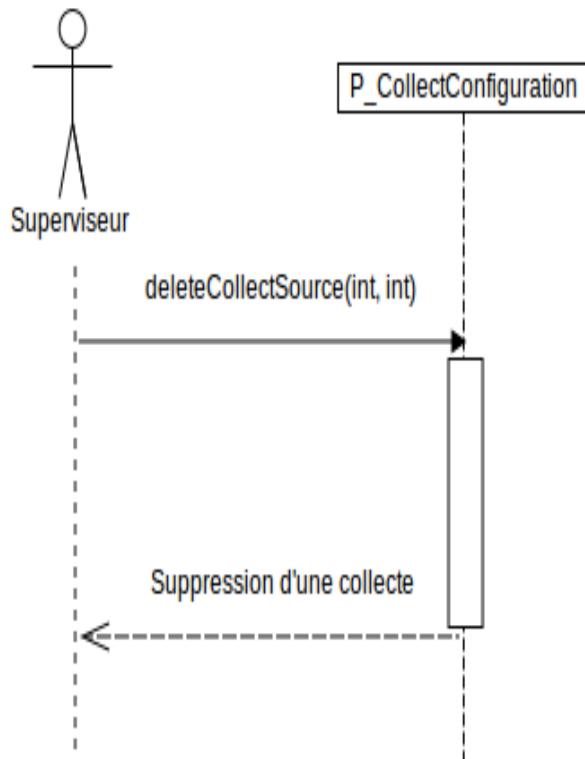
c Ajouter une collecte

Ce diagramme correspond au fonctionnement du portlet *P_CollectConfiguration* qui permet au superviseur d'ajouter une nouvelle collecte.



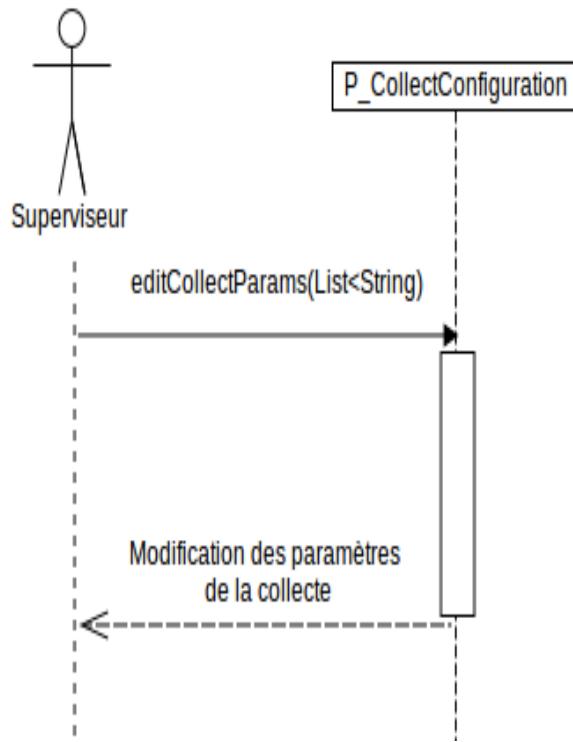
d Supprimer une collecte

Ce diagramme correspond au fonctionnement du portlet *P_CollectConfiguration* qui permet au superviseur d'ajouter une nouvelle collecte.



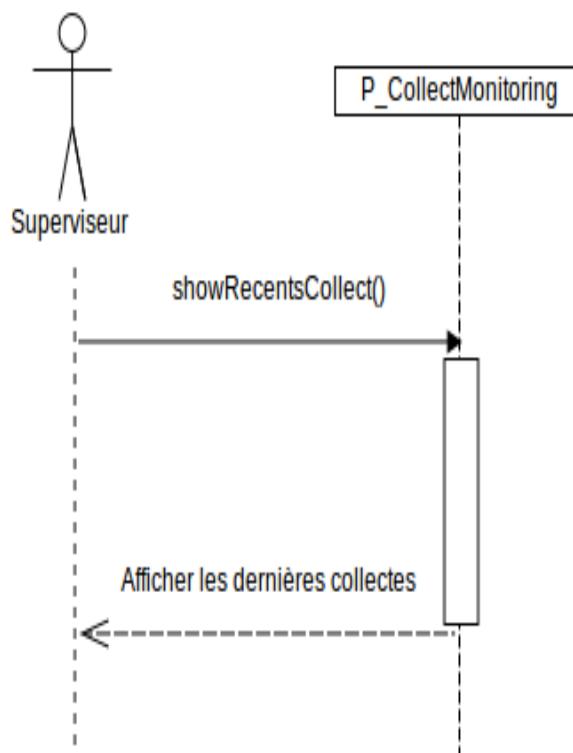
e Modifier les paramètres d'une collecte

Ce diagramme correspond au fonctionnement du portlet *P_CollectConfiguration* qui permet au superviseur de modifier les paramètres d'une collecte.



5 Consulter l'historique des collectes : *P_CollectMonitoring*

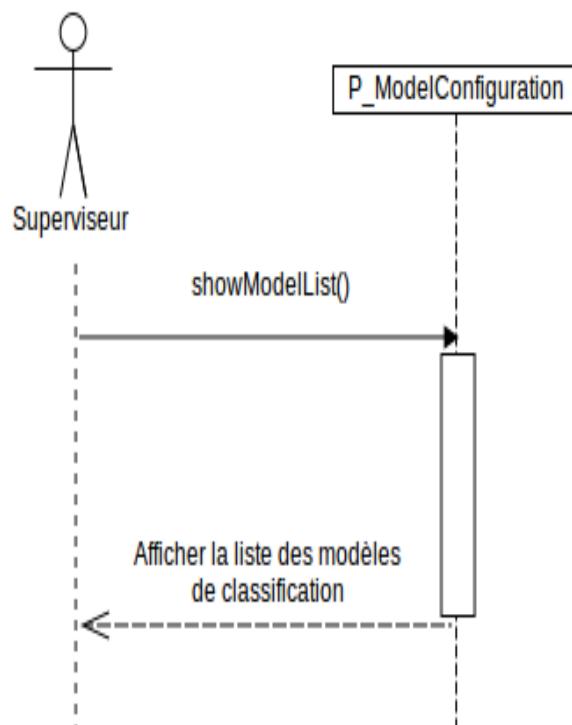
Ce diagramme correspond au fonctionnement du portlet *P_CollectMonitoring* qui permet au superviseur de consulter l'historique des dernières collectes.



6 Gérer les modèles de classification : *P_ModelConfiguration*

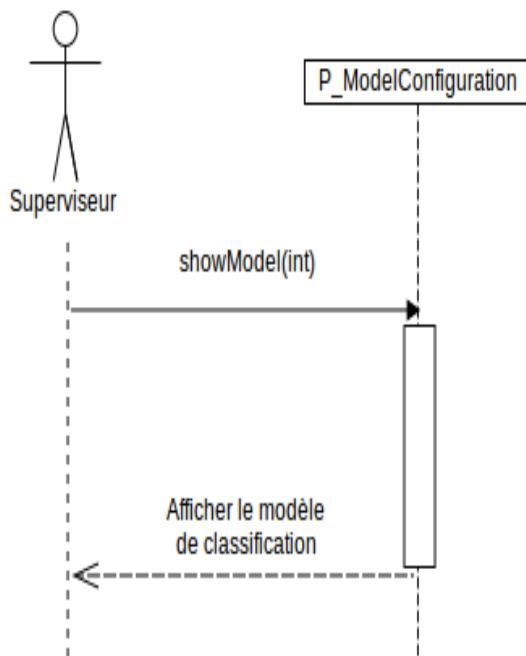
a Consulter la liste des modèles

Ce diagramme correspond au fonctionnement du portlet *P_ModelConfiguration* qui permet au superviseur de consulter la liste des modèles de classification.



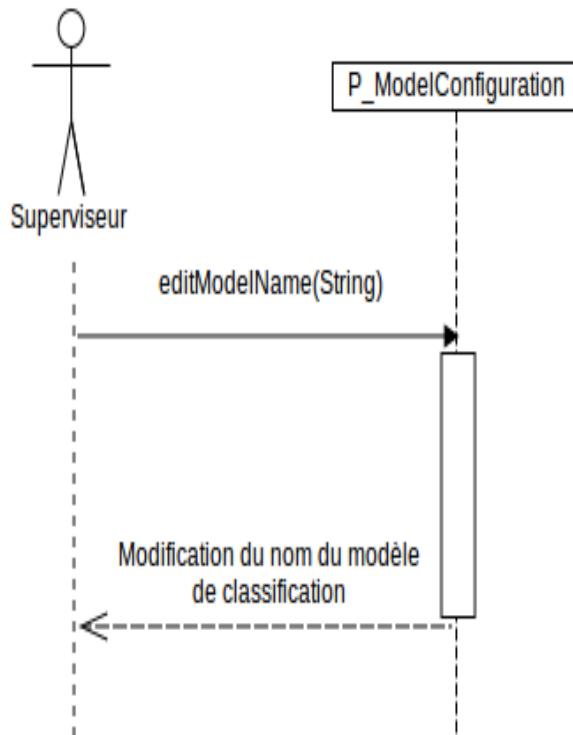
b Voir un modèle de classification

Ce diagramme correspond au fonctionnement du portlet *P_ModelConfiguration* qui permet au superviseur de consulter un modèle de classification.



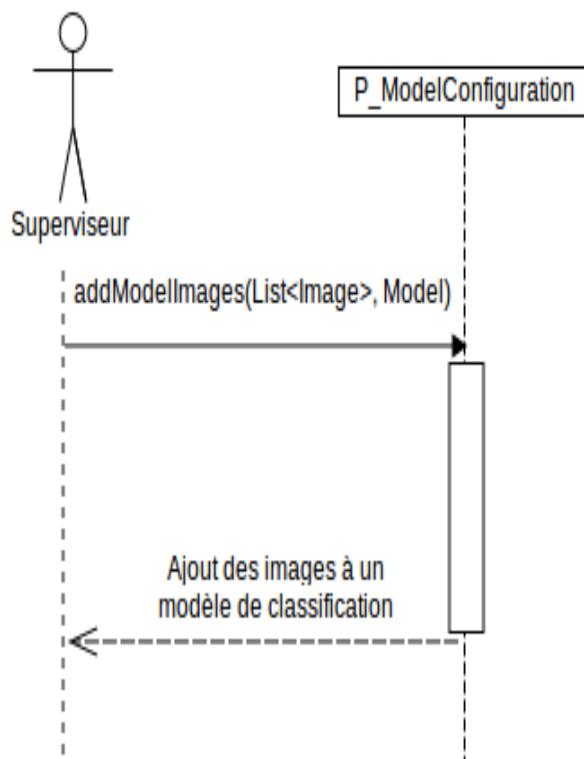
c Modifier le nom d'un modèle de classification

Ce diagramme correspond au fonctionnement du portlet *P_ModelConfiguration* qui permet au superviseur de modifier le nom d'un modèle de classification.



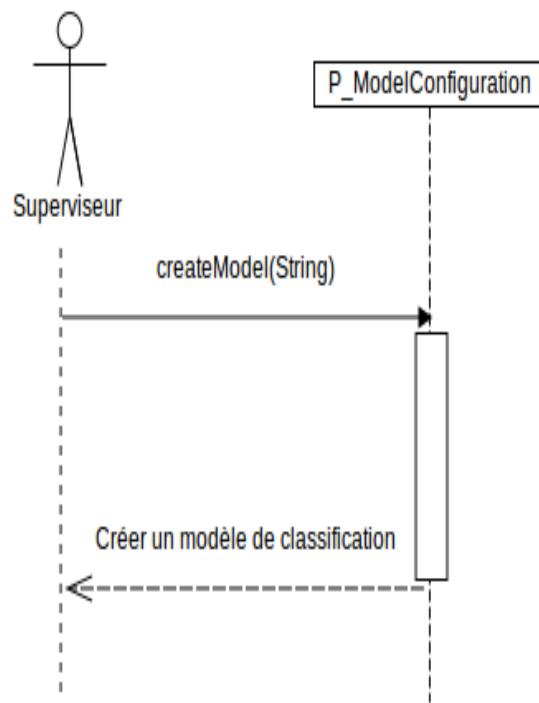
d Ajouter des images à un modèle de classification

Ce diagramme correspond au fonctionnement du portlet *P_ModelConfiguration* qui permet au superviseur d'ajouter des images à un modèle de classification.



e Créer un modèle de classification

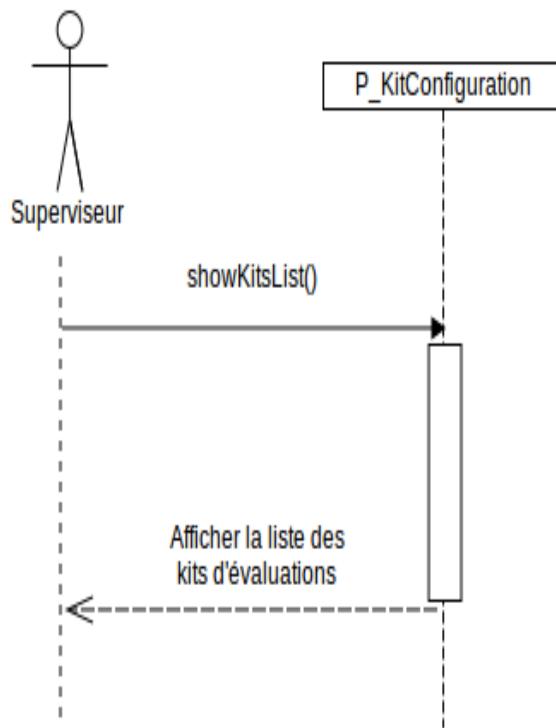
Ce diagramme correspond au fonctionnement du portlet *P_ModelConfiguration* qui permet au superviseur de créer un modèle de classification.



7 Gérer les kits d'évaluations : *P_KitConfiguration*

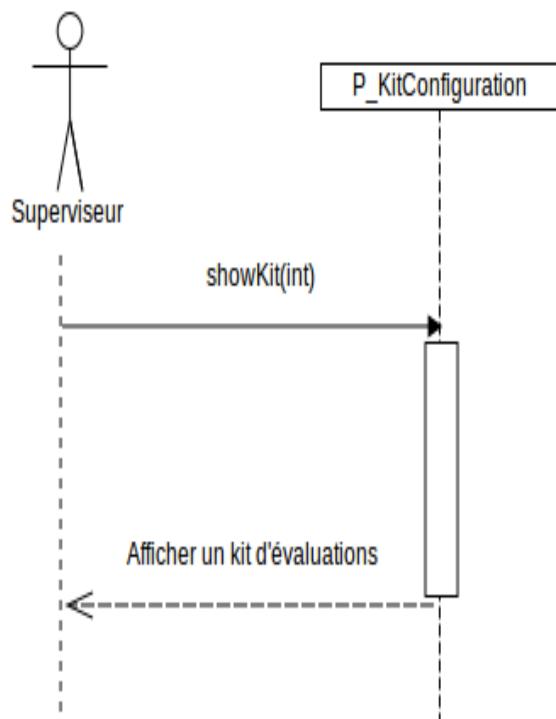
a Afficher la liste des kits d'évaluations

Ce diagramme correspond au fonctionnement du portlet *P_KitConfiguration* qui permet au superviseur d'afficher la liste des kits d'évaluations.



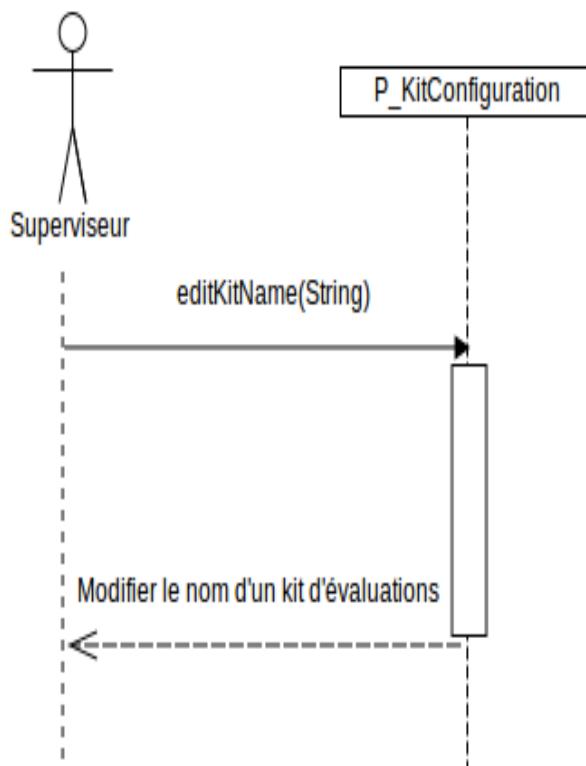
b Afficher un kit d'évaluations

Ce diagramme correspond au fonctionnement du portlet *P_KitConfiguration* qui permet au superviseur d'afficher en détail un kit d'évaluations.



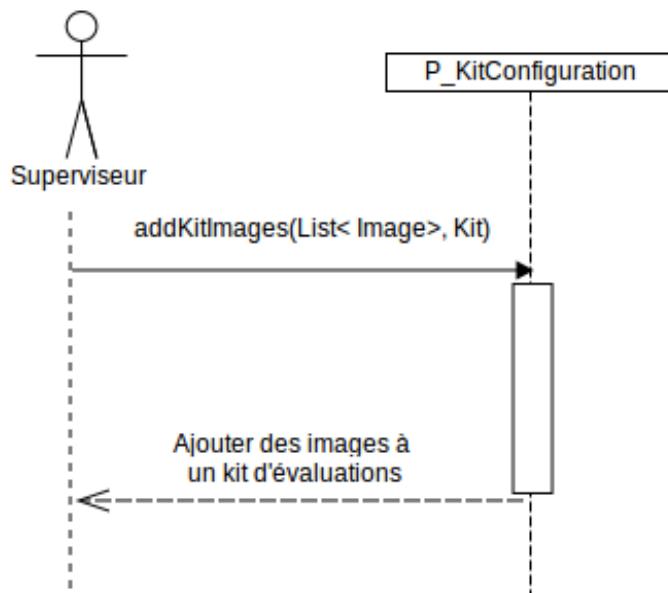
c Modifier le nom d'un kit d'évaluations

Ce diagramme correspond au fonctionnement du portlet *P_KitConfiguration* qui permet au superviseur de modifier le nom d'un kit d'évaluations.



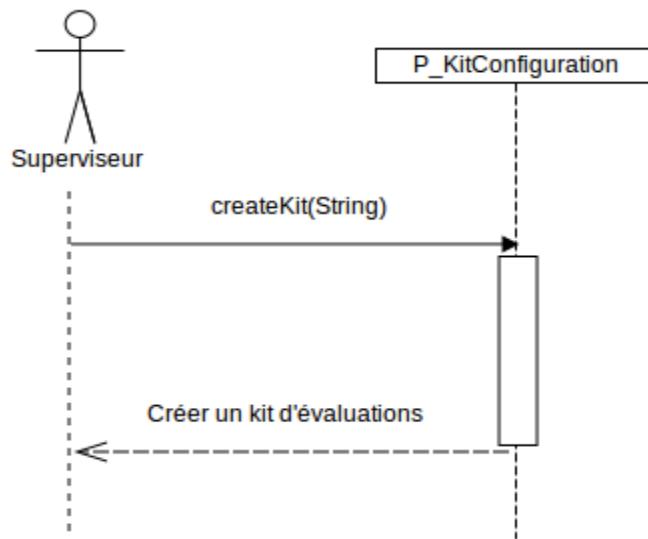
d Ajouter des images à un kit d'évaluations

Ce diagramme correspond au fonctionnement du portlet *P_KitConfiguration* qui permet au superviseur d'ajouter des images à un kit d'évaluations.



e Créer un nouveau kit d'évaluations

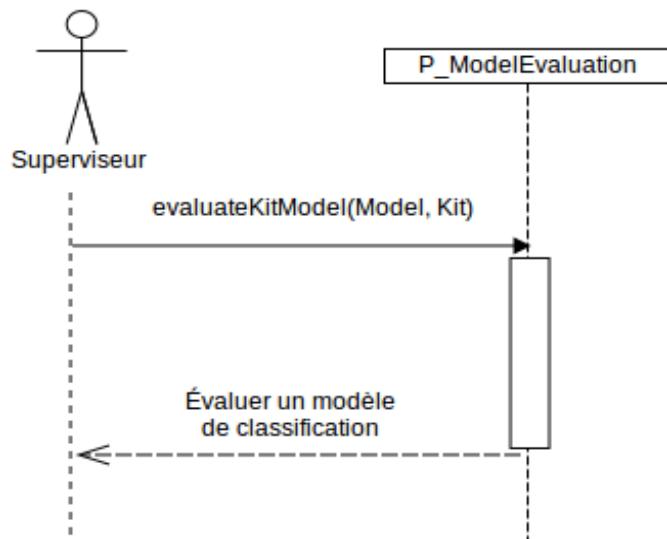
Ce diagramme correspond au fonctionnement du portlet *P_KitConfiguration* qui permet au superviseur de créer un nouveau kit d'évaluations.



8 Gérer l'évaluation des modèles de classification : *P_ModelEvaluation*

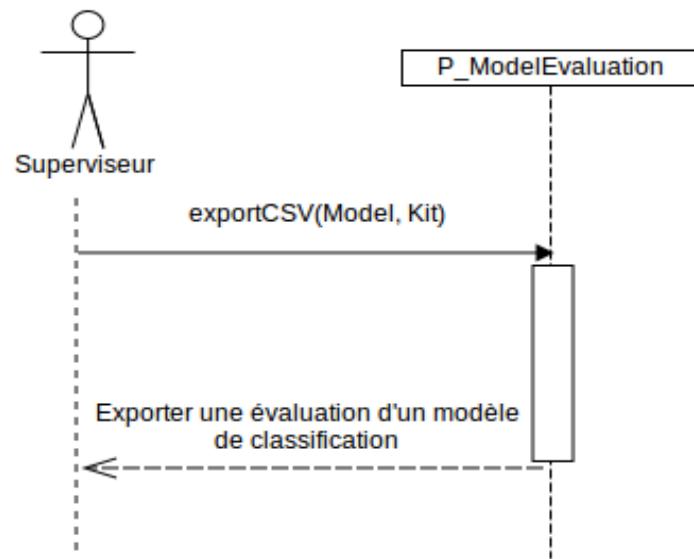
a Évaluer un modèle de classification

Ce diagramme correspond au fonctionnement du portlet *P_ModelEvaluation* qui permet au superviseur d'évaluer un modèle de classification en fonction d'un kit d'évaluations.



b Exporter une évaluation en CSV

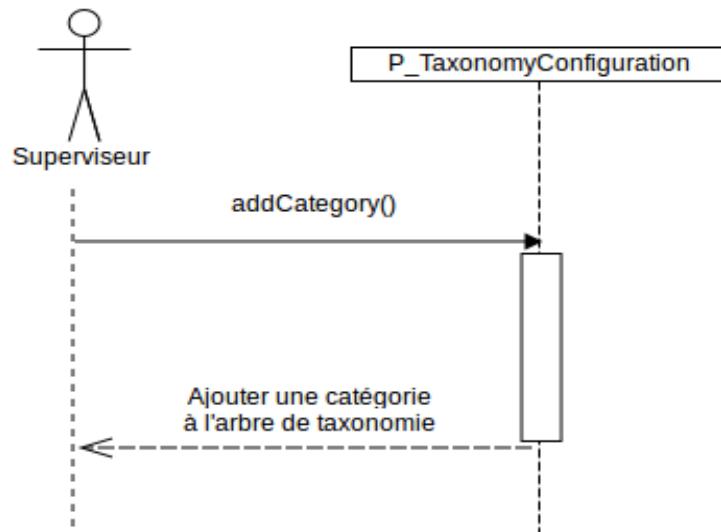
Ce diagramme correspond au fonctionnement du portlet *P_ModelConfiguration* qui permet au superviseur d'exporter un modèle de classification au format CSV.



9 Gérer la taxonomie : *P_TaxonomyConfiguration*

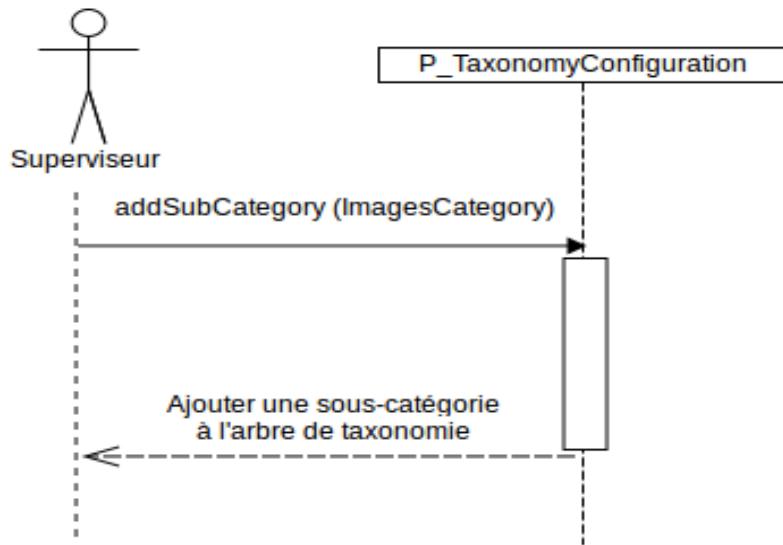
a Ajouter une catégorie

Ce diagramme correspond au fonctionnement du portlet *P_TaxonomyConfiguration* qui permet au superviseur d'ajouter une catégorie à l'arbre de taxonomie.



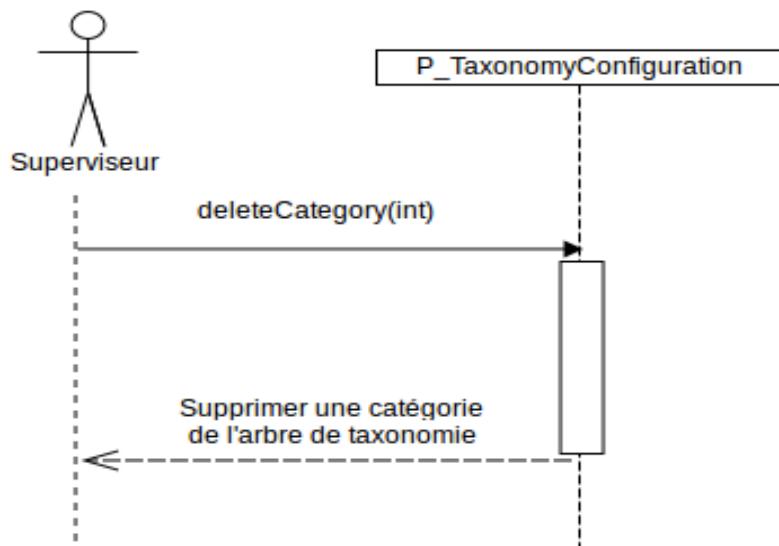
b Ajouter une sous-catégorie

Ce diagramme correspond au fonctionnement du portlet *P_TaxonomyConfiguration* qui permet au superviseur d'ajouter une sous-catégorie à l'arbre de taxonomie.



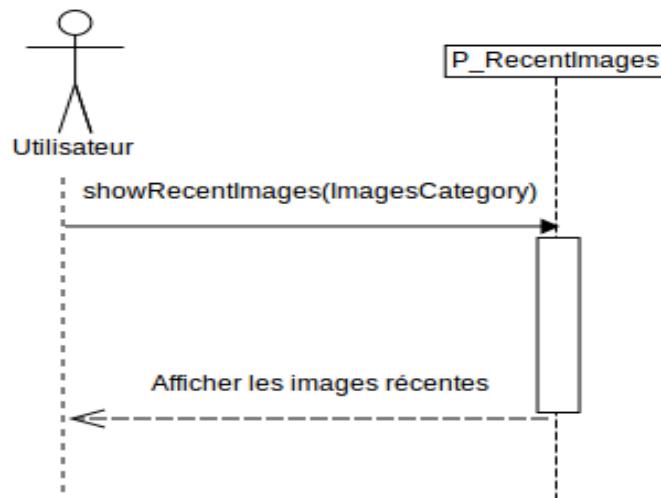
c Supprimer une catégorie

Ce diagramme correspond au fonctionnement du portlet *P_TaxonomyConfiguration* qui permet au superviseur de supprimer une catégorie de l'arbre de taxonomie.



10 Gérer les images récentes : *P_RecentImages*

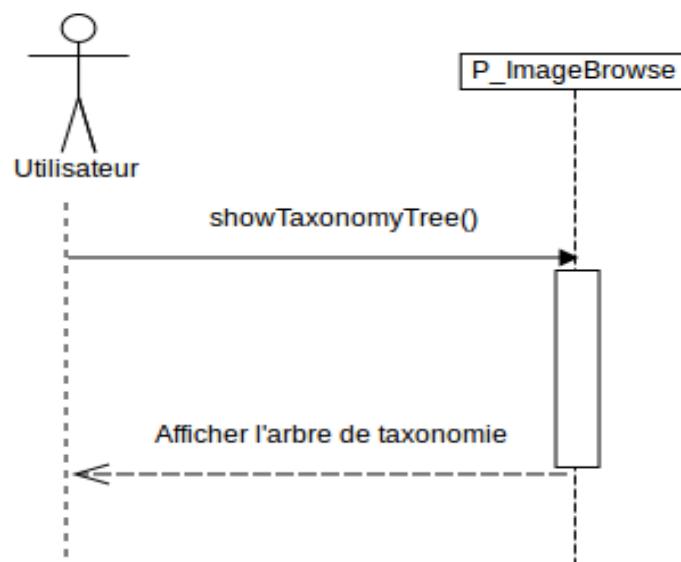
Ce diagramme correspond au fonctionnement du portlet *P_RecentImages* qui permet à l'utilisateur d'afficher les images récentes.



11 Naviguer dans le corpus d'images : *P_ImageBrowse*

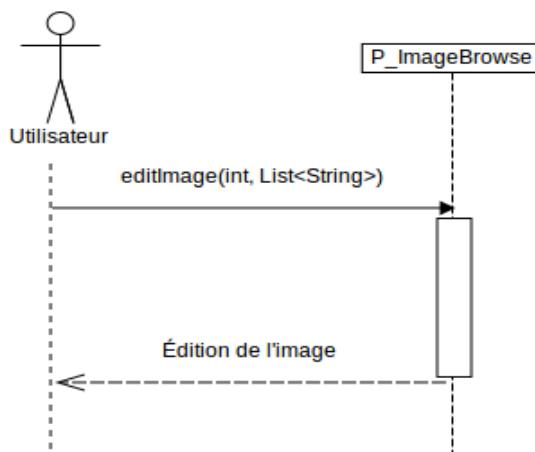
a Afficher l'arbre de taxonomie

Ce diagramme correspond au fonctionnement du portlet *P_ImageBrowse* qui permet à l'utilisateur d'afficher l'arbre de taxonomie.



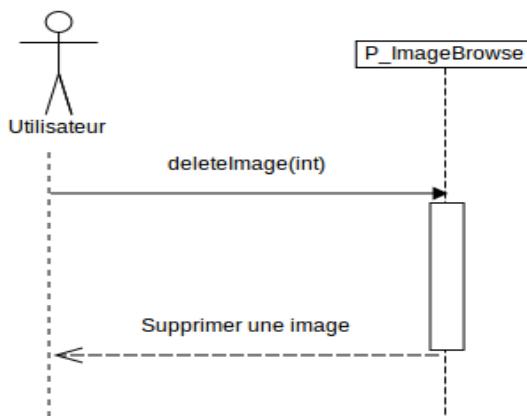
b Modifier une image

Ce diagramme correspond au fonctionnement du portlet *P_ImageBrowse* qui permet à l'utilisateur de modifier la fiche d'une image.



c Supprimer une image

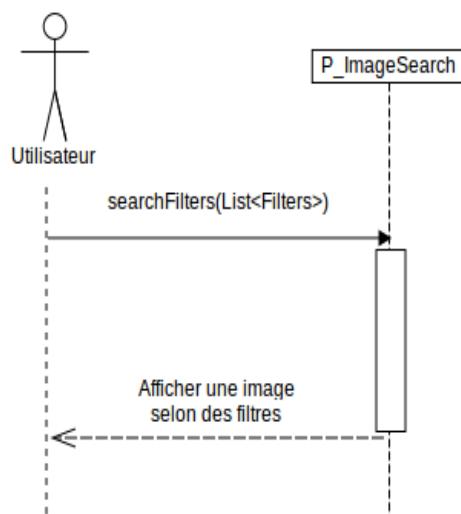
Ce diagramme correspond au fonctionnement du portlet *P_ImageBrowse* qui permet à l'utilisateur de supprimer une image.



12 Rechercher des images : *P_ImageSearch*

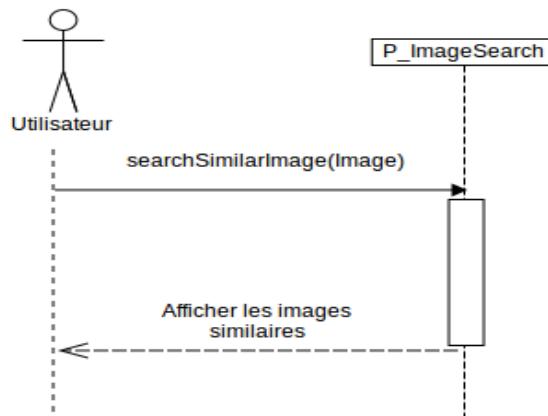
a Rechercher une image selon des filtres

Ce diagramme correspond au fonctionnement du portlet *P_ImageSearch* qui permet à l'utilisateur de rechercher une image selon des filtres.



b Rechercher des images similaires

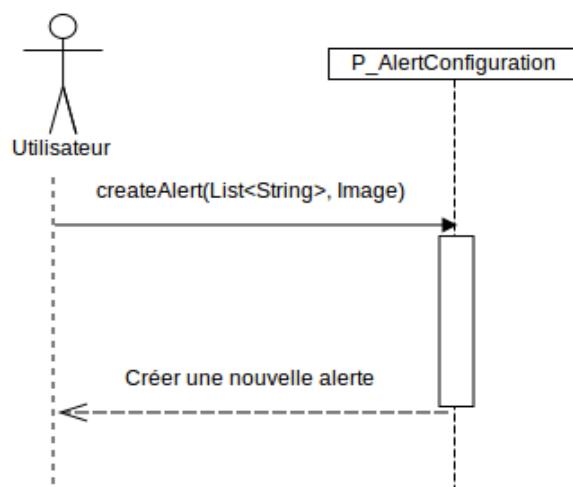
Ce diagramme correspond au fonctionnement du portlet *P_ImageSearch* qui permet à l'utilisateur de rechercher des images similaires à une autre image.



13 Gérer les alertes : *P_AlertConfiguration*

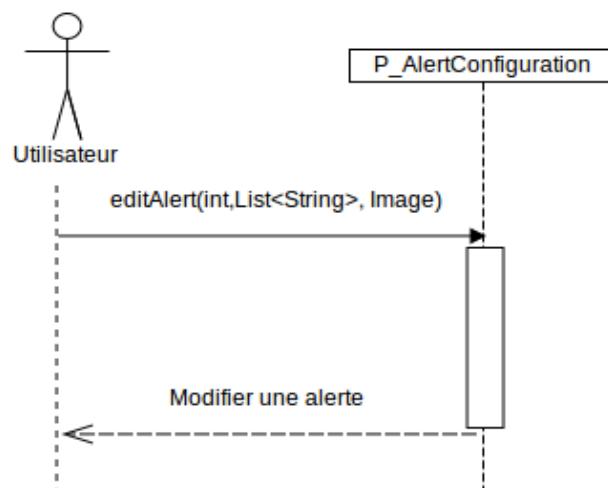
a Créer une nouvelle alerte

Ce diagramme correspond au fonctionnement du portlet *P_AlertConfiguration* qui permet à l'utilisateur de créer une nouvelle alerte.



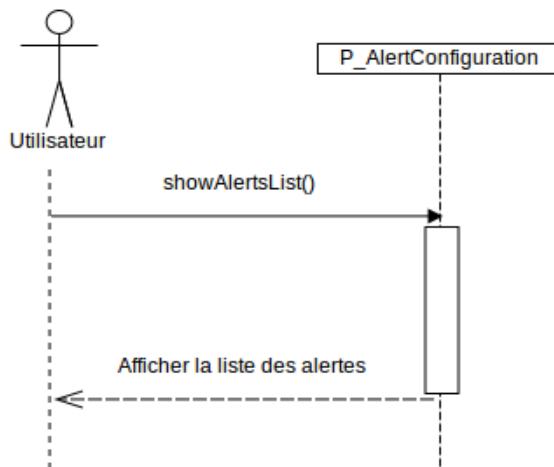
bModifier une alerte

Ce diagramme correspond au fonctionnement du portlet *P_AlertConfiguration* qui permet à l'utilisateur de modifier une alerte.



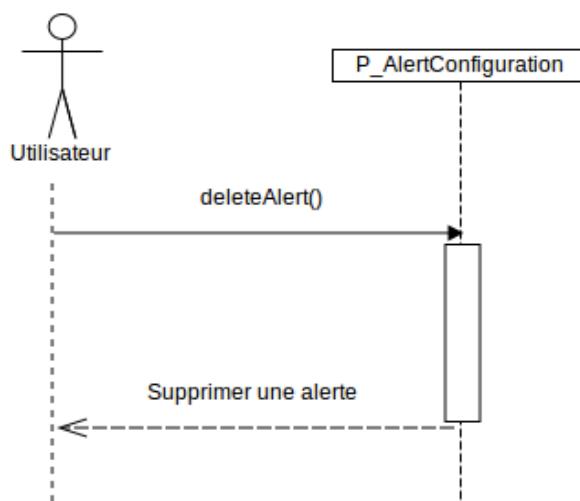
c Afficher la liste des alertes

Ce diagramme correspond au fonctionnement du portlet *P_AlertConfiguration* qui permet à l'utilisateur d'afficher toutes les alertes.



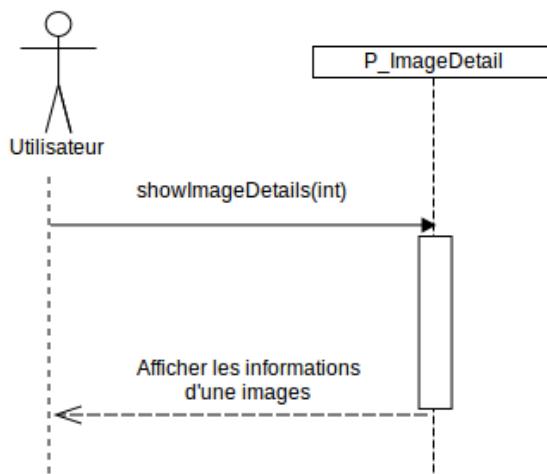
d Supprimer une alerte

Ce diagramme correspond au fonctionnement du portlet *P_AlertConfiguration* qui permet à l'utilisateur de supprimer une alerte.



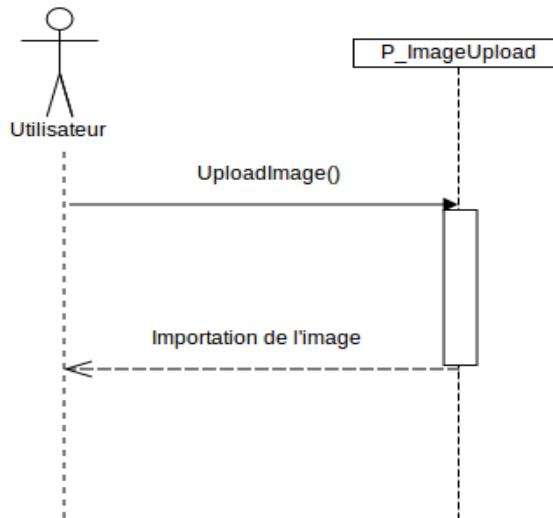
14 Avoir les informations d'une image : *P_ImageDetail*

Ce diagramme correspond au fonctionnement du portlet *P_ImageDetail* qui permet à l'utilisateur d'avoir les informations d'une image.



15 Uploader une image : P_ImageUpload

Ce diagramme correspond au fonctionnement du portlet *P_ImageUpload* qui permet à l'utilisateur d'uploader une image.



C Justifications techniques

1 Présentation du Framework JSF

Java server Faces est un Framework de développement d'applications Web en java. Il est de type MVC, basé sur des composants côté présentation et destiné aux applications web respectant l'architecture J2EE.

2 Composants du Framework JSF

JSF est constitué principalement de :

- un ensemble d'APIs pour la représentation et la gestion des composants, de leur état, des évènements, de la validation des entrées et la conversion des sorties, l'internationalisation et l'accessibilité ainsi que la navigation inter-vues ;
- deux jeux de composants standards HTML et CORE (affichage de texte, saisie de texte, tables, zone à cocher, etc.) ;
- deux bibliothèques de balises JSP (une pour chaque jeu de composants) pour permettre l'utilisation des JSPs pour la construction de vues JSF ;
- un modèle évènementiel côté serveur ;
- des Managed Beans qui forment la couche contrôle de JSF ;
- et un Unified Expression Language (abrégué en EL) ou langage d'expressions unifié pour JSF et JSP 2.0 qui permet de lier les composants aux managed beans.

3 Avantages du Framework JSF

Ce Framework est considéré comme un des standards de JEE. En effet, JSF apporte un grand nombre d'apports et d'avantages aux communautés des développeurs. Parmi ces avantages, on peut citer :

- une séparation entre la couche présentation et les autres couches d'une application web ;
- une mise en place d'un mapping HTML/OBJET ;
- la réutilisation de composants graphiques ;
- une gestion de l'état de l'interface entre les différentes requêtes ;
- une liaison entre les actions côté Client et les actions des objets Java côté Serveur ;
- la création de composants customs grâce à une API ;

- le support de différents clients (HTML, WML, XML, ...) grâce à la séparation des problématiques de construction de l'interface et du rendu de cette interface ;
- un système de navigation statique et dynamique très souple ;
- et une personnalisation des labels (internationalisation, messages d'erreurs).

4 Inconvénients du Framework JSF

JSF a quelques inconvénients dont le plus important est qu'il n'autorise que peu de contrôle sur le code HTML/CSS/JS généré. Cela ne tient en réalité pas à JSF en lui-même, mais simplement au fait que c'est un Framework MVC basé sur les composants, et pas sur les requêtes (actions).

5 Présentation de la bibliothèque PrimeFaces de JSF

Sur un projet JSF il est très utile de sélectionner une bibliothèque de composants graphiques pour gagner en productivité et efficacité. Cela évite de réinventer la roue. Parmi les différentes bibliothèques disponibles et connus, on trouve PrimeFaces.

L'intérêt principal de PrimeFaces réside dans la diversité et la qualité des composants proposés. Ils sont nombreux, plus de 100, et répondent le plus souvent en standard aux besoins des applications. Ce sont des composants graphiques avancés qui possèdent des fonctionnalités prêtes à l'emploi, aidant ainsi à créer aisément des RIA (Rich Internet Application).

Le code généré par cette bibliothèque est simple et lisible. De plus, l'utilisation d'Ajax est très présente dans PrimeFaces.

Il existe plusieurs catégories de composants dans PrimeFaces qui peuvent nous aider dans le développement de nos interfaces graphiques. Parmi eux on trouve :

Tree - Listeners

Tree provides flexible callbacks for expand, collapse and selection events.

- ▼ Node 0
 - ▼ Node 0.0
 - Node 0.0.0
 - Node 0.0.1
 - ▶ Node 0.1
- ▼ Node 1
 - ▼ Node 1.0
 - Node 1.0.0
 - Node 1.1
- Node 2

ImageCropper

ImageCropper is used to extract a certain part of an image and assign it to a new image value. ImageCropper is capable of both cropping local or external images.



LightBox

LightBox is a modal overlay component to display images, inline content and iframes.

Images

Inline

Video Content

iFrame

PrimeFaces

Source

lightbox Sopranos 4

AutoComplete

AutoComplete is used by defining a server side complete method that returns the suggestions.

Simple :

Min Length (3) :

Delay(1000) :

Max Results(5) :

Force Selection :

DropDown :

Cache :

Submit

A dropdown menu is open, showing suggestions starting with 'r'. The suggestion 'r8' is highlighted.

Menubar

Menubar brings the desktop application menubars to JSF. Using menuitems, it is very easy to execute ajax, non-ajax and navigations.

Default Menubar

File ▾ Edit ▾ ? Help ▾ Actions ▾ × Quit

Actions ▾

- ↳ Ajax
- ↳ Non-Ajax

Click Trigger

File ▾ Edit ▾ ? Help ▾ Actions ▾ × Quit

Password

Password component is an extended version of standard inputSecret component with theme integration, strength indicator and match mode.

Enter your passwords

Basic:

Feedback: Please enter a password

Feedback (Turkish):

Inline Feedback:

Match Mode

Password 1: Validation Error: Value is required. Password 1: Validation Error: Value is required.
 Password 2: Validation Error: Value is required. Password 2: Validation Error: Value is required.

Password 1: *

Password 2: *

Save

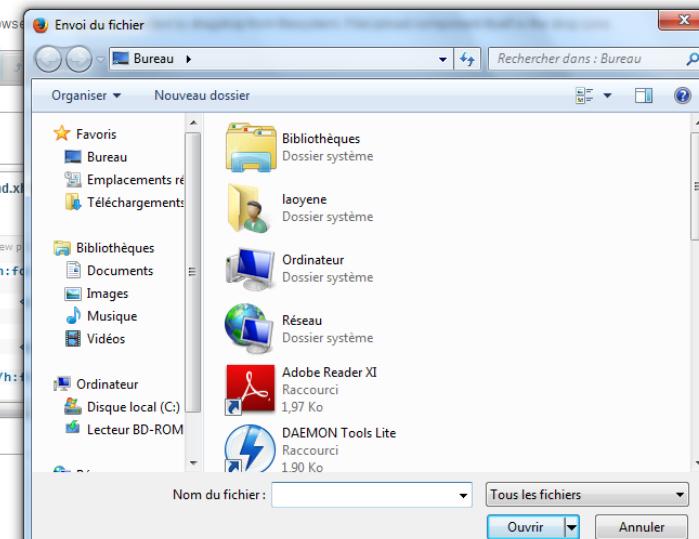
DataTable - Pagination

DataTable has built-in support for ajax pagination.

Model	Year	Manufacturer	Color
6ceccf0f	1964	BMW	Brown
c3c5fecf	1992	Audi	Orange
65450bb0	1981	Ford	Maroon
79931a8b	2007	Volvo	Green
120d6902	2007	Opel	Blue
1b1ec10c	2007	Mercedes	Silver
401b2428	1979	Volvo	Black
613a992f	1965	Renault	White
43dc6fd8	1977	Ford	Silver
9309237c	2002	Volvo	Blue

FileUpload - Drag and Drop

In supported browsers, you can drag and drop files directly onto the input field.

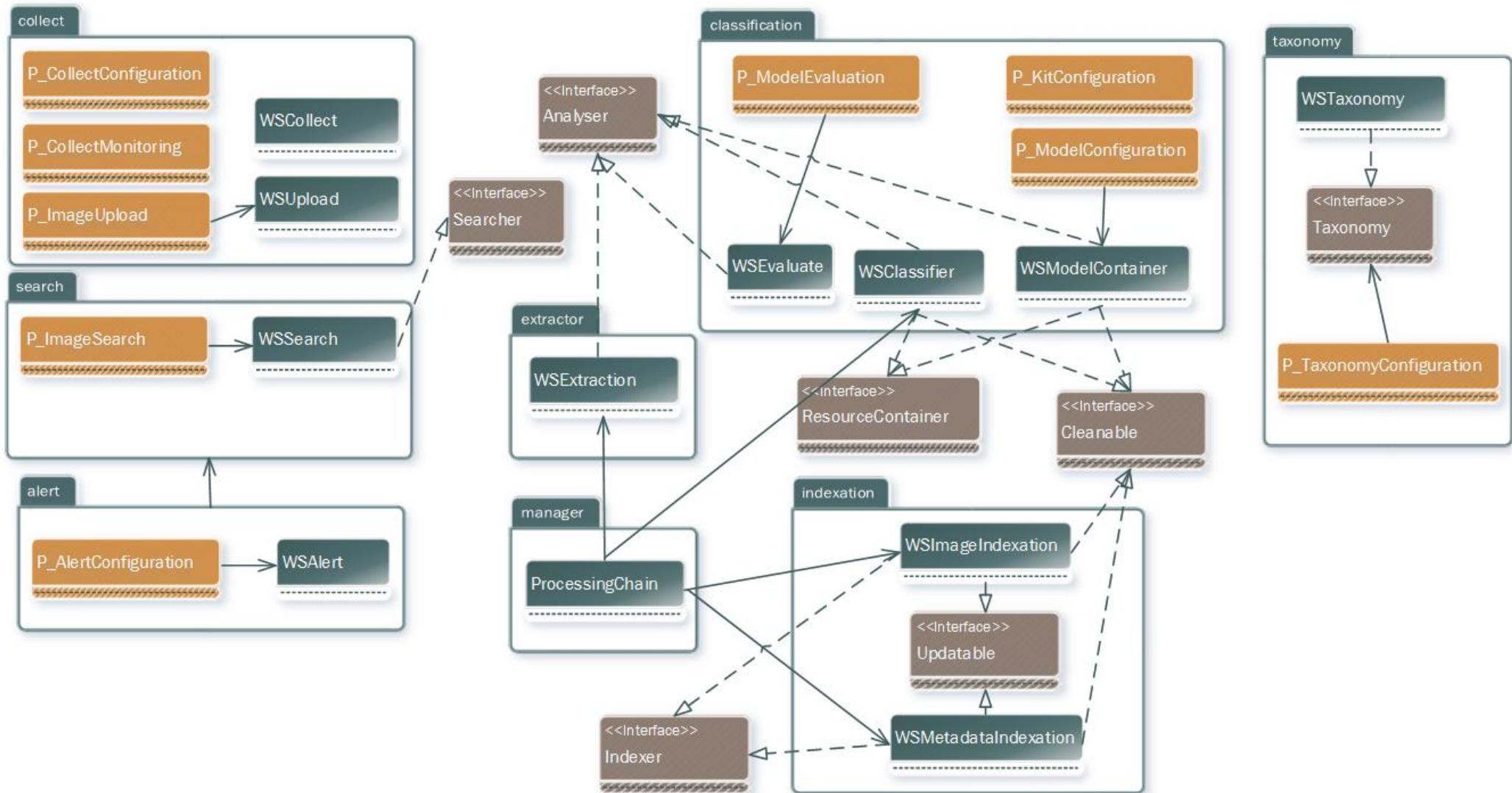


6 Conclusion

Après avoir fait le tour de certains outils de développement des interfaces web, notre choix s'est porté enfin sur l'utilisation de JSF et sa bibliothèque puissante PrimeFaces qui nous permettent de :

- Accroître la productivité des développeurs dans le développement des interfaces utilisateur tout en facilitant leur maintenance;
- Fournir aux développeurs une interface de programmation en lui permettant de manipuler l'interface Web dans un environnement JEE sans avoir à recourir souvent à du code HTML/CSS ou JS;
- Séparer entre la couche présentation et les autres couches de notre système;
- Mettre en place un mapping HTML/OBJET;
- Et fournir de nombreux composants graphiques réutilisables.

X Vue d'ensemble du démonstrateur SPORTIFS



XI Traçabilité des exigences

Exigence	Exigence de la STB	Traçabilité dans le document
E_COL_10	STB_COL_03, STB_COL_06	WSCollect WSCollect
E_COL_20	STB_COL_01, STB_USER_11	WSUpload P_ImageUpload
E_COL_30	EO_COL_02	Voir la STB
E_COL_50	STB_SUP_19, STB_SUP_20, STB_SUP_21	P_CollectConfiguration
E_COL_60	STB_COL_05, STB_SUP_22	P_CollectMonitoring
E_COL_70	STB_COL_02	WSCollect
E_COL_80	STB_COL_04	WSImageIndexation
E_EXT_10	STB_EXT_01	WSExtraction
E_EXT_20	STB_EXT_02	WSExtraction
E_EXT_40	STB_EXT_04	WSExtraction
E_EXT_50	STB_EXT_01	WSExtraction
E_EXT_60	STB_EXT_01	WSExtraction
E_EXT_80	STB_EXT_03	WSExtraction
E_EXT_90	STB_EXT_05	WSExtraction
E_IND_10	STB_IND_01, STB_IND_02, STB_IND_03, STB_IND_04	WSMetadataIndexation WSMetadataIndexation WSImageIndexation WSImageIndexation
E_IND_20	STB_SEA_01, STB_USER_07	WSSearch P_ImageSearch
E_IND_30	STB_SEA_01, STB_USER_08	WSSearch P_ImageSearch
E_IND_40	STB_SEA_03, STB_USER_10	WSSearch P_ImageSearch
E_IND_50	STB_SEA_04, STB_USER_10	WSSearch P_ImageSearch
E_IND_60	STB_USER_09	P_ImageSearch
E_IND_70/80	STB_SEA_06, STB_USER_10	WSSearch WSSearch P_ImageSearch
E_IND_90	STB_SEA_02, STB_USER_10	WSSearch P_ImageSearch
E_IND_100	STB_SEA_05, STB_IND_05, STB_USER_12	WSSearch, P_ImageSearch
E_CLA_10	STB_CLA_05	WSClassifier
E_CLA_20	STB_CLA_05	WSClassifier
E_CLA_30	STB_SUP_6	P_ModelConfiguration
E_CLA_40	STB_CLA_01	WSTaxonomy
E_CLA_50	STB_CLA_05	WSClassifier
E_CLA_60	STB_CLA_03, STB_CLA_06, STB_SUP_07,	WSClassifier WSClassifier P_ModelConfiguration

	STB_SUP_08	P_ModelConfiguration
E_CLA_70	EO_TEC_01	...
E_CLA_80	STB_SUP_09	P_ModelConfiguration
E_CLA_90	STB_CLA_04, STB_CLA_05, STB_SUP_23	WSClassifier WSClassifier P_ModelConfiguration
E_CLA_100	STB_CLA_07, STB_SUP_17	WSEvaluate P_ModelEvaluation
E_CLA_110	STB_CLA_07, STB_SUP_18	WSEvaluate P_ModelEvaluation
E_CLA_120	STB_CLA_06, STB_SUP_12	WSClassifier P_KitConfiguration
E_CLA_130	EO_CLA_06	Voir la STB
E_TAX_10	STB_TAX_01 STB_TAX_02	WSTaxonomy
E_TAX_20	STB_TAX_03, STB_TAX_04	WSTaxonomy
E_TAX_30	STB_TAX_03, STB_TAX_04	WSTaxonomy
E_TAX_40	STB_CLA_01	Voir la STB
E_TAX_50	STB_SUP_03, STB_SUP_04, STB_SUP_05	P_TaxonomyConfiguration P_TaxonomyConfiguration P_TaxonomyConfiguration
E_TAX_60	STB_TAX_01 à STB_TAX_05	WSTaxonomy
E_TAX_70	STB_TAX_02, STB_SUP_02	WSTaxonomy P_UserEditions
E_TAX_80	STB_TAX_03	WSTaxonomy
E_VIS_10	STB_USER_10, STB_USER_16	P_ImageSearch, P_RecentImages
E_VIS_20	STB_USER_03, STB_USER_04	P_ImageSearch P_ImageDetail
E_VIS_30	STB_USER_05	P_ImageDetail
E_VIS_40	STB_USER_06	P_ImageBrowse, P_ImageSearch
E_VIS_50	STB_USER_01	P_ImageBrowse
E_VIS_60	STB_SUP_01	P_UserEditions
E_VIS_70	STB_SUP_02	P_UserEditions
E_ALE_10	STB_USER_13, STB_USER_14, STB_USER_15	P_AlertConfiguration P_AlertConfiguration P_AlertConfiguration
E_ALE_20	STB_ALT_01	Maquette
E_ALE_30	STB_USER_16	IX.A.2.a
E ADM_10	STB ADM_01, STB ADM_02,	P>UserManagement P>UserManagement

	STB_ADMIN_03	P_UserManagement
E_ADMIN_20	STB_LOG_01, STB_ADMIN_04	... P_LogViewer
E_TEC_10		IV V.A
E_TEC_20		V.A
E_TEC_30		V.A V.B
E_TEC_40		V.B
E_TEC_50		V.B
E_TEC_60		Voir les conventions
E_TEC_70		IV.E
E_TEC_80		...
E_TEC_90		IX.A.2
E_TEC_100		IX.A.2
E_TEC_110		IX.A
E_TEC_120		IV.E