Ewald LE CORRE

Ifremer
Pointe du Diable
29280 Plouzané

RAPPORT D'ALTERNANCE

- **Responsable d'alternance:**

  M. Erwan BODERE

- **Enseignant encadrant :**

  M. Alain PLANTEC

Université de Bretagne Occidentale

Master 2 TIIL

Année 2015/2016

REMERCIEMENTS

Je tiens à remercier toutes les personnes de l'Ifremer pour l'accueil qui m'a été réservé depuis le début de mon alternance, et tous particulièrement Thierry CARVAL le responsable du département informatique pour m'avoir accueilli au sein de son équipe.

Mes remerciements vont également à mon tuteur Erwan BODERE, pour la qualité de ses explications, son aide, sa patience, et l'expérience enrichissante qu'il m'a apportée tout au long de mon alternance.

Pour finir, je tiens à remercier toute l'équipe pédagogique de l'Université de Bretagne Occidentale (UBO) et les intervenants professionnels responsables de la formation MASTER TIIL, pour avoir assuré la partie technique de celle-ci.

SOMMAIRE

1 INTRODUCTION

Dans le cadre de ma formation en « Master Technologie de l'information de Ingénierie du Logiciel » à l'Université de Bretagne Occidentale, j'ai effectué mon alternance au sein de la société Ifremer. Cette alternance d'un an à commencer le 7 septembre 2015.

Cette alternance à Ifremer m'a permis d'appréhender certaines technologies concernant les développements J2EE et celles liées au stockage de donnée, dans la mouvance Big Data.

La mission qui m'a été confiée consiste à la définition et au développement logiciel de tableaux de bord pour mieux suivre l'activité des ressources en lignes opérationnels des principaux systèmes d'informations de l'Ifremer.

Le périmètre de l'alternance se limite aux besoins du système d'information Sextant qui fournit des services d'accès à des données géo spatialisées. L'objectif est de développer une IHM Web permettant de restituer l'état courant et l'usage des ressources en lignes.

Durant cette alternance, j'ai été amené à travailler sur trois sujets :

1. Mise en place et étude de solutions Big Data, pour traiter les logs d'activités
2. Développement de traitements pour exploiter les logs d'activités
3. Développement d'une application Web J2EE permettant de restituer les informations.

Une partie de ce rapport permettra de présenter la société dans laquelle j'ai effectué mon alternance, à savoir Ifremer. La suite du rapport détaillera le besoin. Je présenterai ensuite mon travail réalisé et les technologies utilisées. Enfin, j'indiquerais les tâches que j'aurais à réaliser durant la période temps pleins de l'alternance.

2 PRESENTATION DE L'ENTREPRISE

Créé en 1984, l'**Institut français de recherche pour l'exploitation de la mer** (Ifremer) est un établissement public à caractère industriel et commercial (EPIC) sous la tutelle du ministère de l'Écologie, du Développement durable et de l'Énergie.

2.1 MISSIONS DE L'INSTITUT

L'Ifremer contribue, par ses travaux et expertises, à la connaissance des océans et de leurs ressources, à la surveillance du milieu marin et du littoral et au développement durable des activités maritimes. À ces fins, il conçoit et met en œuvre des outils d'observation, d'expérimentation et de surveillance, et gère des bases de données océanographiques.

L'Ifremer travaille en réseau avec la communauté scientifique française, mais aussi des organismes partenaires dans de nombreux pays. La coopération est centrée sur des grands programmes internationaux, sur l'Outre-mer et sur quelques pays-cibles (Etats-Unis, Canada, Japon, Chine, Australie, Russie), et sur une politique méditerranéenne associant l'Europe à la rive Sud de la Méditerranée.

L'Ifremer a des missions de recherche, d'expertise et d'agence de moyens.

Une recherche finalisée afin de répondre aux questions sociétales actuelles (effets du changement climatique, biodiversité marine, prévention des pollutions, qualité des produits de la mer…). Les résultats couvrent la connaissance scientifique, les innovations technologiques ou les systèmes d'observation et d'exploration de l'océan. Le partenariat est public, privé ou associe les deux.

La surveillance des mers et du littoral, en soutien à la politique publique de gestion du milieu et des ressources. A partir d'avis ou de rapports d'études, de campagnes d'évaluation, de réseaux de surveillance ou de suivi du milieu marin, l'Ifremer apporte son expertise sur des grandes questions scientifiques dans les domaines de compétences de l'Institut et en lien avec les professionnels.

Le développement, la gestion et la mise à disposition de grandes infrastructures de recherche – flotte, moyens de calcul, centre de données, moyens d'essais, structures expérimentales – à la disposition de la communauté scientifique nationale et européenne, mais aussi dans le cadre de partenariats de recherche public / privé.

Voici quelques chiffres clés concernant l'institut :

- Un budget annuel de 210 millions d'euros (hors opérations internes)
- Plus de 1 500 salariés Ifremer et 330 salariés de l'armateur Genavir
- 26 sites répartis sur l'ensemble du littoral de la France métropolitaine et en outre-mer
- 8 navires (dont 4 hauturiers), 1 submersible habité, 1 engin téléopéré pour grande profondeur (- 6 000 m) et 2 AUVs.

2.2 ORGANISATION

2.2.1 ORGANISATION GENERALE

Sous la présidence de François Jacq, président-directeur général de l'Ifremer, l'institut est composé :

- D'une Direction Scientifique et de 4 départements scientifiques (Département des Ressources Biologiques et Environnement, Département des Ressources physiques et Écosystèmes de fond de Mer, Département Océanographie et Dynamique des Écosystèmes et Département des Infrastructures Marines et Numériques)

- De 5 Centres (Centre Manche-Mer du Nord, Centre Bretagne, Centre Atlantique, Centre Méditerranée et Centre du Pacifique)
- Des directions fonctionnelles ou de support à la recherche (Direction de l'Information Scientifique, de la Communication, de la Médiation et des Relations Institutionnelles ; Direction des Affaires Européennes et Internationales ; Délégation Générale à l'Outre-Mer ; Direction des Ressources Humaines ; Direction des Affaires Juridiques ; Direction des Affaires Financières et du Contrôle de Gestion ; Direction de la Valorisation ; Direction des Moyens et Opérations Navals).

2.2.2 ORGANISATION DU DEPARTEMENT IMN

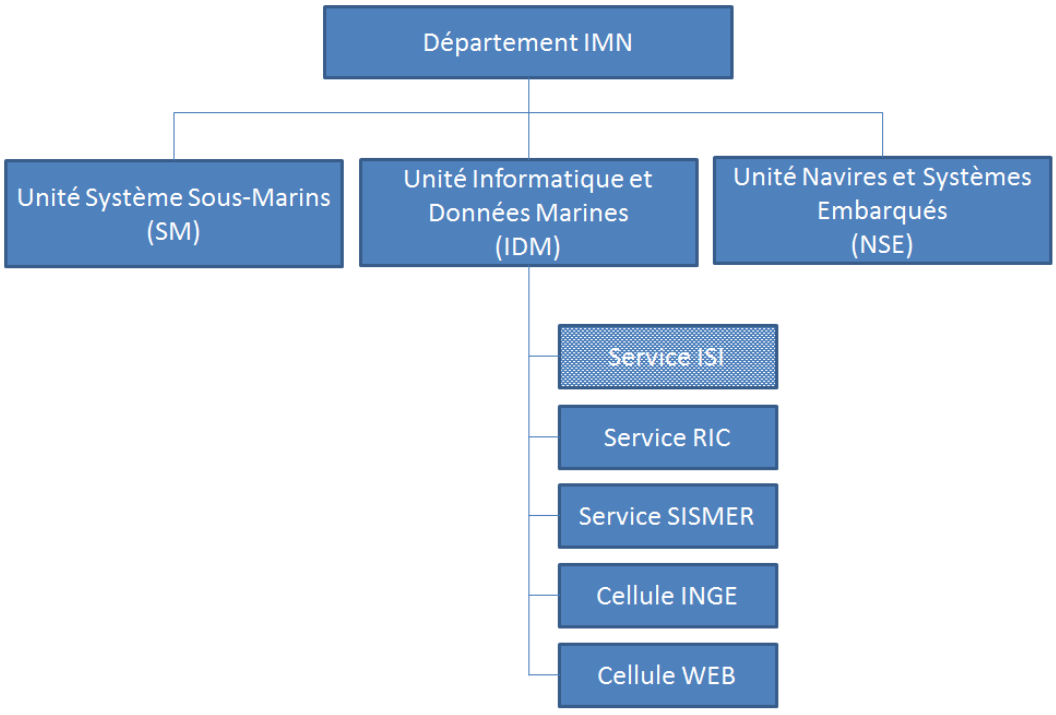
Le département **Infrastructures Marines et Numériques** (IMN) assure des services et réalise des projets dans le but de créer ou d'améliorer en continu des infrastructures de recherche. Il s'attache à promouvoir les infrastructures de recherche dont il a la charge auprès des guichets de financement nationaux et internationaux pertinents.

Les équipes développent et opèrent les grands systèmes d'informations de données marines de l'Ifremer, en collectant, validant, traitant, bancarisant et diffusant les données dans le respect des directives nationales et européennes en vigueur (DCSMM, DCE, INSPIRE, DCF).

Les unités de recherche rattachées au département IMN sont :

- Systèmes Sous-Marins (La Seyne)
- Navires et Systèmes Embarqués (Brest)
- Informatique et Données Marines (Brest et Nantes).

2.2.3 ORGANISATION DE L'UNITE IDM



1. Organigramme d'Ifremer

L'unité de recherche **Informatique et données marines** (IDM) est composée de trois services (Ressources Informatiques et Communications, Ingénierie des Systèmes d'Information, Systèmes d'Informations Scientifiques pour la Mer), et 2 cellules (Webmestre et Informatique de Gestion).

Elle est chargée de développer et de gérer les infrastructures informatiques communes de l'Ifremer et le centre de données océanographiques dans un objectif continu, adapté aux usages et aux capacités de l'Institut.

L'unité participe également au développement des systèmes d'information scientifiques nécessaires aux programmes de l'Ifremer (y compris ceux relevant de partenariats scientifiques), et assure les services de données scientifiques en continu.

Elle développe sous la maîtrise d'ouvrage des services fonctionnels concernés, les systèmes informatiques de gestion et les maintient en service opérationnel.

Ses domaines de compétence sont les systèmes informatiques et les réseaux de télécommunications, l'informatique de gestion, la gestion de données marines, l'ingénierie des systèmes d'informations scientifiques, les services opérationnels, la publication web.

2.2.4 SERVICE ISI

J'ai effectué mon alternance au sein du service **Ingénierie des Systèmes d'Information** (ISI).

Ce service conçoit et développe des systèmes informatiques pour les données scientifiques et techniques des programmes de l'Ifremer. Il assure la conception, le développement et l'évolution des bases de données nécessaires au programme « Centres de données », aux programmes d'Océanographie Opérationnelle et aux autres programmes d'Ifremer en visant à la réalisation de systèmes harmonisés et complémentaires. Il forme le personnel lors de leur mise en exploitation et l'assiste, quand nécessaire, dans leur utilisation. Il contribue à la mise en place des systèmes informatiques de collecte des données et à la réalisation des traitements et produits associés aux bases de données.

Pour ce faire, il maintient et fait progresser ses compétences dans les techniques informatiques et, en particulier, en génie logiciel, en ingénierie des bases de données et Web et en systèmes d'informations géographiques. Il maintient et développe les ateliers de génie logiciel nécessaires avec les méthodes associées.

Ce service s'appuie sur les compétences en systèmes et réseaux du service RIC et celles de SISMER en gestion de données. Les solutions d'architecture et d'exploitation qu'il propose sont définies en collaboration avec ces services, dans le cadre des projets et programmes d'Ifremer.

3 COMPREHENSION DU BESOIN

3.1 CONTEXTE IDM

3.1.1 LES PRINCIPAUX SERVICES GERES À IDM

A travers ses activités scientifiques et ses activités d'observations, l'Ifremer récolte et valorise un grand nombre de donnée très variées (mesures physiques, chimiques, biologiques, observations, cartographie…). Ces données sont archivées dans des bases de données très diverses.

Pour organiser ces différentes ressources, plusieurs systèmes d'information interopérables ont été mis en œuvre.

2. Flux des données marines

3.1.1.1 CORIOLIS

Coriolis est un centre de données d'océanographie opérationnelle, avec en complément des mesures de la surface des océans effectuées au moyen d'appareils embarqués sur des satellites.

Les données recueillies permettent de construire une image instantanée de la structure des masses d'eau et de l'intensité des courants.

3.1.1.2 SISMER

Le Sismer (Systèmes d'Informations Scientifiques pour la MER) a pour activités la compilation, la sauvegarde et la diffusion de données conventionnelles collectées lors de programmes nationaux et internationaux. La gestion des données inclut l'information sur les formats et les méthodes (métadonnées), et les contrôles de qualité pour assurer la cohérence entre les jeux de données provenant de sources diverses

3.1.1.3 SIH

Le SIH (Système d'Informations Halieutiques) constitue le réseau pérenne et opérationnel d'observation des ressources halieutiques et des usages associés (pêches professionnelle et progressivement pêche récréative).

Il est ainsi responsable et dépositaire des cahiers des charges et des spécifications techniques pour les plans d'échantillonnage, la collecte, l'archivage, la mise à disposition et l'accès aux données halieutiques.

3.1.1.4 QUADRIGE

Quadrige est un système d'information qui gère les données de la surveillance du littoral, l'Ifremer a développé le système d'information Quadrige, qui associe à une base de données une panoplie d'outils d'interprétation et d'élaboration de produits d'information. Quadrige constitue un élément du Système d'Information sur l'Eau, et à ce titre, contribue aux travaux du Secrétariat d'Administration National des Données Relatives à l'Eau (Sandre).

La banque thématique Quadrige a eu pour mission première la gestion et la valorisation des données issues des réseaux de surveillance mis en œuvre par l'Ifremer.

3.1.1.5 SEXTANT

Sextant a pour vocation de collecter et mettre à disposition un catalogue de données géo référencées sur le domaine marin. Il vient en soutien de problématiques telles que la biodiversité, les énergies renouvelables en mer, la gestion intégrées des zones côtières, la pêche, l'environnement littoral et profond, l'exploration et l'exploitation des fonds marins...

3.1.1.6 ARCHIMER

Archimer est une Archive Institutionnelle de l'Ifremer, contient désormais plus de 15 000 documents en texte intégral dont plus de 9500 accessibles librement sur Internet.

3.1.2 LES BESOINS IDM

L'unité IDM fournit des services opérationnels utilisés par différents utilisateurs des scientifiques, des partenaires mais également le grand public.

L'unité IDM souhaite mieux connaitre ses utilisateurs et évaluer la performance de ses services, afin :

- D'améliorer le service rendu
- D'effectuer du reporting mensuel/annuel
- De répondre aux engagements de service (SLA)

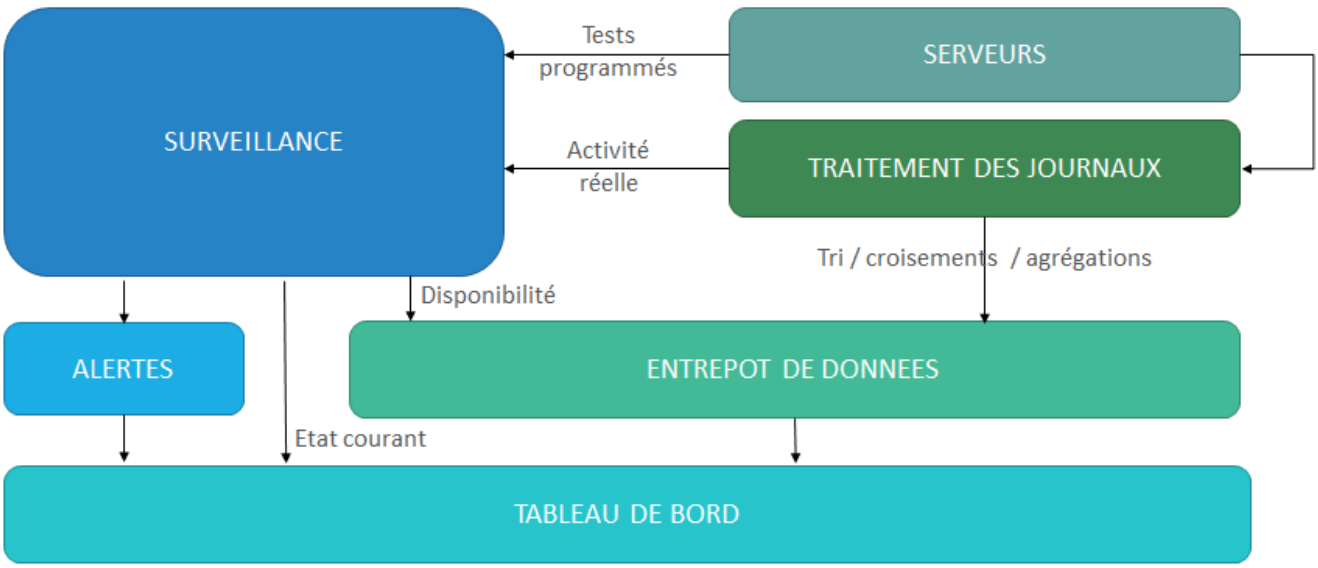
Pour cela, il faut utiliser différentes sources d'informations :

- Monitoring : permet de surveiller les services opérationnels et de connaitre l'état actuel et déterminer le taux de disponibilité
- Logs d'activité
    - Logs serveurs (http, ftp, …) : permet de connaître l'activité réelle des utilisateurs
    - Logs applicatifs : permet de récupérer des informations plus précise sur les différentes briques logicielles des services opérationnels

L'objectif IDM, à travers cette alternance est de mettre en place une infrastructure commune à l'ensemble des systèmes d'informations.

3.1.2.1 SCHEMA INFRASTRUCTURE

Le schéma d'infrastructure ci-dessous présente l'architecture ciblée. Durant mon alternance, je dois étudier, installer, configurer et utiliser différentes briques logicielles pour répondre aux différents besoins.



3. Schéma d'infrastructure

Les services opérationnels tournent sur une infrastructure de production.

Cette infrastructure et les services opérationnels sont monitorés (actuellement via l'outil Icinga). Périodiques, des tests sont effectués. Cela permet :

- De connaître l'état courant (en panne, en fonctionnement avec ou sans avertissement)
- De déduire le taux de disponibilité
- De générer des alertes.

D'autre part, les différents éléments d'architecture et les services opérationnels produisent des logs qui pourront être traités et croisés avec d'autres informations (ex : référentiels des systèmes d'informations), afin de produire des nouvelles informations enrichies que l'on pourra requêter et visualiser, en utilisant potentiellement des outils d'indexation de messages. Ces nouvelles informations nous permettront :

- De produire des indicateurs qui seront stockés dans un entrepôt de métriques (ex : taux de disponibilité mensuel, nombre de visites dans l'année)
- D'alimenter l'outil de monitoring d'erreurs remontées depuis l'activité réelle
- De consulter, via un tableau de bord, l'ensemble des informations de manière simple et le plus proche possible des besoins des administrateurs des systèmes d'information.

3.2 BESOINS SPECIFIQUES SEXTANT

Afin de valider cette infrastructure et lever les risques techniques liés à l'utilisation des technologies Big Data, il a été décidé de se limiter aux attentes du projet Sextant.

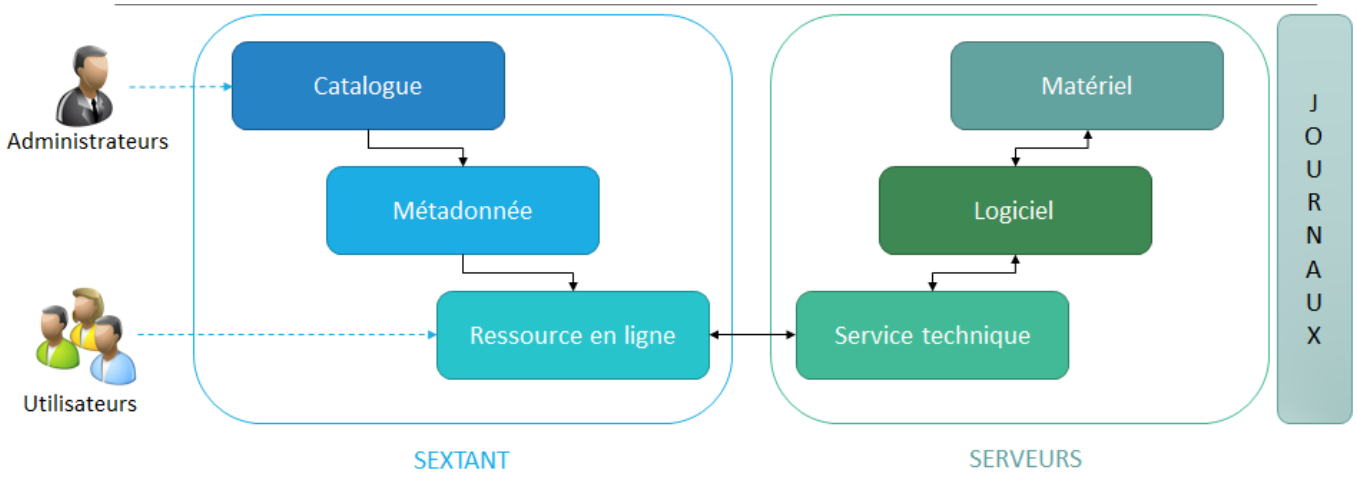
3.2.1 PRESENTATION DE SEXTANT

Sextant est une plateforme logicielle permettant d'accéder à des données géo-spatialisées (Lien : http://sextant.ifremer.fr/fr/accueil). Sextant est composé de trois briques logicielles principales :

1. Catalogue de données (*geonetwork*)
    - Edition de métadonnées
        - Les métadonnées sont des fiches décrivant des données géo-spatialisées (auteurs, description, emprise géographique, mots clés, liens vers les services d'accès aux données, …)
        - Les métadonnées doivent respecter certaines normes ou directives (ex : ISO19115, INSPIRE, ….)
    - Gestion des droits d'accès aux métadonnées
    - Recherche des métadonnées avec le support du protocole OGC « CSW »
2. Visualisation des données (*mapserver, qgis-server, thredds/ncwms,* …)
    - Visualisation compatible avec les standards OGC « WMS » et « WMTS »
3. Téléchargement des données
    - Via les standards OGC « WFS » ou « WCS » (via *mapserver, qgis-server, thredds/ncwms,* …)
    - Via un traitement java « maison » pour accéder aux données présentent dans des SGDB (Oracle, PostgreSQL) ou fichier (shapefile)

3.2.1.1 VUE MACROSCOPIQUE DU FONCTIONNEMENT DE SEXTANT

Ci-dessus une vue macroscopique du modèle de données Sextant.



4. Diagramme de classes

Un **catalogue** Sextant (ex : photos anciennes du littoral) correspondant à un regroupement thématique de métadonnées. Chaque catalogue est administré par une ou plusieurs personnes qui sont des administrateurs.

Une **métadonnée** est une fiche décrivant un produit géo-spatialisé. Cette fiche contient de nombreuses informations permettant de décrire le produit (description, auteurs, mots, …) et les **ressources en lignes** d'accès aux différentes couches en visualisation, téléchargement, … . Certaines métadonnées sont à accès restreint.

Les ressources en lignes peuvent être de simples fichiers téléchargeables (via http, FTP) ou des services web, notamment des services OGC (WMS, WFS, …). Ces services web s'appuient sur des **services techniques** pour fonctionner. Un serveur technique est généralement une configuration d'un logiciel de SIG (ex : Mapserver) tournant sur un serveur. Chaque accès aux ressources en ligne est tracé dans des journaux (ex : apache, ftp, …).

3.2.1.2 METADONNEE SEXTANT

Voici un exemple d'une métadonnée Sextant :

Identifiant du système de référence

Code	27572

Identification des données

Résumé	Il s'agit de clichés noir et blanc du littoral métropolitain. Les prises de vue s'échelonnent entre les mois de juillet et octobre 1919. Du fait d'un archivage correct dans une salle climatisée, leur état de conservation est resté remarquable. Une partie d'entre elles sont rangées en album avec un numéro et un plan d'assemblage. D'autres sont stockées dans des boîtes de carton plus ou moins organisées. Le reste est en vrac. Elles existent en général sous forme de clichés. Mais on trouve dans cette collection également quelques photos sur plaques de verre qui sont en fait les originaux et quelques films négatifs.
But	L'objectif de ces traitements est : - De garantir la précision spatiale, - De respecter la radiométrie, - D'élaborer de la documentation (méta données), - De leur donner le statut de Données de référence.
Reconnaissance	© SHOM, Ifremer, Photothèque nationale; Traitements financés par Brest Métropole Océane
gmd:MD_ProgressCo...	Mis à jour continue

Mots clés

Non classé

anciens
photographie
avion
guerre
photo
scan
Bretagne
Finistère

external.theme.inspire-theme

Ortho-imagerie

gmd:MD_SpatialRepr...	Raster
Code ISO de la langue	Français
gmd:MD_CharacterS...	Utf8
Informations supplé...	Elles sont particulièrement intéressantes car elles ont été prises à marée basse pour deux raisons principales (renseignements fournis par le SHOM) : - dans un but de défense du territoire, - et surtout pour remettre à jour les cartes marines du SHOM.

Informations de référence

Titre	Photos anciennes - Bretagne - Finistère 1919 - Rade de Brest
Publication	22-12-2011 13:12:00
Identificateur	SXT_PHOTOANCIENNE_BZH29_LOCMARIA_KERHUON19
Forme de la présenta...	Carte numérique

5. Métadonnée Sextant – rade de Brest en 1919

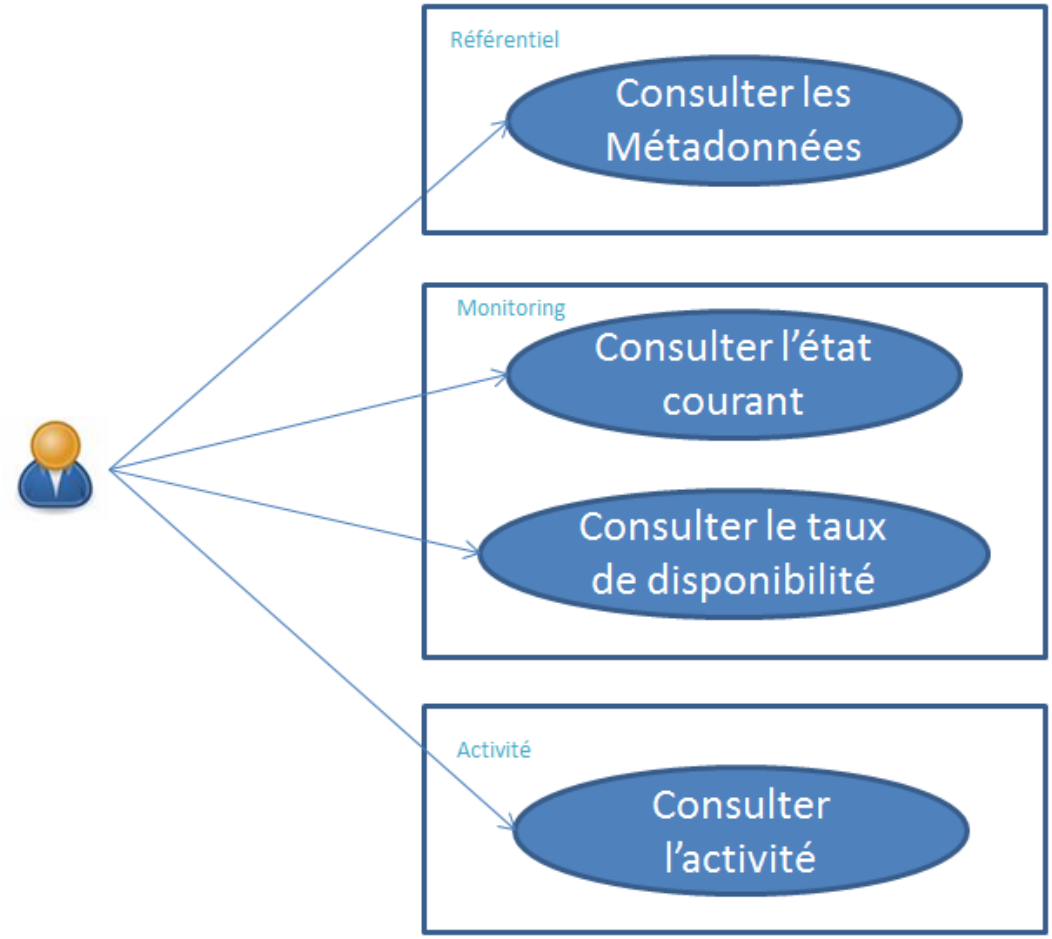
6. Visualisation de Rade de Brest 1919

3.2.2 PRESENTATION DES BESOINS SEXTANT

L'équipe Sextant a exprimé différents besoins, qui varient selon les interlocuteurs et leurs activités.

3.2.2.1 CAS D'UTILISATION

Le diagramme de cas d'utilisation ci-dessous représente les quatre grands besoins :



7. Cas d'utilisation d'un administrateur

3.2.2.2 REFERENTIEL DES DONNEES

L'utilisateur doit pouvoir consulter les métadonnées d'un catalogue. Un métadonnée contient un titre, et plusieurs indicateurs. Les premiers indicateurs montrent si la métadonnée est conforme à des tests auxquels elle a été soumise (Valid, Geon, Inspire, ISO, OpenData). Les deux autres indicateurs permettent de déterminer si la métadonnée contient des thèmes (Inspire, Sextant).

3.2.2.3 MONITORING DES RESSOURCES EN LIGNES

Un monitoring des ressources en lignes Sextant est déjà en place via l'outil Icinga. Il permet de surveiller :

- La conformité des métadonnées par rapport aux différentes normes/directives
- Le bon fonctionnement de l'accès aux données en visualisation et téléchargements

Icinga ne permet pas de créer des vues spécifiques. Ainsi l'équipe Sextant souhaite pouvoir consulter l'état courant (Ok, KO) et le taux de disponibilité (quotidien, mensuel et annuel) de chaque ressource en ligne, métadonnée, catalogue et service technique.

3.2.2.4 SUIVI DE L'ACTIVITE

A partir de différents logs, il faudrait pouvoir récupérer :

- Un maximum d'informations sur les utilisateurs (localisation, user agent, …)
- Leurs actions pour avoir des statistiques d'usages :
- Le nombre d'accès à un produit
    - Principaux type de ressources en lignes (WMS, WMTS, …)
    - Cibler une zone géographique

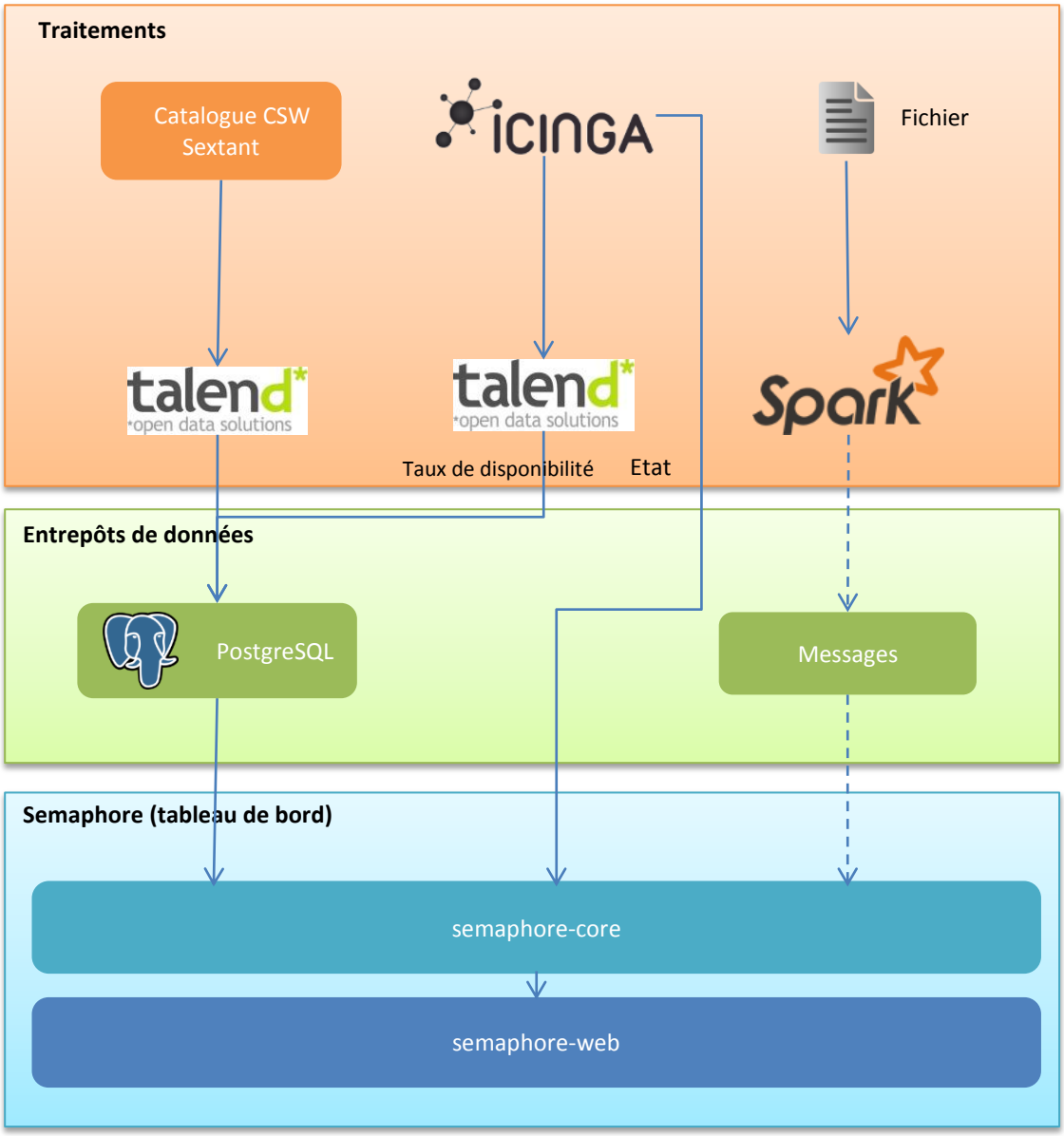
Il faudrait pouvoir produire les indicateurs standards suivants :

- Nombre de requêtes
- Nombre d'adresses IP distinctes
    - Si possible différencier les utilisateurs récurrents des nouveaux utilisateurs
- Nombre de sessions utilisateurs. Une session est une période de temps durant laquelle l'utilisateur reste actif (i.e. sans inactivité de plus de x minutes, avec x à définir),c'est-à-dire qu'il est toujours en train de consulter des ressources en lignes.

4 TRAVAIL REALISE

Dans cette partie nous verrons en détail le travail réalisé durant mon d'alternance.

4.1 SCHEMA GLOBAL



8. Schéma global

Mon travail peut être décomposé en trois activités :

- Etudie, installation et configuration de briques logicielles
- Développement des traitements permettant de traiter et charger des données
- Développement d'une application web J2EE permettant de consulter les données

Le travail d'analyse des logs d'activité n'ayant pas pu être terminé, nous avons identifié cette tâche en pointillée sur le schéma ci-dessus.

4.2 METHODOLOGIE

Pour travailler et communiquer avec mon maitre d'alternance. Nous avons utilisé Mantis pour le suivi des actions.

4.2.1 MANTIS

Mantis est un système de suivi d'anomalies logicielles (bugs) basé sur une interface web. Il est écrit en PHP. Pour chaque tâche à réaliser, mon tuteur d'alternance ouvre une fiche Mantis. Je mets à jour la fiche en fonction de mes avancements. Ce qui permet de tracer ce qui est réalisé.

Référence : https://www.mantisbt.org/

4.3 MISE EN PLACE DE L'INFRASTRUCTURE

4.3.1 MONITORING – ICINGA

Un monitoring des ressources en lignes contenant toutes les informations nécessaires est déjà mis en place via Icinga.

4.3.1.1 ICINGA

Icinga est un système informatique de surveillance réseau open source. Il a été créé à l'origine comme un dérivé de Nagios. Icinga propose plus de fonctionnalités, des connecteurs de base de données supplémentaires (pour MySQL, Oracle, et PostgreSQL), et une API REST qui permet aux administrateurs d'intégrer de nombreuses extensions sans modification du noyau.

Référence : https://www.icinga.org/

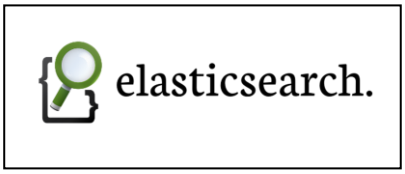
4.3.2 MESSAGES – ELASTICSEARCH

J'ai étudié, installer et configurer la pile ELK (Elasticsearch, Logstash, Kibana) en début d'alternance. L'idée initiale était de stocker sous Elasticsearch l'ensemble des données : référentiel Sextant, logs Icinga, logs d'activité.

Au fur et à mesure de nos tests, on a identifié qu'Elasticsearch n'était pas le bon choix pour stocker le référentiel Sextant, ainsi que les logs Icinga. Les développements réalisés sur cette technologie ne seront donc pas présentés dans ce rapport.

Elasticsearch servira certainement à stocker les logs d'activité. Aussi, nous vous présentons ci-dessous le fonctionnement de la pile ELK.

4.3.2.1 ELASTICSEARCH



Elasticsearch est un serveur utilisant Lucene pour l'indexation et la recherche des données. Il fournit un moteur de recherche distribué et multi-entité à travers une interface REST. C'est un logiciel libre écrit en Java et publié en open source sous licence Apache.

Elasticsearch est le serveur de recherche qui utilise le moteur Apache Lucene. Le cluster Elasticsearch mis en place est composé de quatre nœuds.

Un « cluster » (en français « **grappe** ») est une architecture composée de plusieurs ordinateurs formant des nœuds, où chacun des nœuds est capable de fonctionner indépendamment des autres.

Référence : https://www.elastic.co/products/elasticsearch

4.3.2.1.1 STOCKAGE ET INDEXATION DES DOCUMENTS

Elasticsearch traite uniquement des documents JSON. Il faut donc convertir les autres formats en amont via Logstash par exemple.

Les documents sont stockés dans une arborescence index/type/id :

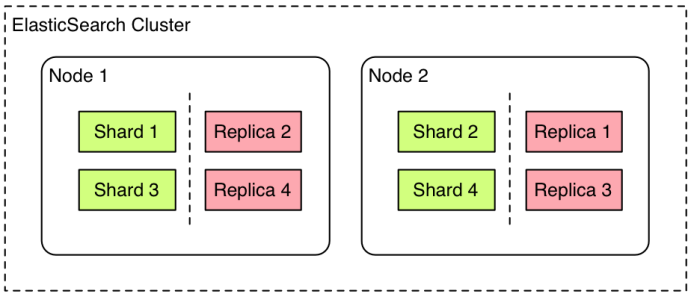
- Index : on peut comparer un index à une base de données.
- Type : on peut comparer un type de document à une table
- Id : identifiant du document (clé naturelle ou auto-générée)

La recherche est rapide car les documents sont gérés à plat (sans liens entre eux, pas de verrous) contrairement à une base de données. Cela implique d'avoir un maximum d'informations dans les documents pour pouvoir des recherches avancées.

La recherche peut se fait sur un/plusieurs index et index_type. Ces possibilités permettent plusieurs stratégies de stockage :

- Un seul gros index
- Un index par unité de temps (jour, mois ou année)
- Un index par système d'information, ou par utilisateur

La puissance de recherche peut être encore décuplée par l'organisation physique des données au sein du cluster Elasticsearch.



9 Cluster Elasticsearch

- Un cluster peut contenir plusieurs nœuds ElasticSearch
- Un nœud ElasticSearch stocke les données et participe à l'indexation et recherche
- Un Shard est un moteur Lucène qui permet de subdiviser des index en plusieurs blocs
    - Cela permet de contourner des limitations physiques et accroitre les performances d'indexation et de recherche
- Le Réplica est une copie d'un shard pour contourner les problèmes d'interruption d'un ou plusieurs nœuds

Tous les champs d'un document sont indexés par défaut. A chaque type, elasticsearch associe un mapping décrivant les propriétés des documents à indexer, les types de données et le choix de l'analyseur (pour du texte on peut analyser en minuscule, majuscule, …).

4.3.2.2 LOGSTASH

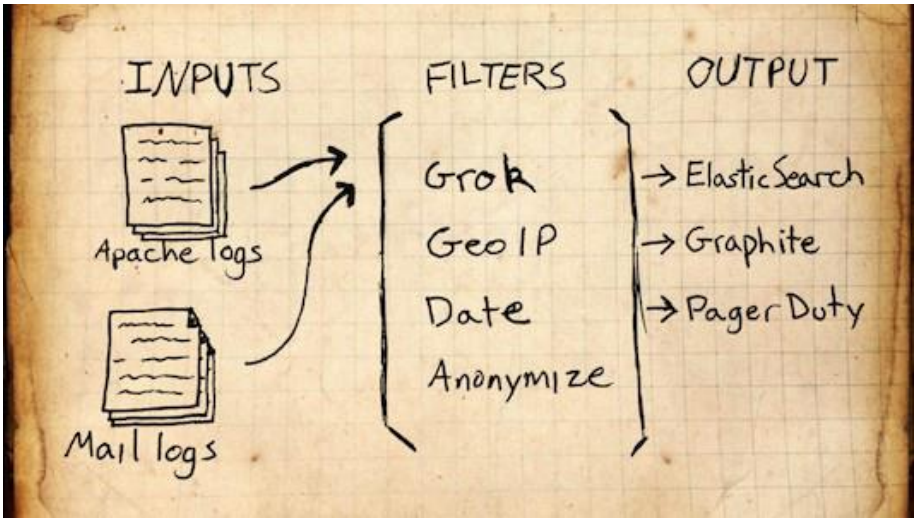
Logstash est le premier agent responsable du cheminement des données. Il fait partie des technologies dites d'**Extract-Transforme Load** (ETL).

ETL est la technologie assurant l'opération de « transvasement » d'une source de données vers une autre. Composée de connecteurs pour assurer les échanges, elle assure également un rôle de transformation des données : conversion, ajout/suppression de données, modifications, calculs…

Le processus opéré par Logstash est découpé en trois parties :

- Entrée de la donnée
- Filtres
- Sortie de la donnée

Chacune de ces parties est composée de plugins de base. Ceux-ci permettent d'effectuer des techniques d'ETL d'une grande diversité. Cependant, il faut savoir qu'il est aussi possible de développer ses propres plugins afin que ceux-ci correspondent mieux à nos besoins.



10. Schéma simplifié du fonctionnement de Logstash

Référence : https://www.elastic.co/products/logstash

4.3.2.2.1 FONCTIONNEMENT GLOBAL

Logstash peut être exécuté en ligne de commande de deux façons :

- Les options sont spécifiées dans la commande
- Le process d'ETL suit une logique déterminée par un script

Le script permet d'effectuer des opérations plus complexes, de multiplier les sources de données et leurs destinations. L'apport majeur est également la possibilité d'effectuer des branchements conditionnels, apportant une multiplicité des possibilités de traitements selon les sources.

Ce script s'appuie sur des plugins. Chaque plugin est une classe Ruby. Il peut comporter de nombreuses méthodes publiques, utilisables avec passage d'arguments. Il peut aussi ne comporter aucune méthode publique et exécuter une méthode privée choisie par le programmeur, sans passage d'argument.

Il est possible d'utiliser autant de plugins qu'on le souhaite. Il est possible d'utiliser des branchements conditionnels (if … else), d'ajouter des tags/champs arbitrairement, afin d'opérer d'une façon plus spécifique.

4.3.2.2.1.1 ENTREE D'UNE DONNEE

Logstash peut recevoir une donnée depuis de nombreuses sources, en voici quelques-unes :

- **Stdin** -> Saisie au clavier
- **File** -> fichier présent sur le disque
- **Elasticsearch** -> lire un message depuis une requête vers Elasticsearch
- **Sqlite** -> lire des données depuis une base de données SQLite
- **Exec** -> exécuter une commande et obtenir sa sortie en tant que donnée entrante
- …

Une donnée entrante est appelée **message**. Ce message consiste en général en ligne séparée des autres par des caractères de fin de ligne.

Chaque message est envoyé vers la seconde partie : le filtre.

4.3.2.2.1.2 TRAITEMENT D'UNE DONNEE

Le message entrant est brut. Selon sa source et ce que l'on souhaite en faire, il existe des filtres qui effectuent une analyse sur des données normalisées (logs systèmes UNIX). Il existe aussi des filtres qui permettent de personnaliser les traitements.

Voici une liste (non-exhaustive) des filtres disponibles et leur utilité :

- **Grok** -> parsing de texte et structuration des données sur la base de **patterns/regular expressions.** Il est possible de créer soi-même ses propres patterns. Il procède par la jointure de type/syntaxes et de données. Chaque association créée un couple clé/valeur
- **Mutate** -> permet de modifier n'importe quelle donnée : conversion de types, séparation d'un texte selon un motif, modification massive de caractères suivant un motif, jointures de champs…
- **Drop** -> suppression d'un message
- **Geoip** -> ajout d'informations concernant l'emplacement géographiques d'adresse IP
- **Ruby** -> exécution d'un code ruby ajouté en argument

4.3.2.2.1.3 SORTIE DE DONNEES

Après que la partie filtrée a été traversée par le message, celui-ci, structuré, peut être livré par Logstash vers le nouveau conteneur. Plusieurs plugins sont possibles au sein d'un même script mais une donnée ne peut en employer qu'un à la fois. En voici quelques-uns :

- **Stdout** -> sortie standar système : écran, notamment …
- **Elasticsearch** -> stockage en base de données Elasticsearch
- **Graphite** -> envoi des données vers graphite pour du monitoring en temps réel sans stockage de données
- **Exec** -> exécution d'une commande
- …

Référence : https://www.elastic.co/products/logstash

4.3.2.3 KIBANA

Troisième couche de la pile ELK, celle-ci constitue l'outil privilégié pour explorer les données stockées en base. Connectée aux instances ElasticSearch, elle nous permettra d'explorer nos index et de leur donner un aspect graphique grâce à l'utilisation du langage de requêtes issu du projet Apache Lucene.

Kibana peut recevoir plusieurs configurations, chacune se rapportant à un index ou un groupe d'index identifiés par un pattern.

Il est donc possible de créer des vues graphiques mélangeant les données de plusieurs index.

Accessible depuis une interface web, Kibana offre trois modes de fonctionnement :

- **Discover** : permet de parcourir chaque document d'un index et d'en afficher les champs. Possibilité de filtrer selon des requêtes et également de sauvegarder ces requêtes pour les intégrer à des visualisations.
- **Visualize** : Permet de créer des visualisations graphiques de données sur la base de requêtes. Plusieurs types de graphiques (histogrammes, pie charts, line charts, tables …) sont disponibles.
- **Dashboard** : Voué à être une page utilisable au quotidien, il s'agit d'un tableau de bord contenant des visualisations graphiques enregistrées au préalable. Il est possible d'enregistrer plusieurs dashboards afin de les regrouper par thème.

Référence : https://www.elastic.co/products/kibana

4.3.2.4 VERSIONS

Les trois composants étaient déjà installés sur un seul serveur avec les spécifications suivantes :

- Java : version 7
- Elastisearch : version 1.5.2
- Logstash : version 1.5.0
- Kibana : version 4.0.1

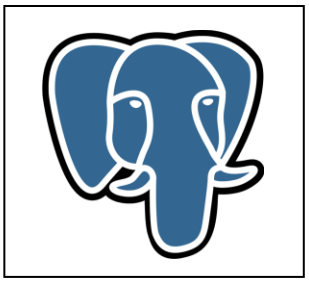
J'ai donc mis à jour ces logiciels, voici la spécification actuelle :

- Java : version 8
- Elastisearch : version 2.1.1
- Logstash : version 2.2.2
- Kibana : version 4.3.1

4.3.3 STOCKAGE DES DONNEES

Nous stockons finalement le référentiel Sextant dans une base de données. Ce choix est plus appropriée que l'utilisation d'Elasticsearch qui lui sert principalement à faire de la recherche « full text ».

4.3.3.1 POSTGRESQL



PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD.

Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB, MySQL et Firebird), ou propriétaires (comme Oracle, Sybase, DB2, Informix et Microsoft SQL Server).

Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.

Référence : http://www.postgresqlfr.org/

4.4 DEVELOPPEMENT DES TRAITEMENTS

J'ai développé deux traitements batch avec l'ETL Talend :

- Récupération du référentiel Sextant
- Calcul des taux de diposnibilité

4.4.1 ETL – TALEND

Talend est un éditeur de logiciel Open Source spécialisé dans l'intégration et la gestion des données, entre autres Big Data. C'est un outil dit ETL (Extraction, Transformation et Chargement), la particularité de ces outils c'est que le développement se fait de manière graphique. L'intérêt d'un tel outil permet d'éviter un grand nombre de script.

Référence : https://fr.talend.com/

4.4.2 RECUPERATION DU REFERENTIEL SEXTANT

L'objectif est de récupérer les informations concernant les catalogues, les métadonnées, les ressources en ligne ainsi que les services techniques. Il faut que le traitement tourne tous les jours. Il doit donc mettre à jour et ne pas faire un annule et remplace.

Ces informations sont nécessaires pour le calcul des indicateurs, ainsi qu'à la présentation des informations dans l'application web « semaphore ».

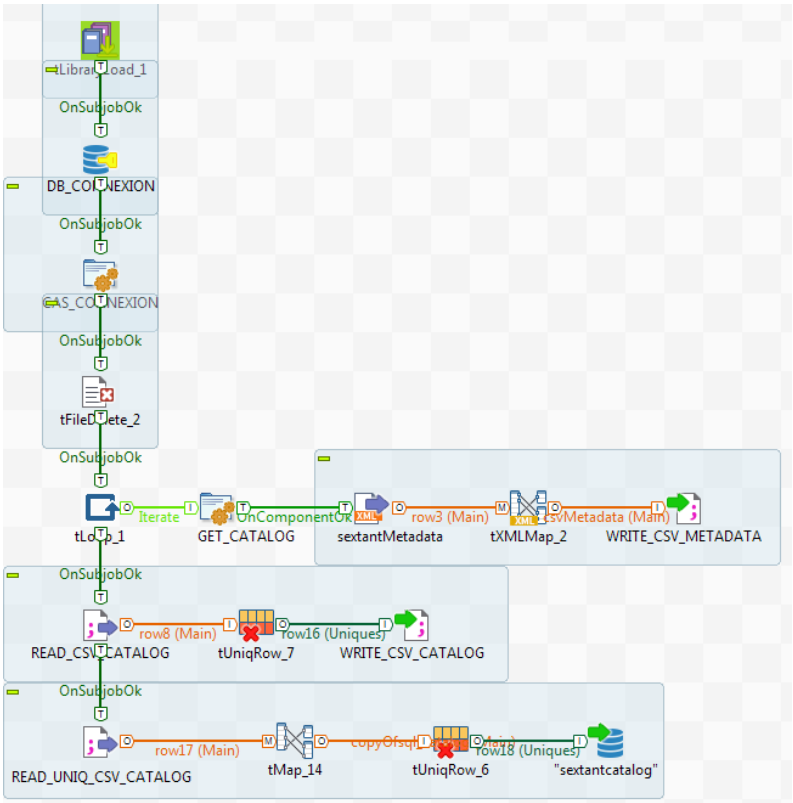
4.4.2.1 CONCEPTION

Voici les différentes actions du traitement pour récupérer le référentiel :

- Interrogation du catalogue CSW (Catalog Service for the Web) protégé via le système de SSO CAS
  - Développement d'un client http qui se charge d'effectuer la connexion à CAS
  - Interrogation du service CSW, pour récupérer les métadonnées vingt par vingt au format XML

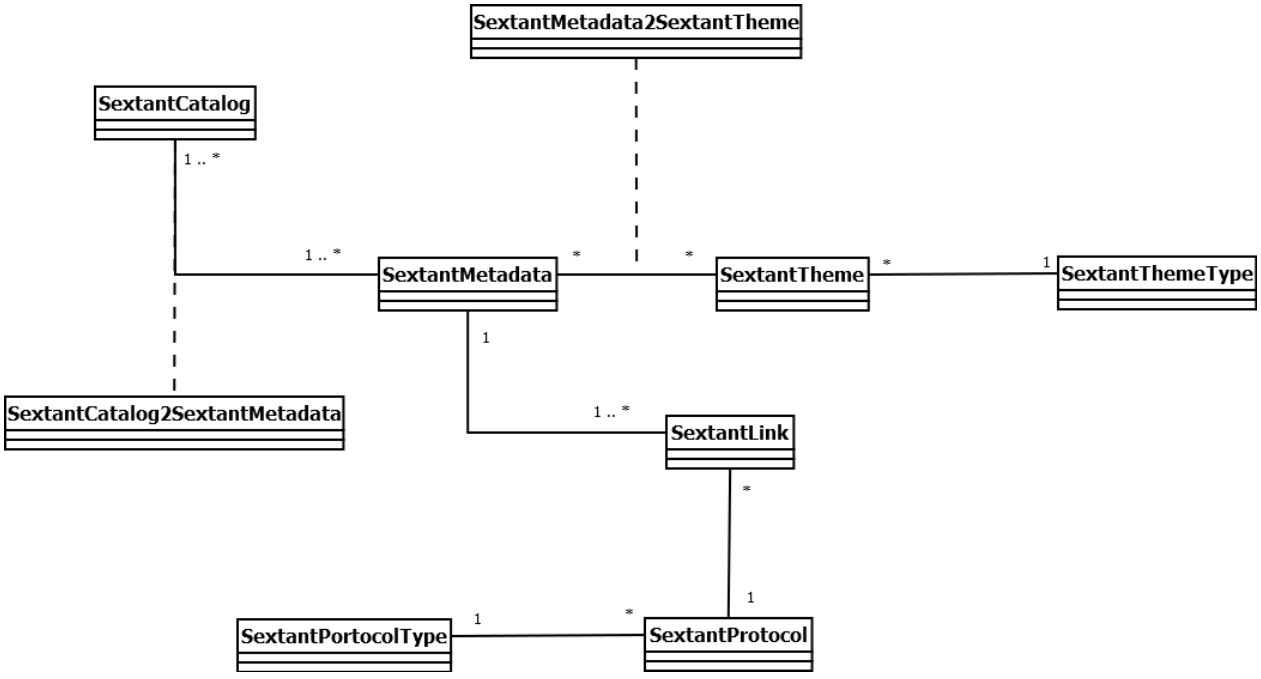
- Pour chaque métadonnée, on extrait uniquement les informations dont on a besoin, à l'aide de requêtes XPATH.
- Des traitements sont effectués sur ces données (ex : pour calculer certains indicateurs)
- Insertion des données dans la base PostgreSQL

Voici un extrait du job Talend :



11. Job Talend référentiel

4.4.2.2 MODELISATION DE LA BASE DE DONNEES



12. Diagramme de classes

4.4.2.3 AUTHENTIFICATION CAS

Le Central Authentication Service (CAS) est un système d'authentification unique (SSO) pour le web développé par Shawn Bayern de l'Université Yale, partenaire majeur dans le développement de uPortal. Ce logiciel est implanté dans plusieurs universités et organismes dans le monde.

Référence : https://www.apereo.org/projects/cas

4.4.3 AGREGATION DES TAUX DE DISPONIBILITE

L'objectif est de récupérer le taux de disponibilité d'un jour donnée des ressources en ligne auprès d'Icinga puis de calculer le taux de disponibilité quotidien, mensuel et annuel pour chaque ressource en ligne , métadonnée, catalogue et services techniques.

Ce traitement doit tourner tous les jours pour mettre à jour les indicateurs.

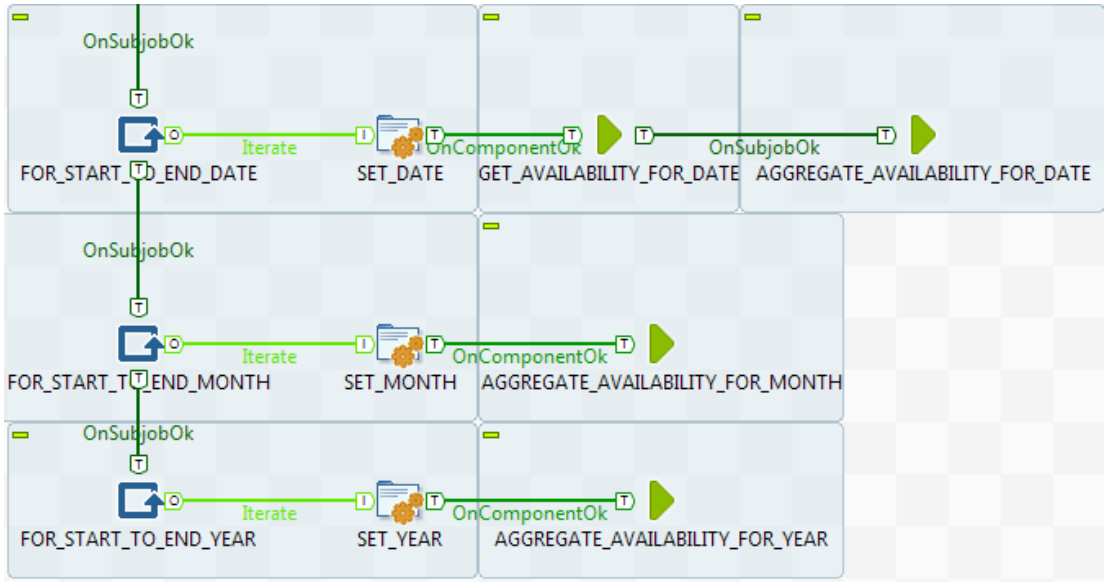
4.4.3.1 CONCEPTION

J'ai développé plusieurs job Talend :

1. Récupération des informations auprès d'Icinga
   - Utilisation de l'API Rest Icinga pour récupérer un flux XML contenant le taux de disponibilité de l'ensemble des ressources en ligne
   - Stockage des informations en base de données
2. Calcul et mise à jour en base des agrégats quotidiens, mensuels et annuels des ressources en ligne
3. Calcul et mise à jour en base des agrégats quotidiens, mensuels et annuels des métadonnées
4. Calcul et mise à jour en base des agrégats quotidiens, mensuels et annuels des catalogues
5. Calcul et mise à jour en base des agrégats quotidiens, mensuels et annuels des services

Ces jobs sont pilotés par un job parent qui permet d'itérer sur une période au besoin.

Voici un extrait du job Talend :



13. Job Talend Main calcul disponibilité

4.4.3.2 MODELISATION DE LA BASE DE DONNEES

Cette table agréage le taux de disponibilité d'une ressource en ligne au niveau jour, mois, année et cela pour chaque objet(catalogue, métadonnée, services..)

object_type	object_id	Period_type	Period_id	Parameter_type	Parameter_value	Process_date
SEXTANT_CATALOG	403	DAY	20150922	MONITORING_AVAILABILITY_RATE	0.994334277620397	2016-07-05 14:07:06.555041+02
SEXTANT_CATALOG	403	YEAR	2016	MONITORING_AVAILABILITY_RATE	0.994330314844476	2016-08-26 03:06:13.339803+02
"SEXTANT_CATALOG"	404	DAY	20150922	MONITORING_AVAILABILITY_RATE	1	2016-07-05 14:07:06.555041+02

4.4.4 RECUPERATION DE L'ETAT COURANT

On avait initialement chargé les informations d'Icinga sous Elasticsearch, via un script Logstash (en chargent les logs Icinga). On s'est aperçu qu'il ne s'agissait pas de la meilleure méthode car Icinga fournit des API REST permettant d'accéder directement à l'information.

On récupère directement l'état depuis l'application web « Sémaphore ».

4.4.5 TRAITEMENT ET CHARGEMENT DES LOGS APACHE

Comme indiqué précédemment dans ce rapport, nous avons essayé, au début de l'alternance, de réaliser ces actions avec la suite ELK, en utilisant notamment logstash. Pour manipuler et charger les logs Apache sous Elasticsearch, j'avais créé :

- Un plugin logstash, afin d'établir une typologie des clients (robot web, outil de monitoring, « vrai » utilisateur).
- Un script Logstash pour traiter et charger les logs apache sous Elasticsearch.

Le volume de chaque fichier de log étant très important nous avons refait cette partie traitement et chargement dans le but d'améliorer les performances et réduire le temps d'exécution.

Après une étude de trois logiciels pour traiter les données (Spark, Drill, Pig). Apache Spark correspondait plus à nos besoins. J'ai utilisé une plateforme déjà existante à l'Ifremer.

4.4.5.1 APACHE SPARK

Spark (ou Apache Spark) est un framework open source de calcul distribué, initialement développé à Berkeley par AMPLab et maintenant un projet de la fondation Apache. Contrairement à Hadoop qui utilise le patron d'architecture MapReduce sur des disques, Spark travaille en mémoire vive ce qui est potentiellement cent fois plus rapide.

Spark reconnait trois langages de programmation (Scala, Java, Python). J'ai décidé de codé en Python, pour découvrir et monter en compétence sur ce langage.

Référence : http://spark.apache.org/

4.4.5.2 CONCEPTION

Les différentes actions à réaliser pour ce traitement :

- Lire les fichiers « bruts »
- Extraire et enrichir les informations
  - Géolocalisation : à partir de l'adresse IP
  - Définition du type de client : robot web, outil de monitoring, application ou vrai utilisateur
  - Identifications des sessions utilisateurs Une session est une période de temps durant laquelle l'utilisateur reste actif c'est-à-dire qu'il est toujours en train de consulter les ressources en lignes.
- Faire le lien avec le référentiel Sextant, pour faciliter la génération d'indicateurs
  Générer de nouveaux fichiers pour exploitation par d'autres traitements

4.4.5.3 UTILISATION DE SPARK AVEC LE LANGUAGE PYTHON

Pour faciliter le développement, j'ai utilisé un environnement de développement web appelé Jupyter.

```
In [216]: Returns a dictionary containing the parts of the Apache Access Log
f parse_apache_log_line(logline):
  from datetime import datetime
  from dateutil.parser import parse

  # apache log pattern
  pattern = '^([^\s]+)\s+([^\s]+)\s+([^\s]+)\s+\[(.*)\]\s+"((?:[^"]|\\")*)"\s+([0-9]*|-)\s+([0-9]*|-)\s+"((?:[^"]|\\")*)"\s+"(

  # execute regexp
  match = re.search(pattern, logline)

  # if pattern doesn't match, throw exception
  if match is None:
      raise Exception("Invalid logline: %s" % logline)

  # if pattern match, prepare and return a dictionnary
  logline_row = {
      'user_ip': match.group(1),
      'user_ident': match.group(2) if match.group(2) != '-' else None,
      'user_auth': match.group(3) if match.group(3) != '-' else None,
      'datetime_raw': match.group(4),
      'request_raw': match.group(5),
      'response_code': int(match.group(6)) if match.group(6) != '-' else None,
      'response_size': long(match.group(7)) if match.group(7) != '-' else None,
      'referrer': match.group(8) if match.group(8) != '-' else None,
      'user_agent_raw': match.group(9)
  }

  # parse date
  parsed_raw_datetime = parse_apache_datetime(logline_row['datetime_raw'])
  if parsed_raw_datetime:
      logline_row.update(parsed_raw_datetime)

  # extract data from request
  parsed_raw_request = parse_apache_log_raw_request(logline_row['request_raw'])
  if parsed_raw_request:
      logline_row.update(parsed_raw_request)
```

14. Parser apache

Ce bloc de code permet à partir d'une ligne de log d'extraire les informations sous la forme d'un dictionnaire.

Une expression régulière nous permet d'extraire les différents champs. Dans les étapes suivantes, on :

- Formate la date
- Décompose le champ «Request» pour en extraire plus d'informations.
- Extrait les informations du champ «User Agent»
- Définit le type de client : un robot, du monitoring ou un utilisateur
- Récupère les informations de géolocalisation à partir de l'adresse IP via la librairie « Geoip2 »

Le bloc de code suivant, montre comment exécuter lire le fichier et utiliser la fonction développé précédemment via l'API spark :

```
In [217]: sc
Out[217]: <pyspark.context.SparkContext at 0x3b79150>

In [218]: apache_logs_rdd = (sc
              .textFile('/home/logsweb/vpublic/2016/access.log.07')
              .map(parse_apache_log_line)
              .cache())

In [219]: %time apache_logs_rdd.first()
          CPU times: user 16 ms, sys: 12 ms, total: 28 ms
          Wall time: 4min 31s

Out[219]: {'client_geoip_country_code_ISO3': u'FR',
           'client_geoip_country_code_name': u'France',
           'client_geoip_position_latititude': 48.4,
           'client_geoip_position_longitude': -4.4833,
           'datetime_raw': u'01/Jul/2016:00:01:45 +0200',
           'referrer': None,
           'request_method': u'GET',
           'request_protocol_version': u'1.1',
           'request_raw': u'GET / HTTP/1.1',
           'request_url': u'/',
           'response_code': 301,
           'response_size': 293L,
           'time_received_isoformat': '2016-07-01T00:01:45',
           'time_received_tz_isoformat': '2016-07-01T00:01:45+02:00',
           'time_received_utc_isoformat': '2016-06-30T22:01:45+00:00',
           'user_agent_raw': u'check_http/v1.5 (nagios-plugins 1.5)',
           'user_auth': None,
           'user_ident': None,
           'user_ip': u'134.246.155.170'}
```

15. Exécution du code Spark en Python

4.5 DEVELOPPMEMENT DE L'APPLICATION WEB

4.5.1 OUTILS UTILISES

Pour développer l'application nous avons décidé d'utiliser l'IDE Eclipse (version Mars) avec pour cible Java 8 et Tomcat 8.

Nous utiliserons également Maven pour gérer les dépendances, ainsi que SVN pour gérer les différentes versions du cde.

4.5.1.1 ECLIPSE MARS

Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la Fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.

J'ai utilisé les plugins subclipse (gestion de SVN) et m2eclipse (pour les actions Maven).

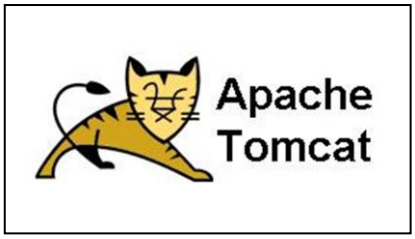
Référence : https://eclipse.org/

4.5.1.2 APACHE

Le logiciel libre Apache HTTP Server (Apache) est un serveur HTTP créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du World Wide Web. Il est distribué selon les termes de la licence Apache. Utilisé en production.

Référence : https://httpd.apache.org/

4.5.1.3 TOMCAT



Apache Tomcat est un conteneur web libre de servlets et JSP Java EE. Issu du projet Jakarta, c'est un des nombreux projets de l'Apache Software Foundation. Il implémente les spécifications des servlets et des JSP du Java Community Process, est paramétrable par des fichiers XML et des propriétés, et inclut des outils pour la configuration et la gestion.

Il comporte également un serveur HTTP que j'utilise en développement.

Référence : http://tomcat.apache.org/

4.5.1.4 MAVEN

Apache Maven est un outil de gestion et d'automatisation de projet logiciel. Il est basé sur le concept d'un modèle d'objet du projet (POM). Chaque projet est configuré par un POM qui contient les informations nécessaires à Maven pour traiter le projet. Ce POM se matérialise par un fichier pom.xml à la racine du projet.

Pour son déploiement Maven peut interagir avec un gestionnaire de code source, un serveur d'intégration continue, un logiciel de surveillance et un repositories manager.

Référence : https://maven.apache.org/

4.5.1.5 SUBVERSION

Subversion (en abrégé svn) est un logiciel de gestion de versions, distribué sous licence Apache et BSD.

Subversion fonctionne donc sur le mode client-serveur, avec un serveur informatique centralisé et unique où se situe les fichiers constituant la référence (le « dépôt » ou « référentiel », ou « repository » en anglais). Un logiciel client, sous forme d'exécutable standalone (ex : SmartSVN) ou de plugin (ex : TortoiseSVN, Eclipse Subversive), permet la synchronisation, manuelle et/ou automatisée, entre chaque client et le serveur de référence.

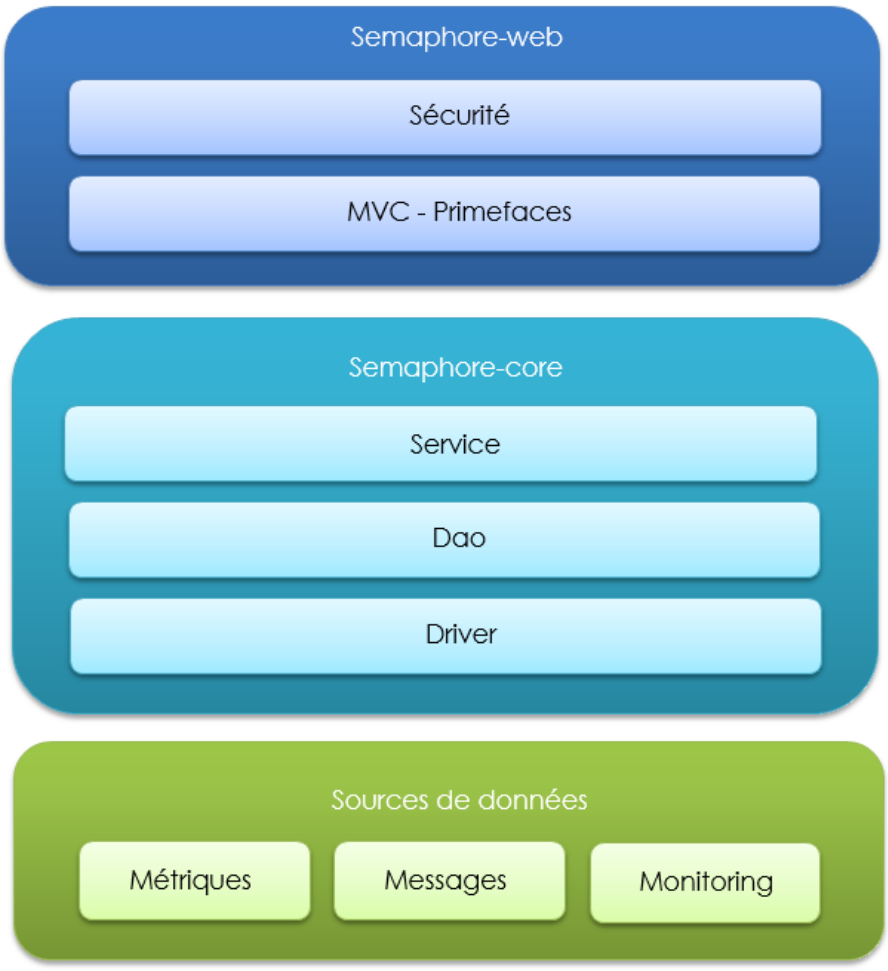
Référence : https://subversion.apache.org/

4.5.2 ARCHITECTURE DE L'APPLICATION

L'application Sémaphore est découpée en deux modules Maven :

- Module « semaphore-core » : gestion de l'accès aux données
- Module « semaphore-web » : application web

Ci-après, un schéma présentant le découpage en couche de l'application.



16. Couches Applicatives

4.5.2.1 MODULE SEMAPHORE-CORE

Ce module fournit l'API pour accéder aux données. Pour cela, j'ai développé trois couches :

- **Driver :** fournit des méthodes pour interroger les différents entrepôts de données (PostgreSQL, Incinga et Elasticsearch)
- **Dao :** fournit les méthodes pour instancier des beans à partir des données présentent dans les différentes sources. Cette couche utilise la couche « Driver ». (ex : SextantCatalogDAO pour récupérer des objets SextantCatalog en passant une requête SQL ainsi qu'un objet permettant d'effectuer le mapping base de données/objet java au le Driver « Base de données »)
- **Service :** fournit des méthodes de haut niveau pour effectuer des traitements sur les données. Cette couche utilise la couche « Dao » (ex : SextantService pour récupérer l'ensemble des données Sextant : catalogue, métadonnée, …)

Nous allons maintenant voir les principaux framework utilisés dans ce module.

4.5.2.2 GESTION DES TRACES APPLICATIVES

La gestion des traces de l'application Sémaphore est effectué grâce au couple **Logback** et **SLF4J.**

Voici comment configurer **SLF4J** pour tracer les messages de niveau DEBUG et supérieur dans la console :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <configuration>
4
5    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
6      <encoder>
7        <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
8      </encoder>
9    </appender>
10
11
12   <logger name="net.sf.ehcache" level="WARN" />
13   <logger name="org.elasticsearch" level="WARN" />
14   <logger name="fr.ifremer.semaphore.core" lavel="DEBUG" />
15
16
17   <!-- Strictly speaking, the level attribute is not necessary since -->
18   <!-- the level of the root level is set to DEBUG by default. -->
19   <root level="DEBUG">
20     <appender-ref ref="CONSOLE" />
21   </root>
22
23 </configuration>
```

17. Fichier de configuration SLF4J

4.5.2.3 CONFIGURATION DE L'APPLICATION

Plutôt que d'utiliser les traditionnels fichiers properties (ensemble de paire clé/valeur), j'ai utilisé une configuration de l'application au format YAML.

YAML est un langage permettant de représenter des données structurées, comme le ferait XML par exemple, mais de manière naturelle et moins verbeuse.

Pour charger la configuration, j'ai utilisé la bibliothèque **SnakeYAML.** Il s'agit d'un outil qui permet de sérialiser/désérialiser un ou plusieurs fichiers YAML dans des objets Java.

Voici un exemple de configuration :

```yaml
# ============================================================
# Build
# ============================================================
build:
  artifact: ${project.artifactId}
  name: ${project.name}
  description: ${project.description}
  version: ${project.version}

# ============================================================
# ELASTICSEARCH
# ============================================================
elasticsearch:
  clusterName: elasticifr
  clusterNodes:
    - visi-common-index1
    - visi-common-cluster1
    - visi-common-cluster2
    - visi-common-cluster3
  repositories:
    sextant:
        index: sextant
        document_type: metadata

# ============================================================
# ICINGA
# ============================================================
icinga:
  credentials:
    enabled: true
    user: icingaadmin
    password: icinga
  repossitory:
    baseurl: http://visi-common-test/cgi-bin/icinga/
    url:
      status: status.cgi?style=detail&jsonoutput

```

18. Fichier de configuration YAML

4.5.3 MODULE SEMAPHORE-WEB

Ce module fournit les interfaces graphiques permettant de consulter les différentes données.

4.5.3.1 IHM

4.5.3.1.1 LISTE DES CATALOGUES

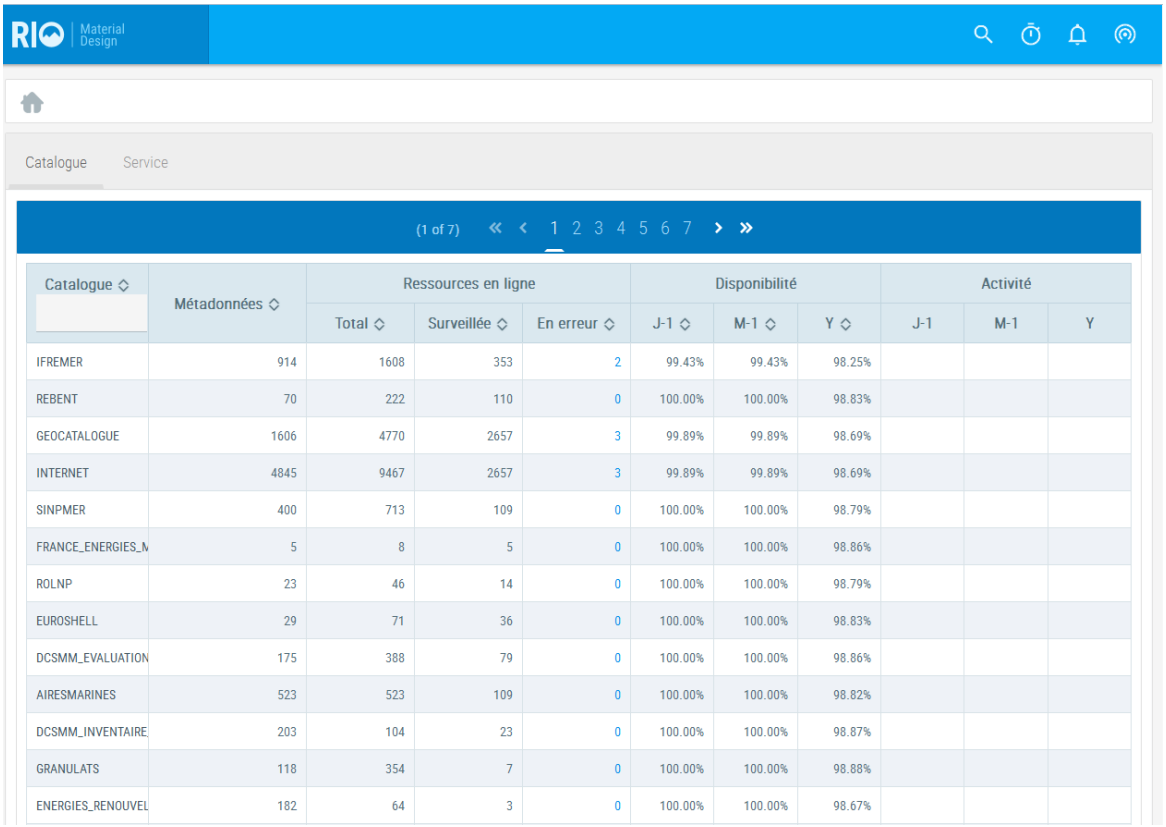
On peut visualiser la liste des catalogues sur la page d'accueil de l'application.

Pour chaque catalogue, l'utilisateur peut :

- Visualiser
  - Le nombre de métadonnées qu'il contient
  - Le nombre de ressources en ligne (« total », « surveillée » et « en erreur »)
  - Le taux de disponibilité de la veille, du mois précédent et année courante
- Cliquer
  - Sur le nom du catalogue pour voir plus en détail ce catalogue
  - Sur le nombre de ressource en ligne pour voir directement le détail des erreurs (quelle ressource en ligne et le message d'erreur).

On s'aperçoit que la partie activité est vide car elle n'a pas pu être développée au cours de l'alternance.

Voici la page d'accueil des catalogues, accessible via l'url **Erreur ! Référence de lien hypertexte non valide.**



Catalogue	Métadonnées	Ressources en ligne			Disponibilité			Activité		
		Total	Surveillée	En erreur	J-1	M-1	Y	J-1	M-1	Y
IFREMER	914	1608	353	2	99.43%	99.43%	98.25%			
REBENT	70	222	110	0	100.00%	100.00%	98.83%			
GEOCATALOGUE	1606	4770	2657	3	99.89%	99.89%	98.69%			
INTERNET	4845	9467	2657	3	99.89%	99.89%	98.69%			
SINPMER	400	713	109	0	100.00%	100.00%	98.79%			
FRANCE_ENERGIES_M	5	8	5	0	100.00%	100.00%	98.86%			
ROLNP	23	46	14	0	100.00%	100.00%	98.79%			
EUROSHELL	29	71	36	0	100.00%	100.00%	98.83%			
DCSMM_EVALUATION	175	388	79	0	100.00%	100.00%	98.86%			
AIRESMARINES	523	523	109	0	100.00%	100.00%	98.82%			
DCSMM_INVENTAIRE	203	104	23	0	100.00%	100.00%	98.87%			
GRANULATS	118	354	7	0	100.00%	100.00%	98.88%			
ENERGIES_RENOUVEL	182	64	3	0	100.00%	100.00%	98.67%			

19. Liste des catalogues

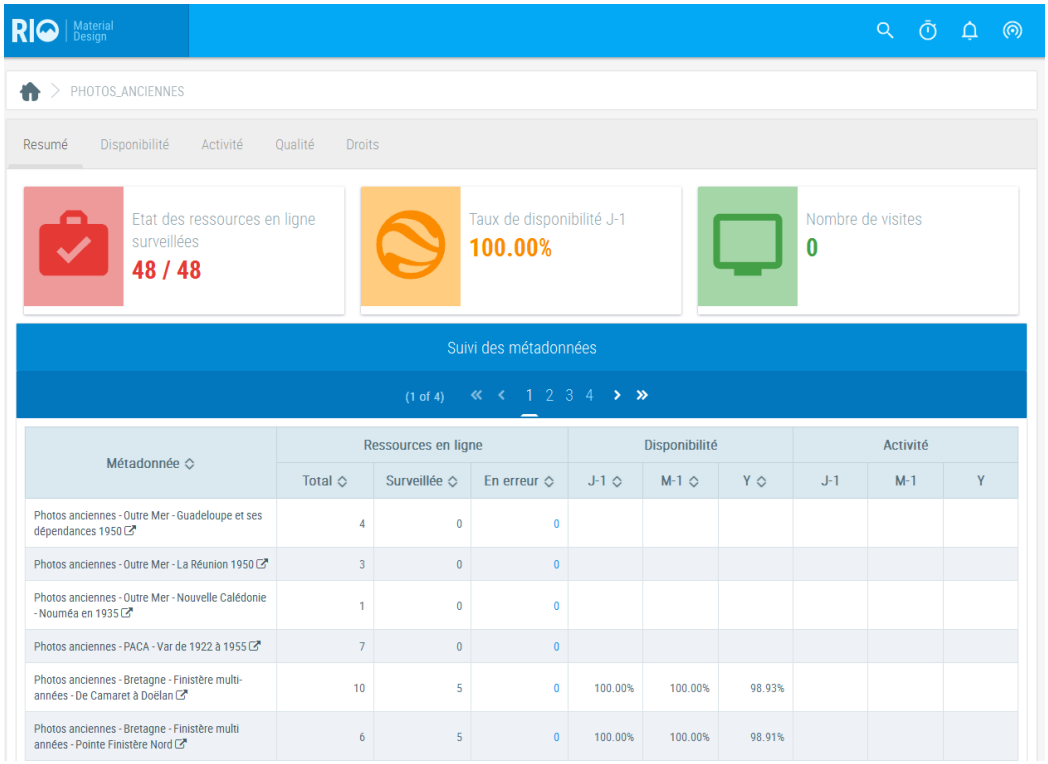
On peut cliquer sur le catalogue que l'on souhaite voir plus en détail. Par exemple le catalogue « PHOTO_ANCIENNES », accessible via l'url :

Erreur ! Référence de lien hypertexte non valide.

4.5.3.1.2 VUE CATALOGUE

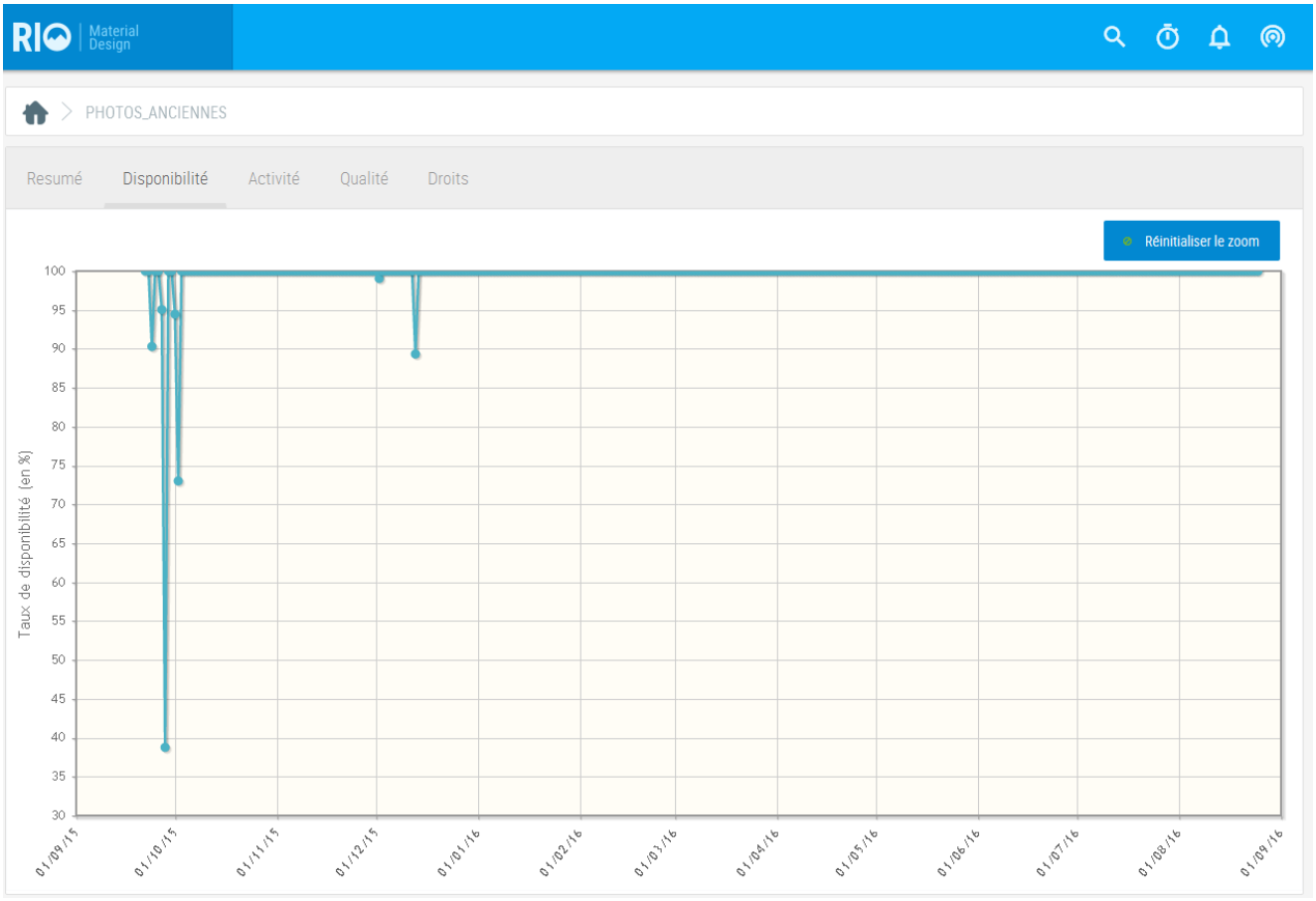
Sur cette page, on voit :

- Un fil d'ariane, permettant de naviguer dans l'application
- Cinq onglets
  - « Résumé » : permet de voir les indicateurs principaux du catalogue ainsi que la liste des métadonnées. En cliquant sur une métadonnée, on arrive sur une page similaire mais présentant la métadonnée et la liste de ces ressources en lignes.
  - « Disponibilité » : permet de voir l'historique des informations de monitoring sous forme de graphique.
  - « Activité » : permet de voir l'historique des informations d'activité (non développé actuellement)
  - « Qualité » : permet de voir en détail si la métadonnée correspond à des normes (ex : norme ISO)
  - « Droits » : permet de savoir qui pourra consulter et modifier la métadonnée. (non développé actuellement)



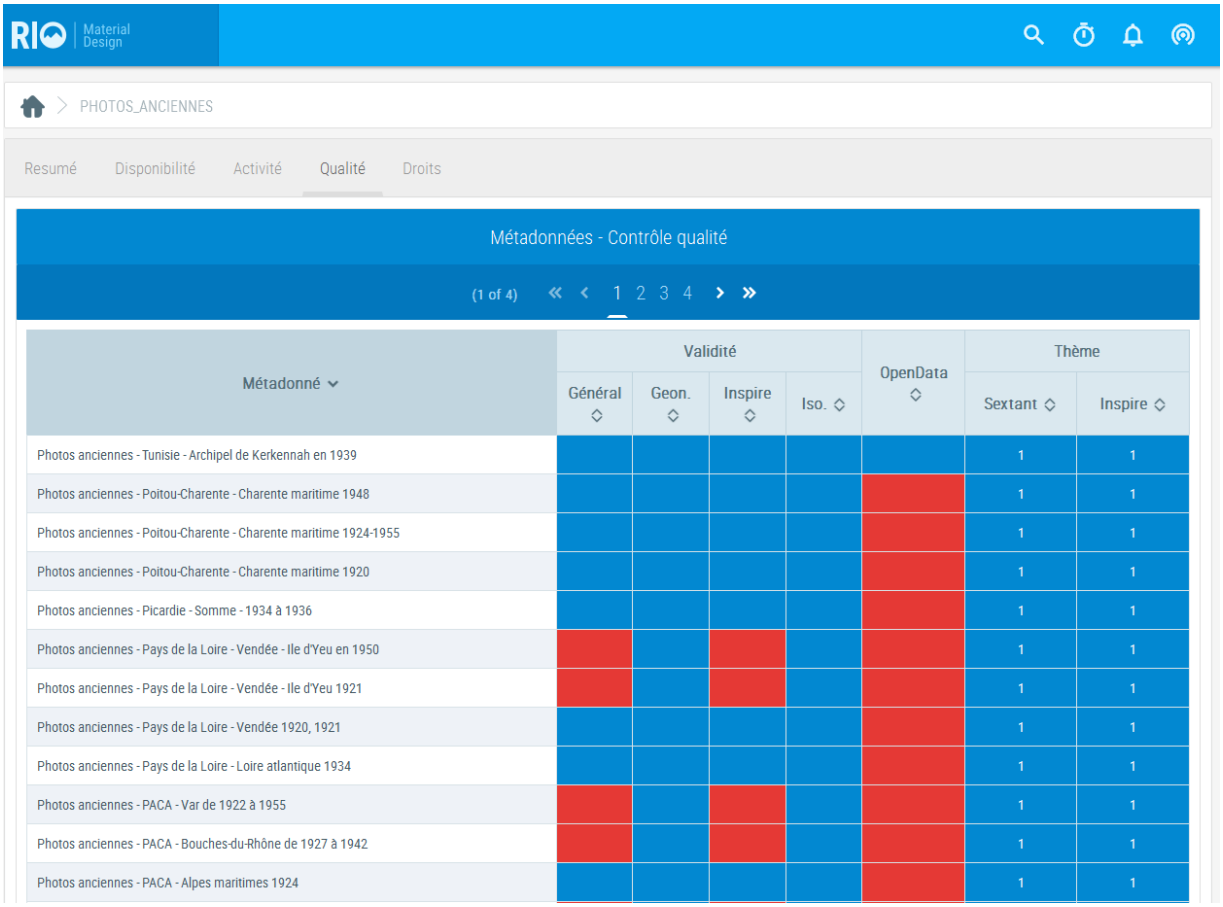
20. Vue catalogue - catalogue PHOTO_ANCIENNES

Voici la vue de l'onglet « Disponibilité » qui est le graphique du taux de disponibilité. On peut faire un zoom sur une période souhaitée en sélectionnant directement cette zone sur le graphique en sachant que par défaut il est affiché sur une année. Un bouton permet de réinitialiser le zoom.



21. Vue catalogue - onglet disponibilité

L'onglet « Qualité » permet de voir les indicateurs qualités de chaque métadonnées du catalogue. Les colonnes « validités » permettent de voir si la métadonnée correspond bien à une norme (ex : la norme « INSPIRE »). On peut voir si la métadonnée possède un thème et si elle en possède un, on peut cliquer pour voir le nom de ce thème.



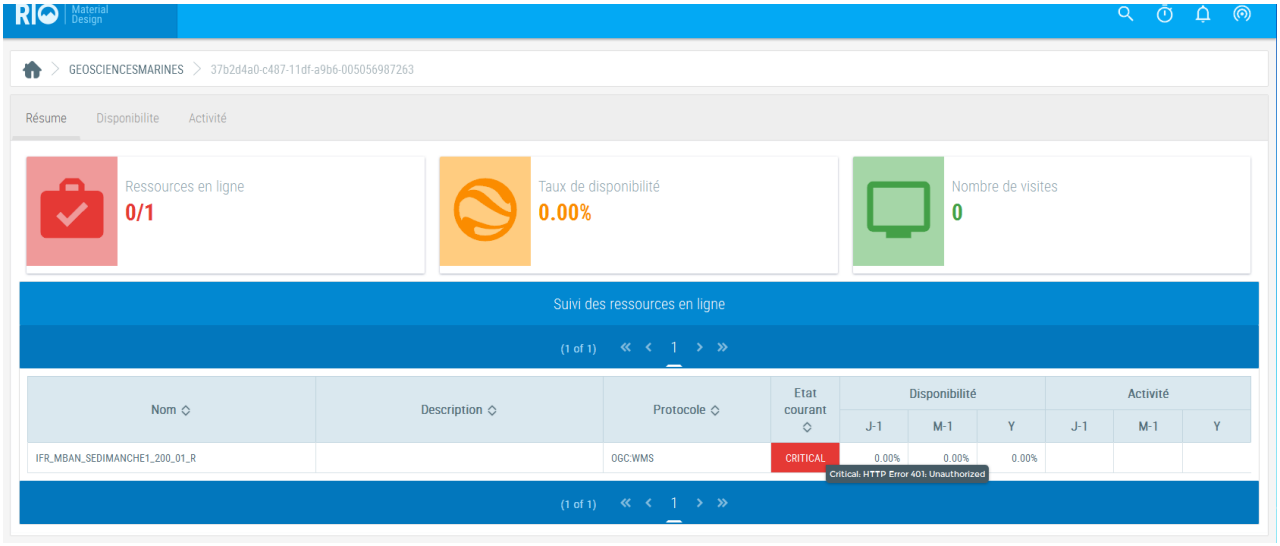
22. Vue catalogue - onglet qualité

4.5.3.1.3 VUE METADONNEE

Au même titre que pour le catalogue, on peut consulter le détail d'une métadonnée, via une url du type :

Erreur ! Référence de lien hypertexte non valide.

Sur l'exemple suivant le taux de disponibilité de la ressource est à 0%. On voit également que son état courant est « CRITICAL », quand l'on glisse la souris sur cet état on voit le détail de l'erreur. Dans ce cas c'est une erreur 401.



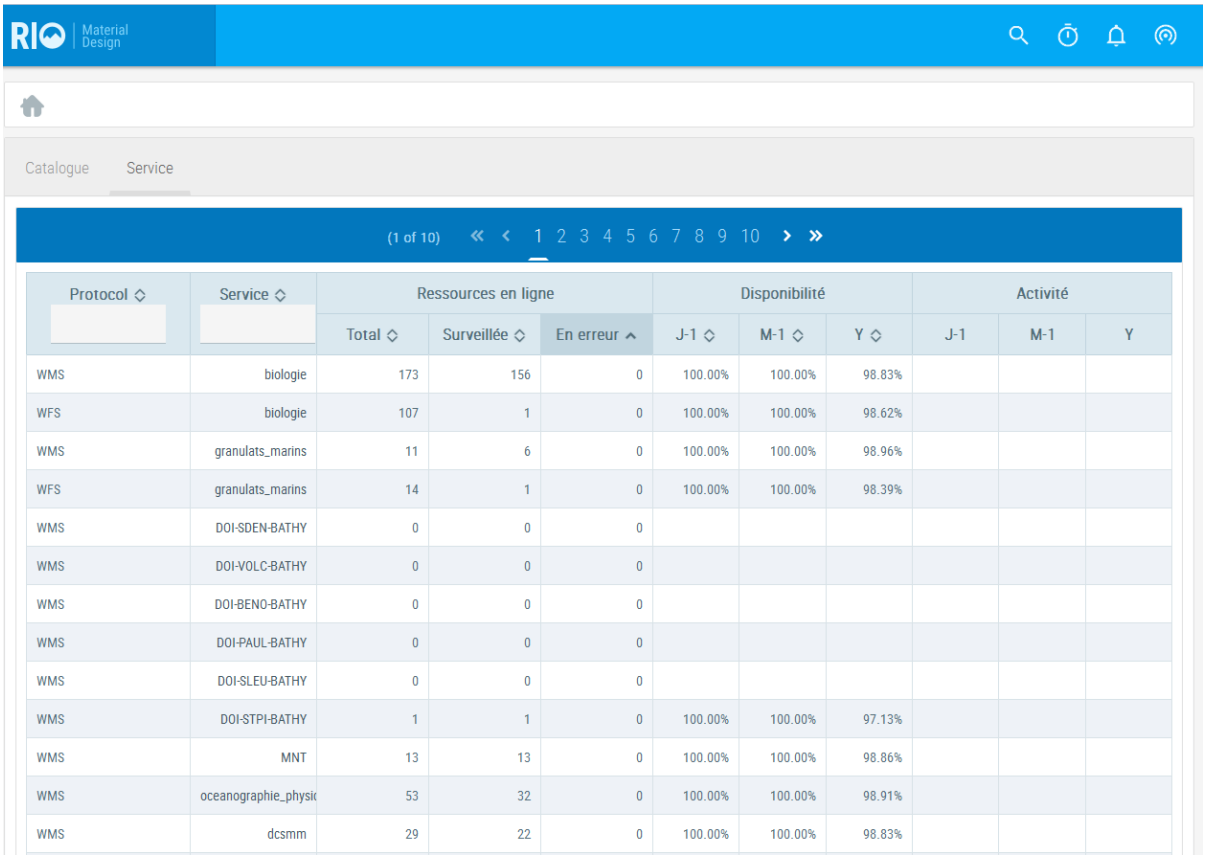
23. Vue métadonnée

4.5.3.1.4 LISTE DES CATALOGUES

Sur la page d'accueil, on peut changer d'onglet pour afficher la liste des ressources en lignes.

Pour chaque ressource en ligne, l'utilisateur peut visualiser le nom et le protocole de chaque service ainsi que les mêmes informations que pour les catalogues (nombre de ressources en ligne et le taux de disponibilité).

On peut voir que certaines ressources en ligne n'ont pas de taux de disponibilité, cela signifie simplement que ce service n'est pas monitoré par Icinga. Comme pour la liste de catalogue la colonne « Activité » est en cours de développement.



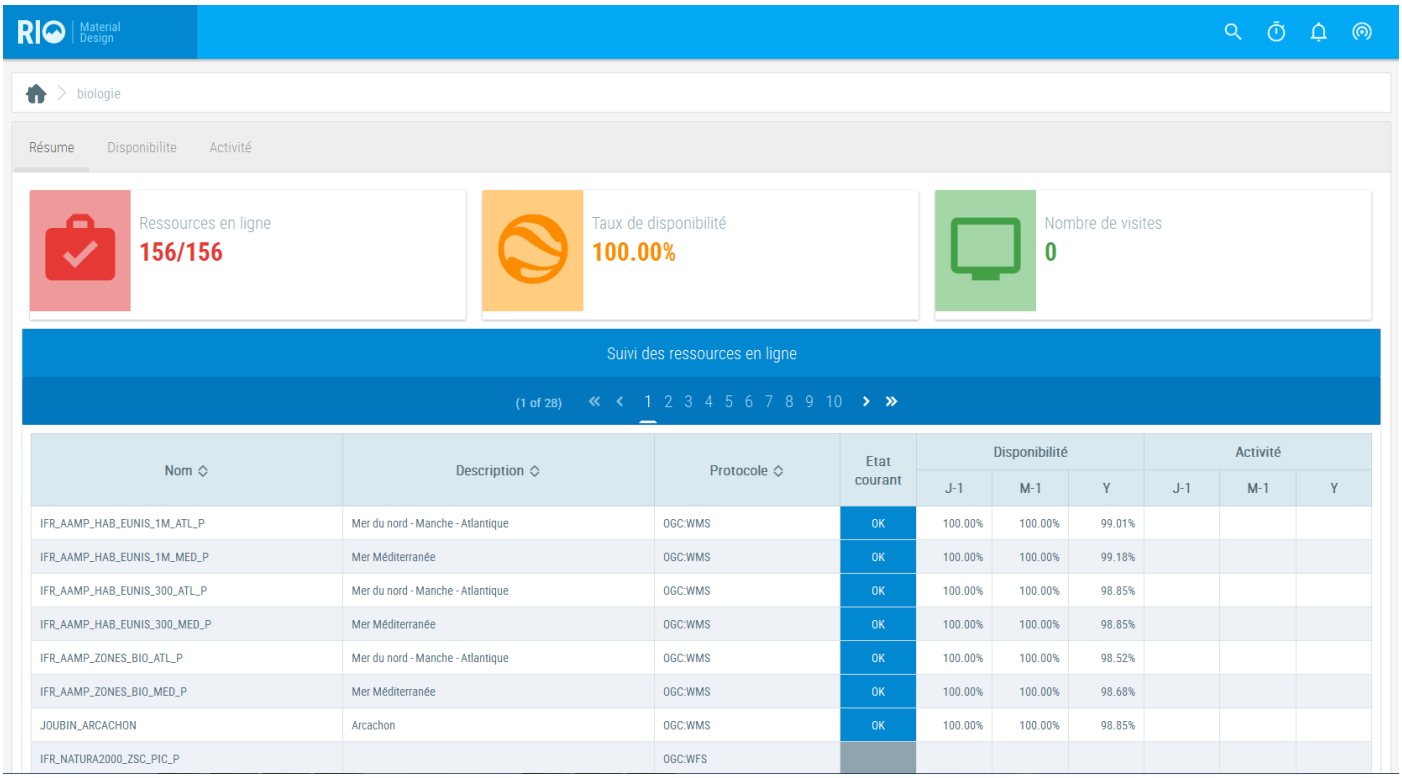
24. Liste des ressources en lignes

4.5.3.1.5 VUE RESSOURCE EN LIGNE

Sur cette page on retrouve toujours un fil d'ariane ainsi que les onglets « Résume », « Disponibilité » et « Activité ».

On y accède via une url du style :

Erreur ! Référence de lien hypertexte non valide.



Nom	Description	Protocole	Etat courant	Disponibilité			Activité		
				J-1	M-1	Y	J-1	M-1	Y
IFR_AAMP_HAB_EUNIS_1M_ATL_P	Mer du nord - Manche - Atlantique	OGC:WMS	OK	100.00%	100.00%	99.01%			
IFR_AAMP_HAB_EUNIS_1M_MED_P	Mer Méditerranée	OGC:WMS	OK	100.00%	100.00%	99.18%			
IFR_AAMP_HAB_EUNIS_300_ATL_P	Mer du nord - Manche - Atlantique	OGC:WMS	OK	100.00%	100.00%	98.85%			
IFR_AAMP_HAB_EUNIS_300_MED_P	Mer Méditerranée	OGC:WMS	OK	100.00%	100.00%	98.85%			
IFR_AAMP_ZONES_BIO_ATL_P	Mer du nord - Manche - Atlantique	OGC:WMS	OK	100.00%	100.00%	98.52%			
IFR_AAMP_ZONES_BIO_MED_P	Mer Méditerranée	OGC:WMS	OK	100.00%	100.00%	98.68%			
JOUBIN_ARCACHON	Arcachon	OGC:WMS	OK	100.00%	100.00%	98.85%			
IFR_NATURA2000_ZSC_PIC_P		OGC:WFS							

25. Vue service – service « WMS biologie »

4.5.3.2 PRINCIPAUX FRAMEWORK

Nous allons maintenant voir les principaux framework utilisés dans ce module.

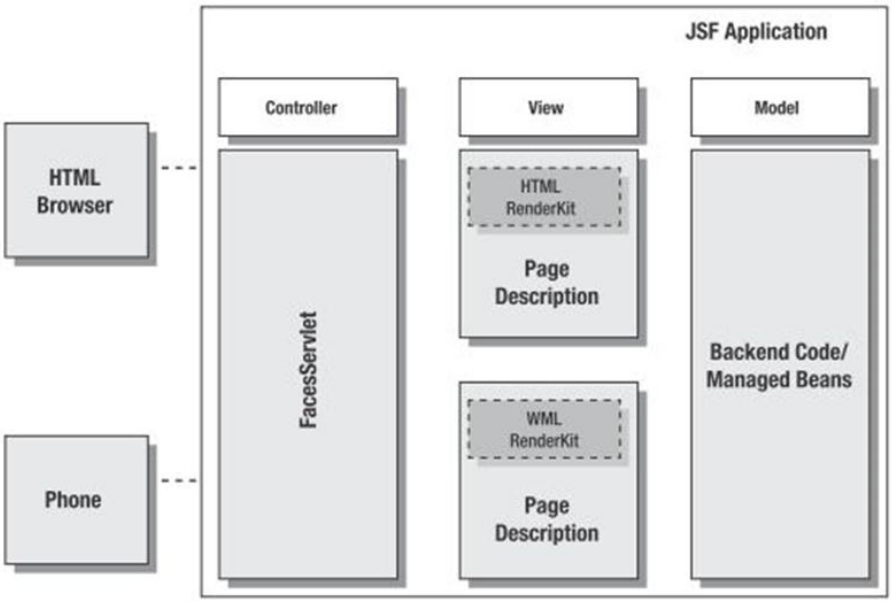
4.5.3.2.1 JSF2.0 – PRIMEFACES

Il a été décidé d'utiliser Primefaces (surcouche JSF2.0) pour la réalisation des IHM. Nous avons également acheté le thème Primefaces «RIO » pour ne pas perdre de temps sur le design de l'IHM.

Le framework JSF2 se base sur le pattern MVC (Modèle – Vue – Contrôleur).

Voici comment cela fonctionne :

- La vue se charge du rendue de la page envoyée au client
- Le contrôleur vérifie la cohérence des données
- Le modèle s'occupe des transactions avec les données



26. Modèle MVC

4.5.3.2.2 GESTION DES URL

Pour que l'application Sémaphore fournisse des URL compréhensibles par les utilisateurs, on a créé une classe qui permet de faire la redirection.

Voici la configuration :

```
package fr.ifremer.semaphore.web.common.rewrite;

import javax.servlet.ServletContext;


@RewriteConfiguration
public class SemaphoreRewriteConfigurationProvider extends HttpConfigurationProvider
{

    private static final Logger logger = LoggerFactory.getLogger(SemaphoreRewriteConfigurationProvider.class);

       @Override
       public int priority()
       {
         return 10;
       }

    @Override
    public Configuration getConfiguration(ServletContext context) {
        logger.debug("Load pretty faces configuration");
        return ConfigurationBuilder.begin()
                // A basic join
                .addRule(Join.path("/sextant/").to("/pages/sextant/index.xhtml"))

                .addRule(Join.path("/sextant/catalog/{name}").to("/pages/sextant/catalog.xhtml"))

                .addRule(Join.path("/sextant/catalog/{name}/metadata/{uuid}").to("/pages/sextant/metadata.xhtml"))

                .addRule(Join.path("/sextant/service/{protocolType}/{protocol}").to("/pages/sextant/service.xhtml"));
    }
}
```

27. Classe URL REST

4.5.3.2.3 GESTION DE LA SÉCURITÉ

Dans l'éventualité d'une gestion des droits d'accès sur l'application, j'ai intégré la bibliothèque **spring-security.** Cette bibliothèque permet de définir les modes d'authentification (http, LDAP, SSO, …) et les droits d'accès à différents ressources.

Plutôt que de configurer spring-security dans un fichier SML, j'ai opté pour une configuration par classe Java annoté.

Ci-après un exemple de configuration de spring-security :

```
1  package fr.ifremer.dashboard.web.security;
2
3  import org.slf4j.Logger;
12
13 @EnableWebSecurity
14 public class DashboardSecurityConfig {
15
16     private static final Logger logger = LoggerFactory.getLogger(DashboardSecurityConfig.class);
17
18     @Autowired
19     public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
20         auth
21             .inMemoryAuthentication()
22                 .withUser("user").password("password").roles("USER").and()
23                 .withUser("admin").password("password").roles("ADMIN", "USER");
24
25     }
26
27     @Configuration
28     @Order(1)
29     public static class LandingWebSecurityConfigurationAdapter extends WebSecurityConfigurerAdapter{
30         protected void configure(HttpSecurity http) throws Exception {
31             http
32                 .authorizeRequests()
33                     .antMatchers("/semaphore/**").access("hasRole('USER')")
34                     .and().formLogin()
35                     .and().httpBasic()
36                     .and().exceptionHandling().accessDeniedPage("/error.xhtml")
37                     .and().csrf().disable().logout().logoutUrl("/404.xhtml").logoutSuccessUrl("/messages.xhtml");
38
39         }
40     }
41 }
```

28. Configuration de la sécurité

5 BILAN

Actuellement une version exploitable des tableaux de bord est disponible auprès des administrateurs de Sextant. Elle permet de consulter des informations relatives à la qualité de la saisie des métadonnées dans le logiciel Sextant et également de consulter les informations de monitoring (état courant et taux de disponibilité).

N'ayant pas pu aller au bout de l'analyse des technologies big-data pour exploiter les logs applicatifs, le tableau de bord ne permet pas encore de consulter les informations d'activité.

De même, certaines demandes exprimées mais ne rentrant pas dans le cadre de ce sujet (ex : consultation des droits associés aux différentes catalogues) n'ont pas été développées.

6 CONCLUSION

Mon alternance dans une des équipes de l'Ifremer m'a permis de travailler dans un domaine qui me plaît particulièrement à savoir le développement logiciel.

Cette alternance m'a permis d'appréhender les technologies nécessaires au développement logiciel et ainsi pouvoir mettre en place des tableaux de bords.

J'ai amélioré ma compétence en travail d'équipe en collaborant avec mon maitre d'alternance. Cette expérience d'un an m'a également permis de me faire une nouvelle expérience du monde de l'entreprise.

Enfin sur le plan humain, l'alternance s'est particulièrement bien déroulée. L'intégration dans l'équipe a été rapide.

7 BIBLIOGRAPHIE ET WEBOGRAPHIE

- Primefaces : http://www.primefaces.org/showcase/
- Java Doc : https://docs.oracle.com/javase/8/docs/api/
- ELK : https://www.elastic.co/
- Spark : http://spark.apache.org/
- Ifremer : http://wwz.ifremer.fr/